X-Hypercubes의 연결성과 그 응용

권 경 희† Zheng, Si-Qing††

요 약

Hypercubes와 유사한 구조를 가진 X-hypercubes는 hypercubes와 같은 수의 node와 edge를 갖고 있다. 그러나 node들을 연결하는 방법을 약간 바꾸어 줌으로써 X-hypercubes는 diameter를 hypercubes의 약 반으로 줄일 수 있었다. 따라서 X-hypercubes 내의 node들 간의 통신시의 delay는 hypercubes의 그것보다 훨씬 적어지는 것을 기대할 수 있다.

본 논문에서는 X-hypercubes를 새롭게 정의함으로써 두 node들 간의 연결에 관한 조건들을 명확히 해 준다. 이 정의에 대한 응용으로서, 본 논문은 hypercubes를 X-hypercubes로 그리고 X-hypercubes를 hypercubes로 embedding 시키는 algorithm을 보여준다. 이는 이들 두 network에서 운용되는 program들이 최소한의 overhead 만으로써 서로 호환될 수 있음을 말 해준다. 또한 본 논문은 hypercubes 에서의 bitonic merge sort를 simulate 함으로써, X-hypercubes 에서도 운용될 수 있는 bitonic merge sort도 보여주고 있다.

Connectivity of X-Hypercubes and Its Applications

Kyung Hee KWON† and Si-Qing Zheng††

ABSTRACT

The hypercube-like interconnection network, X-hypercubes, has the same number of nodes and edges as conventional hypercubes. By slightly changing the interconnecton way between nodes, however, X-hypercubes reduces the diameter by almost half. Thus the communication delay in X-hypercubes can be expected to be much lower than that in hypercubes.

This paper gives a new definition of X-hypercubes establishing clear-cut condition of connection between two nodes. As application examples of the new definition, this paper presents simple embeddings of hypercubes in X-hypercubes and vice versa. This means that any programs written for hypercubes can be transported onto X-hypercubes and vice versa with minimal overhead. This paper also present bitonic merge sort for X-hypercubes by simulating that for hypercubes.

1. Introduction

Many multi-processor computers have been proposed over the last decade. Among these, the hypercube machine has been recognized as one of the most important parallel computers due to its high-bandwidth, logarithmic diameter and regular topological properties. In particular, many of other networks can be embedded in it.

Recently a slightly modified hypercube network called *X-hypercubes* was introduced [4]. *X-hypercubes* have the same structural complexity as conventional hypercubes, i.e. a n-dimensional *X-hypercubes* has the same number of nodes, and the same number of links per nodes as hypercubes, but the diameter of a *X-hypercubes* is about the half of the paramter of hypercubes of the same dimen-

† 정 회 원 : 단국대학교 전자계산학과 전임강사
†† 비 회 원 : Louisiana State Univ. Associate Professor
논문접수 : 1994년 2월 3일, 심사완료 : 1994년 3월 7일

sion. This implies that *X-hypercubes* have the advantage over hypercube when data communication is of major concern, especially under the condition of improving the system performances without or with a little increase in the system costs. Indeed, as shown in [7], the data broadcasting and census operations on a *X-Hypercube* takes about half of data communication steps of the same operations performed on a hypercube.

It is well known that for multi-processor systems, the data communication cost dominates the computation cost. Therefore, it is worthwhile to make comparative studies on hypercubes and *X-hypercubes*, and explore the advantages provided by *X-hypercubes*.

When a hypercube machine of dimension is abstracted as a graph, processors are treated as vertices and data links are treated as edges, where each vertex is given a unique label and the connectivity between vertices can be easily determined by inspecting the labels associated with vertices. In contrast, *X-hypercubes* are less regular. In fact, the original definition of *X-hypercubes* appeared in [4] is not so formal. In [7], a formal definition of *X-hypercubes* is introduced. However, this definition does not provide explicit conditions for the connectivity of vertices. The analysis of the algorithmic aspects and topological properties of *X-hypercubes* in [7] are based on a notion array arrangement of vertices, which is used to derive the connections between vertices. In addition to the inconvinience, finding the connections by using the array arrangement involves computing exponents.

Compared to hypercubes, one of major disadvantages of *X-hypercubes* is the fact that it is hard to use due to its more complicated connectivity than that in hypercubes. In this paper, we give an alternative definition for *X-hypercubes*. We show that using this definition, the connectivity between any two vertices in *X-hypercubes* can be easily determined by scanning the labels of the vertices. We also show how to use this definition to implement simulations between hypercubes and *X-hypercubes*.

2. Definitions of X-hypercubes

A n-dimensional *X-hypercubes*, which is denoted as Q_n^T, is a graph of 2^n vertices. To simplify our presentation, we define the n-dimensional companion *X-hypercubes*, denoted as Q_n^C, in parallel with Q_n^T. Each node in or Q_n^C is labeled by a distinct n-bit binary number in B_n, by which we denote the set of all possible n-bit binary numbers $b_n b_{n-1} \cdots b_1$. We use # to denote the concatenation operation on two binary numbers, i.e., for two binary numbers b_1 and b_2, $b_1 \# b_2$ is the binary number of $|b_1| + |b_2|$ bits obtained by concatenating them, where $|b|$ is the number of bits in b. We use b# B_n, to denote the set of binary numbers obtained by concatenating the binary number b with all numbers in B_n, i.e. b# $B_n = \{b' \mid b' \in B_n\}$. The formal recursive definition of Q_n^T (and Q_n^C) given in [7] is as follows:

Definition 1:

$Q_1^T = (V_1^T, E_1^T)$, where

$V_1^T = \{0, 1\}$; and

$E_1^T = \{[0,1]\}$.

$Q_1^C = (V_1^C, E_1^C)$ is identical to Q_1^T.

For $n > 1$ and n is odd,

$Q_n^T = (V_n^T,\ E_n^T)$, where

$V_n^T = B_n$; and

$$E_n^T = \{[0\# v_i, 0\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^T\}$$
$$\cup \{[1\# v_i, 1\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^C\}$$
$$\cup \{(0\# v_i, 1\# v_j) \mid v_i, v_j \in B_{n-1} \text{ and } v_i = v_j\}.$$

$Q_n^C = (V_n^C,\ E_n^C)$, where

$V_n^C = B_n$; and

$$E_n^C = \{[0\# v_i, 0\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^C\}$$
$$\cup \{[1\# v_i, 1\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^T\}$$
$$\cup \{[0\# v_i, 1\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } v_i = v_j\}.$$

For $n > 1$ and n is even,

$Q_n^T = (V_n^T,\ E_n^T)$, where

$V_n^T = B_n$; and

$$E_n^T = \{[0\# v_i, 0\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^T\}$$
$$\cup \{[1\# v_i, 1\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^C\}$$
$$\cup \{[00\# v_i, 10\# v_j], [01\# v_i, 11\# v_j] \mid v_i, v_j \in B_{n-2} \text{ and } v_i = v_j\}.$$

$Q_n^C = (V_n^C,\ E_n^C)$, where

$V_n^C = B_n$; and

$$E_n^C = \{[0\# v_i, 0\# v_j], [1\# v_i, 1\# v_j] \mid v_i, v_j \in B_{n-1} \text{ and } [v_i, v_j] \in E_{n-1}^T\}$$
$$\cup \{[00\# v_i, 11\# v_j], [01\# v_i, 10\# v_j] \mid v_i, v_j \in B_{n-2} \text{ and } v_i = v_j\}.$$

In figure 1 and figure 2, we show several hypercubes and *X-hypercubes* of low dimensions. For reasons that will soon be apparent, we define a finite state automata $A_n = (S, B, T, q_0, F)$, where $S = \{T_e, T_o, C_e, C_o\}$ is the set of states in A_n; $B = \{0, 1\}$ is the input alphabet; q_0 is the initial state ($q_0 = T_e$ if n is even, and T_0 if n is odd); T is the transition function $S \times B$, and $F = S$ is the set of final states. The transition function T is defined in the transition diagram shown in figure 3. One additional constraint on A_n is that the binary strings that can be accepted by A_n have length no longer than n. We define prefix(n, v;d) as the substring of v obtained by deleting the rightmost d bits of v. We say that *prefix*(n, v;d) is of type T_e (resp. T_0, C_e and C_0) if by left-to-right scanning prefix(n,v;d) the state T_e (resp. T_0, C_e and C_0) of A_n is reached after $n-d$ state transitions. For two distinct binary strings $u = u_n u_{n-1} \cdots u_1$ and $v = v_n v_{n-1} \cdots v_1$, we define d($n$,$u$,$v$) as the maximum i such that $u_i \neq v_i$.

Definition 2:

X-hypercube of dimension n is a graph with 2 ^{n} vertices, each is labeled by a distinct n-bit binary number, and any two vertices $u = u_n u_{n-1} \cdots u_1$ and $v = v_n v_{n-1} \cdots v_1$ are connected by an edge if and only if one of the following two conditions holds:

(1) *prefix*(n,u;d(n,u,v)) is of type C_e, $u_{d(n,u,v)} u_{d(n,u,v)-1} = \overline{v_{d(n,u,v)}}\ \overline{v_{d(n,u,v)-1}}$ and $u_i = v_i$ for $i \neq d(n,u,v)$, and $i \neq d(n,u,v)$a$-$1;

(2) *prefix*(n,u;d(n,u,v)) is not of type C_e $u_i = v_i$ for$\neq d(n,u,v)$.

To verify the equivalence of definition 1 and definition 2, let us look closely at the structure of X-hypercubes. For Q_n^T, we called the subgraph induced by a vertex subset $\{b_n b_{n-1} \cdots b_1 \mid b_n b_{n-1} \cdots b_{d+1} = c_n c_{n-1} \cdots c_{d+1}\}$, where $c_n c_{n-1} \cdots c_{d+1}$ is a constant and $1 \leq d < n$, as a *d-dimensional subcube of* Q_n^T. Clearly, Q_n^T is recursively defined by its subcubes. We say that a *d-dimensional subcube of* Q_n^T induced by a vertex subset $\{b_n b_{n-1} \cdots b_1 \mid b_n b_{n-1} \cdots b_{d+1} = c_n c_{n-1} \cdots c_{d+1}\}$ is of type $T_e(C_e)$ if by looking at the d least significant bits of the labels, the connections of vertices satisfy the definition of $Q_d^T(Q_d^C)$ and d is an even number. Similarly, we define types T_o and C_e of d-dimensional subcubes of Q_n^T. Note that the only difference between T_o and $T_e(C_o$ and $C_e)$ is that d is an odd number for $T_o(C_o)$. Directly following definition 1, we know that two vertices u and v of Q_n^T are connected by an edge if and only if they are connected by an edge in the subcube of the smallest dimension that contains both of u and v. Thus, the problem of determining whether or not u and v are connected is reduced to determining whether or not they are connected in the smallest subcube containing them.

It is easy to see that $d(n,u,v)$ indicate the dimension of the smallest subcube of Q_n^T that contains u and v, and the type of $prefix(n,v; d(n,u,v))$ tells the type of such a subcube. By definition 1, we conclude that

Theorem 1:
Definition 1 and definition 2 for X-hypercubes are equivalent.

It should be pointed out that the type of $prefix(n,u;d)$ can be effectively computed using A_n. To determine the type of $prefix(n, u;d)$, we need to scan $u=u_n u_{n-1} \cdots u_1$ from left to right and make $n-d$ state transitions in A_n. The automata A_n, together with the notion $prefix(n,v;d)$, is not only useful for determining whether or not two vertices in Q_n^T are connected by an edge, it can also be used efficiently solving the following decision problems:

(i) Given a vertex u in Q_n^T, determine all its adjacent vertices in Q_n^T;

(ii) Given a vertex $u = u_n u_{n-1} \cdots u_1$ in Q_n^T, determine the vertex $v = v_n v_{n-1} \cdots v_1$ that is connected to u such that $u_i = v_i$ for $n \geq i > d$ and $u_d \neq v_d$; and

(iii) Given a vertex u in Q_n^T, determine the types of all subcubes of Q_n^T that contain u.

All these operations are useful for investigating the algorithmic aspects and combinatoric structures of X-Hypercube machines. For example, we may define an edge connecting two vertices $u = u_n u_{n-1} \cdots u_1$ and $v = v_n v_{n-1} \cdots v_1$ in Q_n^T such that $u_i = v_i$ for $n \geq i > d$ u_d as a *d-dimensional edge* of Q_n^T. The operation (ii) can be used to find all d-dimensional edges of Q_n^T. The above listed operations are very useful for divide-and-conquer paradigm for designing efficient parallel algorithms on X-Hypercube machines, as indicated in the previous investigations on conventiional hypercube machines.

3. Application Examples

In this section, we show how the new definition can be used to derive results in the computational aspects of X-hypercubes and conventional hypercubes. Firstly, let us consider embeddings between hypercubes and X-hypercubes.

Let G and H be two simple undirected graphs. An *embedding* of G in H is a one-to-one mapping of the vertices of G into the vertices of H, together with a specification of paths in H connecting the images of the endpoints of each edge in G. The *dilation* of the embedding is the maximum length of these paths in H and the *congestion* of the embedding is the maximum number of edges of G whose corresponding mapped paths in H include a single edge in H. Graph embeddings can be used for a model simulating one computer architecture by another. The parameters dilation and congestion are used to measure the efficiencies of such simulations. The following algorithm embeds Q_n into Q_n^T.

```
procedure EMBED1( Q_n,  Q_n^T)
   for every edge [x, y] do
      d = d(n,x,y);
      if the type of prefix(n,x:d) is C_r then
         case ( x_d x_{d-1}, y_d y_{d-1}) of
            (00, 10) : z_d z_{d-1} = 01;
            (01, 11) : z_d z_{d-1} = 10;
         endcase
         let z_i = x_i for i ≠d and i ≠d-1.
         asociate [x,z] and [z,y] in Q_n^T with [x,y] in Q_n;
      else
         associate [x,y] in Q_n^T with [x,y] in Q_n;
      endif
   endfor
end EMBED1
```

For embedding Q_n^T into a Q_n, we give the following algorithm:

```
procedure EMBED2( Q_n^T,  Q_n)
   for every edge [x, y] do
      d = d(n,x,y);
      if the type of prefix(n,x:d) is C_r then
         case ( x_d x_{d-1}, y_d y_{d-1}) of
            (01, 10) : z_d z_{d-1} = 00;
            (00, 11) : z_d z_{d-1} = 10;
         endcase
         let z_i = x_i for i ≠d and i ≠d-1;
         asociate [x,z] and [z,y] in Q_n with [x,y] in Q_n^T;
      else
         associate [x,y] in Q_n with [x,y] in Q_n^T;
      endif
   endfor
end EMBED2
```

Theorem 2 : Q_n can be embedded into Q_n^T with dilation 2 and congestion 2 and Q_n^T can be embedded into Q_n with dilation 2 and congestion 2.

Proof:

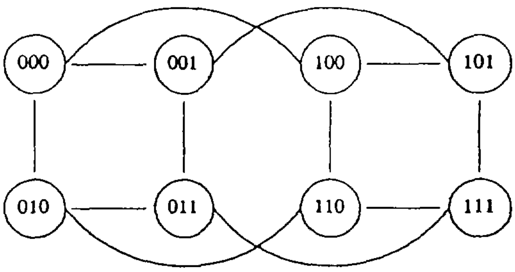
Since the proof for two parts of the theorem are similar, we only give the proof for the first part, i.e. Q_n can be embedded into Q_n^T with dilation 2 and congestion 2. Obviously, the embedding constructed by algorithm *EMBED1* is of dilation 2. By algorithm *EMBED1*, we know that any d-dimensional edge in Q_n is either mapped to the edge in Q_n^T connecting the vertices with the same labels, or mapped to two d-dimensional edges in a d-dimensional subcube of type C_t of Q_n^T that contains x and y. Thus, we only need to consider mappings of d-dimensional edges $[x, y]$ to d-dimensional edges. The edge $[u,v]$ in Q_n^T, where $u_d u_{d-1}=01$, $v_d v_{d-1}=10$, and $u_i=v_i$ for $i \neq d$ and $i \neq d-1$, is used exactly twice in the embedding, and the edge $[u, v]$ in Q_n^T, where $u_d=0$, $v_d=1$, and $u_i=v_i$ for $i \neq d$, is also used exactly twice in the embedding. Therefore, the congestion of the embedding constructed by algorithm *EMBED1* is 2.

By theorem 2, we know that any algorithm on hypercube machines can be implemented on X-Hypercube machines with the same performance. For example, sorting 2^n numbers on a n-dimensional hypercube machine requires $O(n^2)$ time. This can also be achieved on a n-dimensional X-hypercube by simulation shown below. The symbol $\Leftarrow$ denotes such a communication of a data item from an adjacent processor's local memory into the active local memory.

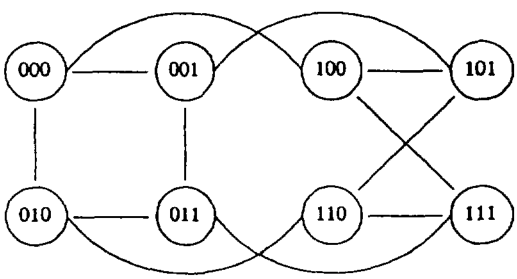
```
procedure BITONIC MERGE SORT
   for i=0 to n-1 do
      for j=i downto 0 do
         d= 2^j;
         for all P_k where 0 ≤k ≤ 2^n-1 do
            if k mod 2d <d then
               if prefix(n,k:j+1) is of type C_e then
                  q= 2^(j-1) ;
                  t_{k+q} ← a_{k+d};
                  t_k ← t_{k+q};
               else
                  t_k ← a_{k+d};
               endif
               if k mod 2^(i+2) < 2^(i+1) then
                  b_k=max( t_k, a_k);
                  a_k=min( t_k, a_k);
               else
                  b_k=min( t_k, a_k);
                  a_k=max( t_k, a_k);
               endif
            if k mod 2d ≥d then
               if prefix(n,k:j+1) is of type C_e then
                  q= 2^(j-1) ;
                  t_{k-q} ← b_{k-d};
                  a_k ← t_{k-q};
               else
                  a_k ← b_{k-d};
               endif
            endif
         endfor
      endfor
   endfor
end BITONIC MERGE SORT
```

4. Concluding Remarks

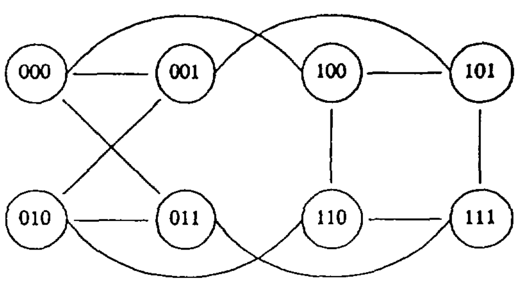
We presented an alternative formal definition for X-hypercubes. As the definition of conventional hypercubes, this concise definition explicitly provide the conditions of connectivity of vertices. As examples, we showed how to derive simple proofs of some known results on embeddings between hypercubes and X−hypercubes. We also showed how to use the connectivity conditions given in the definition to express the bitonic merge sort algorithm for X-hypercubes. We believe that this new definition will be very useful for further research in the parallel computation on X-Hypercube interconnection networks and multi-processor computer systems.



a) 3-dimensional hypercube

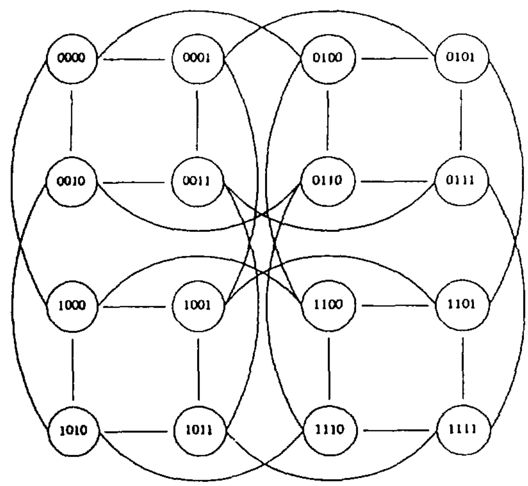


b) Q_3^T

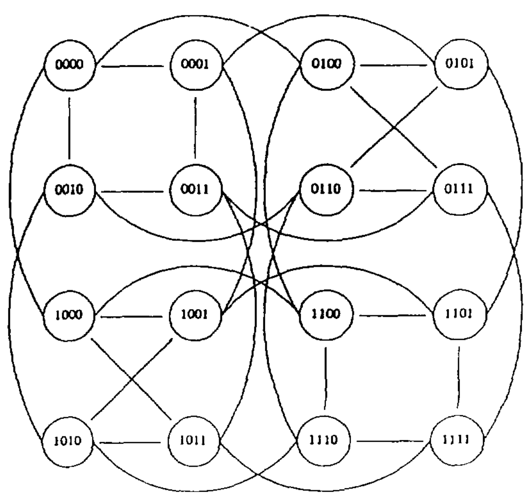


c) Q_3^C

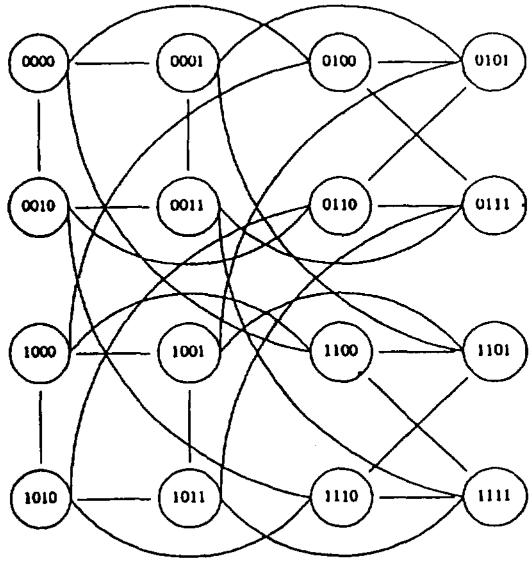
(Fig. 1) 3-dimensional hypercube, Q_3^T and Q_3^C



a) 4-dimensional hypercube

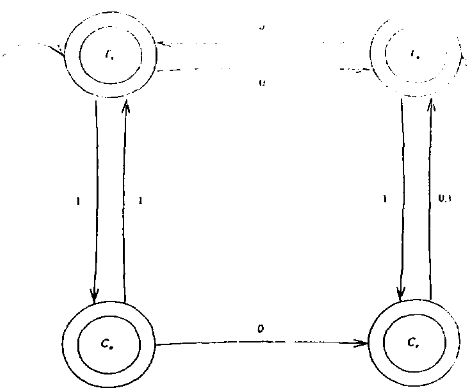


b) Q_4^T



c) Q_4^C

(Fig. 2) 4-dimensional hypercube, Q_4^T and Q_4^T



(Fig. 3) Automata A_n

References

[1] Bhuyan, L.N and D. P. Agrawal, "*Generalized Hypercube and Hyperbus structures for a Computer Network*", IEEE Transactions on Computers, Vol. c-33, No. 4, pp. 323-333, April 1984.

[2] Knuth, D., "*The Art of Computer Programming*, Vol.3, Sorting and Searching", 722 pages, Addison Wesley, 1973.

[3] Quinn, M.J., "*Designing Efficient Algorithms for Parallel Computers*", 288 pages McGraw-Hill,1987.

[4] Sung, Y.Y, "*X-hypercube: A Better Interconnection Network*", Proc. 26th Annual Southeast Regional ACM Conf., pp. 557-561, 1988.

[5] Saad, Y. and Schultz M. H., "Topological Properties of Hypercubes", IEEE Transactions on Computers, Vol. 37, No. 7, pp. 867-872, July 1988.

[6] Ullman, J.D., *The Computational Aspects of VLSI*, 495 pages, Computer Science Press,1984.

[7] Zheng, S.-Q., "Data Broadcasting and Census Algorithms for Twisted Hypercubes" Technical Report #89-011, Dept. of Computer Science, Louisiana State Univ.,1989.

권 경 희

1976년 고려대학교 물리학과 졸업 (학사)

1986년 Old Dominion Univ. Computer Science Dept. (M.S.)

1991년 Louisiana State Univ. Computer Science Dept. (Ph.D.)

1979～1984년 한국산업연구원 연구원

1992년～현재 단국대학교 전자계산학과 전임강사

관심분야 : 병렬 및 분산 처리, 알고리즘 분석 및 설계, 네트워크

Zheng Si-Qing

1973년 중국 길림대학교 전기공학과 졸업 (학사)

1982년 Univ. of Texas, Dallas Mathematical Sience Dept. (M.S.)

1987년 Univ. of California, Santa Barbara Computer Science Dept. (Ph.D.)

1987～1993년 Louisiana State Univ. Assistant Professor

1993년～현재 Louisiana State Univ. Associate Professor

관심분야 : V.L.S.I., Computational Geometry, 병렬 및 분산 처리, 네트워크