An Algorithm for Constructing Hamiltonian Cycle in Metacube Networks

Yamin Li and Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{yamin, speng}@k.hosei.ac.jp

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

Abstract

The high-performance supercomputers will consist of several millions of CPUs in the next decade. The interconnection networks (INs) in such supercomputers play an important role. Metacube (MC) is an attractive IN that can connect extremely large number of nodes with small number of links, meanwhile it holds a short diameter and keeps the simplicity of routing algorithm. An MC(k, m) network can connect 2^{m2^k+k} nodes with $m+k$ links per node, where k is the dimension of the high-level cubes (classes) and m is the dimension of the low-level cubes (clusters). For example, an MC(3,3) with 6 links per node can connect 2^{27}, or 134,217,728, nodes. In this paper, we show that the Metacube is Hamiltonian and give an efficient algorithm to construct a Hamiltonian cycle in Metacube networks.

1. Introduction

The modern high-performance supercomputers consist of hundreds of thousands of CPUs. In the near future, the number of CPUs in supercomputers will reach to several millions. How to connect these extremely large number of CPUs is an important issue for achieving high performance of the supercomputers. A "good" interconnection network should use a small number of links and meanwhile keep the diameter as shorter as possible. The symmetric structure and efficient routing should also be considered.

The binary hypercube has been widely used as the interconnection network in a wide variety of parallel systems such as Intel iPSC, the nCUBE [4], the Connection Machine CM-2 [10], and SGI Origin 2000 [9]. A hypercube network of dimension n, or n-cube, contains up to 2^n nodes and has n edges per node. If a unique n-bit address is assigned to each node of the hypercube, then an edge connects two nodes if and only if their binary addresses differ in a single bit. Because of its elegant topological properties and the ability to emulate a wide variety of other frequently used networks, the hypercube has been one of the most popular interconnection networks for parallel computer/communication systems.

However, the conventional hypercube has a major shortage, that is, the number of edges per node in a system increases logarithmically as the total number of nodes in the system increases. Since the number of links is limited to eight per node with current IC technology, the total number of nodes in a hypercube parallel computer is restricted to several hundreds. Therefore, it is interesting to develop an interconnection network which keeps most of topological properties of the hypercube, and has more nodes in the system than the hypercube with the same number of links per node.

Several variations of the hypercube have been proposed in the literature [1, 2, 3, 8, 11]. Among these variations, only the CCC (cube-connected cycles) can keep the number of links per node within a small constant. However, CCC suffers from quite large diameter, that is about twice of the diameter of hypercubes.

A new interconnection network for extremely large parallel systems called *Metacube* (MC) has been introduced recently [7] [5]. The MC network shares many desired properties of the hypercube, and can be used as an interconnection network for a parallel computer system of almost unlimited size with just a small number of links per node. For example, an MC(2,3) with 5 links per node has 16384 nodes, and an MC(3,3) with 6 links per node has $2^{27} = 134,217,728$ nodes. Compared with the CCC, the MC has shorter diameter, length of the routing path, and the broadcasting time.

A *Hamiltonian cycle* of an undirected graph G is a simple cycle that contains every node in G exactly once. A *Hamiltonian path* in a graph is a simple path that visits every node exactly once. A Hamiltonian path can be obtained from a Hamiltonian cycle by removing any one link from that cycle. A graph that contains a Hamiltonian cycle is said to be *Hamiltonian*. In this paper, we show how to construct a Hamiltonian cycle in Metacube networks.

The rest of this paper is organized as follows. Section 2 describes the Metacube network architecture. Section 3 gives an algorithm for constructing a Hamiltonian cycle in Metacube networks. Section 4 concludes the paper and presents some future research directions.

2. Metacube Interconnection Networks

The MC network has a 2-level cube structure: a high-level cube represented by the leftmost k bits of the binary node address of $m2^k + k$ bits (these k bits serve as a class indicator); and low-level cubes, called clusters that forms the basic components in the network, represented by the m bits of the remaining $m2^k$ bits, which occupy the different portions of the $m2^k$ bits for the different classes.

More specifically, there are two parameters in an MC network, k and m. An MC(k, m) contains 2^k *classes*. Each class contains $2^{m(2^k-1)}$ *clusters*, and each cluster contains 2^m *nodes*. Therefore, an MC(k, m) uses $m2^k + k$ bits to identify a node and the total number of nodes is 2^n where $n = m2^k + k$. The value of k affects strongly the growth rate of the size of the network. An MC$(1, m)$ containing 2^{2m+1}nodes is called a *dual-cube*. Similarly, an MC$(2, m)$, an MC$(3, m)$, and an MC$(4, m)$ containing 2^{4m+2} nodes, 2^{8m+3} nodes, and 2^{16m+4} nodes, are called *quad-cube*, *oct-cube*, and *hex-cube*, respectively. Since an MC$(3, 3)$ contains 2^{27} nodes, the oct-cube is sufficient to construct practically parallel computers of very large size. The hex-cube is of theoretical interest only. Note that an MC$(0, m)$ is a hypercube.

The $(m2^k + k)$-bit node address in MC(k, m) is divided into three parts: a k-bit class ID, an $m(2^k - 1)$-bit cluster ID, and an m-bit node ID. The leftmost k-bit number defines a class of clusters (class ID). There are 2^k classes. Of each class, there are $2^{m(2^k-1)}$ clusters, an $m(2^k - 1)$-bit number identifies a cluster of the class (cluster ID). An m-bit number, located in a special portion of the $m2^k$-bit (will be explained in the next paragraph) identifies a node within the cluster (node ID).

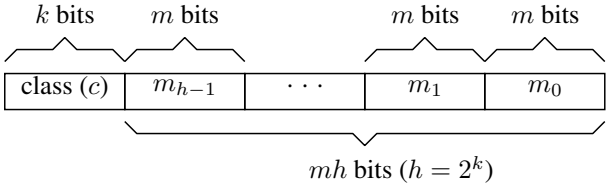


Figure 1. Format of a node address for an MC(k, m)

In the following discussion, we use the format of $(c, m_{h-1}, \ldots, m_1, m_0)$ to denote the node address where $h = 2^k$, c is the k-bit class ID, m_c is the m-bit node ID, and $(m_{h-1}, \ldots, m_{c+1}, m_{c-1}, \ldots, m_0)$ is the $m(2^k - 1)$-bit cluster ID. Figure 1 shows the format of the address.

An MC(k, m) is constructed with the following method. Within a cluster, the m-bit node ID forms an m-cube with m links or *cube-edges*. For a cluster of class i ($i = 0, 1, \ldots, h-1$), the node ID within the cluster is m_i, while $(m_{h-1}, \ldots, m_{i+1}, m_{i-1}, \ldots, m_0)$ is the cluster ID. The links that connect nodes in different clusters are defined as follows. For any two nodes whose addresses differ only in a bit position in the class field, there is a link, a *cross-edge*, connecting these two nodes. This is, the k-bit in the class field defines a high-level k-cube which connects those nodes whose addresses except class field are the same. A cluster then can be considered as a low-level m-cube.

The addresses of two nodes connected by a cross-edge differ only in one bit within the k-bit class field and there is no direct connection among the clusters of the same class. Therefore, a node in an MC(k, m) has $m + k$ links: m links construct an m-cube cluster and k links construct a k-cube. For example, the three neighbors within the cluster of the node with address (01,111,101,110,000) in an MC$(2, 3)$ have addresses (01,111,101,111,000), (01,111,101,100,000), and (01,111,101,010,000). The underlined bits are those that differ from the corresponding bits in the address of the referenced node. The two neighbors in the high-level cube are (00,111,101,110,000) and (11,111,101,110,000).

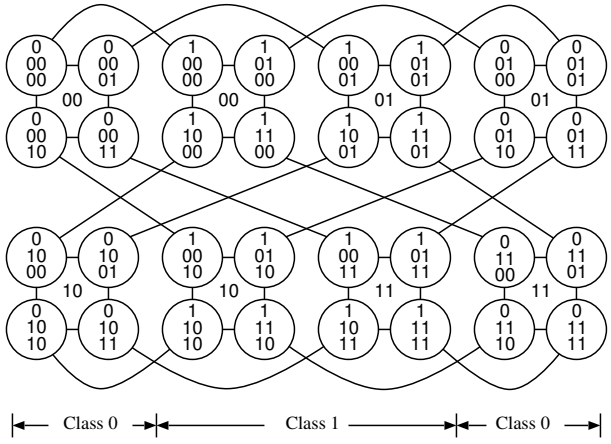


Figure 2. A Metacube MC(1,2)

Figure 2 shows the structure of an MC$(1, 2)$, where the cluster is a 2-cube and there are two classes. Each node has a cross-edge attached to a node of the other class. The binary number shown in the center of a cluster is the cluster ID. Figure 3 shows the structure of an MC$(2, 2)$, where the clusters in the same square are of the same class. In Fig-

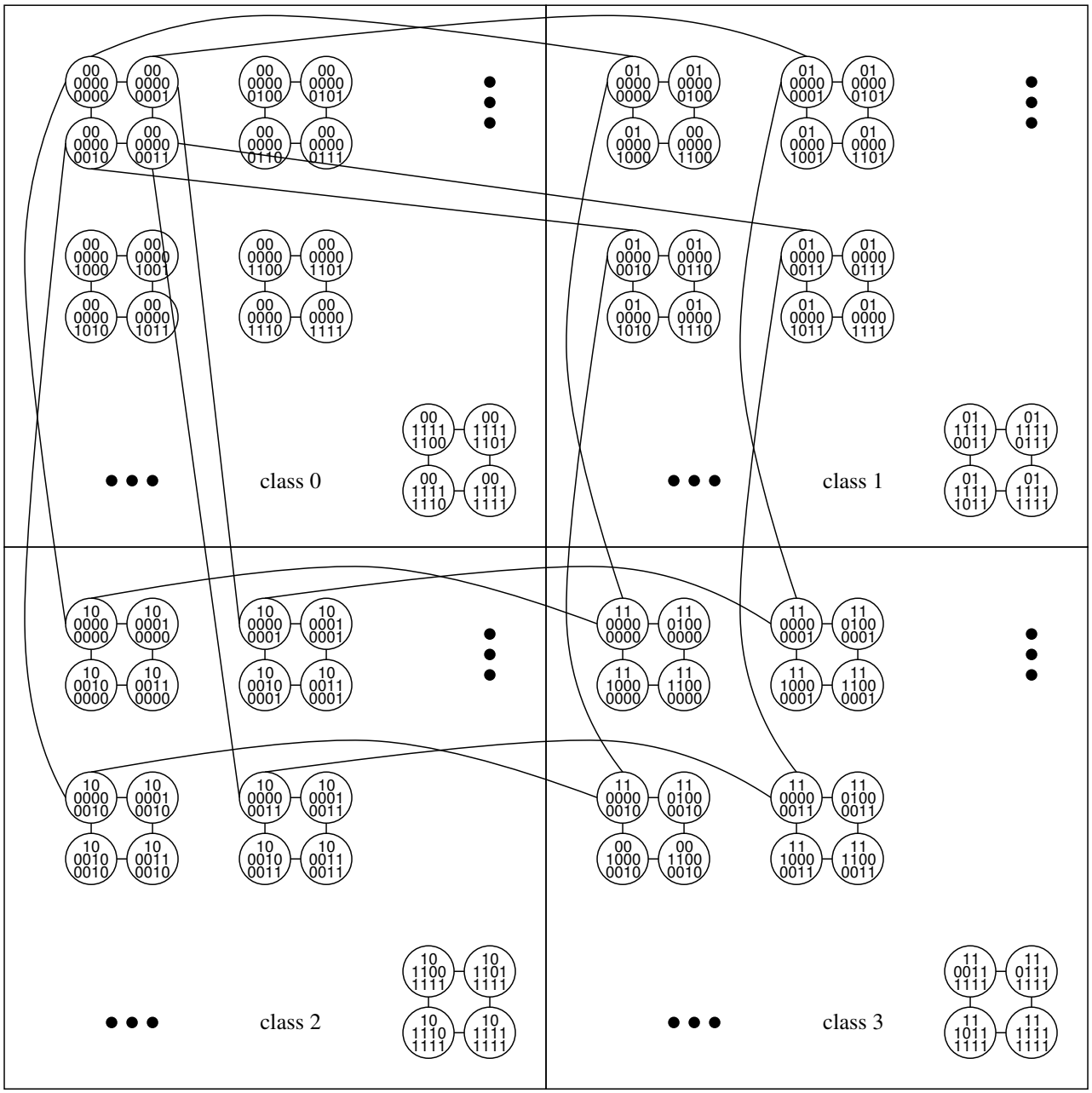


Figure 3. A Metacube MC(2,2)

ure 3, there are $2^{2(2^2-1)} = 64$ clusters in each square, and each cluster is a 2-cube. The figure shows only 4 high-level cubes which contain 4 nodes in the cluster 0 of the class 0, respectively.

The ratio of the total number of links in a hypercube to the total number of links in an MC network is equal to $n/(m+k)$, where $n = m2^k+k$. For example, for $k = 2$ and $m = 3$ ($n = 14$), each of the two networks contains 16384 nodes: a hypercube contains $2^{14} \times 14/2 = 114688$ links, whereas an MC network contains $2^{14} \times (3+2)/2 = 40960$ links. The reduction in the total number of links for this example is 73728 links or about 64%.

Table 1 lists the number of nodes a Metacube can connect. We can see that an MC$(2, 3)$ which requires 5 links per node is sufficient for building an extremely large parallel computer system. To build a system of the same size with a hypercube, each node requires 14 links. The diameter of an MC$(2, 3)$ is 16, only 2 large than that of the hypercube.

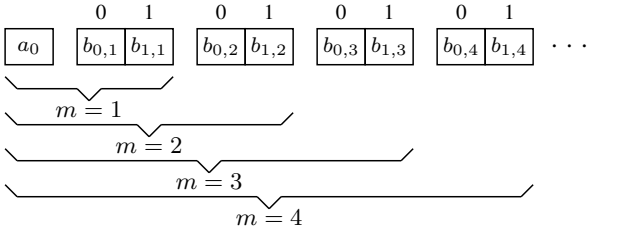
Table 1. Total number of nodes

Links/node	3	4	5	6	7	8
Hypercube	8	16	32	64	128	265
MC$(1, m)$	32	128	512	2,048	8,192	32,768
MC$(2, m)$	64	1,024	16,384	2^{18}	2^{22}	2^{26}
MC$(3, m)$	–	2,048	2^{19}	2^{27}	2^{35}	2^{43}
MC$(4, m)$	–	–	2^{20}	2^{36}	2^{52}	2^{68}

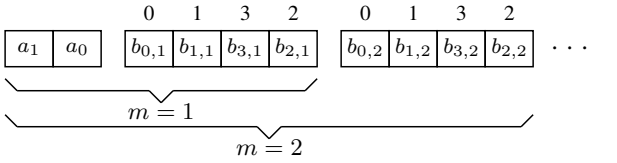
3. Hamiltonian Cycle of Metacube

An algorithm for constructing a Hamiltonian cycle on MC$(1, m)$, or dual-cube, was given in [6]. The algorithm given in this paper is not an extension of [6]; we use a complete new algorithm to construct a Hamiltonian cycle on a general Metacube, MC(k, m). In the following discussion, we start from $k = 2$. Note that the algorithm is also applicable to $k = 1$.

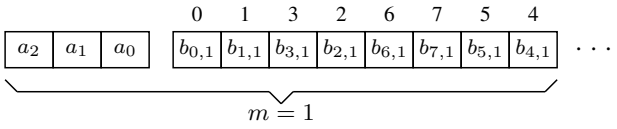
In the following discussion, we use the format of node address as shown as in Figure 4.



(a) $k = 1$



(b) $k = 2$



(c) $k = 3$

Figure 4. Node address format of MC(k, m)

The k-bit class ID is $a_{k-1} \dots a_0$. The rest (cluster ID and node ID) are divided to m parts and each part contains 2^k bits. We use $b_{i,j}$ to denote a bit in these IDs where $i = 0, 1, \dots, 2^k - 1$ and $j = 1, 2, \dots, m$. Note that the positions of the IDs' bits in each part are organized in the order of a Reflected Gray Code. For examples, the IDs' bit order in an MC$(2, m)$ is $0, 1, 3, 2$ and the order in an MC$(3, m)$ is $0, 1, 3, 2, 6, 7, 5, 4$. For a class $a = a_{k-1} \dots a_0$, the m-bit node ID within the low-level m-cube is $b_{a,1}, b_{a,2}, \dots, b_{a,m}$ and the rest bits, $\{b_{i,j}\}|(i \neq a)$, serve as the cluster ID.

The algorithm for constructing the Hamiltonian cycle on Metacube can be described as the following 3 stages:

1. Find a Hamiltonian cycle on k-cube with the Reflected Gary Code.
2. Find a Hamiltonian cycle on MC$(k, 1)$ by incrementally extending the cycle in 1 by routing additional 2-bit per step: We cut the cycle generated in stage 1 along a selected edge and make 4 copies, each with different ID at that 2 bits. Then we link the 4 paths to generate a cycle.
3. Find a Hamiltonian cycle on MC(k, j), $j = 2, \dots, m$, by incrementally extending the cycle on MC$(k, j - 1)$. While extending the cycle on MC$(k, j - 1)$, we work on the additional 2^k bits, one bit per step.

A step example of constructing a Hamiltonian cycle on an MC$(2, 3)$ is shown in Table 2.

Table 2. The steps of Hamiltonian cycle construction

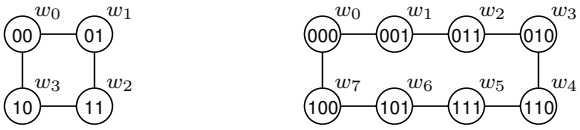
ID: a_1a_0 $b_{0,1}b_{1,1}b_{3,1}b_{2,1}$ $b_{0,2}b_{1,2}b_{3,2}b_{2,2}$ $b_{0,3}b_{1,3}b_{3,3}b_{2,3}$		
1	Build a Hamiltonian cycle on k-cube	
	1.1	Route a_1a_0
2	Build a Hamiltonian cycle on MC$(k, 1)$	
	2.1	Route $b_{0,1}, b_{1,1}$
	2.2	Route $b_{3,1}, b_{2,1}$
3	Build a Hamiltonian cycle on MC(k, m)	
	3.1	Build a Hamiltonian cycle on MC$(k, 2)$
		3.1.1 Route $b_{0,2}$
		3.1.2 Route $b_{1,2}$
		3.1.3 Route $b_{3,2}$
		3.1.4 Route $b_{2,2}$
	3.2	Build a Hamiltonian cycle on MC$(k, 3)$
		3.2.1 Route $b_{0,3}$
		3.2.2 Route $b_{1,3}$
		3.2.3 Route $b_{3,3}$
		3.2.4 Route $b_{2,3}$

3.1. Hamiltonian Cycle of Hypercube

We use $0^{\{i\}}$ to denote a bit pattern $0 \dots 0$ of i bits. The Hamiltonian cycle in an n-cube can be constructed by the binary Reflected Gray Code. A Gray code for binary numbers is a listing of all n-bit numbers so that successive numbers, including the first and last, differ in exactly one bit

position. The best known example of the Gray codes is the binary Reflected Gray Code which can be described as follows. If $P(n)$ denotes the listing for n-bit numbers, then $P(1) = (0, 1)$. For n greater than 1, $P(n)$ is formed by taking the list for $P(n-1)$ with each number prefixed by 0 then following that list by the reverse of $P(n-1)$ with each number prefixed by 1. For example, $P(2) = (00, 01, 11, 10)$, $P(3) = (000, 001, 011, 010, 110, 111, 101, 100)$, and so on. Since the first and last numbers of $P(n)$ also differ in one bit position, the code is in fact a cycle. In an n-cube, there is a link connecting two nodes if their numbers differ in one bit position: connecting the adjacent nodes, also the first and last nodes, in the binary Reflected Gray Code list with links, a Hamiltonian cycle is formed.

First, we generate a Hamiltonian cycle C_0 on a k-cube by using a Reflected Gray Code such that node 0 and node 1 are neighbors. Starting from node 0, in the direction to node 1, we name the nodes in the cycle $w_0,\ w_1,\ \ldots,\ w_{2^k-1}$, as shown as in Figure 5.



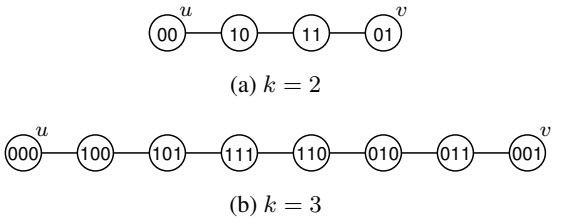
(a) $k = 2$ (b) $k = 3$

Figure 5. C_0: Hamiltonian cycle on a k-cube

3.2. Hamiltonian Cycle of MC$(k, 1)$

Now, we got a cycle of length 2^k. Next, consider the cluster ID and node ID of MC$(k, 1)$. Because $m = 1$, there are totally 2^k bits. We use a loop to generate a Hamiltonian cycle on MC$(k, 1)$ by working on 2 bits of cluster IDs in each iteration of the loop. Therefore, there are 2^{k-1} iterations in total.

We use i to count the iterations. For $i = 1$, generate a Hamiltonian path P_0 by removing edge (u, v) from C_0, where $u = w_0$ and $v = w_1$. For examples, $u = 00$, $v = 01$ for $k = 2$ and $u = 000$, $v = 001$ for $k = 3$, as shown as in Figure 6.



(a) $k = 2$

(b) $k = 3$

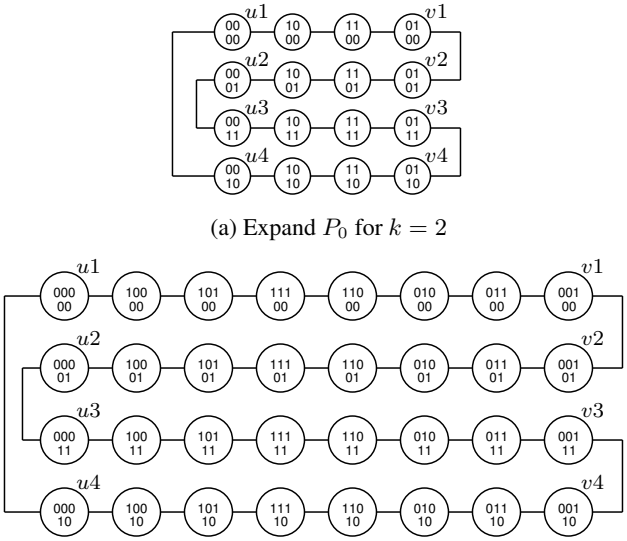
Figure 6. P_0: Removing an edge from C_0

The two terminate nodes in P_0 are nodes 0 and 1, which are the class IDs in an MC$(k, 1)$. From these two terminate nodes, we can expand the path to form a larger cycle. Because the class IDs are 0 and 1, we can modify the bits $b_{0,1}$ and $b_{1,1}$ (node IDs) to do the expansion. This is the reason why 2 bits are dealt with in each iteration.

Now we route 2 bits $b_{0,1}b_{1,1}$. Four paths $Q_1,\ Q_2,\ Q_3$, and Q_4 are generated based on P_0 by attaching 2 bits 00, 01, 11, and 10 to each node address in P_0, respectively:

$Q_1 = P(u_1, v_1)$, where $u_1 = u\ 00$ and $v_1 = v\ 00$;
$Q_2 = P(v_2, u_2)$, where $v_2 = v\ 01$ and $u_2 = u\ 01$;
$Q_3 = P(u_3, v_3)$, where $u_3 = u\ 11$ and $v_3 = v\ 11$;
$Q_4 = P(v_4, u_4)$, where $v_4 = v\ 10$ and $u_4 = u\ 10$;

Then, a new cycle, C_1, can be generated: $C_1 = Q_1 \cup (v_1, v_2) \cup Q_2 \cup (u_2, u_3) \cup Q_3 \cup (v_3, v_4) \cup Q_4 \cup (u_4, u_1)$. Figure 7 shows the cases of $k = 2$ and $k = 3$.



(a) Expand P_0 for $k = 2$

(b) Expand P_0 for $k = 3$

Figure 7. C_1: Intermediate cycle ($i = 1$)

For $i = 2$, bits $b_{3,1}b_{2,1}$ will be routed. We generate a Hamiltonian path P_1 by removing edge (u, v) from C_1, where $u = w_2\ 00$ and $v = w_3\ 00$. For examples, $u = 11\ 00$, $v = 10\ 00$ for $k = 2$ and $u = 011\ 00$, $v = 010\ 00$ for $k = 3$. Because the class IDs of the two terminate nodes in P_1 are 3 and 2, we can expand the path by modifying bits $b_{3,1}b_{2,1}$ (node IDs) of an MC$(k, 1)$.

Four paths $Q_1,\ Q_2,\ Q_3$, and Q_4 are generated based on P_1 by attaching 00, 01, 11, and 10 to each node address in P_1, respectively. The cycle C_2 then is generated by connecting the four paths together in a similar manner as in the case of $i = 1$. If $k = 2$, C_2 is a Hamiltonian cycle of MC$(2, 1)$, shown as in Figure 8.

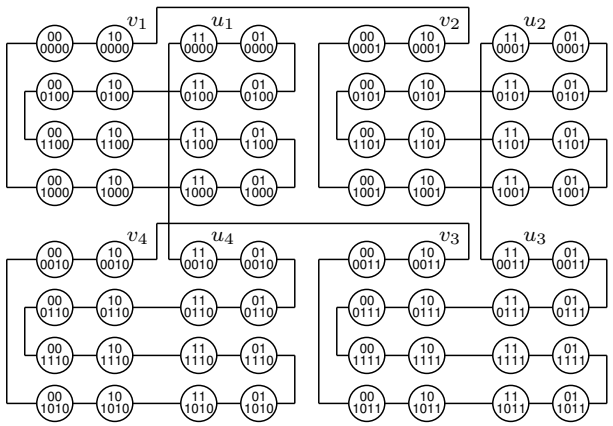


Figure 8. Hamiltonian cycle on an MC(2,1)

For $k = 3$, it needs to do $i = 3$ and $i = 4$ continuously to build a Hamiltonian cycle for MC$(3, 1)$.

For $i = 3$, bits $b_{6,1}b_{7,1}$ will be routed. We generate a Hamiltonian path P_2 by removing edge (u, v) from C_2, where $u = w_4\ 00$ and $v = w_5\ 00$. For example, $u = 110\ 0000$, $v = 111\ 0000$ for $k = 3$. Because the class IDs of the two terminate nodes in P_2 are 6 and 7, we can expand the path by modifying bits $b_{6,1}b_{7,1}$ (node IDs) of an MC$(k, 1)$. Four paths Q_1, Q_2, Q_3, and Q_4 are generated based on P_2 by attaching 00, 01, 11, and 10 to each node address in P_2, respectively. The cycle C_3 then is generated by connecting the four paths together in a similar manner as in case of $i = 2$.

For $i = 4$, bits $b_{5,1}b_{4,1}$ will be routed. We generate a Hamiltonian path P_3 by removing edge (u, v) from C_3, where $u = w_6\ 00$ and $v = w_7\ 00$. For example, $u = 101\ 000000$, $v = 100\ 000000$ for $k = 3$. Because the class IDs of the two terminate nodes in P_3 are 5 and 4, we can expand the path by modifying bits $b_{5,1}b_{4,1}$ (node IDs) of an MC$(k, 1)$. Four paths Q_1, Q_2, Q_3, and Q_4 are generated based on P_3 by attaching 00, 01, 11, and 10 to each node address in P_3, respectively. The cycle C_4 then is generated by connecting the four paths together in a similar manner as in case of $i = 3$. C_4 is a Hamiltonian cycle of MC$(3, 1)$.

3.3. Hamiltonian Cycle of MC(k, m)

Next, we show the case of building a Hamiltonian cycle for MC(k, m) in which $m > 1$.

Consider MC(2,2). We already built a Hamiltonian cycle for MC(2,1) as shown as in Figure 8. For $m = 2$, there are 2^k, or 4 new bits $b_{0,2}b_{1,2}b_{3,2}b_{2,2}$ that need to be routed. A loop generates a Hamiltonian cycle, and this time, one bit is dealt with in each iteration.

In the iteration of $i = 1$, we route bit $b_{0,2}$. A path P_0 is generated by removing an edge (u, v) from the Hamiltonian cycle of MC$(k, 1)$, where $u = w_0\ 00\ldots0$ and $v = w_0\ 10\ldots0$. For examples, $u = 00\ 0000$, $v = 00\ 1000$ for $k = 2$, and $u = 000\ 00000000$, $v = 000\ 10000000$ for $k = 3$. The two terminate nodes in P_0 have a same class ID 0.

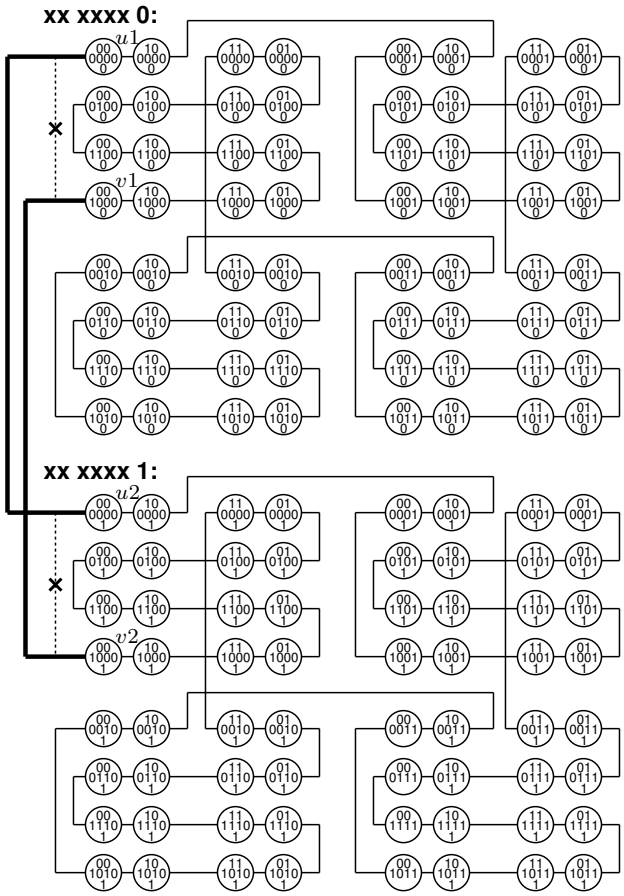


Figure 9. First iteration for building HC of MC(2,2)

Two paths, Q_1 and Q_2, are generated based on P_0 by attaching 0 and 1 to each node address in P_0, respectively:

$$Q_1 = P(u_1, v_1), \text{ where } u_1 = u\ 0 \text{ and } v_1 = v\ 0;$$
$$Q_2 = P(v_2, u_2), \text{ where } v_2 = v\ 1 \text{ and } u_2 = u\ 1;$$

A new cycle C_1 can be generated: $C_1 = Q_1 \cup (v_1, v_2) \cup Q_2 \cup (u_2, u_1)$, as shown as in Figure 9.

In the iteration of $i = 2$, we route bit $b_{1,2}$. A path P_1 is generated by removing an edge (u, v) from C_1, where $u = w_1\ 000\ldots0$ and $v = w_1\ 010\ldots0$. Note that nodes u and v contain one more bit than that in the first iteration. For examples, $u = 01\ 0000\ 0$, $v = 01\ 0100\ 0$ for $k = 2$, and $u = 001\ 00000000\ 0$, $v = 001\ 01000000\ 0$ for $k = 3$.

Similar to the first iteration, two paths, Q_1 and Q_2, are generated based on P_1 by attaching 0 and 1 to each node

Algorithm 1 (HC-Metacube)
Input: MC(k, m)
Output: Hamiltonian cycle $HC(k, m)$ in MC(k, m)
begin
/ Build a Hamiltonian cycle in k-cube: */*
Generate a Hamiltonian cycle C on a k-cube using Reflected Gray Code such that nodes 0 and 1 are neighbor;
Name the nodes in the cycle from node 0, in the direction to node 1, as $w_0, w_1, \ldots, w_{2^k-1}$;
/ Build a Hamiltonian cycle in MC($k, 1$): */*
$C_0 \leftarrow C$;
for $i \leftarrow 1$ to 2^{k-1} **do**
1. Generate a Hamiltonian path P_{i-1} by removing edge (u, v) from C_{i-1}, where
$u = w_{2(i-1)}\ 0 \ldots 0$,
$v = w_{2(i-1)+1}\ 0 \ldots 0$,
where $0 \ldots 0$ has $2(i-1)$ bits.
2. Generate four paths $Q_1,\ Q_2,\ Q_3$, and Q_4 based on P_{i-1}:
$Q_1 = P(u_1, v_1)$, where $u_1 = u\ 00$ and $v_1 = v\ 00$; */* attach 00 to each node address in P_{i-1} */*
$Q_2 = P(v_2, u_2)$, where $v_2 = v\ 01$ and $u_2 = u\ 01$; */* attach 01 to each node address in P_{i-1} */*
$Q_3 = P(u_3, v_3)$, where $u_3 = u\ 11$ and $v_3 = v\ 11$; */* attach 11 to each node address in P_{i-1} */*
$Q_4 = P(v_4, u_4)$, where $v_4 = v\ 10$ and $u_4 = u\ 10$; */* attach 10 to each node address in P_{i-1} */*
3. Generate a new cycle $C_i = Q_1 \cup (v_1, v_2) \cup Q_2 \cup (u_2, u_3) \cup Q_3 \cup (v_3, v_4) \cup Q_4 \cup (u_4, u_1)$.
endfor */* Hamiltonian cycle in MC($k, 1$) was built */*
/ Build a Hamiltonian cycle in MC(k, m): */*
$C_{1,0} \leftarrow C_{2^{k-1}}$;
for $j \leftarrow 2$ to m **do**
for $i \leftarrow 1$ to 2^k **do**
1. Generate a Hamiltonian path $P_{j-1,i-1}$ by removing edge (u, v) from $C_{j-1,i-1}$, where
$u = w_{i-1}\ 0 \ldots 0$, where $0 \ldots 0$ has $2^k(j-1) + (i-1)$ bits,
v is the neighbor of u with the same class ID w_{i-1} in $C_{j-1,i-1}$;
2. Generate two paths Q_1 and Q_2 based on $P_{j-1,i-1}$:
$Q_1 = P(u_1, v_1)$, where $u_1 = u\ 0$ and $v_1 = v\ 0$; */* attach 0 to each node address in $P_{j-1,i-1}$ */*
$Q_2 = P(v_2, u_2)$, where $v_2 = v\ 1$ and $u_2 = u\ 1$; */* attach 1 to each node address in $P_{j-1,i-1}$ */*
3. Generate a new cycle $C_{j-1,i} = Q_1 \cup (v_1, v_2) \cup Q_2 \cup (u_2, u_1)$;
endfor
endfor */* Hamiltonian cycle in MC(k, m) was built */*
end

address in P_1, respectively. A new cycle C_2 can be generated: $C_2 = Q_1 \cup (v_1, v_2) \cup Q_2 \cup (u_2, u_1)$, as shown as in Figure 10. Note that for the simplicity, some nodes were not drawn in the figure.

We can route bits $b_{3,2}b_{2,2}$ in the iterations of $i = 3$ and $i = 4$ in a similar manner. These are shown in Figure 11 and Figure 12. For the simplicity, some nodes were not drawn in the figures.

After finishing the 4th iteration, we get a Hamiltonian cycle of MC(2,2). We can do for $m > 2$ and $k > 2$ in a similar manner to build a Hamiltonian cycle of MC(k, m). A general algorithm for building a Hamiltonian cycle of MC(k, m) is given in Algorithm 1.

The algorithm consists of two parts. The first part constructs a Hamiltonian cycle for MC($k, 1$) and the second part constructs a Hamiltonian cycle for MC(k, m) when $m > 1$. In the first part, 2^k-bit cluster and node IDs are routed by a loop that iterates $2^k/2$ times and 2 bits are

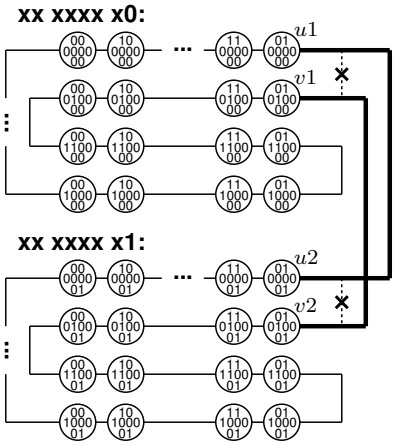


Figure 10. Second iteration for building HC of MC(2,2)

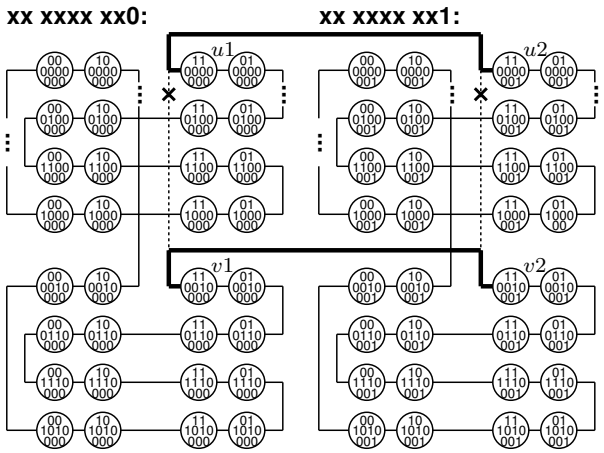


Figure 11. Third iteration for building HC of MC(2,2)

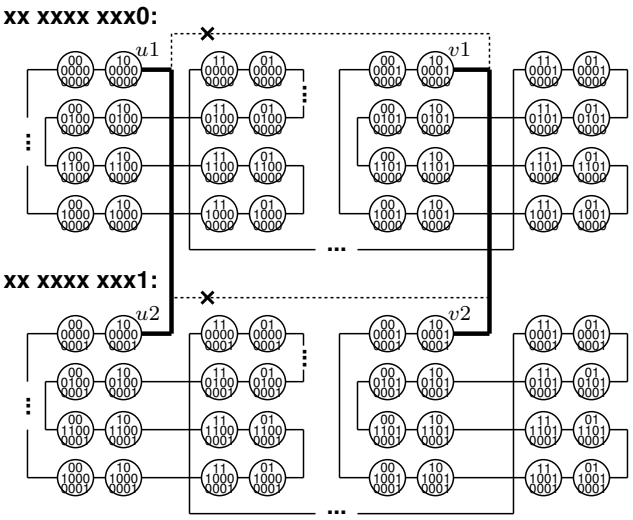


Figure 12. A Hamiltonian cycle of MC(2,2)

routed in each iteration. Initially, the cycle C_0 is of length 2^k. The ith iteration of the loop generates a cycle of length $4^i \times 2^k$ using the cycle created at the $(i-1)$th iteration. In the second part, the outer loop ($j = 2, \ldots, m$) counts the value of m and the inner loop ($i = 1, \ldots, 2^k$) creates a cycle $C_{j-1,i}$ by doubling the size of the cycle $C_{j-1,i-1}$ created from the previous iteration.

4. Concluding Remarks

In this paper, we showed that a Hamiltonian cycle exists in Metacube and gave an algorithm for constructing the Hamiltonian cycle on Metacube. Recently, several high-performance supercomputers were built with hundreds of thousands of microprocessors. Designing a high-performance interconnection networks with a small number of links is critical in such supercomputers. Metacube can be considered as a candidate IN for building such large-scale supercomputers. Hamiltonian is an important feature in interconnection networks. If a Hamiltonian cycle exists, many algorithms, originally based on ring or linear array, can be applied to the interconnection network.

References

[1] A. E. Amawy and S. Latifi. Properties and performance of folded hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):31–42, 1991.

[2] Kemal Efe. The crossed cube architecture for parallel computation. *IEEE Transactions on Parallel and Distributed Systems*, 3(5):513–524, Sep. 1992.

[3] K. Ghose and K. R. Desai. Hierarchical cubic networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):427–435, April 1995.

[4] J. P. Hayes and T. N. Mudge. Hypercube supercomputers. *Proc. IEEE*, 17(12):1829–1841, Dec. 1989.

[5] Y. Li, S. Peng, and W. Chu. Efficient communication in metacube: A new interconnection network. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN 2002)*, pages 165–170, Manila, Philippines, May 2002.

[6] Y. Li, S. Peng, and W. Chu. Hamiltonian cycle embedding for fault tolerance in dual-cube. In *Proceedings of the IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications (NPDPA 2002)*, pages 1–6, Tsukuba, Japan, October 2002.

[7] Y. Li, S. Peng, and W. Chu. Metacube – a new interconnection network for large scale parallel systems. In *Australian Computer Science Communications*, volume 24, pages 29–36, 2002.

[8] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Commun. ACM*, 24:300–309, May 1981.

[9] SGI. *Origin2000 Rackmount Owner's Guide, 007-3456-003*. http://techpubs.sgi.com/, 1997.

[10] L. W. Tucker and G. G. Robertson. Architecture and applications of the connection machine. *IEEE Computer*, 21:26–38, August 1988.

[11] S. G. Ziavras. Rh: a versatile family of reduced hypercube interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1210–1220, November 1994.