The Hamiltonicity of Crossed Cubes in the Presence of Faults

E. Abuelrub, *Member, IAENG*

Abstract—**The crossed cube is considered as one of the most promising variations of the hypercube topology, due to its ability of preserving many of the attractive properties of the hypercube and reducing the diameter by a factor of two. In this paper, we show the robustness capability of the crossed cube in constructing a Hamiltonian circuit despite the presence of faulty nodes or edges. Our result is optimal in the fact that it constructs the Hamiltonian circuit by avoiding only faulty nodes and edges in a crossed hypercube of dimension n. Our algorithm can tolerate up to 2^{n-3} faults with the restriction that each sub cube CQ_3 has at most one faulty node.**

Index Terms—**Embedding, Hamiltonian circuit, crossed cube, cycle, fault-tolerance, hypercube.**

I. Introduction

Parallel architectures based on the hypercube topology have gained widespread acceptance in parallel computing. The hypercube offers a rich interconnection topology with large bandwidth, logarithmic diameter, simple routing and broadcasting of data, maximally fault-tolerance, recursive structure that is naturally suited to divide and conquer applications, and the capability to simulate other interconnection networks with minimum overhead. The hypercube has been the focus of many research activities. Extensive work has been done to show that the hypercube is a powerful architecture capable of simulating other interconnection networks such as rings, meshes, trees, stars, and others with minimum overhead [1]-[3]. In addition, it has been shown that the hypercube architecture is robust and has the ability to simulate, route, and reconfigure itself despite the presence of either faulty links or nodes [4]-[12].

Hypercube based topologies are becoming more popular due to many of their attractive features in parallel computing [1], [13]-[15]. The crossed cube that was introduced by Efe [13] is considered as one of the most attracted variations of the hypercube, due to its attractive properties. Preliminary studies indicate that the crossed cube preserves many of the attractive properties of the hypercube and more importantly reduces the diameter by a factor of two. The crossed cube has been the focus of many recent researches. Some researchers studied the topological properties of the crossed cube compared with other hypercube-like topologies [16]-[17]. Various researchers have gone to great length to demonstrate the ability of the crossed cube to simulate other interconnection networks [18]-[20]. Other researchers have shown the robustness and fault-tolerance of the crossed cube, focusing on its ability to route, compute, and reconfigure itself despite the presence of faulty nodes or edges [21]-[22].

Many important computational problems in parallel processing can be formulated as graph embedding problems. Many variations of embeddings in interconnection networks have been studied in the literature. These variations differ principally in the optimization measures used in the embeddings. The problem of embedding one interconnection network into another is very important in the area of parallel computing for portability of algorithms across various architectures, layout of circuits in VLSI, mapping logical data structures into computer memories, and mapping task graphs into parallel machines. The problem of embedding rings or cycles into other interconnection networks has been studied by many researchers. It is well known that rings can be embedded into hypercubes using cyclic gray codes [3]. Latifi and Zheng [23] generalized the cyclic Gray code method to embed rings into twisted cubes. Many researchers have addressed the problem of embedding rings into hypercubes in the presence of faults. Provost and Melhem [12] have given distributed algorithms despite single, double, and multiple faults wasting up to 50% of the non-faulty processors in the worst case. Chan and Lee [6] improved the previous result by wasting only one processor for every faulty processor with some restriction on the number of faulty processors. Abuelrub [4] presented a recursive technique to embed a ring in a hypercube in the presence of faults that wastes a non-faulty processor for every faulty processor and can tolerate up to 2^{n-3} faults. On the other hand, other researchers addressed the problem of embedding rings into fault-free and faulty topologies [4], [9]-[10], [12], [18], [21], [23]-[28] or the Hamiltonicity of such structures in fault-free and faulty environments [7], [10], [20], [29].

In this paper, we show the robustness capability of the crossed cube in constructing a Hamiltonian circuit despite the presence of faulty nodes or edges. Our result is optimal in the fact that it constructs the Hamiltonian circuit by avoiding only faulty nodes and edges and can tolerate up to 2^{n-3} faults. The remainder of this paper is organized as follows. In section 2, we establish a few definitions and notations. Section 3 describes our scheme to construct a Hamiltonian circuit of size 2^n into a fault-free crossed cube of the same size. Section 4 addresses constructing the Hamiltonian circuit in the presence of faulty nodes. Section 5 concludes the paper and discusses future possible work.

Manuscript received March 1, 2008.

E. Abuelrub is with the Department of Computer Science, Zarqa Private University, Jordan (phone: +962-5-3821100/ ext. 1400; fax: +962-5-3821117; email: abuelrub@zpu.edu.jo).

II. Definitions and Notations

In this paper, we use undirected graphs to model interconnection networks. Each vertex represents a processor and each edge a communication link between two processors. The *embedding* or *mapping* of a guest graph G = (V_G, E_G) into a host graph H = (V_H, E_H) is an injective mapping f from V_G to V_H, where V_G, E_G and V_H, E_H are the vertex and edge sets of G and H, respectively, and where $|V_H| \geq |V_G|$. We consider a cycle with 2^n nodes, denoted C_2n, as the guest graph and a crossed cube with 2^n nodes, denoted by CQ_n, as the host graph. A *hypercube* of dimension n, denoted by CQ_n, is an undirected graph consisting of 2^n vertices labeled from 0 to 2^n-1 and such that there is an edge between any two vertices if and only if the binary representation of their labels differs by exactly one bit position.

Next, we define a *crossed cube* as follows. Let G be any undirected labeled graph, then G^b is obtained from G by prefixing every vertex label with b. Two binary strings x = x_1x_0 and y = y_1y_0, each of length two, are *pair-related* if and only if (x, y) ∈ {(00, 00), (10, 10), (01, 11), (11, 01)}. Now, we define a *crossed cube* of dimension n, denoted CQ_n, as an undirected graph consisting of 2^n vertices labeled from 0 to 2^n-1 and defined recursively as following:

1. CQ_1 is the complete graph on two vertices with labels 0 and 1.
2. For n > 1, CQ_n consists of two copies of CQ_{n-1} one prefixed by 0, CQ^0_{n-1}, and the other by 1, CQ^1_{n-1}. Two vertices u = $0u_{n-2}...u_0 \in CQ^0_{n-1}$ and v = $1v_{n-2}...v_0 \in CQ^1_{n-1}$ are adjacent, if and only if:

a. $u_{n-2} = v_{n-2}$, if n is even, and

b. For $0 \leq i < \lfloor(n-1)/2\rfloor$, $u_{2i+1}\ u_{2i}$ and $v_{2i+1}\ v_{2i}$ are pair-related.

Fig. 1 shows a hypercube and a crossed cube of dimension 3. CQ_n is constructed recursively based on the construction of CQ_{n-1} by pasting together a copy of CQ^0_{n-1} and the mirror image of CQ^1_{n-1}, then adding the appropriate links between the two copies according to the pair-related relationship. For clarity, we view the crossed cube CQ_n as a [2 x 2^{n-1}] grid. If the grid is partitioned horizontally into two equal parts, then all nodes above the horizontal line have a 0 as a prefix while all nodes below the horizontal line have a 1 as a prefix. An *upper node* is a node that lies in the upper part of the crossed cube, its right most significant bit is a 0. A *lower node* is a node that lies in the lower part of the crossed cube, its right most significant bit is a 1. An *upper link* is a link that connects two upper nodes and a *lower link* is a link that connects two lower nodes. A *cycle* of size n, denoted C_n, is an undirected graph consisting of n vertices labeled from v_1 to v_n, such that node v_i is a neighbor with node $v_{(i+1) \bmod n}$, $1 \leq i \leq n$. A *path* (v_o, v_1, v_2, …, v_{n-1}) is a sequence of nodes such that each two consecutive nodes are adjacent. A path in a graph G is a *Hamiltonian path* if all its nodes are distinct and they span G. A cycle or a circuit is called a *Hamiltonian circuit* if it traverses every node of G exactly once. The *vertex Hamiltonicity* of a graph G, denoted $H_v(G)$, measures the performance of the Hamiltonian property of a graph in the presence of faulty nodes. $H_v(G)$ is defined to be the maximum integer k such that G-F remains Hamiltonian for the set of faulty nodes F contained in V(G) with $|F| \leq k$ if G is Hamiltonian, and undefined if otherwise [29]. As the number of processors in parallel machines becomes larger, models without faults are becoming increasingly unrealistic. A *fault* is a processor or a link that fails. We use a strong model where a faulty node can neither compute nor communicate with its neighbors. A node fault will completely destroy the node and all links incident to it. We model a faulty link by making one of the nodes incident to the link faulty.

The quality of an embedding is often guided by some constraints that may differ from one application to another. The most common measures are dilation, expansion, edge congestion, and load factor. If u and v are two nodes in G then the distance from u to v, d = (u, v), is the length of the shortest path from u to v. The *dilation* D is the maximum distance in H between the images of adjacent vertices of G, D = max {d(f(u), f(v)), where u-v ∈ E_G}. The *expansion* E is the ratio of the cardinality of the host vertex set to the cardinality of the guest vertex set, E = $|V_H|/|V_G|$. There is a trade off between dilation, which measures the communication delay, and expansion, which measures processor utilization, such that one can achieve lower expansion at the cost of greater dilation and vice versa. Another cost measure is *congestion*, which is the maximum number of edges of the guest graph routed through a single edge of the host graph. Edge congestion is a measurement of possible degradation due to communication delay. If a particular link in the host network is needed for several different communication messages, then the messages will suffer some delay time since the link can't pass more than one message at a time. In embeddings that are many-to-one maps, an important measure is *load factor*, which is the maximum number of guest processors to be simulated by a single processor in the host interconnection network. An unbalanced processor load will degrade the simulation time, as lightly used processors must wait for heavily used processors to finish their tasks. Our embedding of the cycle into the faulty crossed cube in this paper is unit dilation, expansion, edge congestion, and load factor.

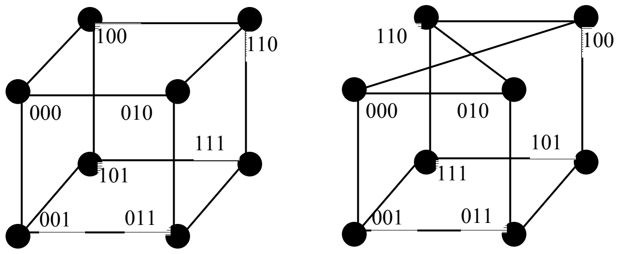
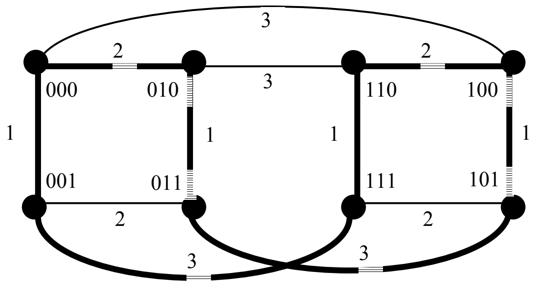


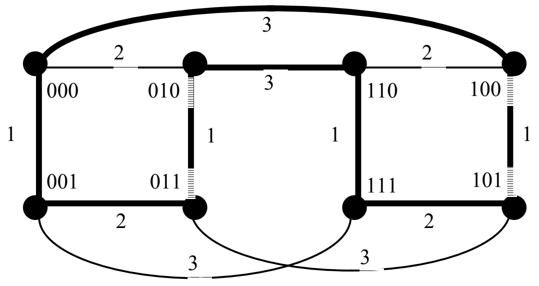
Fig. 1: Hypercube and crossed cube of dimension 3.

III. Hamiltonicity of a Faulty-Free Crossed Cube

Given a cycle C_2n with 2^n nodes, consider the problem of assigning the cycle nodes to the nodes of the crossed cube CQ_n such that adjacency is preserved. Now, given any two adjacent nodes in the cycle, their images by this embedding should be neighbors in the crossed cube through some dimension i, where $1 \leq i \leq n$. We can view such an embedding as a sequence of dimensions crossed by adjacent nodes. We call such a sequence the *embedding sequence*, denoted by ES = (d_1, d_2, ..., d_2n), where $d_i \in \{1, ..., n\}$ for all $1 \leq i \leq 2^n$. Fig. 2 shows two different embeddings of the cycle C_23 into the crossed cube CQ_3. It is more convenient to view the embedded cycle as will as the crossed cube in the way shown in Fig. 2. The embedding sequence of C_{2^3} is ES = (1, 3, 1, 2, 1, 3, 1, 2). For example, in Fig. 2-a, notice that nodes 000 and 001 are connected by a link through dimension 1, 001 and 111 are connected by a link through dimension 3, 111 and 110 are connected by a link through dimension 1, 110 and 100 are connected by a link through dimension 2, and so on. The embedding sequence ES can be generated using Algorithm ES.



(a) Node 000 initiating the embedding sequence.



(b) Node 001 initiating the embedding sequence.
Fig. 2: The embedding sequence.

Algorithm ES
Let n be the dimension of the crossed cube and let the vertical bar be the concatenation operator.

```
1. ES ← 1
2. For i ← 3 to n do
     ES ← ES | i | ES
3. ES ← ES | 2 | ES | 2
End
```

The embedding sequence is generated by applying Algorithm ES on n, where n is the dimension of the crossed cube. The number of nodes in the crossed cube is equal to the number of nodes in the embedded cycle, which is 2^n nodes. Thus, the embedding sequence of the cycle C_{2^4} is ES = (1, 3, 1, 4, 1, 3, 1, 2, 1, 3, 1, 4, 1, 3, 1, 2) and the embedding sequence of the cycle C_{2^5} is ES = (1, 3, 1, 4, 1, 3,1, 5, 1, 3, 1, 4, 1, 3, 1, 2, 1, 3, 1, 4, 1, 3, 1, 5, 1, 3, 1, 4, 1, 3, 1, 2). The embedding sequence corresponds to the binary-reflected Gray code embedding of a cycle into a crossed cube [1]. The binary-reflected Gray code is the most common technique to embed a cycle into a fault-free cube, but it is not suitable in the presence of faults. The embedding sequence will be generalized on the proceeding sections to embed a cycle into a crossed cube in the presence of faults. Notice that the same embedding sequence may result in different embeddings of C_{2^n} into CQ_n depending on the crossed cube node that initiates the cycle construction. Among all different embeddings, we are interested in two kinds. The first embedding is when the node that initiates the cycle construction in the crossed cube is the upper leftmost node, node with label 0. The second embedding is when the node that initiates the cycle construction in the crossed cube is the lower leftmost node, node with label 1. This will not violate the generalization of the technique since the crossed cube is node and vertex symmetric [13], which means that we can relabel the nodes, where any node can be labeled as node 0, and hence initiates the construction of the cycle. In Fig. 2-a, the cycle is initiated by node 000, while in Fig. 2-b the cycle is initiated by node 001.

Theorem 1: For every n, Algorithm ES will generate the embedding sequence to construct a cycle of size 2^n in a fault-free crossed cube of dimension n.
Proof: We prove this by induction on the dimension of the crossed cube. Our induction basis is CQ_2, a cycle of size 4 can be easily constructed in CQ_2 using the embedding sequence ES = (1, 2, 1, 2). Assume the theorem is true for the construction of a cycle of size 2^{n-1} in a crossed cube of dimension n-1. We now prove that the theorem is true for the construction of C_{2^n} in CQ_n. Let G be any undirected labeled graph, then G^b is obtained from G by prefixing every vertex label with b. Consider the two crossed sub cubes CQ^0_{n-1} and CQ^1_{n-1}. By induction hypothesis, we can construct a cycle of size 2^{n-1} in both CQ^0_{n-1} and CQ^1_{n-1}. Let their embedding sequence be ES = $S_{n-1}|2|S_{n-1}|2$, where, S_n is a sequence of dimensions recursively defined as follows: $S_2 = 1$ and $S_{n-1} = S_{n-2}|n|S_{n-2}$. Now, we combine two cycles, each of size 2^{n-1}, to come up with a cycle of size 2^n. This is done by replacing the first link that goes through dimension 2 of the first cycle and the second link that goes through dimension 2 of the second cycle by two links that go through dimension n. The embedding sequence of the new cycle C_{2^n} is ES = $S_{n-1}|n|S_{n-1}|2|S_{n-1}|n|S_{n-1}|2$ = $S_n|2|S_n|2$, which is the same embedding sequence generated by Algorithm ES. □

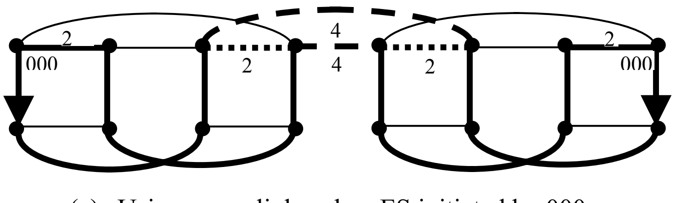
Next, we present Algorithm Fault-Free Hamiltonian Circuit (FFHC) algorithm uses a recursive divide and conquers technique to embed a cycle C_{2^n} into a fault-free crossed cube CQ_n.

Algorithm FFHC

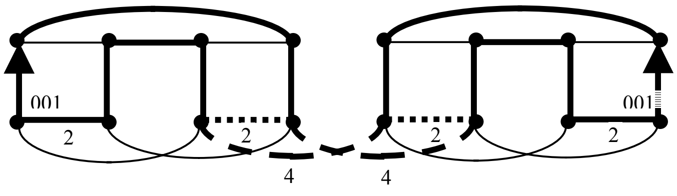
```
1. Partition $CQ_n$ into $2^{n-3}$ disjoint crossed cubes, each of dimension 3.
2. Embed the cycle $C_{2^3}$ into each sub cube using the embedding sequence ES = (1, 3, 1, 2, 2, 1, 3, 1, 2).
3. Connect the $2^{n-3}$ cycles, each of size 8, through the upper,
     or lower, links to come up with a cycle of size $C_{2^n}$..
End
```

Theorem 2: For every n, Algorithm FFHC will embed a cycle of size 2^n in a fault-free crossed cube of dimension n.
Proof: We prove this by induction on the dimension of the crossed cube. Our induction basis is CQ_3, the embedding of a cycle of size 2^3 into a crossed cube of dimension 3 is shown in Fig. 2. Assume the theorem is true for the construction of a cycle of size 2^{n-1} in a crossed cube of dimension n-1. We now prove that the theorem is true for the construction of C_{2^n} in CQ_n. Consider the two crossed sub cubes CQ^0_{n-1} and CQ^1_{n-1}. By induction hypothesis, we can construct a cycle of size 2^{n-1} in both CQ^0_{n-1} and CQ^1_{n-1}.

Now, we combine two cycles, each of size 2^{n-1}, to come up with a cycle of size 2^n. This is done by replacing the first link that goes through dimension 2 of the first cycle constructed in CQ^0_{n-1} and the first link that goes through dimension 2 in of the second cycle constructed in CQ^1_{n-1} by links that go through dimension n. Note the use of the upper links of dimension n when the embedding sequence is generated by node 0, while the lower crossed links of dimension n are used when the embedding sequence is generated by node 1, as shown in Fig. 3. □



(a) Using upper links when ES initiated by 000.



(b) Using lower links when ES initiated by 001.
Fig. 3: The recursive construction of the cycle C_{2^n} in a fault-free environment.

IV. Hamiltonicity of a faulty crossed cube

One of the special significant features of the crossed cube is its ability to simulate other interconnection networks in the presence of faults. In this section, we are interested in answering the following question. Given that some nodes of the crossed cube are faulty, does it have the ability to construct a Hamiltonian circuit efficiently by excluding only faulty nodes? The crossed cube is maximally fault-tolerant, while even one faulty processor will degrade its overall performance; it is still capable of simulating cycles optimally by avoiding only faulty processors during the construction of the Hamiltonian circuit. Next, we extend the Algorithm FFHC to handle a single faulty processor.

The basic idea behind our technique to embed a cycle into a faulty crossed cube is to use some of the unused links to skip a faulty node. As an illustration, in Fig. 1, CQ_3 has 12 links, 4 of them are not used in the construction of the cycle and might be considered spare links; the links between nodes 000 and 100 through dimension 3, 001 and 011 through dimension 2, 111 and 101 through dimension 2, and 110 and 010 through dimension 3 are unused links. We can use these unused links to avoid a faulty node, and since the crossed cube contains odd cycles, we can avoid only the faulty processor by constructing a cycle of size 7 from a crossed cube of dimension 3 that contains a 8 processors with one of them faulty. Therefore, if node v_i in a crossed cube CQ_n is faulty, a cycle C_{2^n-1} can be constructed by using some of the unused links to skip only the faulty node without disturbing the construction of the rest of the cycle. A faulty node is either an upper node or a lower node.

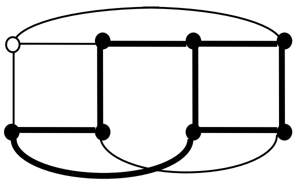
Now, we present the concept of blocks and cubes that are fundamental to the construction of the Hamiltonian circuit in the presence of faults. A *block* is a set of four nodes in a crossed cube that form a cycle of size 4 that has the embedding sequence ES = (1, 2, 1, 2). A *cube* is a sub crossed cube of dimension 3, CQ_3, that consists of two adjacent blocks. Notice that crossed cube of dimension n, Q_n, contains 2^{n-3} cubes and 2^{n-2} blocks. A cycle of size 8, C_{2^3}, can be embedded into a cube by the embedding sequence ES = (1, 3, 1, 2, 1, 3, 1, 2). Our technique is based on identifying the faulty node and the sub cube CQ_3 that contains it, and then avoids the faulty node by using the unused links. Fig. 4 shows all possible locations of a faulty node within the sub cube CQ_3 and the links that need to be used to avoid it in the process of constructing the Hamiltonian circuit. The location of the sub cube that contains the faulty node might be the first, the last, or some where in the middle. Our technique works for all three cases by using the appropriate links. Next, we present Algorithm Faulty-Node Hamiltonian Circuit (FNHC) that embeds a cycle C_{2^n-1} into a crossed cube CQ_n in the presence of a faulty node.

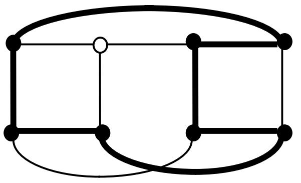
```
Algorithm FNHC
1. Partition $Q_n$ into $2^{n-3}$ disjoint cubes.
2. Locate the cube that contains the faulty node and identify
   whether it is an upper or a lower node.
3. If the faulty node is an upper node then
      a. Choose the appropriate embedding from Fig. 4-a.
      b. Embed the cycle $C_{2^3}$ into each of the fault-free
         cubes using the embedding sequence ES = (1, 3, 1,
         2, 1, 3, 1, 2) that is initiated by node 001 in the sub
         cube $CQ_3$.
      c. Connect all the cycles, one of size 7 and the rest of
         size 8, using the lower links to come up with the
         cycle $C_{2^n-1}$.
   Else {the faulty node is a lower node}
      a. Choose the appropriate embedding from Fig. 4-b.
      b. Embed the cycle $C_{2^3}$ into each of the fault-free
         cubes using the embedding sequence ES = (1, 3, 1,
         2, 1, 3, 1, 2) that is initiated by node 000 in the
         cube $CQ_3$.
      c. Connect all the cycles, one of size 7 and the rest of
         size 8, using the upper links to come up with the
         cycle $C_{2^n-1}$.
End
```

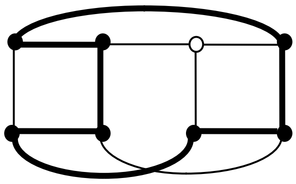
Theorem 4: For every n, Algorithm FNHC will embed a cycle of size 2^n-1 into a crossed cube of dimension n in the presence of a faulty node.

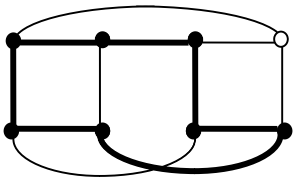
Proof: The theorem can be proven easily by induction by extending the proof of Theorem 3. □

Next, we will generalize our technique to embed a cycle into a faulty crossed cube with multiple faults, and hence constructing the Hamiltonian circuit by avoiding only faulty processors. Now, we describe our technique to embed a cycle C_{2^n-f}, where f is the number of faults, into a crossed cube CQ_n in the presence of f faults such that each cube CQ_3 within CQ_n has at most one faulty node. The maximum number of faults that can be handled by our technique is f = 2^{n-3}. The idea is to generalize Algorithm Faulty Node Hamiltonian to handle multiple faults. The following algorithm, Algorithm Multiple-Faults Hamiltonian Circuit (MFHC), embeds a cycle C_{2^n-f} into a crossed cube CQ_n in the presence of multiple faults and can tolerate up to 2^{n-3} faulty nodes.

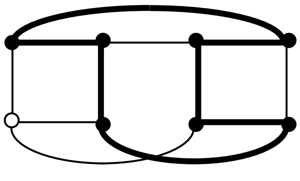


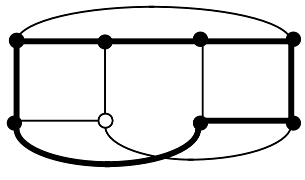


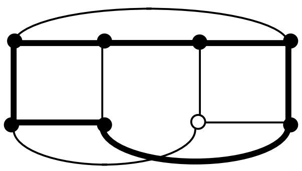


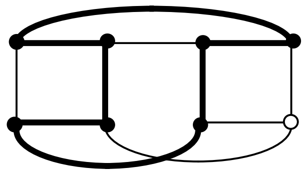


(a) Upper faulty node.







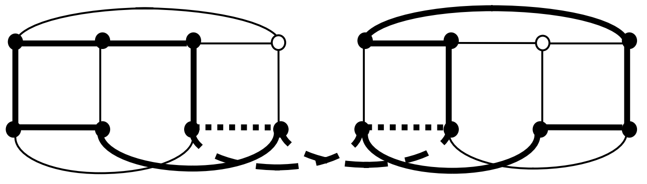


(b) Lower faulty node.
Fig. 4: All possible locations of faulty node within a cube CQ_3.

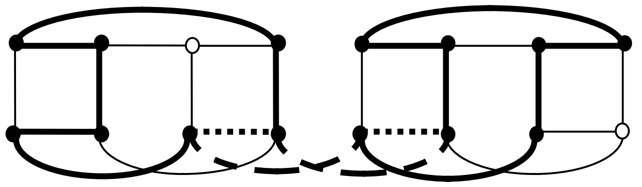
Algorithm MFHC
1. Partition CQ_n into 2^{n-2} disjoint blocks, sub crossed cubes of dimension 2, CQ_2.
2. Identify the blocks with faulty nodes.
3. Group each faulty block with the adjacent non-faulty block to its left to form a faulty cube of dimension 3, CQ_3.
3. Embed a cycle of size 7 into each of the faulty cubes CQ_3 by choosing an appropriate embedding from Fig. 4, and embed a cycle of size 8 into each of the non-faulty cubes.
4. Construct a cycle of size 2^n-f by connecting the cycles, either C_7 or C_8, using the appropriate links, either upper or lower links.

Theorem 5: For every n, Algorithm MFHC will embed a cycle of size 2^n-f into a crossed cube of dimension n, CQ_n , in the presence of f faulty nodes such that each sub cube CQ_3 has at most one faulty node.
Proof: Without loss of generality and for clarity, we assume that the leftmost block has no faulty node. The existence of an adjacent non-faulty block to the left of any faulty block follows directly from our assumption that each cube has at most one faulty node. In the process of constructing the cycle C_{2^n-f}, any two adjacent cubes with a faulty node are one of the following cases:
1. A cube with an upper faulty node followed by a cube with an upper faulty node.
2. A cube with an upper faulty node followed by a cube with a lower faulty node.
3. A cube with a lower faulty node followed by a cube with a lower faulty node.
4. A cube with a lower faulty node followed by a cube with an upper faulty node.

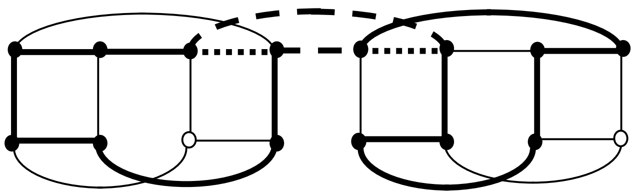
Fig. 5 shows all four cases in the process of constructing the Hamiltonian circuit. We use the crossed lower links with an upper faulty cube followed by either an upper or a lower faulty cube, and the upper links with a lower faulty cube followed by either a lower or an upper faulty cube. The way we grouped the faulty blocks with non-faulty blocks to form cubes always guarantee the existence of such links. The other cases are an upper or a lower faulty cube followed by a block, and a block followed by a block or a faulty cube. We use the crossed lower links with an upper faulty cube followed by a block and the upper links with a lower faulty cube followed by a block or a faulty cube. For the case of a block followed by a block or a faulty cube, we use the appropriate links, either upper or crossed lower links, since both are available. □



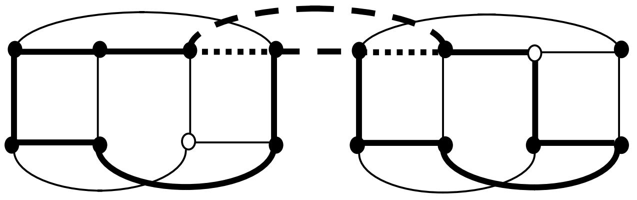
(a) Upper faulty node followed by upper faulty node.



(b) Upper faulty node followed by lower faulty node.



(c) Lower faulty node followed by lower faulty node.



(d) Lower faulty node followed by upper faulty node.
Fig. 5: All possible cases of faults in the process of constructing the Hamiltonian circuit.

V. Conclusions and Future Work

The recent advances in massive parallel architectures have also increased their complexities and thus the more need for reliability. As progress in VLSI has led to small size, low cost, and high performance processors, it has become practical to build parallel computers containing a very large number of processors. A main concern in the development of such a system is fault-tolerance. Since the probability of one or more processor faulting in such systems is quite large, it is desirable to build some fault-tolerance features into them. The hypercube is emerging as one of the most effective and popular network architectures for massive parallel machines. Hypercube based topologies are becoming more popular due to many of their attractive features in parallel computing. The crossed cube is considered as one of the most attracted variations of the hypercube, due to preserving many of the attractive properties of the hypercube and reducing the diameter by a factor of two.

This paper presented efficient techniques for constructing a cycle into a crossed cube with fault-free nodes, single faulty node, and multiple faulty nodes. We show the robustness capability of the crossed cube in constructing a Hamiltonian circuit despite the presence of faulty nodes or edges. Our result is optimal in the fact that it constructs the Hamiltonian circuit by avoiding only faulty nodes. Our method constructs the Hamiltonian circuit C_2n_{-f} within the crossed cube CQ_n despite the presence of $f \leq 2^{n-3}$ faults. Our algorithm can tolerate up to 2^{n-3} faults with the restriction that each sub cube CQ_3 has at most one faulty node. A good problem will be to generalize the algorithm to work for multiple faults with no restrictions and to adapt the same technique on other interconnection networks.

References

[1] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, Pearson Education, Upper Saddle River, 2003.

[2] T. Leighton, Introduction to Parallel Algorithms and Architecture: Arrays, Trees, Hypercubes, *Morgan Kaufmann*, 1992.

[3] Y. Saad and M. Schultz, "Topological properties of the hypercube," *IEEE Transactions on Computers*, vol. 37, no. 7, 1988, pp. 867-872.

[4] E. Abuelrub, "Fault-tolerance embedding of rings into hypercubes," *Al-Manarah*, vol. 4, no. 2, 1999, pp. 119-136.

[5] B. Becker and H. Simon, "How robust is the n-cube?," *Information and Computation*, no. 2, 1988, pp. 162-178.

[6] M. Y. Chan and S. Lee, "Distributed fault-tolerance embeddings of rings in hypercubes," *JPDC*, no. 11, 1991, pp. 63-71.

[7] J. Fu and G. Chen, "Hamiltonicity of the hierarchical cubic network," *Theory of Computer Systems*, vol. 35, 2002, pp. 59-79.

[8] J. Hastad, F. T. Leighton, and M. Newman, "Fast computation using faulty hypercubes," *Proceedings of the 19th Annual ACM STOC*, 1987.

[9] H. Keh and J. Lin, "On fault-tolerance embedding of Hamiltonian cycles, linear arrays, and rings in a flexible hypercube," *Parallel Computing*, vol. 26, no. 6, 2000, pp. 769-781.

[10] J. Lin, "Embedding Hamiltonian cycles, linear arrays, and rings in a faulty supercube," *International Journal of High Speed Computing*, vol. 11, no. 3, 2000, pp. 189-201.

[11] J. Lin and H. Keh, "Reconfiguration of complete binary trees in full IEH graphs and faulty hypercube," *International Journal of High Performance Computing Applications*, vol. 15, no. 1, 2001, pp. 55-63.

[12] F. Provost and R. Melhem, "Distributed fault-tolerant embedding of binary trees and rings in hypercubes," *Proceedings of the International Workshop on Defect and Fault Tolerance in VLSI Systems*, 1989.

[13] K. Efe, "The crossed cube architecture for parallel computation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, 1992, pp. 513-524.

[14] B. Javada, J. Abawjy, and M. Akbari, "A comprehensive analytical model of interconnection networks in large-scale cluster systems," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 1, 2008, pp. 75-97.

[15] P. Loh, W. Hsu, and Y. Pan, "The exchanged hypercube," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 9, 2005, pp. 866-874.

[16] C. Chang, T. Sung, and L. Hsu, "Edge congestion and topological properties of crossed cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 1, 2000, pp. 64-80.

[17] J. Fan, X. Lin, and X. Jia, "Non-pancylicity and edge-pancyclicity of crossed cubes," *Information Processing Letters*, vol. 93, no. 3, 2005, pp. 133-138.

[18] J. Fan, X. Lin, and X. Jia, "Optimal path embedding in crossed cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 12, 2005, pp. 1190-1200.

[19] P. Kulasinghe and S. Bettayeb, "Embedding binary trees into crossed cubes," *IEEE Transactions on Computers*, vol. 44, no. 7, 1995, pp. 923-929.

[20] M. Ma and J. Xu, "Panconnectivity of locally twisted cubes," *Applied Mathematical Letters*, vol. 17, no. 7, 2006, pp. 674-677.

[21] E. Abuelrub and S. Bettayeb, "Embedding rings into faulty twisted hypercubes," *Computers and Artificial Intelligence*, vol. 16, no. 4, 1997, pp. 425-441.

[22] E. Abuelrub, "Fault-tolerance mappings of complete quad trees into crossed cubes, *Journal of the Institute of Mathematics and Computer Science*, vol. 18, no. 1, pp. 17-25, 2007.

[23] S. Latifi and S. Q. Zheng, "Optimal simulation of linear array and ring architectures on multiply-twisted hypercube," *Proceedings of the 11th International IEEE Conference on Computers and Communications*, 1992.

[24] J. Jwo, S. Lakshmivarahan, and S. Dhall, "Embedding of cycles and grids in star graphs," *Research Report*, Department of Electrical Engineering and Computer Science, University of Oklahoma, 1990.

[25] R. Klasing, "Improved compressions of cube-connected cycles networks", *IEEE Transactions on*

Parallel and Distributed Systems, vol. 19, no. 8, 1998, pp. 803-812.

[26] J. Kwak and C. John, "Torus ring: improving performance of interconnection network by modifying hierarchical ring," *Parallel Computing*, vol. 33, no. 1, 2007, pp. 2-20.

[27] S. Lee and H. Ho, "A 1.5 approximation algorithm for embedding hyperedges in a cycle," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, 2005, pp. 481-487.

[28] Y. S. Myung, "An efficient algorithm for the ring loading problem with integer demand splitting," *SIAM Journal of Discrete Mathematics*, vol. 14, no. 3, 2001, pp. 291-298.

[29] S. Hsieh, G. Chen, C. and Ho, "Fault-free Hamiltonian ring embedding in faulty arrangement graphs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 2, 1999, pp. 223-237.

E. Abuelrub is an associate professor and dean of the Faculty of Science and Information Technology at Zarqa Private University, Jordan. He received his Bachelor degrees in computer engineering and computer science from Oklahoma State University, USA, in 1984 and 1985, respectively. He then joined the Alabama A&M University, USA, where he obtained his MSc degree in computer science in 1987. He completed his PhD degree in computer science from Louisiana State University, USA, in 1993. He is the Secretary General of the Colleges of Computing and Information Society and the Editor-in-Chief of the International Arab Journal of Information Technology. He is in the editorial board of many international journals and involved in the organization of many international conferences in computing and information technology. His areas of interest include parallel and distributed systems, interconnection networks, fault-tolerance computing, parallel algorithms, parallel architectures, e-commerce, and quality assurance in higher education. He has over 50 books and publications. He is a member of the IEEE, ACM, JEA, and IAENG.