International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:3, 2013

Independent Spanning Trees on Systems-on-chip Hypercubes Routing

Eduardo Sant'Ana da Silvá, Andre Luiz Pires Guedes, and Eduardo Todt

Abstract—Independent spanning trees (ISTs) provide a number of advantages in data broadcasting. One can cite the use in fault tolerance network protocols for distributed computing and bandwidth. However, the problem of constructing multiple ISTs is considered hard for arbitrary graphs. In this paper we present an efficient algorithm to construct ISTs on hypercubes that requires minimum resources to be performed.

Keywords—Hypercube, Independent Spanning Trees, Networks On Chip, Systems On Chip.

I. Introduction

THE hypercube, generally denoted as Q_k, is a k-connect graph with a special characteristic in which each vertex and its adjacent differ in only one bit in their binary representations. Among the advantages of this topology it may be emphasized the use on k-reliable broadcasts and distributed diagnosis [1].

Let G be a graph, a set of spanning trees rooted at a vertex r in G is said vertex/edge independent if for each vertex v in G, $v \neq r$, the paths of r to v in any pair of trees are vertex/edge disjoint. There is an interesting conjecture about ISTs that states:

Conjecture 1.1: For any k-connected graph G there exist k independent spanning trees($k-ISTs$) of G with any vertex v of G as root of the tree [2].

The conjecture was proved only for k-connected graphs with $k \leq 4$ and still open for arbitrary graphs when $k \geq 5$.

Itai and Rodeh [3] proved it for k=2. For k=3 it was independently proved by Zehavi and Itai[2], and Cheriyan and Maheshwari [4]. However, several algorithms are known in some classes of graphs such as product graphs [5], planar graphs [6], chordal rings [3], De Brujin and Kautz graphs [7], [8], hypercubes [9], [10], folded hypercubes [11], folded hyper-stars [12], multi dimensional thorus [13] and circulant recursive graphs [14].

Tang et al [9] developed an algorithm to construct ISTs on Q_k recursively from Q_{k-1}, i.e., it is necessary to construct $k-1$ trees in advance to obtain the $k-ISTs$. In [10] the authors presented an algorithm based on latin square distance, which is not recursive like the one presented in [9] and can, therefore, be parallelized.

However, the algorithms used to construct the trees generally require high computational power, even to those classes of graphs mentioned above. Such construction becomes prohibitive in many existing network architectures with limited resources. For such architectures simpler alternatives should be used instead, due to limited space and processing power.

Systems-on-chip (*SoCs*) designs usually employ several distinct types of components such as memories, processors, peripherals, and external IP (intellectual property) blocks that need to communicate with each other in some way. Several architectures have been used to accomplish this needs using bus based architectures such as: ARM Micro-controller Bus Architecture (AMBA) [15], IBM Core-Connect [16], Sonics SMART Interconnect [17], OpenCores Wishbone [18], and Altera Avalon [19].

Although widely used such alternatives are not standardized and their use requires deep knowledge on the manner that each technology works. Networks-On-Chip (*NoCs*) [20], [21], [22] have been proposed as a manner to provide abstraction in the way that *SoCs* blocks communicate among them. Therewith the time to market a new product is reduced significantly, turning the semiconductor companies more competitive.

Network-on-Chip is seen as an evolutionary approach to provide high performance and scalability in addition to a robust infrastructure for communication on-chip. The *NoCs* interconnect architectures used in *SoCs* must meet the requirements of these systems simultaneously providing scalability, re-usability and parallelism in communication, in addition to covering issues such as power consumption restrictions and use of distributed clock. [23], [22], [24].

The computational resources like memory, I/O and logical units in *NoCs* are interconnected by routers. In such networks the resources communicate with each other using data packets managed by routers deployed on the same integrated circuit.

The knowledge coming from the computer networks and distributed systems provides a wide range of results to be mapped to the field of Networks on Chip. This mapping is not easy mainly due to the restrictions on the implementation of network infrastructures in silicon.

Addressing the problem of limited resources in *NoCs* demands the use of new approaches to old problems such as routing. Due to space constraints components as routers and network interfaces should be simple to minimize memory requirements and processing power. Algorithms that require intensive computation to construct ISTs are not suitable for implementation in *NoCs*. Thus we present an algorithm that solves the problem using straightforward circuits with low memory consumption.

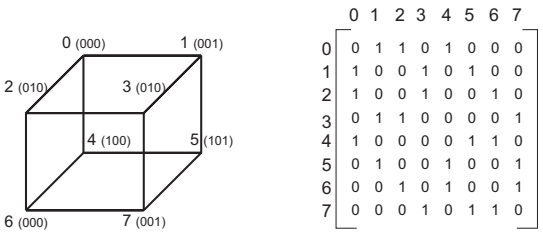
E. Silva, A. Guedes and E. Todt are with the Department of Informatics, Federal University of Paraná, Curitiba, PR, 19018 Brazil e-mails: eduardo.santanadasilva@gmail.com andre@inf.ufpr.br todt@inf.ufpr.br.

II. Preliminaries

Before presenting the developed method it is necessary to define some terms that are used throughout this article as well as some conventions proposed here.

Let $G = (V, E)$ be a graph with a vertex set $V = \{v_1, \ldots, v_n\}$ and an edge set E, the adjacency matrix $M_G = (m_{ij})$, of the same order of G, has non-zero value at the value at coordinate i, j (generally value 1 when the graph is not valued) iff v_i and v_j have an edge connecting them, otherwise the value at coordinate i, j is 0. All matrix coordinates m_{ij} satisfying the condition $i = j$ have 0 as value.

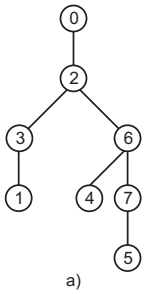
Figure 1 shows the adjacency matrix corresponding to the hypercube Q_3 with 8 vertices shown in the left side.



	0	1	2	3	4	5	6	7
0	0	1	1	0	1	0	0	0
1	1	0	0	1	0	1	0	0
2	1	0	0	1	0	0	1	0
3	0	1	1	0	0	0	0	1
4	1	0	0	0	0	1	1	0
5	0	1	0	0	1	0	0	1
6	0	0	1	0	1	0	0	1
7	0	0	0	1	0	1	1	0

Fig. 1. Q_3 adjacency matrix

In this paper an alternative matrix is used instead of the adjacency matrix. We also present a reading convention of the alternative matrix when dealing with trees. Thus the alternative matrix (from now on denoted as M_T) shown in Figure 2 (b) must be read as follows: The values of the row i in the matrix indicate that the vertices of the column index j are children of the vertexes with index i. In this way one can obtain the spanning tree of Figure 2 (a) by reading the adjacency matrix as described above.



	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1	0
3	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	1
7	0	0	0	0	0	1	0	0

b)

Fig. 2. Q_3 tree representation matrix M_T

One of the consequences of this convention is that all leave vertices have zero valued rows (vertices 1, 4, 5, Figure 2). It is valid to mention that in the reading convention of M_T, the edge $x-y$ is considered distinct of the edge $y-x$, which does not occur in the conventional adjacency matrix. Therewith the graph becomes directed and each edge oriented in a specific direction must belong to a distinct tree, meaning that each vertex has a different parent in each tree.

III. Constructing independent spanning trees on Q_k

The trees constructed by the algorithm presented here start from vertex 0 and vertices are incrementally added obeying a simple rule in the first phase in which the edges are added to the tree only if the child vertex has index greater than the parent vertex. In the second phase the edges are added in the reverse order to complete the tree. Algorithm 1 is used to generate the first tree. It is optimized to execute using logic operators such as OR and XOR, whose task is to compare vertices indexes and avoid edge repetitions.

```
input : k
output: first tree edge list, 0 as root
k: 2^k is the hypercube order
edgesOrder: (normal=0 or inverse=1)
vertices: number of vertices (2^k)
bitwise operators OR: ∨, and XOR: ⊕
for edgesOrder ← 0 to 1 do
    for n ← 0 to vertices do
        for b ← 1 ; i ← 1 to k do
            n2 ← n ∨ b;
            if n ≠ n2 then
                if edgesOrder = 0 then
                edge ←Edge(n,n ⊕ b); else
                edge ←Edge(n ⊕ b, n);
                if IsTheFirstEdge(edge) then
                    AddToTree(t, edge);
                end
                if ParentIsInTheTree(edge)
                then
                    AddToTree(t, edge);
                end
            end
            b ← b(rll)1;
        end
    end
end
```

Algorithm 1: Algorithm to create the first tree.

The algorithm is straightforward, it uses only one bit to mark the addition of the vertices in the tree. There is a special treatment to the first edge addition as vertex 0 has only one child in each tree. After that, any edge respecting the order mentioned before can be added to the tree if the parent was already added. The parent existence test is performed by checking a bit array that has size n. All other trees can be generated by the bit-wise left rotate operation with the index tree as the number of shifts to be performed. The left rotate operation is equivalent to the equation of conjecture 4.1.

After the trees with vertex zero as root are created, any other root can be calculated simply by performing an exclusive-or ($\oplus$) operation of each vertex of the trees using the desired root vertex. Figure 3 shows how to obtain a tree with root 5 from a tree with root 0.

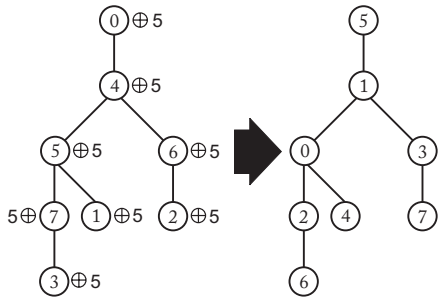


Fig. 3. Tree with root 5 obtained from the tree with root 0

IV. Experimental results

Table I presents the data about the memory consumption of the proposed algorithm compared to the algorithm presented by Yang et al. [10]. As our memory space is measured in bits, to compare with the algorithm presented by Yang et al., that is $O(kN)$, it is needed to multiply the previous equation by k, since it is necessary k bits to represent N. So the new equation to memory consumption of Yang's algorithm should be $O(k^2N)$. Compared with our method that requires N bits, our algorithm consumes 0.34% of the memory required by Yang et al. to build the trees on Q_{17} (131,070 vertices).

TABLE I
Memory consumption (in bits) during the trees construction process

k	Yang	Ours	Ours/Yang perc.
3	72	8	11.11
4	256	16	6.25
5	800	32	4.0
6	2,304	64	2.77
7	6,272	128	2.04
8	16,384	256	1.56
9	41,472	512	1.23
10	102,400	1,024	1.0
11	247,808	2,048	0.82
12	589,824	4,096	0.69
13	1,384,448	8,192	0.59
14	3,211,264	16,384	0.51
15	7,372,800	32,768	0.44
16	16,777,216	65,536	0.39
17	37,879,808	131,072	0.34

It were performed tests until Q_{17} so far, the independence verification process is being done computationally though we believe that the algorithm works to any N. So we proposed the following conjectures regarding hypercubes:

From an initial tree, T_0, rooted at vertex 0, k trees $T_1, T_2, ..., T_k$, all rooted at vertex 0, can be generated by permutations of the vertices. Let π_i be the permutation that generates T_i. Given a vertex x from T_0, $\pi_i(x)$ is the equivalent vertex in T_i. And π_i is given by:

$$\pi_i(x) = \begin{cases} 2^i \times x \mod 2^k - 1, & if \quad x \neq 2^k - 1 \\ x, & if \quad x = 2^k - 1 \end{cases} \quad (1)$$

Conjecture 4.1: If T_0 is constructed by the algorithm 1, and the trees $T_1, T_2, ..., T_k$ constructed with the operation π above then the set $T_0, T_1, ..., T_k$ is a set of ISTs of the Q_k.

Conjecture 4.2: All independent spanning trees, constructed by the algorithm presented in this paper are optimal in terms of average length of the paths [9].

Both conjectures above were confirmed until Q_{17} and we are currently working on the proofs.

V. Circuit simulation

The circuit simulation was done through Matlab Simulink to Q_4 (figure 4). Each custom block is depicted (figures 5 and 6) with the exception of the bit-array implementation since it is trivial. For Q_4 we need a bit-array with $16(2^4)$ bits of size. In the figure 7 we can see that the edges of the first tree are generated in about 120 cycles.

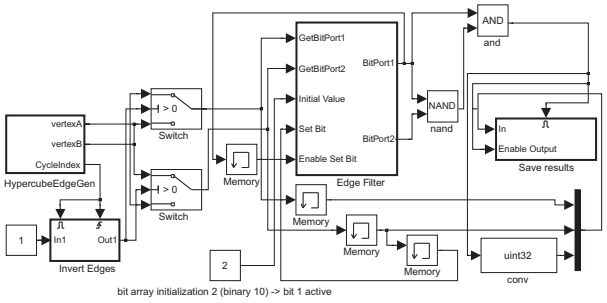


Fig. 4. Block to calculate first tree on Q_4

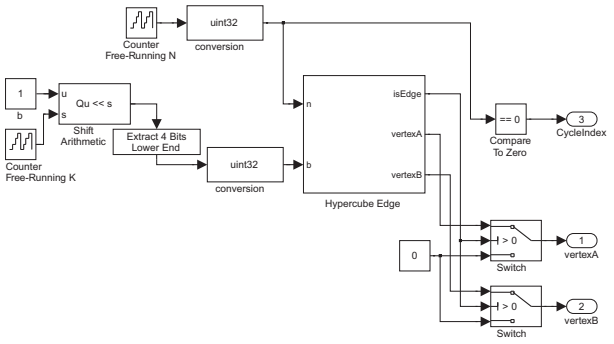


Fig. 5. Hypercube edges build block

VI. Conclusion

The algorithm presented in this paper is done in $O(kN)$ time and its main advantage is the memory space used to construct the trees, $O(N)$ bits. All independent spanning trees, constructed by the algorithm presented in this paper are optimal in terms of average length of the paths [9]. Our construction method is straightforward and very suitable to devices with limited computational power as well.

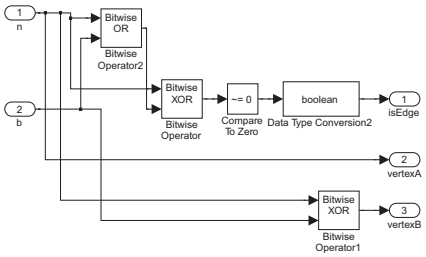


Fig. 6. Hypercube individual edge block

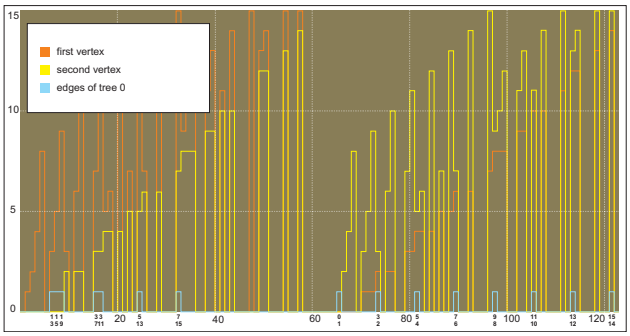


Fig. 7. Wave simulation diagram

REFERENCES

[1] J. Duarte, E.P., A. Brawerman, and L. Albini, "An algorithm for distributed hierarchical diagnosis of dynamic fault and repair events," in *Parallel and Distributed Systems, 2000. Proceedings. Seventh International Conference on*, 2000, pp. 299 –306.

[2] A. Zehavi and A. Itai, "Three tree-paths," *Journal of Graph Theory*, vol. 13, pp. 175–188, 1988.

[3] A. Itai and M. Rodeh, "The Multi-Tree Approach to Reliability in Distributed Networks," *Information and Computation*, vol. 79, pp. 43–59, 1984.

[4] J. Cheriyan and S. N. Maheshwari, "Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs," *J. Algorithms*, vol. 9, no. 4, pp. 507–537, 1988.

[5] K. Obokata, Y. Iwasaki, F. Bao, and Y. Igarashi, "Independent spanning trees of product graphs," in *Graph-Theoretic Concepts in Computer Science*, ser. Lecture Notes in Computer Science, F. d'Amore, P. Franciosa, and A. Marchetti-Spaccamela, Eds. Springer Berlin / Heidelberg, 1997, vol. 1197, pp. 338–351.

[6] A. Huck, "Independent Trees in Planar Graphs Independent trees," *Graphs and Combinatorics*, vol. 15, pp. 29–77, 1999. [Online]. Available: http://dx.doi.org/10.1007/s003730050030

[7] Z. Ge and S. L. Hakimi, "Disjoint rooted spanning trees with small depths in deBruijn and Kautz graphs," *SIAM Journal on Computing*, vol. 26, no. 1, pp. 79–92, 1997. [Online]. Available: www.scopus.com

[8] T. Hasunuma and H. Nagamochi, "Independent spanning trees with small depths in iterated line digraphs," *Discrete Applied Mathematics*, vol. 110, no. 2-3, pp. 189 – 211, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X00002699

[9] S.-M. Tang, Y.-L. Wang, and Y.-H. Leu, "Optimal Independent Spanning Trees on Hypercubes," *J. Inf. Sci. Eng.*, vol. 20, no. 1, pp. 143–156, 2004.

[10] J.-S. Yang, S.-M. Tang, J.-M. Chang, and Y.-L. Wang, "Parallel construction of optimal independent spanning trees on hypercubes," *Parallel Comput.*, vol. 33, no. 1, pp. 73–79, 2007.

[11] J.-S. Yang, J.-M. Chang, and H. Chan, "Independent Spanning Trees on Folded Hypercubes," *Parallel Architectures, Algorithms, and Networks, International Symposium on*, vol. 0, pp. 601–605, 2009.

[12] A. A. Rescigno, "Vertex-disjoint spanning trees of the star network with applications to fault-tolerance and security," *Information Sciences*, vol. 137, no. 1-4, pp. 259 – 276, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025501001219

[13] S.-M. Tang, J.-S. Yang, Y.-L. Wang, and J.-M. Chang, "Independent Spanning Trees on Multidimensional Torus Networks," *IEEE Transactions on Computers*, vol. 59, pp. 93–102, 2010.

[14] J.-S. Yang, J.-M. Chang, S.-M. Tang, and Y.-L. Wang, "On the independent spanning trees of recursive circulant graphs g(cdm,d) with d¿2," *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 2001 – 2010, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397508009122

[15] ARM, "AMBA Specification and Multilayer AHB specification (rev2.0)," http://www.arm.com/armtech/AXI/.

[16] IBM, "CoreConnect Specification," http://www.ibm.com/.

[17] Sonics, "SMART interconnect," http://www.sonicsinc.com/.

[18] "Wishbone Specification," http://www.opencores.org/wishbone/.

[19] "Altera Avalon Interface Specification," http://www.altera.com/.

[20] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, pp. 70–78, 2002.

[21] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *Micro, IEEE*, vol. 22, no. 5, pp. 36 – 45, sep/oct 2002.

[22] J. Owens, W. Dally, R. Ho, D. Jayasimha, S. Keckler, and L.-S. Peh, "Research Challenges for On-Chip Interconnection Networks," *Micro, IEEE*, vol. 27, no. 5, pp. 96 –108, sept 2007.

[23] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist, "Lowering power consumption in clock by using globally asynchronous locally synchronous design style," in *Design Automation Conference, 1999. Proceedings. 36th*, 1999, pp. 873 –878.

[24] K. Srinivasan, K. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 407 –420, april 2006.

Eduardo Sant'Ana da Silva received the BSc degree in computer science from Federal University of Paraná, Curitiba, Brazil in 2002, and the MSc degree at the same university in 2005. He currently is a PhD candidate at Federal University of Paraná and member of the Vision, Robotics and Image Research Group (VRI). His research interests are primarily in the area of graphs and fault tolerant algorithms applied to interconnection networks.

André Luiz Pires Guedes received his DSc in Computer Science from Federal University of Rio de Janeiro, Brazil, in 2001. He is a professor at Federal University of Paraná and works at the Algorithm Research Group (ARG). His research interests are graphs and algorithms.

Eduardo Todt received the BE degree in electrical engineering from Federal University of Rio Grande do Sul, Porto Alegre, Brazil in 1985, and the MSc degree at the same university in 1990. He received his PhD in Advanced Automation and Robotics - Polytechnic University of Cataluña in 2005 and he currently is a professor of the Department of Informatics Federal University of Paraná and leader of the Vision, Robotics and Image Research Group (VRI). His research interests are primarily in the area of image processing, robotics and embedded systems.