Design of a Highly Reconfigurable Interconnect for Array Processors

Lizy Kurian and Daniel Brewer
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620

Eugene John
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802

Abstract

Different interconnection topologies have different desirable properties and different algorithms run best on different networks. If the network topology is reconfigurable, it can be tailored to suit the particular algorithm being executed. Reconfigurability of processor arrays is also important from fault-tolerance point of view. An array processor that is programmable by the user any number of times to yield a different topology or to isolate faults is envisaged in this paper. The system has a host or controller that broadcasts a command to the interconnect to configure itself into a particular fashion. The interconnect uses static-RAM programming technology and can be programmed to different configurations by sending a different set of bits to the configuration RAM in the interconnect. We present three designs reconfigurable into array, ring, mesh, or Illiac mesh topologies. The first design is reconfigurable to different topologies, but provides no redundancy or fault tolerance. The second and third designs are capable of graceful degradation by eliminating faulty elements.

Keywords: Array processors, Polymorphism, Network embedding, Fault-tolerance, Programmability.

1 Introduction

Array processors and systolic arrays consisting of large numbers of identical processing elements have been a very popular architecture in VLSI and WSI technology. Reconfigurability of processor arrays is important due to two reasons (i) to efficiently execute different algorithms, and (ii) to isolate faulty processors. Different algorithms run more efficiently on different networks - for instance, hexagonally connected meshes are used for LU decomposition; binary trees are used for sorting; torus is used for transitive closure; doubles trees are good for searching; linear array is good for convolution and so on.

The major work in the past addressing functionally reconfigurable interconnection pattern is from Snyder [10]. The CHiP architecture proposed by Snyder consists of switches interspersed with processor elements or PEs to form a switch lattice. Snyder's paper contained several very interesting ideas about a polymorphic computer. Despite being a very fascinating idea, the advancement in the area of reconfigurable architectures has been puny and tardy due to the problems associated with their implementation. With the success and popularity of static-RAM based FPGAs and Field Programmable Interconnects from several manufacturers such as Xilinx, Plessey, Algotronix, Aptix [2] [12] [6] etc, it looks promising to design a static-RAM based field programmable interconnect for array processors.

Fault-tolerance in array processors and VLSI/WSI systems also calls for reconfiguration. Reconfiguration of systolic arrays has been extensively studied from the perspective of fault-tolerance [1] [9] [8] [11]. Most of the previous research considered systolic arrays with laser programmable interconnect, which is programmed to increase the yield of VLSI/WSI arrays.

Reconfigurability always comes with a price. Any programmable device contains several switches and switch matrices for reconfigurability purposes. These switches introduce additional parasitic capacitance and longer propagation paths. These effects tend to reduce speed. Our objective is to design an efficient and flexible programmable interconnect for array processors.

The objectives of the paper are two-fold. Our first objective is to design a network that can be programmed any number of times to yield the desired topology to suit the specific algorithm being used. The second objective is to design a network that can be reconfigured to isolate faults.

1.1 Overview

The rest of this paper is organized as follows. In section 2, we discuss fundamental principles of the static-RAM programming technology. In section 3, a network configurable into linear array, ring, mesh, Illiac mesh etc is presented. In section 4, reconfiguration for fault-tolerance is examined. We first present a design with capability to gracefully degrade by bypassing faulty columns and/or rows. Then we present a design which can bypass and reroute row and column connections.

2 The Static-RAM Programming Technology

The programmable interconnect designed in this paper uses the static RAM programming technology. A pass transistor together with a RAM cell can be used to realize a programmable connection between two wires. The pass transistor turns on or off depending on the content of the SRAM cell. When off, it presents a very high resistance and thereby an open connection between the two wires. When turned on, it forms a relatively low resistance and thereby a closed connection between the two wires. Thus the connection between the two wires can be established or removed by loading a different value into the SRAM cell and this is the basic principle being used in Field Programmable Gate Arrays being manufactured by Xilinx, Algotronix, Concurrent Logic and Plessey Semiconductors [3]. Instead of pass transistors, a transmission gate could be employed and the RAM cell can control both the n-channel and p-channel transistors of the full transmission gate. Alternatively, a multiplexer may be used to optionally connect one of several input wires to an output wire. Since the static RAM cell contents can be altered several times, the configuration of the interconnect can be varied any number of times. The RAM cell could be configured very quickly while in circuit.

3 A Polymorphic Network

This section presents the design of an array processor with a reconfigurable topology. The basic idea is to construct an array of processors where each processing element is surrounded by programmable switches, the switches being controlled by static RAM cells. The host of the array processor system would broadcast configuration bits to the static RAM cells resulting in the desired configuration for the interconnect. If the network has to support only a mesh/ring configuration with no fault-tolerance, the switching elements can be only pass transistors or transmission gates, controlled by a single RAM cell. If more connectivity has to be incorporated for polymorphic reconfigurability or isolation of faults, the switching elements should be multiplexers with the various inputs connected to the alternate connections that are desirable from reconfigurability point of view, and controlled by one or more SRAM cells depending on the number of input lines. The most flexible network following this ideal would be a completely connected network, which requires $N(N-1)$ programmable points. Each processor will have a programmable connection to every other processor. One could program such a network to realize any topology. But such a network is impractical for large values of N and hence we will look in detail on restricted versions of programmable interconnects. We first present the design of an interconnect chip that can connect processors to linear arrays, rings, mesh, and Illiac mesh, by broadcasting a different set of configuration bits to the system.

Theorem: In order to configure a set of N processors to a linear array, ring, mesh, or Illiac mesh interconnects, it is necessary and sufficient to provide $2N$ pass gates, N gates providing a connectivity between ith and $i+1$ th elements, and the other N between ith and $i+\sqrt{N}$th elements.

Proof: If the ith processor is connected to the processor at $(i+1) mod N$, ring or linear array can be realized. For an Illiac mesh, the equivalent functions are $(i+1) mod N$ and $(i+\sqrt{N}) mod N$. Hence, if every node could be connected to two successors, a mesh/Illiac mesh can be realized. By utilizing only one of the mesh connections, linear array/ring can be realized. Since there are N processors, and each processor needs 2 connections, $2N$ pass gates will be required.

□

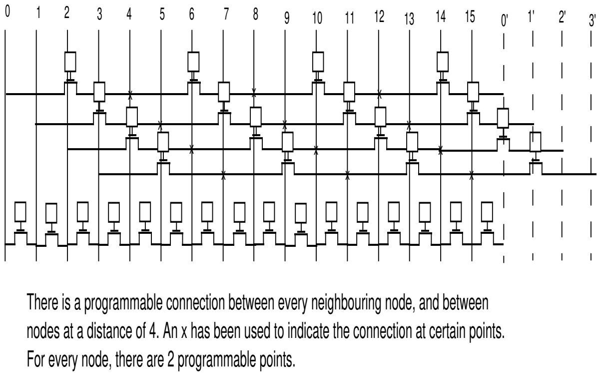


Figure 1: Design 2N: A 16-element network reconfigurable into array, ring, mesh or Illiac mesh. There are 32 programmable points with 1 SRAM cell each. This network does not incorporate redundancy or fault-tolerance.

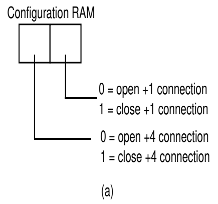
A 16 element network that can be configured to linear arrays, rings, mesh, Illiac mesh, etc can be constructed by incorporating 32 programmable switches as illustrated in Fig 1. The mesh is shown unfolded here, for sake of clarity in the illustration. To avoid round-about interconnections in the illustration, dotted lines are used to represent shadows of some processors. Again, for simplicity in drawing, a pass transistor is used as the switching device. A full transmission gate should be used if bidirectional connectivity is desired. The 32 programmable points involve 32 SRAM cells (two cells for each node). The network can be configured into several topologies by loading different bits into the RAM.

Fig 2 presents details about programming the configuration RAM to different topologies. The format of the configuration RAM bits for each node is illustrated in Fig 2.a. The user opens or closes each programmable point to realize a particular topology. All the connections are closed for the Illiac mesh as illustrated in Fig. 2.b. Fig. 2.c illustrates the RAM bits in hex for typical topologies such as array, ring, mesh and Illiac mesh. Programming the topology to the desired one involves loading the configuration RAM with the corresponding group of bits.

For each node, 2 static RAM cells are required to control the two outgoing connections. Each static RAM cell typically consumes 6 transistors. Depending on whether unidirectional or bidirectional connectivity is desired, one pass transistor or gate is required for each programmable link. If there are N processors and every channel is w bits wide, it is necessary to provide $2Nw$ pass gates and $2N$ static RAM cells in order to provide any ring/mesh topology.

4 Reconfiguration in presence of faults

Reconfiguration of array processors to eliminate faulty nodes is important from reliability point of view. When



Node	+4	+1	Hex eqvt
0	1	1	3
1	1	1	3
2	1	1	3
3	1	1	3
4	1	1	3
5	1	1	3
6	1	1	3
7	1	1	3
8	1	1	3
9	1	1	3
10	1	1	3
11	1	1	3
12	1	1	3
13	1	1	3
14	1	1	3
15	1	1	3

(b)

Configuration	Configuration RAM Contents (16 2-bit elements) (Hex)
Array	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
Ring	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Mesh	3 3 3 2 3 3 3 2 3 3 3 2 1 1 1 0
Illiac Mesh	3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

(c)

Figure 2: Details on the configuration RAM for Design 2N (a) The format for the configuration RAM for each node (b) RAM bits for each node for Illiac mesh connection (c) Configuration RAM bits for array, ring, mesh and Illiac mesh (in hex).

faulty processors are present, reconfiguration may be performed by maintaining the original size of the array (fixed-structure approach) or by reducing the number of processors (graceful degradation). To maintain the original size, spare nodes and extra interconnect have to be incorporated in the design so that the faulty elements can be replaced by the spares. The spares could be provided on a global basis, or a local basis. Fixed structure approach has the advantage that performance of the system is maintained throughout its life. However, spare elements are usually inactive during normal operation, and hence the utilization of processors is not good. The alternate approach is to network as many good processors as possible into a new network. This scheme does not dedicate any processors as spares. Whenever faulty elements have to be isolated, the entire row/column with the faulty processor can be bypassed, or rerouting can be used to isolate only the faulty elements and utilize the good elements in the row/column with the faulty element. The interconnect for rerouting will be more complex than the one required for mere bypassing.

Abraham et. al. [1] developed a 4-tuple notation (D, S, R, T) for reconfigurable arrays, where D denotes the dimension after reconfiguration (fixed structure or graceful degradation), S denotes whether the switches are local to the PE or independent, R denotes the replacement strategy (direct replacement or shifted replacement) and T denotes the reconfiguration technology (static or dynamic). For static-RAM based arrays, T stays fixed at the value for dynamic systems, and one could devise a design for the various combinations of D, S, and R. And each design could be done for various compensation distances, with increasing hardware cost. In this paper we shall devise designs for the following very typical configurations.

(i) Graceful degradation, row/column bypassing.

(ii) Graceful degradation, row bypass and column rerouting, or column bypass and row rerouting.

Definition: Compensation distance: The largest possible difference between the row (column) indices of two connected fault-free cells in a reconfigured array minus one. In the original physical array the difference is one, and the compensation distance is said to be zero. If one row or column can be bypassed, the compensation distance is one. If larger compensation distance is to be allowed more hardware resources in terms of switches are required.

4.1 Design 4N: Graceful Degradation, with entire Row and Column Bypass

In this subsection, we investigate a design called Design 4N that bypasses the entire row or column with the faulty element.

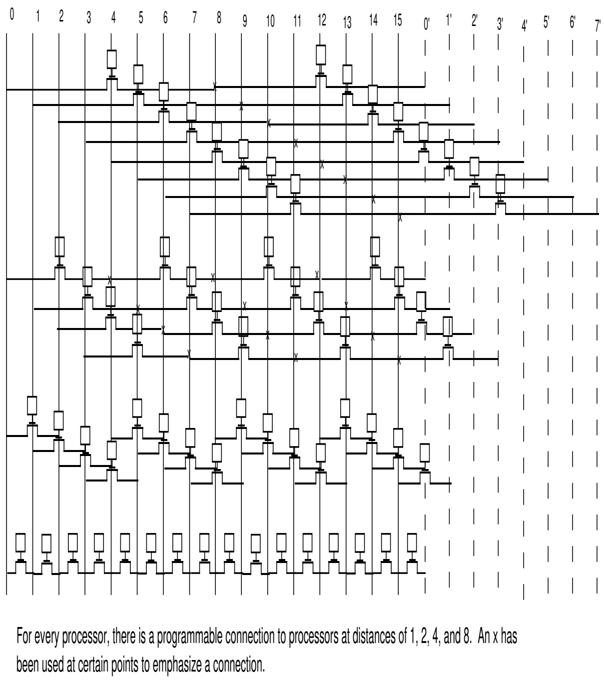


Figure 3: Design 4N: A 16-element performing graceful degradation with row/column bypass. There are 64 programmable points with 1 SRAM cell each. One faulty row and column can be bypassed to yield a fault-free network.

Theorem: In order to provide a network of N processors, bypass capability with a compensation distance of d, it is necessary and sufficient to provide $(2+2d)N$ bidirectional pass gates.

Proof: Basic mesh connectivity requires each processing element to be connected to two neighboring processors, and hence $2N$ connections are required for basic mesh connectivity. If the successor is faulty, the faulty element has to be bypassed. For bypassing one element, the column bypass link needs to connect ith processor to the

$(i+2)$th processor and the row bypass link needs to connect ith processor to $(i+2\sqrt{N})$th processor. If the compensation distance is d, up to d columns have to be bypassed, and hence there needs a connection between ith processor and $(i+p)$th processor where p ranges from 2 to $(d+1)$. Similarly, for bypassing up to d consecutive rows, connections need to exist between the ith processor and $(i+p\sqrt{N})$th processor, where p ranges from 2 to $d+1$. Thus dN programmable interconnections are required between ith and $i+2, i+3, i+4, ...i+d+1$th elements, for column bypass and another dN are required between ith and $i+2\sqrt{N}, i+3\sqrt{N}, i+4\sqrt{N}, ...i+(d+1)\sqrt{N}$th elements, for row bypass. Thus $(2+2d)N$ programmable connections are required. Often $d < \sqrt{N}$, which means that the compensation distance is less than the maximum number of rows or column in a square array. □

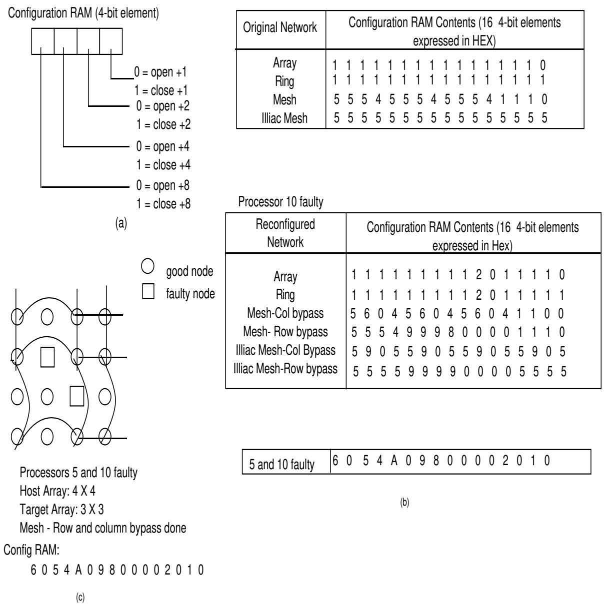


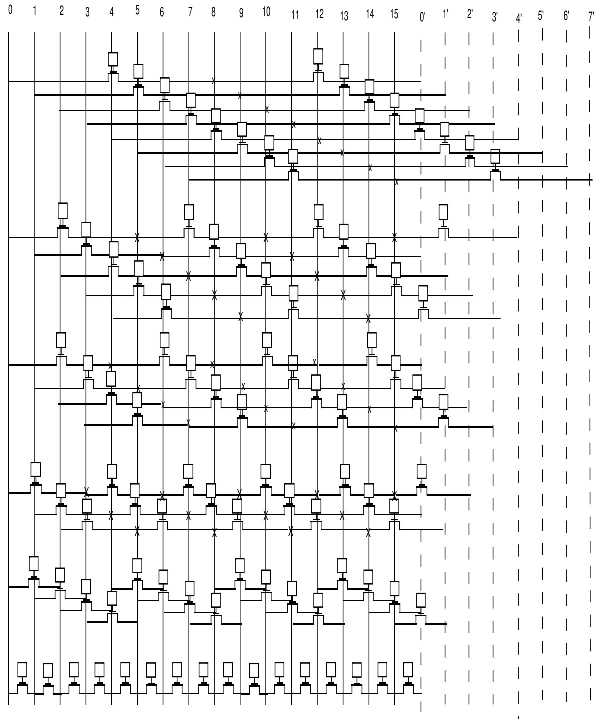
Figure 4: Details on configuration RAM for Design 4N (a) Format on configuration RAM (b) Configuration RAM contents in hex for several fault-free and degraded configurations (c) One example reconfiguration

In order to provide a network of N processors, the capability to bypass a single row and/or column, it is necessary and sufficient to provide $4N$ bidirectional pass gates, $2N$ providing basic mesh connectivity, N providing a connectivity between ith and $i+2$th element, and the other N between ith and $i+2\sqrt{N}$. A case with 16 nodes with a compensation distance of one is illustrated in Fig 3.

Fig 4 provides details on the configuration RAM for design 4N. Four bits are required for each processor. The format of the configuration RAM is illustrated in Fig. 4.a. The configuration RAM contents for original and reconfigured array, ring, mesh, and Illiac mesh are illustrated in Fig. 4.b. Several example reconfigurations with faulty processor(s) are illustrated in Fig. 4.c.

4.2 Design 6N: Graceful Degradation with Rerouting (Shifted Replacement)

Elimination of an entire column/row due to the presence of a single faulty element contributes to severe degradation. More efficient use of the processors can be achieved by incorporating redundant paths into the network. Bypassing just the faulty element in one dimension and rerouting in the other direction can increase the utilization of processors significantly. In this subsection, we present a design called design 6N which can eliminate only the faulty element rather than the entire row or column.



2N connections provide functional connectivity and 4N extra connections required to provide rerouting. For each processor there are 6 programmable connections. Dotted lines are used to represent shadows of processors, only to avoid round-about connections in the drawing.

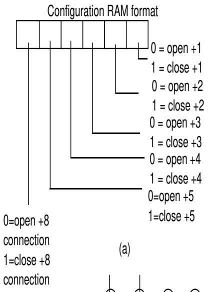
Figure 5: Design 6N: A 16-element network capable of graceful degradation with rerouting. There are 96 programmable connections. One faulty row and column element can be bypassed/rerouted to yield a fault-free network.

Theorem: In order to provide a network of N processors, the capability to bypass row and reroute column connections (or bypass column and reroute row connections) with a compensation distance of d, it is necessary and sufficient to provide $(2+4d)N$ bidirectional pass gates.

Proof: $2N$ connections are required to provide basic mesh connectivity. Bypassing up to d faulty elements in the row require dN switches to provide a connectivity between ith and $(i+2), (i+3), (i+4), ..., (i+d+1)$th elements. When only the faulty element rather than the entire column is bypassed rerouting of column connections is required. For rerouting, dN switches are required between ith and $i+$

$2\sqrt{N}, i+3\sqrt{N}, i+4\sqrt{N}, ...i+(d+1)\sqrt{N}$th elements, dN between ith and $i+2\sqrt{N}-1, i+2\sqrt{N}-2, ...i+2\sqrt{N}-d$, and another dN switches between ith and $i+2\sqrt{N}+1, i+2\sqrt{N}+2, ...i+2\sqrt{N}+d$. Altogether the circuit requires $4dN$ connections for fault-tolerance purposes and $2N$ connections for basic mesh connectivity. $\square$

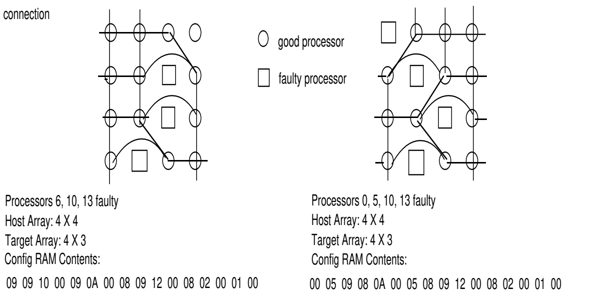
In order to provide a network of N processors, rerouting capability with a compensation distance of one, it is necessary and sufficient to provide $6N$ bidirectional pass gates, $2N$ providing basic mesh connectivity, and $4N$ providing fault-tolerance. A case with 16 nodes (called Design 6N) which allows bypassing for rows and rerouting for columns, with a compensation distance of one is illustrated in Fig 5. A circuit with row rerouting and column bypassing will be equivalent to this design in complexity and is not separately described. Fig 6 provides details on the configuration RAM for Design 6N. Six bits are required in the configuration RAM for each processor. The format of the configuration RAM is illustrated in Fig. 6.a. The configuration RAM contents for original and reconfigured array, ring, mesh, and Illiac mesh are illustrated in Fig. 6.b. Several example reconfigurations with faulty processor(s) are illustrated in Fig. 6.c.



(a)

Configuration	Config RAM Contents (16 6-bit elements expressed in Hex)
Array	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 00
Ring	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
Mesh	09 09 09 08 09 09 09 08 09 09 09 08 01 01 01 00
Illiac Mesh	09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
Mesh-6,10,13 faulty->4 X 3	09 09 10 00 09 0A 00 08 09 12 00 08 02 00 01 00
Mesh-0,5,10,13 faulty->4 X 3	00 05 09 08 0A 00 05 08 09 12 00 08 02 00 01 00

(b)



(c)

Figure 6: Details on the configuration RAM for Design 6N. (a) Format of the configuration RAM (b) Configuration RAM contents in hex for several fault-free and degraded configurations (c) A few example reconfigurations

5 Concluding Remarks

In this paper we explored several designs for a highly reconfigurable interconnect for an array processor. The designs exploit the reprogrammability of static-RAM programming technology as in some of the popular commercial FPGAs. Different topologies are best suited for different algorithms and we presented a design using $2N$ programmable connections that can connect N processors to ring, linear array, mesh or Illiac mesh configurations by loading a different set of bits into the configuration RAM on the system. Then we presented a fault-tolerant design using $(2+2d)N$ programmable interconnections capable of graceful degradation by bypassing upto d consecutive rows and columns with faulty elements. Finally, a design using $(2+4d)N$ programmable interconnects that performs row/column bypass and column rerouting, for compensation distance d is designed. A multi-chip module with several processors and a field programmable interconnect as designed in the paper could be used to construct a highly configurable array processor. Such a system in which every processor itself is built with programmable logic that could be configured to the architecture that specially suits the application, would result in a polymorphic computer.

References

[1] J. A. Abraham, P. Banerjee, C-Y Chen, W. K. Fuchs, S-Y Kuo, and A. L. N. Reddy, "Fault Tolerance Technique for Systolic Arrays", IEEE Computer, July 1987, pp. 65- 75.

[2] Aptix, The Programmable Interconnect Company Data Book, February 1993.

[3] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, Field Programmable Gate Arrays, Kluwer Academic Publishers, 1992.

[4] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks", IEEE Transactions on Computers, Vol. 36, No. 6, pp. 775-785, 1990.

[5] I. Dobbelaere, A. E. Gamal, D. How, and B. Kleveland, "Field Programmable MCM Systems – Design of an Interconnection frame", ACM/SIGDA Workshop on Field Programmable Gate Arrays, 1992, pp. 52-56.

[6] W. J. Hsieh, "Reconfigurable Hardware", Computer Design, March 1993, pp. 27-30.

[7] Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw Hill, 1993.

[8] J. H. Kim, and S. M. Reddy, "On the Design of Fault-Tolerant Two-Dimensional Systolic Arrays for Yield Enhancement", IEEE Transactions on Computers, Vol. 38, No. 4, April 1989, pp. 515 - 525.

[9] S. Kuo and I. Chen, "Efficient Reconfiguration Algorithms for Degradable VLSI/WSI Arrays", IEEE Transactions on Computer Aided Design, Vol 11, No. 10, pp. 1289 - 1300, 1992.

[10] L. Snyder, "Introduction to the Configurable Highly Parallel Computer", IEEE Computer, January 1982, pp. 47-56.

[11] T. A. Varvarigou, V. P. Roychowdhury, and T.Kailath, "Reconfiguring Processor Arrays Using Multiple-Track Models: The 3-Track-1-Spare Approach", IEEE Transactions on Computers, Vol. 42, No. 11, November 1993, pp. 1281 - 1293.

[12] Xilinx Programmable Gate Array Data Book, 1994.