Power Efficient Sub-Array in Reconfigurable VLSI Meshes

Ji-Gang Wu (武维刚) and Thambipillai Srikantan

Centre for High Performance Embedded Systems, School of Computer Engineering, Nanyang Technological University
Singapore 639798
E-mail: {jigw, astsikan}@ntu.edu.sg
Revised July 4, 2005.

Abstract Given an $m \times n$ mesh-connected VLSI array with some faulty elements, the reconfiguration problem is to find a maximum-sized fault-free sub-array under the row and column rerouting scheme. This problem has already been shown to be NP-complete. In this paper, new techniques are proposed, based on heuristic strategy, to minimize the number of switches required for the power efficient sub-array. Our algorithm shows that notable improvements in the reduction of the number of long interconnects could be realized in linear time and without sacrificing on the size of the sub-array. Simulations based on several random and clustered fault scenarios clearly reveal the superiority of the proposed techniques.

Keywords degradable VLSI mesh, reconfiguration, heuristic algorithm, fault-tolerance, NP-complete

1 Introduction

Area-time efficiency is one of the major considerations in VLSI designs. In recent years, the growth of personal computing devices (portable computers and real time audio and video-based multimedia products) and wireless communication systems (personal digital assistants and mobile phones) has forced designers to make high performance systems that consume less power. This necessitates the need to minimize the number of switches employed to implement the interconnection between two processing nodes during rerouting. Hence, degradable arrays that involve minimal number of switches will provide for less routing cost, less capacitance and less dynamic power consumption while improving the overall reliability.

Mesh is one of the most thoroughly investigated network topologies for multi-processor systems. It is of importance due to its simple structure and its good performance in practice and is becoming popular for reliable and high-speed communication switching. It has a regular and modular structure and allows fast implementation of many signal and image processing algorithms. With the advancement in VLSI technologies, integrated systems for mesh-connected processors can now be built on a single chip. As the density of VLSI arrays increases, probability of the occurrence of defects in the arrays during fabrication also increases. These defects obviously affect the reliability of the whole system. Thus, fault-tolerant technologies must be employed to enhance the yield and reliability of these mesh systems.

There are mainly two methods for reconfiguration in a mesh structure, namely, redundancy approach and degradation approach. In the redundancy approach, a system is populated with some spare elements so that they could be employed to replace faulty elements. Various reconfiguration schemes have been proposed in many papers, e.g., [1–8]. In the degradation approach, all elements in a system are treated in a uniform way, and there are no spare elements. This approach employs as many fault-free elements as possible to construct a target system. The problems of two-dimensional degradable arrays under different routing constraints are reported in [9, 10]. They have shown that most reconfiguration problems are NP-complete. A special case of the reconfiguration problem with row bypass and column rerouting capabilities is also discussed in [11]. Another heuristic algorithm, referred to as RCRS [12], performs row-exclusion and compensations with the excluded row for its neighbors. Subsequently, an improved version of RCRS was reported in [13]. However, no previous work has been carried out in order to provide power efficient sub-mesh for degradable VLSI arrays with faults.

This paper proposes the problem of finding a fault-free sub-array which demonstrates low power in a two-dimensional mesh-connected VLSI array. We first show that the problem of finding the power efficient sub-array is equivalent to a classical optimization problem, and then we propose a simple but efficient heuristic algorithm that has low power requirements, while maintaining the same harvest.

Section 2 provides definitions and notations. Section 3 shows the hardness of finding the power efficient sub-array. Section 4 describes the proposed algorithm. Section 5 shows detailed comparisons in the performance between the proposed algorithm and the most efficient one. Finally, we conclude our work in Section 6.

2 Preliminaries

Let host array $H$ be the original array obtained after manufacturing. Some of the elements in this array may be defective. A target array $T$ is a fault-free sub-array of $H$ after reconfiguration. The rows (columns) in the host array and target array are called the physical row (columns) and logical rows (columns), respectively.
In a host array, if \( e(i,j + 1) \) is a faulty element, then \( e(i,j) \) can communicate with \( e(i,j + 2) \) directly and data will bypass \( e(i,j+1) \). This scheme is called row bypass scheme. If \( e(i,j) \) can connect directly to \( e(i,j') \) with external switches, where \( |j' - j| \leq 1 \), this scheme is called column rerouting scheme. The problem turns out to be very difficult if rerouting in both row and column directions, i.e., row and column rerouting, is considered at the same time.

In this paper all the assumptions in architecture are the same as that in [10-13]. Neighboring elements are connected to each other by a four-port switch. A target array \( T \) is said to contain the set of the rows \( \{R_1, R_2, \ldots, R_k\} \) if each logical column in \( T \) contains exactly one fault-free element from each of the selected rows. The previous problem and the related algorithms are as follows.

**Problem \( \mathcal{R} \).** Given an \( m \times n \) mesh-connected host array, find a maximal sized target array under the row and column rerouting scheme that contains the selected rows.

The problem \( \mathcal{R} \) is optimally solved in linear time. Low proposed an algorithm, called Greedy Column Rerouting (GCR), to solve \( \mathcal{R} \) [11,12]. Let \( \text{col}(u) \) and \( \text{col}(v) \) denote the physical column index of the element \( u \) and \( v \), respectively. All operations in GCR are carried out on the adjacent sets of each fault-free element \( u \) in the row \( R_i \), defined as

\[
\text{Adj}(u) = \{ v : v \in R_{i+1}, v \text{ is fault-free and } |\text{col}(u) - \text{col}(v)| \leq 1 \}.
\]

The elements in \( \text{Adj}(u) \) are ordered in increasing column numbers for each \( u \in R_i \) and for each \( i \in \{1, 2, \ldots, k - 1\} \).

GCR constructs the target array in a left-to-right manner. It begins by selecting the leftmost fault-free element, say \( u \), of the row \( R_1 \) for inclusion into a logical column. Next, the leftmost element in \( \text{Adj}(u) \), say \( v \), is connected to \( u \). This process is repeated until a logical column is fully constructed. In each iteration, GCR produces the current leftmost logical column. A detailed description of GCR can be found in [12]. Theorem 1 describes the properties of the GCR algorithm.

**Theorem 1.** GCR solves the problem \( \mathcal{R} \) in linear time and produces the maximal sized target array\(^1\).\(^{12}\)

3 Power Efficient Target Array

As shown in Fig.1, there are 6 possible types of link-ways for a target array. They can be categorized into two classes based on the number of the switches used.

In this paper, a link-way that uses only one switch to connect neighboring elements in a target array is called a short link-way, while a link-way using two switches is called a long link-way. In Fig.1, (a) and (d) are short link-ways, but the others are long. (a), (b) and (c) are used for row rerouting, while (d), (e) and (f) are used for column rerouting. Obviously, the smaller the number of long link-ways the target array has, the lesser the routing cost, capacitance and dynamic power consumption.

**Definition 1.** The maximal sized target array with the minimal number of long link-ways is called the power efficient target array or power efficient solution.

The problem that we proposed is described as follows.

**Problem \( \mathcal{P} \).** Given an \( m \times n \) mesh-connected host array, find a power efficient solution under the row and column rerouting scheme that contains the selected rows.

The remaining part of the section shows the hardness of the problem \( \mathcal{P} \).

Given a host array \( H \) of size \( m \times n \), let \( U \) be the set of logical columns that pass through each of the rows. Thus, each logical column in \( U \) contains exactly one fault-free element from each of the rows. Furthermore, the \( i \)-th element in each logical column resides in row \( R_i \), for each \( i = 1, 2, \ldots, m \). We define a partial order on the logical columns in \( U \) as follows.

**Definition 2.** For any two logical columns, \( C_p \) and \( C_q \) in \( U \), and \( p \neq q \),

1. We say that \( C_p < C_q \) if the \( i \)-th element in \( C_p \) lies to the left of the \( i \)-th element in \( C_q \), for \( 1 \leq i \leq m \);
2. We say that \( C_p < C_q \) if the \( i \)-th element in \( C_p \) lies to the left of, or is identical to, the \( i \)-th element in \( C_q \), for \( 1 \leq i \leq m \);
3. We say that \( C_p \) and \( C_q \) are independent, if \( C_p < C_q \) or \( C_p > C_q \).

Let \( k \) be the maximal number of the independent logical columns that pass through each of the rows. According to [12], GCR can produce a target array of \( k \) independent logical columns. In addition, the \( j \)-th leftmost logical column is produced in the \( j \)-th iteration of GCR, \( 1 \leq j \leq k \). (see the proof of Theorem 1 in [12]). This is because that the fault-free elements are routed to form logical columns in a left-to-right manner, and the elements in \( \text{Adj}(u) \) are ordered in increasing column numbers for each \( u \in R_i \) and for each \( i \in \{1, 2, \ldots, m - 1\} \).

Symmetrically, we can obtain the \( j \)-th rightmost logical column in the \( j \)-th iteration by employing a new reconfiguration procedure, denoted as GCR'. The GCR' has the same approach as GCR except that it forms logical columns in the right-to-left manner, and the el-

---

\(^1\)Note: The reference [12] is not shown in the text. It seems to be a citation error or missing reference.