J. Parallel Distrib. Comput. 74 (2014) 3026–3036

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Flexible rerouting schemes for reconfiguration of multiprocessor arrays

Guiyuan Jiang [a], Jigang Wu [b,*], Jizhou Sun [a], Yiyi Gao [a]

[a] School of Computer Science and Technology, Tianjin University, Tianjin, 300072, China

[b] School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin, 300387, China

HIGHLIGHTS

- We develop a flexible rerouting scheme to improve the efficiency of utilizing fault-free PEs.
- We propose a heuristic to construct maximum logical arrays in linear time.
- We develop an efficient algorithm to reduce the interconnection redundancy of the logical array.
- We propose a tight lower bound on the total interconnection length.

ARTICLE INFO

Article history:
Received 20 June 2013
Received in revised form
7 June 2014
Accepted 23 June 2014
Available online 30 June 2014

Keywords:
Processor array
Reconfiguration
Fault tolerance
Rerouting scheme
Interconnection length
Algorithm

ABSTRACT

In a multiprocessor array, some processing elements (PEs) fail to function normally due to hardware defects or soft faults caused by overheating, overload or occupancy by other running applications. Fault-tolerant reconfiguration reorganizes fault-free PEs to a new regular topology by changing the interconnection among PEs. This paper investigates the problem of constructing as large as possible logical array with short interconnects from a physical array with faults. A flexible rerouting scheme is developed to improve the efficiency of utilizing fault-free PEs. Under the scheme, two efficient reconfiguration algorithms are proposed. The first algorithm is able to generate the maximum logical array (MLA) in linear time. The second algorithm reduces the interconnect length of the MLA, and it is capable of producing nearly optimal logical arrays in comparison to the lower bound of the interconnect length, that is also proposed in this paper. Experimental results validate the efficiency of the flexible rerouting schemes and the proposed algorithms. For 128 × 128 host arrays with 30% unavailable PEs, the proposed approaches improve existing algorithm up to 44% in terms of logical array size, while reducing the interconnection redundancy by 49.6%. In addition, the proposed algorithms are more scalable than existing approaches. On host arrays with 50% unavailable PEs, our algorithms can produce logical arrays with harvest over 56% while existing approaches fail to construct a feasible logical array.

1. Introduction

The quest for high-performance and low-power consumption leads to the development of multi-core architectures in which a large number of parallel processing elements (PEs) are integrated on a single chip in a tightly coupled fashion. However, faults often occur with the increased integration density due to overheating during massive parallel computing. The faulty PEs destroy the regular structure of the communication networks, and thus they reduce the processing capabilities of the multiprocessor array. This leads us to reconstruct the network topology using *reconfiguration techniques*, which open up new possibilities to improve the computing capabilities and the reliability of multiprocessor systems.

In application-aware topology reconfiguration, a multiprocessor array is customized to a topology that matches the traffic pattern of the application to reduce power consumption and message latency. There are two important application-specific reconfiguration methods, one is optimization of network topology [24,23,2], and the other is mapping of cores to network [13,22]. However, this becomes a big burden for programmers, because an optimized network topology working well for one application may not work well for another application. Differently, the *fault-tolerant reconfiguration* tries to reorganize fault-free PEs of a faulty array into a logical

* Corresponding author.
E-mail addresses: jguiyuan@gmail.com (G. Jiang), asjgwu@gmail.com (J. Wu), jzsun@tju.edu.cn (J. Sun), yygao17@gmail.com (Y. Gao).

http://dx.doi.org/10.1016/j.jpdc.2014.06.009
0743-7315/

array with regular topology (standard structured), which ensures well-controlled parameters for communication.

Two types of fault-tolerant architectures are mostly investigated for mesh connected processor arrays, i.e., *router-based architecture* and *switch-based architecture*. Router-based architecture consists of network nodes connected by links in mesh topology [28,6,18,36,5]. Each network node contains a conventional router, which provides routing and arbitration capabilities for packet messages. Header information of packet messages is required in forwarding data from the source to the destination. In case of faults, the router based network only needs to map the irregular network topology to a regular logical topology. Although the router-based architecture is easier to reconfigure in the algorithm aspect, it requires a complex router circuit, which increases the hardware cost, power consumption and circuit faults. In addition, the routing procedure is usually time-consuming. In switch-based fault-tolerant architecture, switches are laid between neighboring PEs, and the switches are also interconnected with one another [23,1,27]. In this type of architecture, once the connection is set up, signal messages can be transferred through the connection without any header information. Furthermore, the time delay is negligible since no routing or arbitration is needed. Hence, switch-based architecture is superior in terms of hardware cost, time delay, power consumption and probability of circuit faults. But the challenge in switch based architecture lies in the design of efficient reconfiguration algorithms. In this paper, we focus on developing efficient reconfiguration algorithms for the switch-based fault tolerant architecture.

Published solutions on the switch-based processor arrays can be classified into two categories, i.e., the redundancy approach and the degradation approach. The former intends to obtain a target logical array with a guaranteed size by replacing faulty PEs using spare ones [17,30,4,12,35], while the latter tries to provide a target array as large as possible. In the degradation approach, a fault-free logical subarray of $m' \times n'$ is formed from a faulty array of $m \times n$ ($m' \leq m, n' \leq n$), such that the original application can still work on the $m' \times n'$ subarray. Many approaches for reconfiguring processor arrays have been proposed under three different rerouting constraints [16], namely (1) *row and column bypass*, (2) *row bypass and column rerouting*, and (3) *row and column rerouting*. Most problems that arise under the above-mentioned constraints are NP-complete. Many methods that use different tracks [9,21,29], switches [34] and rerouting schemes [12,33,8,14,15] have been proposed to increase the utilization rate of fault-free PEs on the fault-tolerant processor arrays. However, these approaches intend to find as large as possible logical arrays without considering the network length. The work in [20,19] investigated a special case, i.e., reconfiguration on selected rows (which will be discussed in detail later), and obtained the maximum logical array (MLA). Then, the MLA is further optimized by a dynamic programming approach to reduce the communication cost, capacitance and dynamic power dissipation [31].

These mentioned approaches limit the rerouting distance to 1 such that two fault-free PEs with column/row distance exceeding 1 cannot connect as logical neighbors. The rational is that the network link capacity of d units is required if the rerouting distance is set to d, thus d must be kept small in order to reduce hardware cost. We will discuss this in detail in Section 2.2. However, the limitation on rerouting distance leads to a large number of unused fault-free PEs in constructing logical arrays. On the other hand, in reducing the interconnection length (inter-length for short) of the logical array, the order of revising each logical column significantly affects the inter-length of the resultant logical array. But in [31], columns are revised from right to left, resulting in a considerable amount of interconnection redundancy.

In this paper, our first goal is to improve the efficiency of using fault-free PEs to construct logical arrays. For this purpose, we develop a flexible rerouting scheme which does not limit the rerouting distance. We utilize the algorithm to guarantee that the resultant logical array can be implemented on the original physical array. We propose an efficient heuristic approach for constructing MLAs and prove that it is optimal in finding the maximum logical array size. Our second goal is to minimize the inter-length of the proposed MLA. For this purpose, we propose an efficient algorithm to reduce the inter-length of each column of the MLA. Unlike [31], the logical columns are revised in a recursive order. The main contributions of this paper are as follows. (1) A flexible rerouting scheme is developed to improve the utilization of fault-free PEs. (2) An efficient heuristic is proposed to produce the MLA in linear time under the flexible rerouting scheme. (3) A dynamic programming based algorithm is developed to reduce the inter-length, producing nearly optimal MLA in terms of the total inter-length. (4) A lower bound on the total inter-length of the MLA is proposed to evaluate the performance of the proposed algorithms.

The rest of this paper is organized as follows. In Section 2, we introduce the reconfiguration architecture, reconfiguration schemes and a brief description of previous work. In Section 3, we propose an efficient heuristic, which is capable of producing MLA under flexible rerouting schemes in linear time. In Section 4, we investigate the problem of constructing MLA with short inter-length by modeling it as a shortest path problem. We present the proposed lower bound on the inter-length of the MLA in Section 5. Experimental results and analysis are shown and discussed in Section 6. Finally, we conclude our work in Section 7.

2. Preliminaries

2.1. Fault-tolerant architecture

Let H denote the physical (host) array where some of the PEs are defective. Assume the fault density of the physical array is ρ, then there are $N = (1 - \rho) \cdot m \cdot n$ fault-free PEs in an $m \times n$ physical array. An $m' \times n'$ subarray comprising only fault-free PEs can be constructed by changing the connections among PEs. This subarray is called a target array or logical array, denoted as T. The rows and columns in physical/logical array are called physical/logical rows and columns, respectively. In this paper, $e_{i,j}(e'_{i,j})$ indicates the PE located at (i, j) of the host (logical) array, where i is its row index and j is its column index. $\mathrm{row}(u)(\mathrm{col}(u))$ denotes the physical row (column) index of PE u. $u = v$ indicates that u is identical to v.

Fig. 1(a) shows the structure of a fault-tolerant processor array of 4×4 with 3 faulty PEs. The fault-tolerant reconfiguration is achieved by inserting several 4-port switches in the network, allowing the network to dynamically change the connections among PEs. Each square box in the figure represents a PE, whereas each circle represents a 4-port switch. There are 4 states for each switch, and the link capacity of each link is 1 unit. In Fig. 1(b), a logical array of 4×3 is constructed by changing the states of switches. Throughout the paper, the gray shaded boxes in the figures represent faulty PEs while unshaded ones represent fault-free PEs.

2.2. Reconfiguration schemes

Two types of reconfiguration schemes, i.e., *bypass scheme* and *rerouting scheme*, have been proposed to guide reconfiguration algorithms for constructing logical arrays. As shown in Fig. 2(a), if PE $e_{i,j}$ is faulty, PE $e_{i,j-1}$ can directly communicate with PE $e_{i,j+1}$ and data will bypass $e_{i,j}$ through an internal bypass link without being processed. This scheme is called *row bypass* scheme. Under this scheme, an entire column of PEs can be bypassed if the column contains too many faulty PEs. Note that no external switch is