Microelectronics Reliability 83 (2018) 14–24

Contents lists available at ScienceDirect

Microelectronics Reliability

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

A dynamic partial reconfigurable system with combined task allocation method to improve the reliability of FPGA

Guohua Wang, Song Liu, Jiangtao Sun*

School of Instrumentation Science & Opto-electronics Engineering, Beihang University, Beijing, China

ARTICLE INFO

Keywords:
Resource management
Reliability
Task allocating
Task acceptance ratio
Dynamic partial reconfigurable system

ABSTRACT

Currently most FPGAs use SRAM-based technology, which are susceptible to faults from external electromagnetic radiation or produced by long-time internal overload operation. The dynamic partial reconfigurable (DPR) system, as an emerging technology, provides a promising way to solve this problem by reallocating the tasks in damaged resource areas to non-faulty regions at runtime. Based on such idea, an infrastructure for coordinately executing specialized hardware tasks on a reconfigurable FPGA is presented to achieve the flexibility for tolerating the occurring faults at runtime. Moreover, a method named MER-3D-Contact that combines the maximum empty rectangles (MER) technique with the adjacency heuristic is proposed to allocate tasks in the dynamical partial reconfiguration system for higher resource utilization, higher task acceptance ratio and lower fragmentation ratio. At last, experiments are carried out to evaluate the performance of the proposed system, results show that the proposed system can make the highest improvement 36% without damaged areas and the highest improvement 58% with damaged resources in terms of task acceptance ratio. Thus, the proposed system is expected a wide application in the field of more reliable FPGAs.

1. Introduction

As most commercial field programmable gate arrays (FPGAs) use SRAM-based technology, which are susceptible to faults from external electromagnetic radiation or produced by long-time internal overload operation, such as electro migration (EM), time-dependent dielectric breakdown (TDDB), single event upset (SEU) and hot carrier injection (HCI) [1–3]. These faults may cause abnormalities or failures in digital system, even bring serious consequences. The dynamic partial reconfiguration (DPR) technology based on FPGAs recently emerged and brought a promising solution to this problem by reallocating the functional modules in damaged resource areas to non-faulty regions during runtime without stopping running the FPGA device or interrupting the overall system operation [4–6].

Andreas et al. presented a reconfigurable system called ReconOS, which utilizes a unified multi-threaded programming model and extends the support of hardware threads to the operating system [7]. One advantage of the system is that it can get the optimal combination of hardware and software; the other is that it can improve the portability by providing a fixed interface for threads synchronization and communication. However, this system belongs to the traditional slot-based reconfigurable systems, whose reconfigurable areas are fixed to multiple slots. Kizheppatt et al. proposed an open source controller called ZYCAP [8], which can improve the ICAP execution speed effectively and achieve the near peak throughput. It can also reduce the software management overhead greatly because of the ability of running independent tasks by software and accelerators respectively. Although this system is reconfigurable, the task placement requires an additional tool (e. g. PlanAhead) for artificial settings. R3TOs is a good dynamic reconfigurable system, proposed by the University of Edinburgh [9]. The system is able to improve the flexibility by using microblaze as the kernel and dealing with the hardware task as the software task. Meanwhile, they proposed an innovative algorithm for 2D-bin-packing to realize the dynamic placement of hardware tasks. Nevertheless, there is still some room for performance improving in terms of resource utilization. Besides, many researches have been conducted in the field of task allocation which is known as the NP-hard problem to improve the task acceptance ratio in DPR systems [10]. In paper [11,12], Tabero et al. presented a new algorithm involved vertex list set (VLS) data structure, which is used to describe the area contours that are free in FPGAs. Unlike the maximum empty rectangles (MER) algorithm operating on the free area, VLS algorithm pays more attention to the perimeter of the free area. As a result, this task placement algorithm is very complicated. Handa et al. [13] and Cui et al. [14] proposed staircase structure and scan line algorithm (SLA) respectively based on [15]. However, the resource utilization of these algorithms is relatively

* Corresponding author.
E-mail addresses: wgh@buaa.edu.cn (G. Wang), ls2015@buaa.edu.cn (S. Liu), jiangtao_sun@buaa.edu.cn (J. Sun).

https://doi.org/10.1016/j.microrel.2018.02.005
Received 18 August 2017; Received in revised form 3 February 2018; Accepted 6 February 2018
Available online 20 March 2018

limited. Authors of paper [16] pioneered the MER strategy. Based on this method, they proposed two algorithms: keeping all maximal empty rectangles (KAMER) and keeping non-overlapping empty rectangles (KNER). These algorithms prove that a suitable free area could be found as long as it exists, but it may take a lot of time. Walder [17] and Steiger [18] proposed two different improvement programs to solve Bazagan's defections. Though they improved task placement quality, these structures had a high time consumption. More importantly, they are very vulnerable when fault occurs. The storage structure collapses as soon as a fault is emerged in the overlapping regions. Aiming at reducing chip area fragmentation, two allocating algorithms are developed in [19]. The first one, called empty area compaction (EAC), is an innovative allocation algorithm for 2D-bin-packing; the other called empty volume compaction (EAV) applies for 3D volume. The algorithms presented above possess good performance when faults occur in the reconfigurable region. Besides, there are some new algorithms such as hybrid genetic [20–22] and hilbert curve [23]. Nevertheless, there is still promotion space in the task acceptance ratio for these algorithms.

Although many reconfigurable systems have been proposed, the complex development process and runtime control keep this technique away from many applications where it would be beneficial and lead to a reduction of costs and power consumption. Moreover, the task allocating method in DPR systems is required for higher performance, such as higher task acceptance ratio, higher utilization rate of resources, lower chip fragmentation and more efficient allocation in dealing with damaged resources. To circumvent these existing obstacles, a DPR system based on Zynq is presented in this paper to reduce the complexity and increase the operationally of reconfiguration, which is able to realize the relocation of partial bitstreams and exchange the information among the partial reconfiguration tasks to improve the reliability. Moreover, an algorithm called MER-3D-Contact algorithm is proposed that combines MER technology with adjacency heuristic to manage the available free area as much as possible, which is expected to enhance the ability to deal with faults, and increase the placement efficiency of MER algorithms as well.

The remainder of this paper is organized as follows. At first, Section 2 presents the architecture of DPR system. After that, the allocation algorithm used to find the best location for tasks is described in Section 3. Then Section 4 gives the experiments of the proposed system. Finally, the conclusion is given in Section 5.

2. The DPR system architecture

The architecture of the proposed DPR system is shown in Fig. 1, which is composed of bottom hardware and Linux software for the purpose of real-time tasks allocation. According to the characteristics of the Xilinx Zynq FPGA, the system is divided into processing system (PS) part and programmable logic (PL) part.

The PS part is the embedded system processor, which contains the Linux kernel, clock, ACP, DMA and so on. These structures provide a comprehensive management of the system and communicate with the other parts through the AXI bus. The PL part is the resource part of the FPGA including static region and reconfigurable region. The execution parts of the system including memory subsystem, ICAP controller and placement module are placed in the static region, and the hardware tasks that need to be placed suitably are stored in the reconfigurable region according to its own information. The hardware tasks to be allocated are placed in the external memory, which are stored in the form of partial bitstreams files consisting of the ID, length, width and location information of tasks. The following paragraphs detail the bottom hardware of the DPR system including the memory subsystem, ICAP controller and placement module.

2.1. The architecture of the memory subsystem

In the conventional DPR system, there is data interaction among all the software tasks. The DPR system conducts the data transmission by means of shared memory. Before the reconfiguration, the system applies for a piece of memory space and then divides the space according to the hardware tasks. In the runtime, hardware tasks need to store the generated data into the designated memory space and send the access command to memory controller when accessing other hardware tasks. The memory controller will search the data in the memory space according to the hardware tasks' ID and transmit the data to the designated hardware task.

The memory controller connects the memory subsystem to the memory bus of the system as an AXI master, which is responsible for searching the table address entry when the memory manage unit (MMU) cannot find the relevant entry. The searched entry will be sent back to the MMU after searching in the memory controller. Then the MMU will generate a physical address based on the received entry and send back the physical address to the memory controller. After that, the memory controller will inquire the system memory based on the

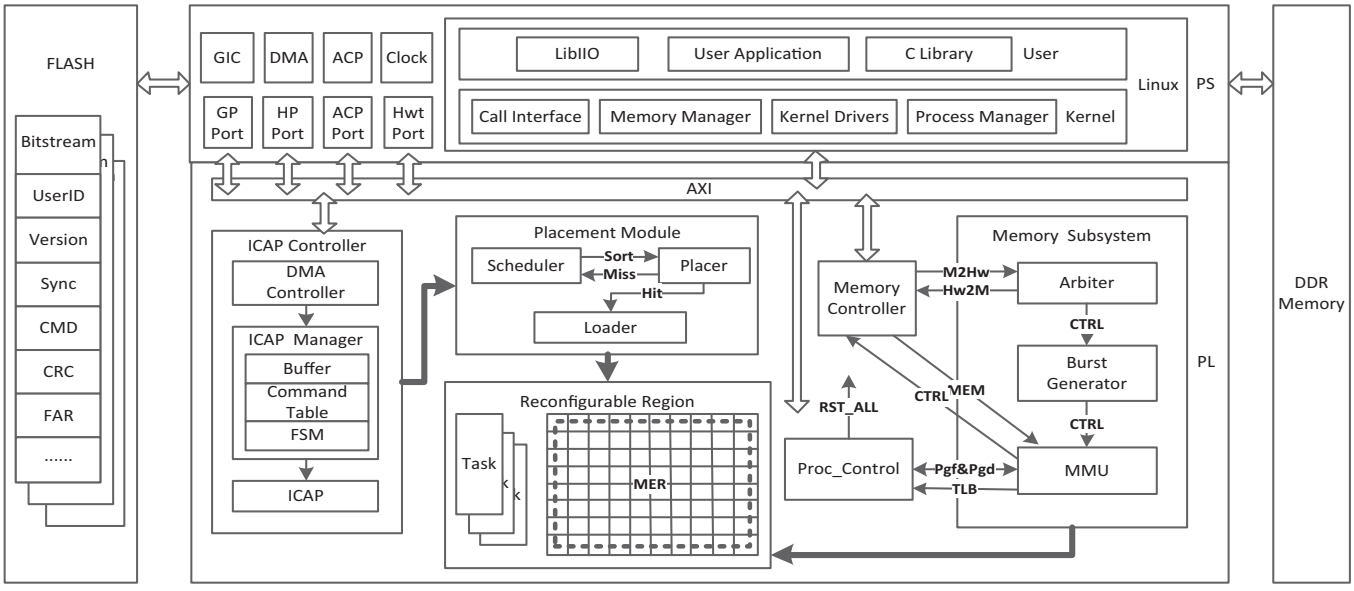

Fig. 1. The architecture of the DPR system.