PLACEMENT AND DEFRAGMENTATION IN RECONFIGURABLE COMPUTING SYSTEMS

Parag K. Lala
Department of Electrical
Engineering/ Texas A&M
University-Texarkana, **USA**
E-mail: plala@tamut.edu

ABSTRACT

This paper begins with a short discussion of the basic concepts of reconfigurable computing. Configuration is the process of programming a field-programmable gate array (FPGA) device for implementing a certain logic function. Configuration can be either static or dynamic. In static configuration the FPGA device retains its configuration state till the task is completed. In dynamic configuration the device is reconfigured repeatedly. In full reconfiguration the previous configuration state of the device is erased completely and the entire device is reconfigured. In partial reconfiguration only one part of the device is reconfigured while the other parts continue their normal operations. The advantage of partially reconfigurable devices is that they can retain more than one configuration provided the total size of these configurations does not exceed the available hardware resources.

Keywords: FPGA, Reconfiguration, Run-time placement, Prefetching.

INTRODUCTION

Field Programmable Gate Arrays (FPGAs) comprise of a matrix of logic blocks and interconnect network. Logic blocks and interconnect can be programmed to implement a particular function by downloading configuration bits onto an FPGA using a process called *mapping*. Because of their ability to provide high levels of performance and flexibility at the same time, FPGAs are now extensively used in a wide variety of computing applications . For example a reconfigurable computing system typically consists of one or more FPGAs connected through programmable links, a processor and a memory. block (Fig.1). The execution of a task is carried out by the FPGA and the processor performs all other operations including the configuration of the FPGA. The memory block stores the configuration data. Any function that is to be implemented by the reconfigurable system is divided into several tasks. The configuration data corresponding to each task is stored in the memory block

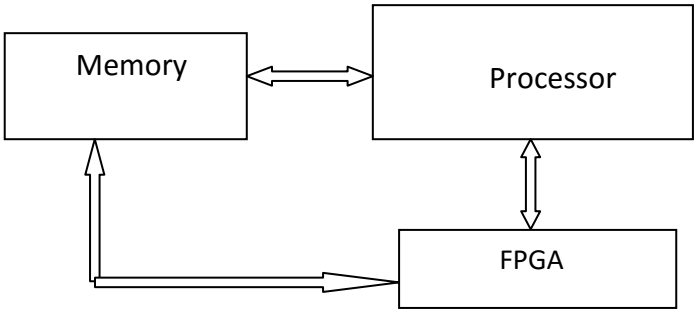

Fig. 1 A reconfigurable computing system

FPGAs can be *fine-grained* or *coarse-grained*. In fine-grained FPGAs logic functions and interconnections can be programmed at the bit-level. In coarse-grained FPGAs on the other hand, logic functions can be programmed at the word level. The method of reconfiguring the FPGA(s) in a reconfigurable system has to be considered in addition to the granularity of the FPGA. Based on the methods of reconfiguration, an FPGA can be one of the following three types: *single context*, *multi-context* and *partially reconfigurable*:

i. Single context FPGAs: The whole device is programmed via some of its input pins. Typically it takes several milliseconds to program such a device. The major disadvantage of this category is that even for any small change in configuration data a whole device has to reconfigured. This type of devices are suitable only in applications where runtime reconfiguration is not necessary.

ii.Multi-context FPGAs: The configuration mechanism for multi-context FPGAs is similar to that of single-context FPGAs. Several configurations are stored in the FPGA, however only one of them is active at any given time. The device can switch from one configuration to another in one clock cycle (usually a few nanoseconds). Thus a multi-context FPGA can be considered as a set of multiplexed single context FPGAs. Configuration data can be loaded onto any of the single contexts provided it is inactive during the system operation

iii. Partially reconfigurable FPGAs: In applications where only a small portion of the configuration data need to be modified, partially reconfigurable FPGAs are more suitable than single or multi-context FPGAs. In addition they are used in systems where runtime reconfiguration is essential. The location of configuration data to be changed is identified by the corresponding address.

Traditionally configuration information are loaded in FPGA devices in bit-serial fashion . Partially reconfigurable FPGAs can be programmed selectively i.e. only certain portions of the device can be modified . Two FPGA family such as Virtex series from Xilinx and the AT6000 series from Atmel Corporation support partial reconfiguration. In these devices the configuration information is loaded during the runtime in an addressable manner; thus address information as well as configuration data need to be supplied to such FPGAs. The advantage of partially reconfigurable FPGAs is that they can accommodate more than one configuration provided the total size of the these configurations does not exceed the available hardware resources. Additionally, in some applications only a portion of the configuration data has to be changed because no changes are necessary in the rest. For example in a filtering operation in signal processing the circuit function does not change except some constant values need to be updated over time [1].

In partially reconfigurable FPGAs tasks corresponding to an application can be replaced with those for another application provided the new one does not overlap with the configuration data of an already present application. In a dynamic environment the sequence of tasks corresponding to an application is not known in advance. Thus tasks are mapped onto a device when they arrive; this is known as *run time placement*. This is also known as *online placement* or *temporal placement*, and plays a major role in reconfigurable computing. The on-line placement is composed of the following two tasks [2]:

i. Locate all possible empty space blocks where the new task may be placed
ii. Select the best block among them to place the task

LOCATION OF EMPTY AREAS ON FPGAS

Several techniques for finding empty areas for online placement have been proposed [3,4,5]. In these techniques the available space on an FPGA device is considered as a set of rectangles; the tasks are also assumed to be of rectangular shape An incoming task is placed in one of these rectangles based on the placement strategy. Once a task has been completed it is removed from the FPGA and another one is mapped onto it. A drawback of this strategy is that incoming tasks are considered as independent and communications among them are ignored**.**

Different configurations that need to be loaded onto a partially programmable FPGA during the system operation are stored in the memory block of the system. The time needed to load data corresponding to a reconfiguration is known as the *reconfiguration overhead*. Since the system processor must remain idle during the reconfiguration process the reconfiguration overhead is a serious concern. The development of a placement algorithm that locates empty space in an FPGA without incurring high reconfiguration overhead is extremely important for the design of efficient dynamic reconfiguration system. A possible approach to reduce reconfiguration overhead and thereby improve the speed of reconfiguration, is to overlap a reconfiguration with the operation of the processor in the system; this approach is known as *configuration prefetching* [6]. In order to maximize the overlap with the computation of the system processor it is necessary to determine when to start a reconfiguration. False prefetching of a configuration has to be avoided as well otherwise the data corresponding to configuration that is next in sequence will be overwritten.

PLACEMENT TECHNIQUES

A placement technique aims to achieve one of the following objectives in order to reduce reconfiguration overhead [7]

i. Reduce the configuration data needed to program the functional units and the interconnections in the FPGA
ii. Reduce the number of reconfigurations neede

Reduction of Configuration Data

In many cases it may not be possible to use configuration prefetching to make the reconfiguration process transparent. This may happen when multiple configurations have to be loaded, one following another, onto an FPGA. In this situation the reconfiguration can be reduced by compressing the configuration bits. As a result the system processor takes less time to transfer the compressed reconfiguration bits to the FPGA. This approach is known as *configuration compression,* and is based on the fact that the configuration bits are highly redundant and hence no information loss results from compaction [8]. Several algorithms have been proposed for compressing configuration [9,10]. The disadvantage of compressed configuration is that additional circuits for decompression must be incorporated on the FPGA. An alternative strategy is to avoid the reconfiguration of components that are common to successive configuration. This strategy is particularly suitable for partially reconfigurable devices, only part of a device that have different configuration values are

reconfigured. Therefore if successive reconfigurations are selected such that their differences are minimum, the reconfiguration time could be reduced [11].

Reduction of the Number of Reconfigurations

The number of required configurations can be reduced either by configuration relocation and defragmentation or by configuration caching:

Configuration relocation and defragmentation approach is suitable only for partially reconfigurable FPGAs. It allows a new configuration to be loaded onto the FPGA while the ones that have already been loaded remain intact in their respective locations in the device. Since only a portion of the device has to be reconfigured less amount of data has to be transferred to the device, thereby resulting in the reduction of reconfiguration time.

A major contributing factor to the low area utilization in dynamically reconfigurable systems is the fragmentation of the programmable resources in an FPGA [12]. Fragmentation does not allow the transfer of configuration data for implementing a task or tasks even though the free space is available in scattered fragments. The utilization of best allocation and replacement techniques cannot prevent fragmentation to occur in reconfigurable systems over a period of time. All the available fragments of empty space need to be consolidated into one block to accommodate configuration data corresponding to a new task on such a system or to relocate an existing one; this process is known as *defragmentation*. The empty space left on an FPGA after some tasks have been placed on the FPGA, can be represented using a set of rectangles known as *empty rectangles* [13] .

A technique for representing empty space in an FPGA using non-overlapping rectangles has been proposed in Ref [5]. The drawback of this strategy is that if two reconfigurations are to be placed at locations that partially overlap, then each time of one these configurations is used the other one has to be removed; this will have a negative impact on the reconfiguration time. A more efficient approach is to place a new configuration during runtime onto a part of the FPGA that is unlikely to be used for some time

As configurations are loaded and unloaded the location of unoccupied space in a partially reconfigurable device become fragmented over time. Consequently there could be enough empty slots in a device but since the slots are scattered, an incoming task cannot be accommodated in them. Therefore the mapping of a task into the device may overwrite some of the valid configurations. It should be noted that the empty area in a device does not give a measure of fragmentation; for example empty areas in two different states of a device may be identical but the fragmentation in each state may be different [11] A method for defragmentation that employs dynamic relocations of module positions during runtime has been proposed in Ref.14; the main goal of the technique is to make the contiguous free space as large as possible to fit in new modules.

An alternative approach for reducing the number of reconfigurations is *configuration caching* [15]. Cache memory is a small amount of fast memory that is used in a general purpose computing system. The system processor can access the cache memory much faster than the main memory. It uses the cache memory to get and store information while the cache reads from and write into the memory in spare time. This allows the processor to operate at full speed most of the time. In a reconfigurable system an FPGA retains configuration data on the device so that data transfers to it can be reduced. A major

problem in configuration caching is to determine which configurations should be retained on the chip and which one among these should be replaced during the reconfiguration process.

So far not much has been published on the affect of defragmentation on reconfigurable system performance [12]. Ref. [11] identifies a set of research issues that needs to be investigated in order to understand the relationship between defragmentation, scheduling and placement; some of the major issues are:

i. derive an accurate quantification and measurement of fragmentation on an FPGA device
ii. determine best time to perform fragmentation
iii. identify the area in the device that can be defragmented
iv. determine the place of defragmentation in the overall system architecture of reconfigurable computing systems

CONCLUSIONS

The basic building block of most reconfigurable computing systems is field programmable gate arrays (FPGAs). The capabilities of FPGA devices have experienced significant growth in recent years primarily because of the inclusion of run-time partial reconfiguration feature onto current SRAM-based FPGAs. This feature allows a new module to be placed on a reconfigurable chip without interfering with the processing of other tasks being executed at that time. During the past decade and a half several techniques have been developed for the management of free spaces in reconfigurable systems. This paper discussed several of these techniques; these techniques have major advantages over full reconfiguration approach. However, as discussed in the paper several additional problems need to be addressed to realize the full potential of partial reconfiguration feature in FPGAs.

REFERENCES

1. Compton, K. and Hauck, S, "Reconfigurable Computing : A survey of systems and software", ACM Computing Survey, Vol.34, No.2., pp. 171-210, June 2002

2. Ahmadinia, et.al. , "A new approach for on-line placement on reconfigurable devices", 18th International Parallel and Distributed Processing Symposium 26-30 April 2004, Santa Fe, New Mexico

3. Walder, H., Steiger, C., and Platzner, H., " Fast online task placement on FPGAs, free space partitioning and 2D hashing ", Proc. Parallel and Distributed Processing Symposium, 2003.

4. Handa, M., and Vemuri, R., "An efficient algorithm for finding empty space for online FPGA placement", Design Automation Conf., June 2004.

5. Bazargan, K., Kastner, R., and Sarrafzadeh, M., "Fast template placement for reconfigurable computing systems,", IEEE Design and Test of Computers, vol. 17, pp. 68-83, January 2000.

6. Hauck, S., " Configuration prefetch for single context reconfigurable coprocessors", ACM / SIGDA International Symp. On FPGAs, 1998.

7. Shoa, A. and Shirani,R., "Run-time Reconfigurable Systems For Digital Signal Processing Applications: A Survey", Jour. of VLSI Signal Processing, March 2005

8. Dandalis,A., and Prassana,V., " Configuration compression for FPGA-based embedded systems", ACM/SIGDA Int.Symp.on FPGAs, pp.173-182, 2001.

9. Hauck, S., Li, Z, and Schwabe,E.,., " Configuration compression for the Xilinx XC6200 FPGA,"IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,vol. 18, Issue:8, pp. 1107-1113, 1999.

10. Turner et.al., "A virtual hardware handler for RTR systems," Proc. Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 262-263, 1999.

11. Ejnioui,A., and Demara,R ., " Area reclamation strategies and metrics for SRAM-based
reconfigurable devices", Proc.Intl.Conf. on Eng. of Reconfig. Systems and Algorithms", Las Vegas, 2005

12. Gericota et.al. "Run-time management of logic resources on reconfigurable systems". 2003 Design Automation and Test in Europe Conference and Exposition (DATE 2003), , Munich, Germany, March 2003

13. M. Handa and R.Vemuri, " Area fragmentation in reconfigurable operating systems", Engineering of Reconfigurable Systems and Algorithms, June 2004.

14 Pekete et. al., "Dynamic defragmentation of reconfigurable devices", ACM Trans. Reconfigurable Technology and Systems,vol.5.No.2,Article 8,June 2012

15. Z. Li et. al., "Configuration caching techniques for FPGAs", IEEE Symp. on 8th IEEE International Symposium on Field-Programmable Custom Computing Machines April 17-19, 2000.