Enabling High Level Design of Adaptive Systems with Partial Reconfiguration

Kizheppatt Vipin, Suhaib A. Fahmy
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore
vipin2@e.ntu.edu.sg, sfahmy@ntu.edu.sg

(PhD Forum Paper)

***Abstract*—Adaptive systems have the ability to respond to environmental conditions, by modifying their processing at runtime. While this is easy to do software systems, modern algorithms can be computationally expensive, requiring powerful processors. At the same time hardware is not as flexible. Field programmable gate arrays (FPGAs) are recognised as being suitable for adaptive systems implementation, due to their flexibility and high performance. The use of partial reconfiguration on FPGAs to implement adaptive systems has been proposed many times in the literature. However the design process for partially reconfigurable systems is complex and requires specialist knowledge on behalf of the application designer. Hence, it has remained a rarely used capability outside of academic circles. We propose a new approach to leverage partial reconfiguration within adaptive systems, by integrating with, rather than circumventing, supported vendor tool flows, while automating many of the steps that have made such designs more difficult in the past. This makes it possible for system designers with less FPGA expertise to use partial reconfiguration when designing adaptive systems.**

I. Introduction

Adaptive systems are able to respond to variations in their environment, leading to more sophisticated applications as well as improved application performance. For example, a software defined radio can change its modulation technique or coding scheme based on variations in channel conditions [1] and a driver assistance system can modify its analysis algorithms based on road conditions [2].

FPGAs appear to be a suitable candidate for adaptive hardware systems, due to their high performance and flexibility. Some of the hindrances associated with the adoption of FPGAs in the design of adaptive systems have been mitigated over the past few years. Specifically, dynamic partial reconfiguration (PR) is now supported in FPGA vendor design flows. PR exploits the fact that functionality implemented in an FPGA can be altered by modifying the contents of its configuration memory. By selectively modifying parts of this memory, portions of the system can be modified, while the remaining portions continue to operate. Vendors such as Xilinx already provide devices as well as tools that support partial reconfiguration and Altera has announced support for PR in their next generation FPGAs. PR has led to the concept of hardware virtualisation, in which unused regions of an FPGA can be repurposed for additional functionality, effectively time-sharing the silicon.

Although PR was introduced over a decade ago, it still remains an expert feature due to tool limitations and limited automation, and hence, system designers who are not FPGA experts find it difficult to exploit this feature. Current FPGA tools that support PR require the user to provide architecture-dependent information for the implementation of designs. The efficiency of the system implementation greatly depends upon the inputs provided by the designer. This mandates that the system designer should have considerable knowledge of FPGA architecture, and the mechanics of the PR operation, in order to efficiently integrate PR in their design. This architecture dependency makes PR less attractive to system designers, who are not FPGA experts.

Through our work, we aim to introduce a design framework and associated tools, which will allow high level system designers to develop efficient adaptive systems that leverage FPGA PR, without the need for detailed inputs related to low-level PR issues and target device architecture. Key design challenges have been identified and solutions and tools are being developed. These tools can be integrated with the vendor supplied tool chain, which makes the solutions portable across architecture developments.

II. Partial reconfiguration for Adaptive Systems

Presently, the only vendor-supported partial reconfiguration toolflow is from Xilinx Inc. Altera has announced upcoming support for PR in their next generation FPGAs. Although some earlier families of Xilinx FPGAs supported PR, it became popular with the introduction of the Virtex-5 family of FPGAs. The Xilinx tool chain supports PR through a software package called PlanAhead [3]. The current tool flow imposes a number of limitations that make designing PR-based adaptive systems difficult. To use this feature effectively, the designer is expected have considerable knowledge about the target hardware and the details of PR operations. This is because PR is treated as an advanced form of floorplanning, as opposed to having an application-centric view of adaptation.

978-1-4577-1740-6/11/$26.00

The main contributions of our proposed work are in two aspects of enabling PR for adaptive systems design. Firstly, providing support for system designers with less hardware expertise at design time, by making the architecture dependent details transparent and automating PR design implementation. Secondly, abstracting the runtime PR configuration details to a higher level. The primary challenge is to automate tasks that are currently manual, but are required to implement PR designs, while automatically minimising overheads.

The primary overhead associated with PR is reconfiguration time. This is the time required for the system to transit from one configuration to another. In order to develop efficient PR systems, hardware utilisation needs to be maximised and reconfiguration time minimised.

Although significant research work has been published on PR, most does not consider current device architectures. Many suggested solutions have practical limitations due to recent FPGA architectures moving away from regular repeating array structures of basic components to highly sophisticated devices with built-in macros and hard processors. Throughout our research, we consider modern FPGA architectures. Our proposed solutions are independent of lower level vendor specific FPGA design stages such as place and route. Our strategy is to integrate the solutions with vendor specific tools, so that the solutions are portable across vendor tools and evolutions in FPGA architecture. It is also important to note that we are interested in adaptive systems that react to external events and hence the sequence and likelihood of adaptation is not known in advance, unlike systems that use PR to schedule fixed task graphs.

A. Designing Adaptive PR Systems

We wish to enable system level designers, who are not FPGA experts, to generate area efficient PR schemes with minimal reconfiguration overhead. This work aims at providing support to the designers during the hardware design stage of the system. Specifically speaking, we aim to automate several operations the designers currently have to perform manually, while not compromising the efficiency of the system. In fact, intelligent automation can improve system efficiency by considering multiple optimisation objectives and it also saves design time.

1) System Partitioning for PR: Our initial task is to develop an efficient partitioning scheme for PR based designs. The partitioning process involves determining the number of reconfigurable regions required for design implementation and assigning modules to these regions. Reconfigurable regions are areas in the FPGA, where different modules are loaded at run time. Currently, the number of regions required in a system and the allocation of design modules to these regions are determined arbitrarily by the designer. In [4], a method for minimising the reconfiguration latency based on communication graphs is presented. However this assumes the number of regions is pre-determined. Determining the number of regions is not straightforward, and current devices and tools do not provide any support in this respect.

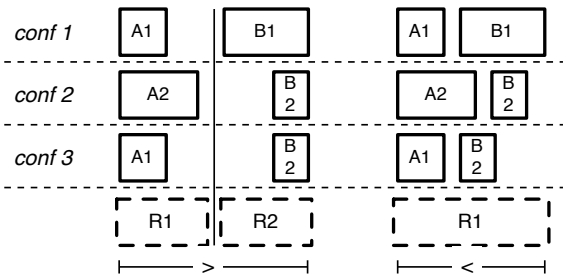

Fig. 1. When assigning modules to separate regions, if some configurations do not exist, combining modules into a single region can save area.

A recent contribution that explores partitioning and floorplanning of PR designs is presented in [5]. It describes a simulated-annealing-based algorithm for determining the module allocation to regions, based on the minimisation of area requirement variance at different time instances. It is difficult to extend this work for adaptive systems because the algorithm presented uses a scheduled task graph. Moreover the impact of reconfiguration time is not accounted for in their method.

The simplest way to partition the FPGA for partial reconfiguration is to divide the whole FPGA into two: one static region and one PR region. This approach has some benefits: the designer only needs to allocate a single region, large enough to hold the most resource hungry configuration; and the tools can optimise area usage and timing across all modules, resulting in the best possible timing performance and area. However, some major drawbacks mean this method is not ideal, such as high reconfiguration time and a large number of bitstreams. Hence, simply allocating all reconfigurable modules to a single region is not ideal for systems that rely on minimising reconfiguration time and bitstream storage.

Generally, a one-region-per-module approach offers the lowest worst-case reconfiguration times, since a region will only reconfigure when its sole module needs to, and the size of the region is only as large as the largest mode of that single module. However, a one-region-per-module approach is the least area-efficient way to allocate regions. This is shown in Fig.1. Here A and B are two modules. A_1 and A_2 are different operating *modes* of module A and B_1 and B_2 are different operating *modes* of module B. *Modes* are mutually exclusive implementations of the module with the same set of inputs and outputs. For example, a filter module can have two *modes*, one acting as a high-pass filter and one acting as a low-pass filter. In the possible system configurations, if the largest modes of the modules do not co-exist, a one-region-per-module scheme causes resource wastage. This shows that the partitioning of the design into regions and allocation of modules to those regions needs to be done intelligently.

We have presented a method for partitioning reconfigurable modules in [6]. We propose an efficient partitioning tool, which generates an area efficient partitioning of the adaptive system, while minimising reconfiguration overhead. Our

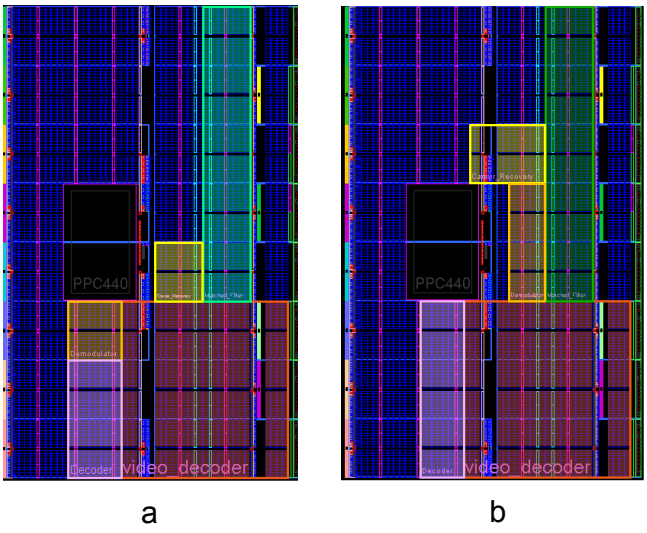

Fig. 2. Example results using the floorplanner. (a) with minimum resource wastage (b) with minimum wirelength

method takes advantage of the fact that not all possible combinations of different modes of modules will be required by the system. By considering only possible combinations of modes, called configurations, efficient partitioning of the design can be performed. A mathematical formulation is developed, aimed at minimising the total FPGA resource consumption as well as the reconfiguration time, while considering the constraints generated by the FPGA architecture and the PR operation. The tool also considers the architecture of the target FPGA, minimising required resources and defining each region in terms of configuration tiles as required by the implementation tools. It determines the total number of reconfigurable regions to use and the allocation of modules to those regions for the most efficient implementation of the system.

2) Floorplanning for PR: Another important factor considered during the design stage is floorplanning. For years, floorplanning was of little concern to FPGA system design engineers. The available vendor tools were sufficiently versatile to perform area-constrained placement and routing, while meeting timing requirements. The introduction of partial reconfiguration has altered this scenario. The tools currently available do not perform floorplanning for PR designs automatically, and require considerable input from the designer. This can be a time consuming activity and the results can be sub-optimal. Also, manual floorplanning requires the designer to have FPGA architecture knowledge.

Although some work has been done in this field ([5], [7], [8]), current solutions have several weaknesses. Most floorplanning methods suggested in the literature make simplistic architectural assumptions, and hence, are not suitable for modern devices, due to evolution in device architecture. The presence of a non-uniform distribution of hard macros such as DSP slices and Block RAMs as well as the presence of embedded processors makes automated floorplanning a challenge. In addition to this, PR brings additional constrains over floorplanning such as the regions being contiguous and rectangular in shape, a white space requirement around the regions for bus-macros, among others. Furthermore, all existing work we have found, focusses on the static properties of a particular placement. Hence, the placement is not optimised for the dynamic behaviour of a partially reconfigurable system.

We have developed an efficient floorplanner, which considers both the costs associated with PR, and the architecture of the FPGA. This algorithm takes into account factors such as required area for reconfiguration, resource wastage, wire length, and communication architecture. The floorplanner minimises the total area of the reconfigurable regions, which leads to reduced reconfiguration time, while also minimising the distance between reconfigurable regions, so that the maximum achievable frequency is improved. The output of the floorplanner is a set of area constraints, which specify the coordinates of the bottom left and top right corners of each region.

The floorplanning is based on a new method we call Dynamic Template Matching (DTM). The floorplanner begins by calculating the resource usage of each region in terms of reconfigurable tiles. When the floorplanner is integrated with the previously mentioned partitioning tool, this information is automatically generated. The floorplanner maintains a database of FPGA architectures that contains information about the location of FPGA resources. The locations of hard processors and transceivers are marked as unavailable in the database. FPGA resources are then merged to create structures, which are the basic unit of floorplanning. There will be different types of structures such as structures with DSP Slices and Block RAMs, structures with DSP Slices alone, structures with Block RAMs alone, etc. Basic structures can be merged to create larger structures.

Regions are floorplanned in terms of the structures based on a floorplanning schedule. Regions are scheduled for planning according to their resource utilisation and the weighting factor given to different resource types, based on their availability in the target FPGA. After floorplanning each region, the available type of structures and their numbers vary. The remaining regions can be floorplanned only using the remaining structures. Hence, we call this Dynamic Template Matching (DTM). Once the whole design is floorplanned, a weight function is calculated based on the resource wastage and total wire length. The floorplanning operation is repeated by considering different structures and the weight function is recalculated. The floorplan which generates the least weight value is selected for final region packing. Two examples of the floorplanner output are shown in Fig.2

The overall design-time framework for adaptive systems using partial reconfiguration is shown in Fig.3. The output of the floorplanner is passed to the vendor-supported PR tools to generate the necessary bitstreams. In this manner, as devices evolve and tool capabilities improve, our toolflow needs minimal adjustment.

B. Managing PR in Adaptive Systems

The design-time contributions detailed above, provide everything necessary for the PR system to be implemented. However, true adaptive systems need intelligent management of adaptation that is typically better done in software. During the design stage, the proposed tools build necessary infrastructure and interfaces for the management of PR. During runtime the system adapts based on operational conditions. Our work in this area will be develop previous work in [1] and [9].

An adaptive system can be considered as having two planes. The data plane implements the processing of data, such as the signal processing in a radio. The system designer uses a high level tool to describe this based on a library of IP cores. This is what is partitioned and floorplanned as described in Section II-A. The control plane implements the management and control functionalities. This plane is implemented in software due to its flexibility. The adaptation of the system can take place at different levels in the data plane:

1) Functional reconfiguration involves completely replacing the functionality of the data plane, e.g., a radio switching from a passive receiver to a transceiver.
2) Structural reconfiguration involves replacing, removing or introducing new components in the data plane, e.g., a radio changing the modulation scheme.
3) Parametric reconfiguration means modifying a parameter of one of the components within the data plane, e.g., changing the gain of a scaling component.

These adaptations are managed by the control plane by determining the appropriate type of configuration and loading the corresponding bitstreams or configuring internal register values. We have shown this technique to be effective for adaptation in cognitive radios [10]. Current PR tools do not provide automated support for different levels of reconfiguration. Adaptation needs to be explicitly coded by the designer, who must explicitly state which bitstreams to load. A design framework needs to be developed so that designers with less hardware expertise can specify complex adaptations using a high level system specification. From this high level description, the runtime should be able to enact the appropriate reconfiguration. The levels of reconfiguration are abstracted away and should be transparent to the system designer, with the mapping of adaptation to physical reconfiguration automated.

This framework is expected to work alongside a library of application-specific hardware IP modules such as signal processing libraries for software defined radios. Non-hardware designers can then use these IP blocks and their own software adaptation description to fully implement the adaptive system.

III. Conclusions and Future Work

Through this research work, we aim to develop a framework for adaptive system design using partial reconfiguration. Key challenging areas and objectives have been identified. An efficient partitioning algorithm has been developed and ground work on the floorplanner is already complete. As the next step, we are integrating the partitioning tool with the floorplanner and vendor place and route tools. Specific domains are identified, for which library components need to be developed. We believe that the availability of better tools and support will lead to wider adoption of PR in adaptive systems, for which we believe FPGAs offer the ideal mix of higher performance and flexibility.

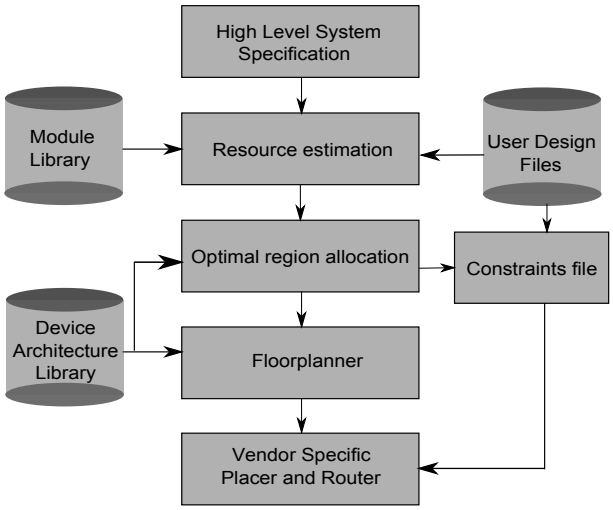

Fig. 3. Proposed design-time framework for PR based adaptive system.

References

[1] S. Fahmy, J. Lotze, J. Noguera, L. Doyle, and R. Esser, "Generic software framework for adaptive applications on FPGAs," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2009.

[2] C. Claus, R. Ahmed, F. Altenried, and W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems," *Reconfigurable Computing: Architectures, Tools and Applications*, vol. 5992, pp. 55–67, 2010.

[3] *UG702: Partial Reconfiguration User Guide*, Xilinx Inc., 2010.

[4] V. Rana, S. Murali, D. Atienza, M. D. Santambrogio, L. Benini, and D. Sciuto, "Minimization of the reconfiguration latency for the mapping of applications on fpga-based systems," in *Proceedings of IEEE/ACM international conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, 2009.

[5] A. Montone, M. D. Santambrogio, D. Sciuto, and S. O. Memik, "Placement and floorplanning in dynamically reconfigurable FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 4, pp. 24:1–24:34, November 2010.

[6] K. Vipin and S. A. Fahmy, "Efficient region allocation for adaptive partial reconfiguration," in *Proceedings of the International Conference on Field Programmable Technology (FPT)*, 2011.

[7] P. Banerjee, M.Sangtani, and S.Sur-Kolay, "Floorplanning for partial reconfiguration in FPGAs," in *Proceedings of International Conference on VLSI Design*, 2009.

[8] L. Singhal and E. Bozorgzadeh, *Multi-layer floorplanning for reconfigurable designs*. IET Computers & Digital Techniques, July 2007, pp. 276–294.

[9] J. Lotze, S. A. Fahmy, J. Noguera, and L. E. Doyle, "A model-based approach to cognitive radio design," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 2, pp. 455–468, February 2011.

[10] J. Lotze, S. Fahmy, J. Noguera, B. Ozgl, L. Doyle, and R. Esser, "Development framework for implementing FPGA-Based cognitive network nodes," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2009.