An Approach to a Fully Automated Partial Reconfiguration Design Flow

Kizheppatt Vipin, Suhaib A. Fahmy
School of Computer Engineering
Nanyang Technological Univesity
Nanyang Avenue, Singapore
{*vipin2,sfahmy*}*@ntu.edu.sg*

***Abstract*—Adoption of partial reconfiguration (PR) in mainstream FPGA system design remains underwhelming primarily due the significant FPGA design expertise that is required. We present an approach to fully automating a design flow that accepts a high level description of a dynamically adaptive application and generates a fully functional, optimised PR design. This tool can determine the most suitable FPGA for a design to meet a given reconfiguration time constraint and makes full use of available resources. The flow targets adaptive systems, where the dynamic behaviour and switching order are not known up front.**

Partial reconfiguration (PR) remains an advanced FPGA design method, where the designer is expected to understand low-level FPGA architecture details as well as factors affecting run-time management of reconfiguration. Available vendor-supported tools are not fully automated and several design steps must be performed manually. Optimising the mapping of adaptive systems to PR designs requires expertise and is time consuming. We propose a fully automated design flow for PR systems, which provides support and automation during the design phase as well as automating the run-time management of reconfiguration.

The proposed flow is shown in Fig. 1. The designer provides a high-level description of the system's behaviour in the form of a finite state-machine (FSM). Each state in the FSM represents a system configuration and the state transitions represent the adaptive behaviour of the system. All modules required in different system configurations should be available as RTL source files. The user also specifies the required timing constraints.

The tool first determines the resource utilisation of individual modules by synthesising them. It then partitions the design. This entails determining the number of reconfigurable regions (RRs) and allocating modules to them. Partitioning should be performed in such a way that it minimises overall reconfiguration time [1]. We use a modified hierarchical clustering algorithm which does this by grouping modules with a higher probability of coexistence into the same RR [2].

The output of the partitioning tool is then passed to the floorplanner, which determines the physical location of RRs on the FPGA. The result is a set of area constraints that can be used by the implementation tools to generate partial bitstreams. The primary objective of the floorplanner is to minimise resource wastage. We adopt the method in [3], that uses kernels of FPGA primitives which can be repeated in the vertical direction to satisfy a RR's resource requirements. The low-level place and route operations are then performed using vendor tools and partial bitstreams corresponding to each configuration of each RR are generated.

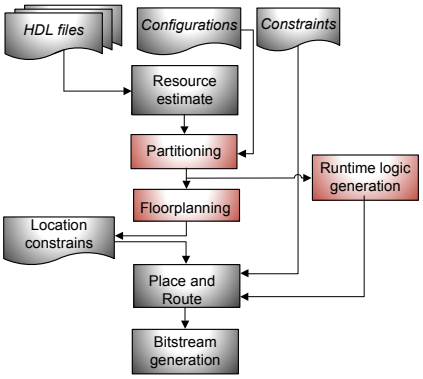

Figure 1. Proposed design flow.

The flow also includes generation of the runtime system that manages reconfiguration and implements the dynamic behaviour described by the designer. A high-performance custom ICAP controller is automatically inserted into the design after partitioning. A transformed finite state machine is implemented, combining the user-described behaviour and the flow-produced partitions. When the required conditions are triggered, the ICAP-controller automatically loads the corresponding partial bitstream(s) to configure the required region(s).

References

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

[2] ——, "Automated partitioning for partial reconfiguration design of adaptive systems," in *Proceedings of IEEE Int. Parallel and Distributed Processing Symp. Workshops and PhD Forum (IPDPSW)*, 2013.

[3] ——, "Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration," in *Reconfigurable Computing: Architectures, Tools and Applications*, 2012.