Towards Automatic Partial Reconfiguration in FPGAs

Fubing Mao[1], Wei Zhang[2], Bingsheng He[1]
[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, HK
fmao001@e.ntu.edu.sg, wei.zhang@ust.hk, bshe@ntu.edu.sg

I. Problem And Motivation

Partial Reconfiguration (PR) is an advanced reconfigurable characteristic for FPGAs and it has the capability to reconfigure specific regions of FPGAs while the other parts are still active or are inactive in a shutdown mode after its initial configuration. It provides many benefits for industry, *e.g.* sharing the same hardware resource for different applications.

It has become a very important and challenging problem to suitably prepare a good initial configuration as a reference for different applications and how to manage the static region and PR region resources for avoiding signal interruption incurred by PR [1]. Static logic means the logic will not be reconfigured during PR operation and PR logic means the logic can be reconfigured during PR operation [2][3]. On the one hand, since the initial configuration solution will be taken as a reference for later PR operation and different reconfigurations can have various properties *e.g.* area, aspect ratio and delay. It is difficult to take them into consideration during placement. On the other hand, it is challenging to determine how to efficiently place the static logics and PR logics in the FPGAs.

In [4], they introduced a global floorplan generation approach to obtain shared positions for common modules across sub-task instances. In [5], they proposed a resource-aware and reconfiguration-aware approach for PR heterogenous FPGAs. However, they all did not consider different shapes and delay for each PR logic during placement. The commercial PR CAD tools, such as PlanAhead [2], and TransFR [6] are used to deal with PR work. However, much more manual efforts are needed for the mapping steps when PR is involved. With respect to the academic CAD tools, few work considered module based placement and PR-aware routing.

The popularly used tool VPR [7] is based on the homogenous tiles [7] [8]. Still there is a missing link between the modern CAD tools and the PR applications. A good module based placer and PR-aware router is urgently needed. In this thesis, we attempt to fill this gap and propose an B*-tree [9] based modular placer to solve the requirement of PR.

II. solution And Contribution

We proposed following two key techniques to address the aboved challenges.

- Since functions (modules) with different shapes may have very different performance, we propose to simultaneously consider different aspect ratios and delay for each PR logic during placement to get the reference of PR logic.
- Module based placement introducing B*-tree representation is convenient to differentiate static regions and partial reconfigurable regions and we take the placement result as a reference for partial reconfigurable operations later.

In our work, we propose a module based placer including the techniques and integrate our placer into a mapping flow that can be used for PR operations. The flow has been developed on top of VPR and the experiment results demonstrate improvement on delay and execution time with acceptable area overhead compared with results of tile-based VPR [7].

Our main contributions can be summarized as follows:

- We provide an automatic placement tool which considers the static and PR modules at the same time. The shape, module size and delay of each context switching subfunction are considered during the placement.
- We utilize B*-tree representation to enable a fast modular placement on both fine-grained and coarse-grained fabric.

III. Current State of Our Work

A. Module-Based Placer

BMP (B*-tree Modular Placer) [10] simultaneously places both the static and reconfigurable modules and introduces B*-tree representation to enable a fast search [9] and convergence. Different module sizes and delay are considered during the placement and these properties can be used to guide simulated annealing (SA) to find a good placement result. We choose SA for its simplicity and effectiveness in finding the good placement. BMP can support the module library structure.

B. Module Library

The input files for the BMP should be prepared in advance. The benchmarks of reconfigurable modules were still unavailable when we proposed BMP. Thus, we selected and modified the cases in the VPR suite [7] for experiment. We decompose the circuit into static logics and PR logics (module) using an existing approach [2][3] and we regard each logic as a module. In order to get the netlist for connecting modules, the module library can be used. For library based design we do not need to integrate all the modules' verilog codes to get a complete netlist, which is commonly used. Instead we only need to provide the modules' list, connection port name and the top

978-1-4799-6245-7/14/$31.00 ©2014 IEEE

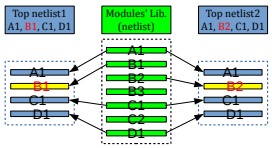


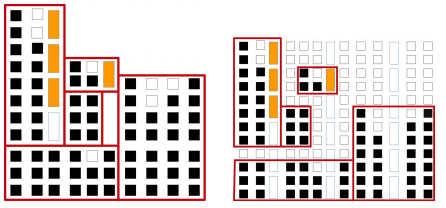
Fig. 1. Library based structure.

module IO information to design a new function. In order to finish the design, it only needs to start from the physical synthesis stage. An example of a library based structure is shown in Fig. 1. We assume that A, B, C and D are four modules, where only B and C are reconfigurable modules with multiple contexts. Currently the designer specify A1, B1, C1 and D1 as a module list and their connections have been collected from the top netlist. Then the placer fetches the corresponding modules from the library to combined into a complete netlist. Our implementation will support the context switch for the reconfigurable module B and C.

B*-tree can efficiently support modules with coarse-grained resources. Fig. 2(a) is a floorplan result in which some modules contain coarse-grained resources. Each part is a module and independent to others. The Fig. 2(b) is the mapping placement result, we can see it matches well with the floorplan result.

IV. Preliminary Results

We briefly present a placement result of benchmark *Boundtop* in Fig. 3 to demonstrate the promising results of our proposal. In Fig. 3(a), the red rectangle is the total floorplan area of the benchmark. The grey area is the total area of all the modules and their topology information is shown in the figure. The white space represents the empty space which is not occupied by any modules. We omit demonstration of IO which is around the red rectangle. There are totally 13 modules in *Boundtop* and we index them starting from 0. The placement considers the different ratios and delay of each context switching sub-function which previous work does not take into account. We use linear weighted function to evaluate the quality of the solution and use the module delay got from VPR routing result to reflect the module property. The placement result on VPR is shown in the Fig. 3(b). Each red rectangle area represents a mapping area for the corresponding module. To evaluate the efficiency of the BMP, we compare the experiment result between the tile-based flow (VPR) and our work (placer tool). For fairness, we use the router of VPR to run our placement result for getting delay result. On different experimental settings, our tool can achieve 3.71% to 21.29% improvement in delay and save the execution time by 77.28% on average with up to 19.89% area overhead.



(a) Coarse-grained module floorplan. (b) Coarse-grained module placement.

Fig. 2. Coarse-grained module floorplan and placement result

V. Conclusions And Future Work

Our work proposes a B*-tree Modular placer (BMP) which supports the partial run-time reconfiguration and heterogenous resources. The proposed module-based placer used the modified B*-Tree representation to optimize the floorplanning and placement of the modules with the consideration of flexible module ratio. The corresponding parameters for cost functions and searching algorithms are explored in the experiments. Compared to the state-of-the-art tile-based placement [7], the results of the proposed B*-tree Modular Placer have improvement on delay and execution time with acceptable area cost due to the empty space. We believe the automatic tool will bridge the gap between the academic research and industry. In the future, the following directions will be considered.

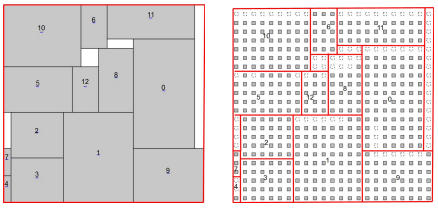
- We plan to investigate PR-aware routing and the context switching in the PR region.
- We will map the result to the commercial FPGA boards, *e.g.,* Xilinx FPGA board and Altera FGPA board.
- We will study other floorplanning representations, such as, Sequence Pair to optimize the floorplan.
- We will introduce other heuristic search algorithms, *e.g.,* Tabu Search, Genetic Algorithms to the problem.

Acknowledgment

This work is partly supported by a MoE AcRF Tier 2 grant (MOE2012-T2-1-126) in Singapore.

References

[1] P. Sedcole, B. Blodget, J. Anderson, P. Lysaghi, and T. Becker, "Modular Partial Reconfigurable in Virtex FPGAs," in *FPL*, 2005, pp. 211–216.
[2] Xilinx, "Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite," 2012, http://www.xilinx.com.
[3] Altera, "Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs," 2010, http://www.altera.com.
[4] P. Banerjee, M. Sangtani, and S. Sur-Kolay, "Floorplanning for Partially Reconfigurable FPGAs," *TCAD*, vol. 30, no. 1, pp. 8–17, 2011.
[5] L. Nan, C. Song, and T. Yoshimura, "Resource-aware multi-layer floorplanning for partially reconfigurable FPGAs," *IEICE Transactions on Electronics*, vol. E96-C, no. 4, pp. 501–510, 2013.
[6] Lattice, "Field Update FPGAs While System Operates," 2005, http://www.latticesemi.com.
[7] J. Rose, J. Luu, and e. Yu, Chi Wai, "The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing," in *FPGA*, 2012, pp. 77–86.
[8] P. Chow, "The Design of a SRAM-Based Field-Programmable Gate Array–Part II: Circuit Design and Layout," *TVLSI*, vol. 7, no. 3, 1999.
[9] T. Chen and Y. Chang, "Modern Floorplanning Based on B*-Tree and Fast Simulated Annealing," *TCAD*, vol. 25, no. 4, pp. 637–650, 2006.
[10] F. Mao, Y.-C. Chen, W. Zhang, and H. Li, "BMP: A Fast B*-Tree based Modular Placer for FPGAs," in *FPGA*, 2014, pp. 248–248.



(a) Boundtop floorplan. (b) Boundtop placement.

Fig. 3. Floorplan and placement results.