

A Design Workflow for the Identification of Area Constraints in Dynamic Reconfigurable Systems

Alessio Montone, Marco D. Santambrogio and Donatella Sciuto
Politecnico di Milano - Dipartimento di Elettronica e Informazione
Via Ponzio 34/5 - 20133 Milano, Italy
alessio.montone@dresd.org, {santambr, sciuto}@elet.polimi.it

Abstract

Nowadays, working in a reconfigurable scenario, the identification of the placement constraints that will be used to implement each module used to define a reconfigurable system, is an important problem that needs to be solved. The proposed approach exploits results of different area constraints assignment techniques in order to optimize a given objective function. This approach can easily be extended to support global optimizations over several (all) configurations of the reconfigurable system under development.

1 Introduction

Floorplanning and area constraints definition is a well known problem and it has been faced for several decades and several metrics have been proposed for ASICs. Depending on the chosen metrics several optimization heuristics have been also proposed [1, 9]. While a lot of effort has been devoted to floorplanning for ASICs, floorplanning studies for FPGA and more particularly for reconfigurable architectures are at early stages and only few results and methodologies have been proposed.

In a reconfigurable architecture there is a static part and a reconfigurable part [2, 3, 10]. The reconfigurable part is the most delicate from the area constraints point of view due to the fact that there is no single configuration to be adjusted, but several of them. Given a set of reconfigurable modules m_i , $i = 1 \dots M$ and a set of time intervals t_k , $k = 1 \dots K$, the system configuration at each time is described through a binary relation $\varphi(i, k)$: i.e. a module m_i has to be configured and running at \bar{k} if and only if $\varphi(i, \bar{k})$ holds. Given a time \bar{k} , $P(\bar{k})$ is defined as the set of all the modules being up and running at \bar{k} , i.e.

$$P(\bar{k}) = \{m_i \mid i = 1 \dots M \mid \varphi(m_i, \bar{k}) = 1\}$$

$P(\bar{k})$ will be referred as *static photo* of the reconfigurable system at instant \bar{k} .

A system, in order to be feasible, has to meet its resource constraints. Given

- a set of resources r_l with $l = 1 \dots L$,
- $b_{m,l}$ the number of resources of type l needed by module m ,
- B_l the number of resources of type l available.

The resource constraints at any time can be written as:

$$\sum_{i=1}^M \varphi(m_i, k) b_{m,l} \leq B_l \quad \forall l, k$$

An area of the FPGA has to be assigned to each module belonging to static photo $P(\bar{k})$ and also, in order to allow a modular reconfigurability a set of area constraints has to be assigned. Aim of this paper is to present an approach to solve the floorplanning problem for a single static photo optimizing a given objective function. One of the most remarkable works proposed in literature on the topic is [7]. In [7] the floorplanning problem of reconfigurable architectures is solved by optimizing the floorplanning of each static photo and then refining each floorplanning between two time-adjacent static photos using simulated annealing. The main drawback of such a technique is that the objective function is minimized on each single static photo and not globally on the entire set of static photos.

Currently, automated tools like PlanAhead [6] provided by Xilinx Inc. support modular floorplanning using brute force techniques that are often much time consuming thus requiring entire running days to optimize non-trivial designs. Even with such brute force approach such tools do not support floorplanning for reconfigurable architectures, due to the fact that modules belonging to different static photos have to be considered at the same time.

Section 2 presents the proposed workflow describing the techniques used to assign the area constraints to each module used to define the reconfigurable system under development. The solution space exploration done using the proposed approach is described in Section 3. In order to verify the proposed flow several components and architectures have been implemented, but in order to provide an experimental support to the proposed approach a SDRAM memory controller will be hereby considered. Finally Section 4 presents the conclusion and future work.

2 The Proposed workflow

The flow proposed in this paper needs as input the set of reconfigurable modules m_i written in a hardware description language such as VHDL or Verilog. Also $\varphi(m_i, k)$ and B_l are provided as input, the former depending on the architecture that is going to be implemented, the latter depending on the device on which the architecture is going to be deployed.

During the preliminary phases of the flow each module is synthesized in order to get the list of required resources. During this preliminary phase the previously introduced a matrix $b_{m,l}$ is built up. According to φ and B_l an early feasibility check can be performed. The first phase of the proposed algorithm consists in the definition of a starting feasible floorplanning for each static photo $\varphi(\cdot, k)$. During the *Resource Assignment* phase all the available resources B_l are assigned to each $m_l \in P(\bar{k})$. Then each module goes through a *Shape Identification* phase: according to the objective function a shape for each module m_l is proposed in order to optimize the given objective function. In this phase we look for an optimal floorplan shape for each module, but adopting optimal shaper for each module of a static photo $\varphi(\cdot, \bar{k})$ is not a feasible solution, hence the optimal shape is used just as a starting point for the core of the algorithm.

All of the state of the art algorithms face the floorplanning problem by optimizing at most a single static photo per time, the improvement given by the proposed work flow is that the entire architecture is optimized with respect to the considered objective function. The core of the optimization process is given by the last phase generating the final placement constraints.

Aim of the latter stage is the identification of the placement constraints that will be used to implement each module. The previous phase generated an optimal shape, but the problem that still need to be solved is the identification of the placement constraints taking into consideration the fact that those modules are not configured as single core on the reconfigurable device, but they have to share the reconfigurable resources with other configuration codes. The UCF Builder and Analyzer (BUBA) tool accept as input the starting area solutions computed by the previous stage, a static

scheduling of the application and the information regarding the reconfigurable device that has to be used to implement the desired design. Due to these parameters BUBA tries to assign a placement constraints to each module using different strategies, as proposed in Section 2.3. At the end of the flow a set of area constraints is provided, one set for each static photo composing the design. In the next section further details of each phase are proposed.

2.1 Design Synthesis

This stage is used to synthesize each functionality description, provided to this stage in VHDL or Verilog, to estimate the resources that will be required to define the corresponding configuration code.

2.2 Floorplanning Reconfiguration Driven

This stage aims at the definition of the area constraints for each configuration code. Once the estimation for the resources required by each configuration code are provided by the design synthesis phase, it is possible to identify a floorplanning constraint that takes into consideration both the resource requirements and the constraints introduced by the reconfigurable scenario (i.e. working with a Xilinx device, a width constraint multiple of 4 slices [4, 5]).

In order to define this area constraint it is possible to use the fragmentation index, that represents a percentage of wasted space (relative to the size of a region) and that can be computed using the following formula:

$$\Phi = \frac{(\#region_slices) - (\#occupied_slices)}{\#region_slices} * 100 \quad (1)$$

In the fragmentation index the $\#region_slices$ parameter represents the number of slices available in the area defined by the area constraint into the UCF, User Constraint File, file, while the $\#occupied_slices$ value defines the actual number of slices used by the physical implementation of the core into its assigned area. To obtain for each module the $\#occupied_slices$ the corresponding VHDL description has to be synthesized¹. The floorplanning reconfiguration driven stage provides as output an area constraint aware of all the constraints introduced by the reconfiguration scenario.

2.3 Placement Constraints

Aim of this stage is the identification of the placement constraints that will be used to implement each configuration code. The floorplanning reconfiguration driven stage

¹The accuracy of the $\#occupied_slices$ value affect dramatically the quality of the placement

provides a set of feasible area constraints, but the problem that still needs to be solved is the identification of the placement constraints taking into consideration the fact that those configuration codes are not configured as single core on the reconfigurable device, but they have to share the reconfigurable resources with other configuration codes. The UCF Builder and Analyzer (BUBA) tool accepts as input the starting area solutions computed by the previous stage, a static scheduling of the application and the information regarding the reconfigurable device that has to be used to implement the desired design. All this information is described in the *system characterization* file. Given these parameters, BUBA tries to assign, in a greedy way, the placement constraints to each module trying to minimize the number of reconfigurations. This is possible due to the fact that a module can be requested to be executed at different times and not only once. In such a scenario there might be a placement solution able to keep configured this configuration code on the reconfigurable device, without affecting the quality of the schedule, without having to reconfigure the same module twice or more just because it is no longer available on the reconfigurable device. Once the new placement constraints are defined this information is stored into the *system characterization* file and in the UCF file.

A *game theory*-based technique is now under development. The idea is to consider each core as a *player* in the game, where the objective of each player is the best placement assignment [8]. The set of all the players identifies the *society*. In this context the best solution for the society may not be equal to the best assignment of each core, that means that a placement assignment solution for a system corresponds to an equilibrium.

3 Experimental results

This section describes the solution space of the proposed approach. In order to verify the proposed flow, several components and architectures have been implemented, and the proposed approach will be exemplified through a SDRAM memory controller.

Several metrics can be used in order to rate the floorplanning of a module, our flow considers three of them:

- the fragmentation factor
- the length of critical path
- the average length of the 10 longest paths

Previously it has been underline how one of the most critical steps in order to obtain a good result consists in the choice of the initial floorplanning for each module.

A single objective function has been defined

$$\Theta = A\Phi^2 + B\left(\frac{\tau_{\max}}{\max_{\text{design}}\{\tau\}}\right)^2 + C\left(\frac{\tau_{10\text{longest}}}{\max_{\text{design}}\{\tau\}}\right)^2 \quad (2)$$

In order to let Θ be the sum of parts belonging to the interval $[0, 1]$ the time components have been normalized using the longest critical path. Hence if the considered module is responsible for the longest critical path then Θ will have a normalized time component equal to 1, and any variation to the critical path due to the floorplanning will be appreciated more than any other improvement obtained i.e., decreasing fragmentation. The square power underlines variations in critical components.

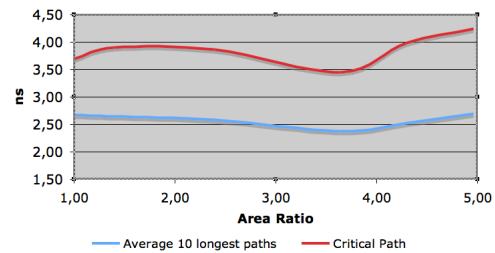


Figure 1. Critical Path variation with respect to a variation of assigned square area

Figure 1 plots how circuit's delay (after technology mapping on a Xilinx Virtex II 1000 FPGA) varies assigning different square areas (i.e. allowing more dead space that is space not occupied by logic). All the areas are normalized to the smallest assignable area (i.e. the area occupied by the logic without allowing dead space). There critical path delay and the average of the 10 longest paths are plotted. As the allowed dead space increases, the resulting routing becomes less knotty (routing resources on FPGA always represents a bottleneck) and critical paths are shortest, this effect decreases as allowed dead space increases. Concurrently, as the assigned area increases the logic becomes more sparse (due to the implementation of placers) critical path increases. In the figure 1 the less-knotty-routing effect causes the critical path to decrease for area ratios below 3.5, while for values above 3.5 the sparse-logic effect leads to an increment of the critical path. Our experience shows that this behavior can be observed in the majority of modules.

Such conclusions can be drawn also for other components of objective function Θ , hence each component of Θ has a parabol-like shape and a local search algorithm can be applied to find an area constraint that minimizes Θ .

Due to its algebraic definition, Θ is given by the sum of functions that can be approximated by parabolic functions along wide intervals, hence also Θ can be approximated locally by polynomial functions. Hence results of

Table 1. SDRAM memory controller technology Mapping results, corresponding to different area constraints assignment

Width	Height	$\frac{Height}{Width}$	Longest Delay	Avg. Conn. Delay Worst 10 Nets	fragm. index
13	13	1.00	3.90	2.75	0.85
17	17	1.00	3.87	2.58	0.49
21	21	1.00	3.45	2.38	0.32
23	23	1.00	3.99	2.52	0.27
9	21	0.43	2.94	2.39	0.76
9	17	0.53	3.20	2.39	0.94
9	25	0.36	4.02	2.67	0.64
17	9	1.89	3.69	2.79	0.94
21	9	2.33	3.88	2.54	0.76
25	9	2.78	4.04	2.42	0.64
5	31	0.16	3.74	2.93	0.93

Lagrange multipliers theory can be applied to find the singularity points of the function. That is why the proposed approach adopts an optimized local search method driven by the discrete gradient of Θ .

We assigned different area constraints to the OpenCores SDRAM memory controller and Table 1 presents the results of the corresponding technology mapping on a Xilinx Virtex II 1000 FPGA. As usual, in order to describe the shape of the assigned rectangular area, a coefficient given by the ratio between height and width of the edges has been associated with each feasible solution. Using such coefficients, the improvements given by scaling assigned area can be easily pointed out. Looking at the table can be notice that an increase in the assigned area does not always result in a better timing result, this is due mapping tools tendency to spread logic along the assigned area without caring, unless explicitly specified, about global timing. While generally a square area is the most conservative one, in this case an area with aspect ratio equal to 0.43 is the preferred from critical path point of view, this result is due the netlist topology of the controller.

The proposed work-flow generates an optimal floorplanning for each module exploiting the described mathematical approach. Each set of modules belonging to the same static photo goes through a final floorplanning phase using a greedy algorithm to schedule as best as possible the entire static photo.

4 Conclusions and Future Works

In this paper it has been shown how the reconfiguration capability of FPGAs introduce another degree of freedom in the floorplanning solution space. While several algorithms

and approaches have been introduced in the past for static architectures floorplanning, such problem has not been yet widely analyzed for reconfigurable architectures.

In this paper a work flow for floorplanning tailored for reconfigurable architectures has been described, such approach is based on the definition of the best area constraint for each module followed by the application of a greedy search strategy driven by the optimization of a given objective function. It has been found that, in order to find a good final floorplanning, one of the most critical point is the identification of the best area constraint for each module with respect an objective function, such identification will directly effect the resulting global floorplanning.

In order to find the best constraints a local-search-like algorithm has been proposed and also an objective function considering both area occupation and critical path timing. The proposed approach has been validated through several examples. In the proposed approach the final schedule is obtained through a greedy search strategy, hence one of the possible future works will consist in the introduction of new algorithms based on game theory or simulated annealing (in both traditional or quantistic version), the latter are widely used for the floorplanning of standard VLSI architectures.

References

- [1] Y. Feng. Heterogeneous floorplanning for fpgas. In *International Conference on Embedded Systems and Design*, 2006.
- [2] F. Ferrandi, M. D. Santambrogio, and D. Sciuto. A design methodology for dynamic reconfiguration: The caronte architecture. In *The 12th Reconfigurable Architectures Workshop (RAW 2005)*, 2005.
- [3] M. P. Heiko Kalte and U. Ruckert. A prototyping platform for dynamically reconfigurable system on chip designs. In *Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip (SoC)*, 2002.
- [4] X. Inc. Two Flows of Partial Reconfiguration: Module Based or Difference Based. Technical Report XAPP290, Xilinx Inc., November 2003.
- [5] X. Inc. *Early Access Partial Reconfiguration Guide*. Xilinx Inc., 2006.
- [6] X. Inc. *PlanAhead User Guide*. Xilinx Inc., July 27, 2007.
- [7] S. L. and E. Bozorgzadeh. Multi-layer floorplanning on a sequence of reconfigurable design. In *Field Programmable Logic and Applications*, pages 1–8, 2006.
- [8] M. J. Osborne. *An introduction to game theory*. Oxford University Press, 2002.
- [9] I. L. M. Saurabh N. Adya. Fixed-outline floorplanning through better local search. In *ICCD*, 2001.
- [10] M. Ullmann, M. Hübner, B. Grimm, and J. Becker. An fpga run-time system for dynamical on-demand reconfiguration. In *Proc. of the 18th International Parallel and Distributed Processing Symposium*, 2004.