2018 IEEE International Parallel and Distributed Processing Symposium Workshops

Multi-Start Simulated Annealing for Partially-Reconfigurable FPGA Floorplanning

François Galea, Sergiu Carpov, Lilia Zaourar
CEA, LIST
91191 Gif-sur-Yvette, France
Email: first.last@cea.fr

***Abstract*—FPGA floorplannig consists in finding a satisfactory placement of the different pre-determined regions of a design, onto the resource matrix that composes the FPGA hardware. Performance is ensured by minimizing the distances between communicating regions, as well as between regions and their I/O ports on the FPGA. To this very challenging problem, additional features can be added, such as taking into account the existence of partially-reconfigurable regions. This paper presents the solution method we proposed in the RAW Floorplanning Desing Contest, organized for the 25th Anniversary of the Reconfigurable Architectures Workshop (RAW), held in conjunction with the IPDPS'18 conference. The solution method is a multi-start simulated annealing procedure, which won the contest.**

I. Introduction

For over 10 years FPGA devices such as the Xilinx Virtex series have been featuring the possibility of partial reconfiguration, in order to modify an operating FPGA design while it is in use. However, floorplanning in FPGA design with both static and Partially Reconfigurable Regions (PRRs) became a very challenging problem since this directly affects the feasibility and the performance of the final solution. In addition, different constraints due to the variety of resources in modern FPGA such as CLB blocks, DRAM, DSP, etc. have to be also considered. Therefore, a plentiful literature about FPGAs floorplaning that explores several methods have been proposed to automate floorplaning with various constraints and formulations. This is also the reason for the floorplanning design contest proposed by the organisers of the Reconfigurable Architectures Workshop [1] leading to this work.

This paper is organized as follows: the rest of this section presents an overview of some of the most significant related works as well as a concise formulation of the problem, then Section II presents our methodology based on multi-start simulated annealing, and Section III is dedicated to the numerical results. Finally a conclusion is given in Section IV.

A. State of the art

In the plentiful literature of FPGA floorplanning few papers deal with both static modules and PRRs and only some of them consider the FPGAs with heterogeneous resource distribution as it is the case in this work. We resume here only few works that are relevant for this contest.

One of the first works that considers the heterogeneity of FPGA resources has been presented in [2]. The algorithm exploits Simulated Annealing (SA). However, the approach assumes homogeneous resources distribution in the FPGA which is not the case in recent devices.

A three-step floorplacer, also based on a SA strategy, has been proposed in [3]. It considers both partitioning and floorplanning. The aim is to optimize external wirelength and area. However, they consider only a uniform resource distribution.

Furthermore works of [4] and more recently in [5] deal with floorplanning on heterogeneous FPGAs. They consider arbitrary resource distribution and aim to optimize the internal and external wirelengths. But they are limited to a suboptimal search space and the objective function is only based on wirelength. In particular, in [5] they suggested treating static modules similar to PRRs which adds unnecessary constraints.

In [6] the authors introduce Columnar Kernel Tesselation to take into account the complex device architecture. Their objective is to optimize both area and internal wirelength but they do not explore potentially promising solutions because they use a fixed placement ordering of the regions. Objective function is also biased towards area optimization only.

Later, two approaches for the identification of area constraints on PRR-enabled FPGAs have been introduced in [7]. In this work, the authors consider arbitrary distribution of heterogeneous resources. Moreover, they also give the possibility to customize the objective function and control on the quality of the desired solution.

More recently, the tool presented in [8] tackles both static and PR regions. It proceeds first to a recursive bipartition based on *Gurobi solver* then determines feasible placements. Moreover, in [9] the authors proposed an explicit enumeration of the possible placements of each region.

They propose a genetic algorithm, enhanced with a local search strategy, to automate the floorplanning activity on the defined direct problem representation.

In this work, we consider partially-reconfigurable FPGAs with heterogeneous resources, and specific constraints on the PRRs. The objective function takes into account the total placement area as well as the total wirelength.

B. Problem formulation

The problem is precisely described on the RAW Floorplanning Design Contest website [1]. We present here a minor reformulation which better suits our method needs.

FPGA description: The FPGA representation consists of a rectangular matrix of *blocks*, each block corresponding to

978-1-5386-5555-9/18/$31.00 ©2018 IEEE
DOI 10.1109/IPDPSW.2018.00208

either an available set of resources of a specified type (CLB, BRAM, DSP), unused resources (NULL), or a forbidden block, which must not be used in a feasible solution. Each block has a dedicated set of row and column coordinates, an may also feature an input-output (I/O) port. Block dimensions are noted B_w (block width) and B_h (block height).

The FPGA description also features two binary vectors v_{LEFT} and v_{RIGHT}, with one bit per column. For a specific column col, the values $v_{LEFT}(col)$ and $v_{RIGHT}(col)$ specify that the respectively left and right borders of partially-reconfigurable regions may be placed at this specific column if the value is 1, and must not if the value is 0.

The FPGA matrix is subdivided in *tiles*, which are rectangles of blocks whose width is one block and whose height in blocks is a constant called *tile height*.

Regions description: The problem features a set of *regions* R, which have to be allocated in the form of disjoint rectangular subparts of the FPGA matrix. Each region $r \in R$ has a set of resource demands $\{d_{r,CLB}, d_{r,BRAM}, d_{r,DSP}\}$, each value corresponding to the demand of blocks for a specific resource type (CLB, BRAM, DSP). This region also features a (possibly empty) set of *I/O ports*, noted P_r. Each port $p \in P_r$ is defined as a 3-tuple $p = (x, y, nc)$, corresponding to the column and row of the I/O port in the FPGA matrix, and a number of connections from the region to the I/O port. The problem also features an *interconnection matrix* IC where each component $IC_{r_1 r_2}$ corresponds to the number of wires connecting region r_1 and region r_2.

A region is either static or partially-reconfigurable. In the latter case, additional placement constraints apply.

Problem variables: One must decide, for each region $r \in R$, where it is placed on the FPGA matrix. As each placement is a rectangular subset of the matrix, it can be represented as a 4-tuple $Pl(r) = (x_0, y_0, x_1, y_1)$, whose components correspond, in order, to the column and row numbers for the upper left corner of the placement, and the numbers for the bottom right corner's column and row (inclusive). A solution of the problem is a set Pl of placements for all regions.

Constraints: All placements in Pl must satisfy the resource demands, meaning each of them must cover a sufficient number of blocks with regard to the demand for each block type. The given model specifies that placements must not overlap, *ie.*, their intersection must be empty.

In addition to these basic constraints, two additional technology-driven constraints have been added for the case of PR regions. Firstly, x_0 and x_1 (leftmost and rightmost columns) placement values for PR regions must be such that $v_{LEFT}(x_0) = 1$ and $v_{RIGHT}(x_1) = 1$. Lastly, two PR regions must not share the same tile.

Objective function: The given objective function is a combination of different costs, depending on the total FPGA usage. One has to maximize a *Score* defined as follows:

$$Score = M - AW \cdot A_{cost} - WW \cdot (IO_{cost} + IC_{cost})$$

. M, AW and WW are constants corresponding to the maximum score, and the weights for area and wirelength costs.

A_{cost} is the area cost. It is a weighted sum of the total number of used CLB, BRAM and DSP blocks in the solution. The corresponding weights are parameters given in the instance file. Minimizing this cost corresponds to reducing the number of unused blocks, also improving the chance to find a feasible solution.

IO_{cost} is the wirelength cost for I/O connections. It can be formulated as follows:

$$\sum_{\substack{r \in R \\ (x_0,y_0,x_1,y_1) \\ = Pl(r)}} \sum_{(x,y,nc) \in P_r} \left(\left| \frac{x_0 + x_1}{2} - \left(x + \frac{1}{2} \right) \right| \cdot B_w + \left| \frac{y_0 + y_1}{2} - \left(y + \frac{1}{2} \right) \right| \cdot B_h \right) \cdot nc$$

. IC_{cost} is the wirelength cost. Its value is

$$\sum_{\substack{r_1 \in R \\ (x_0,y_0,x_1,y_1) \\ = Pl(r_1)}} \sum_{\substack{r_2 \in R \\ (x'_0,y'_0,x'_1,y'_1) \\ = Pl(r_2)}} \left(\left| \frac{x_0 + x_1}{2} - \frac{x'_0 + x'_1}{2} \right| \cdot B_w + \left| \frac{y_0 + y_1}{2} - \frac{y'_0 + y'_1}{2} \right| \cdot B_h \right) \cdot IC_{r_1 r_2}$$

.

Infeasibility cost: In our implementation, as some instances may be very constrained, we decided to relax all of them and model them into an *infeasibility cost* function, noted Inf_{cost}. The infeasibility cost of a solution can be interpreted as how distant the solution is from a feasible solution. If the infeasibility cost is equal to zero, then the solution is feasible.

The infeasibility function is the sum of squared individual constraint violation costs. We measure a constraint violation by the number of FPGA matrix blocks that do not satisfy the constraint. For instance, if two region placements overlap with three blocks, the corresponding violation cost is 3, hence the infeasibility function value is increased by $3^2 = 9$. As another example, if the placement for a region lacks two CLB blocks and one DSP block, the increase in the infeasibility function value will be of $2^2 + 1^2 = 5$. The penalty cost for a region placement not meeting a PRR column alignment constraint is the square of the height of the placement. Finally, the penalty cost for two PRR placements sharing tiles is the square of the number of tiles that intersect both placements.

Squaring individual costs adds more penalty to large constraint violations. This corresponds to a first stage of the penalty method described in [10].

Therefore, instead of the Score function of the problem description, our objective function, to be minimized is

$$Obj = B \cdot Inf_{cost} + AW \cdot A_{cost} + WW \cdot (IO_{cost} + IC_{cost})$$

, where the constant B is chosen as a tradeoff between the needs to obtain a feasible solution with a good probability and to reduce the solution time. In practice, the value of $B = 2^{16}$ has shown good results on all tested problem instances.

II. Methodology

We implemented a solution method based on simulated annealing. In order to improve the quality of the results, this method has been integrated into a multi-start framework.

A. Simulated Annealing

Simulated annealing [11] (SA) is an iterative process, which step-by-step explores solutions by randomly generating a new solution from a current one, using a neigbourhood function. Each generated solution is sumbitted to a probabilistic test based on the solution value (or energy) variation Δ_E and a current temperature value T. The acceptance probability is 1 when $\Delta_E < 0$, meaning the solution value is decreasing, and it is $e^{-\frac{\Delta_E}{T}}$ otherwise. If the test is successful, the new solution becomes the current solution from which new solutions are generated and tested. A cooling scheme regularly decreases the temperature, reducing the probability of accepting solution updates with positive energy variations.

Initial solution: As the initial solution, we choose placements for the region as 1×1 rectangles randomly placed on the FPGA surface. Even though this solution is obviously infeasible (its $Inf_{cost} > 0$), the optimization process is designed to converge to a feasible solution with a sufficiently high enough probability.

Neighborhood function: In our method, we designed two neighbourhood functions, combined in a single function. The first function alters the solution by $+1$ or -1 one of the coordinates of the placement of one randomly chosen region. The second function swaps the placements for two randomly chosen regions. At each iteration of the SA procedure, one of those two functions is selected at random: the swapping function is chosen with a probability of 0.25, and the coordinate-altering function is used the rest of the time.

SA parameters: The parameters for our SA implementation which determine the initial temperature, the cooling scheme and the stop criterion are as follows.

As initial temperature T_0, we choose a value for which we estimate the probability for accepting solutions is $P_0 = 0.9$. This is done by evaluating the average solution value increase $\bar{\Delta}$ by executing a small number of SA iterations from the initial solution, using only the transitions with positive cost increase. Then $T_0 = -\frac{\bar{\Delta}}{\log P_0}$.

The number of steps in each temperature level is set to a problem size value we consider to be the product of the FPGA matrix width, its height, and the number of regions.

From temperature level k with temperature value T_k, we compute the temperature for level $k+1$ with the formula: $T_{k+1} = \alpha T_k$ where the *cooling ratio* α is a constant parameter such as $0 < \alpha < 1$. In our implementation, we set $\alpha = 0.995$, and the algorithm stops when $T_k \leq 1$.

B. Multi-Start

Due to the stochastic nature of the SA process and the very large number of solutions that can be explored, we have no guarantee that neither one execution of the SA method will find a solution with sufficiently good quality, nor it will find a feasible solution. In order to increase the likelyhood that we obtain a relatively good solution, for each instance we run the SA solver several times, with different initialization parameters for the random generator. Since the different SA executions are independent, they can be executed in parallel.

III. Numerical results

The experimental scheme is the following.

A. Instances

A set of 32 instances were provided by contest organizers for algorithm evaluation. Final ranking score was based on solution values of the last 10 instances. In what follows, we use only these instances for illustrating proposed method performance. Instances differ in region number and type (static or PR), FPGA size, objective, etc. Two FPGA types are used: Xilinx Virtex xc7vx485 for instances 10023-10025 and Xilinx Zynq xc7z100 for the other ones. Table I provides an overview of instance characteristics. The number of static ("st") and PR regions ("pr") to floorplan are given in multi-column "#region". The following 2 multi-columns show the number of resources used by instance regions ("#block") and respectively the number of available FPGA resources ("FPGA #block"). In multi-column "avail./used ratio" is depicted the ratio between the number of available FPGA resources and resource count used by regions. Last column describes optimization objective of the instance:"w" – wirelength and "wa" – linear combination between wirelength and area.

B. Joint feasibility and score objective

In this experiment we have tested the execution performance of the proposed method. The objectives are: (i) minimization of regions intersection (i.e. solution feasibility) and (ii) problem maximization objective. These objectives are linearly combined to obtain a joint objective.

Table II illustrates the results obtained by the heuristic method with joint objective. The simulated annealing algorithm is executed 30 times with different random seed values. The number of obtained feasible solutions is shown in column "#feas" and the average execution time (in seconds) in last column. The average, the standard deviation and the maximum (i.e. best) of solution values (the $Score$ function as described in Section I-B) are depicted in multi-column "solution value". Only feasible solutions were used in result aggregation.

For instances 10026–10032 the multi-start heuristic obtains an average solution value which is less than 3% from the best found one, whilst for instances 10023, 10025 this gap is approximatively 11%. Unfortunately, no solution have been found for instance 10024. Heuristic execution times depend mainly on input instance size, in particular on the number of regions to place and the size of FPGA.

The best found solution for instance 10025 is shown in Figure 1 for illustration purposes. We can observe that the found floorplan is quite "compact" and that communicating regions are close to each other.

IV. Conclusion

In this paper, we have described the multi-start heuristic used by our team for the RAW Floorplanning Desgin Contest. Contest goal was to propose new algorithms for region floorplanning on a partially-reconfigurable FPGA target. Besides usual region non-overlap constraints, we had to deal with

TABLE I
FLOOR-PLANNING INSTANCE CHARACTERISTICS.

Instance id	#region		#block			FPGA #block			avail./used ratio			Objective
	st	pr	CLB	BRAM	DSP	CLB	BRAM	DSP	CLB	BRAM	DSP	
10023	25	0	7180	744	840	7590	1030	1400	1.06	1.38	1.67	w
10024	89	0	6596	352	792				1.15	2.93	1.77	w
10025	19	0	7000	450	90				1.08	2.29	15.56	w
10026	40	0	5186	0	0	6920	750	1010	1.33	-	-	w
10027	0	39	4699	0	0				1.47	-	-	wa
10028	55	0	5199	504	838				1.33	1.49	1.21	w
10029	0	53	4616	595	751				1.50	1.26	1.34	w
10030	0	54	4509	554	646				1.53	1.35	1.56	wa
10031	60	30	6920	750	1010				1.47	1.39	1.36	wa
10032	110	0	5896	651	858				1.17	1.15	1.18	w

TABLE II
HEURISTIC PERFORMANCE WITH JOINT OBJECTIVE.

Inst. id	#feas	solution value			avg. exec time (s)
		avg.	std.	max.	
10023	14	6271085.6	510997.6	7061014.0	218.1
10024	0	-	-	-	-
10025	28	3538860.6	319527.1	3991812.0	189.2
10026	30	918579.3	3774.4	923682.0	506.3
10027	30	1840640.2	14775.0	1871868.0	539.2
10028	30	1766438.0	11119.9	1786616.0	855.1
10029	29	3328547.6	45180.9	3405432.0	795.8
10030	30	6180159.5	85230.8	6326632.0	717.7
10031	16	9190050.0	48703.5	9249238.0	1485.1
10032	22	12026599.5	98128.5	12207174.0	2684.0

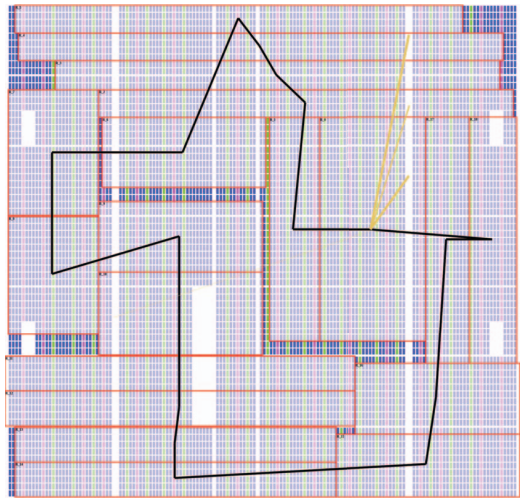

Fig. 1. Illustration of the best found solution for instance 10025.

specific constraints for the partially-reconfigurable regions (e.g. tile non overlap).

Our multi-start heuristic is based on several executions of a simulated annealing algorithm. Decision variables for the SA algorithm are region coordinates. In order to (supposedly) ease feasible solution search we have chosen to include problem constraints into an infeasibility cost to minimize. A possible improvement in the proposed method will be a more fine-grained selection of simulated annealing parameters.

In the majority of cases each execution of the simulated annealing algorithm found a feasible solution. Obtained solution values are rather close to each other. The main difficulty we met was finding feasible solutions for some instances. For example, we could not found a solution for instance 10024. We suppose that instances (e.g. 10023-10025) which contain regions of similar sizes, besides tighter resource constraints, are harder to solve because their feasible solution space is smaller. In a future work, we envisage to apply FPGA and/or region pre-partitioning methods in order to ease the search for feasible solutions for such kinds of instances.

REFERENCES

[1] M. Rabozzi and M. D. Santambrogio, "RAW Floorplanning Design Contest," `http://raw-floorplanning-contest.necst.it/`.

[2] L. Cheng and M. D. Wong, "Floorplan design for multimillion gate FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2795–2805, 2006.

[3] A. Montone, M. D. Santambrogio, and D. Sciuto, "Wirelength driven floorplacement for FPGA-based partial reconfigurable systems," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–8.

[4] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 11, no. 6, pp. 1120–1135, 2003.

[5] C. Bolchini, A. Miele, and C. Sandionigi, "Automated resource-aware floorplanning of reconfigurable areas in partially-reconfigurable FPGA systems," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. IEEE, 2011, pp. 532–538.

[6] K. Vipin and S. A. Fahmy, "Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration," in *International Symposium on Applied Reconfigurable Computing*. Springer, 2012, pp. 13–25.

[7] M. Rabozzi, J. Lillis, and M. D. Santambrogio, "Floorplanning for partially-reconfigurable FPGA systems via mixed-integer linear programming," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*. IEEE, 2014, pp. 186–193.

[8] T. D. Nguyen and A. Kumar, "Prfloor: An automatic floorplanner for partially reconfigurable fpga systems," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 149–158.

[9] M. Rabozzi, G. C. Durelli, A. Miele, J. Lillis, and M. D. Santambrogio, "Floorplanning automation for partial-reconfigurable FPGAs via feasible placements generation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 151–164, 2017.

[10] A. E. Smith and D. W. Coit, "Penalty functions," in *Handbook of Evolutionary Computation*. Oxford University Press and Institute of Physics Publishing, 1996, ch. C 5.2.

[11] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.