Optimal and Greedy Heuristic Approaches for Scheduling and Mapping of Hardware Tasks to Reconfigurable Computing Devices

Zakarya Guettatfi[1,2(✉)], Paul Kaufmann[3], and Marco Platzner[1]

[1] Paderborn University, Paderborn, Germany
zakarya@mail.uni-paderborn.de, platzner@upb.de
[2] Center for Development of Advanced Technology, Algiers, Algeria
zguettatfi@cdta.dz
[3] University of Mainz, Mainz, Germany
paul.kaufmann@uni-mainz.de

Abstract. Executing real-time tasks on dynamically reconfigurable FPGAs requires us to solve the challenges of scheduling and placement. In the past, many approaches have been presented to address these challenges. Still, most of them rely on idealized assumptions about the reconfigurability of FPGAs and the capabilities of commercial tool flows. In our work, we aim at solving these problems leveraging a practically useful 2D slot-based FPGA area model. We present optimal approaches for reconfigurable slot creation, hardware task assignment, and placement creation. We quantitatively compare optimal and heuristics algorithms through simulation experiments and show that the heuristics are rather close to the optimal techniques in terms of solution quality, in particular for reconfigurable slot creation and hardware task assignment. Further, we also derive an indication for the amount of fragmentation of the FPGA surface that is inherent to our 2D area model.

1 Introduction

The FPGA utilization can be maximized if the hardware tasks can be arranged such that there is no simultaneous temporal and geometrical overlap between them. The resulting scheduling and floorplanning problems for two-dimensional resources are NP-hard [7,11]. There is substantial earlier work that deals with the interdependent problems of task scheduling and placement on FPGAs. These works differ in the characteristics of the task sets, i.e., whether tasks have deadlines or not, the optimization goals, whether they deal with off-line or on-line problems, and, most importantly, the area model for the FPGA surface. Many of the presented techniques use an area model with free placement where tasks can be placed rather flexibly on the FPGA fabric. While this model has received a lot of attention in the past, aspects such as reconfiguration schemes of commercial FPGAs, capabilities of commercial tool flows, and the infrastructure needed to

F. Rincón et al. (Eds.): ARC 2020, LNCS 12083, pp. 108–117, 2020.
https://doi.org/10.1007/978-3-030-44534-8_9

connect hardware tasks to the CPU, memory, and I/O, constitute major hurdles for practical realization. In VLSI design, there are works on optimal bin packing and metaheuristics for floorplanning. Still, only a few of these consider problem characteristics important for our work, such as preplaced modules [7,11] and "soft modules" that are specified only by their area instead of a geometric layout.

In previous work [4], we have introduced a tool flow for task scheduling and floorplanning based on a special 2D slot-based reconfiguration model, where reconfigurable slots comprise several micro slots. Micro slots constitute rectangular reconfigurable regions with a complete set of resource types and their creation and partial reconfiguration is supported by commercial tool flows. Under this area model, we have proposed heuristics for scheduling and placement.

In this paper, we present novel *optimal slot creation and task assignment as well as layout generation methods* for the 2D slot model. Based on the insight that slot-based reconfigurable task scheduling becomes a problem similar to mono and multi-processor scheduling, and that slot placement becomes very similar to the VLSI's floorplanning problem, we adopt corresponding approaches and develop optimal slot creation and task assignment (SCTA) as well as layout generation (LG) algorithms. We compare these optimal techniques with our previous heuristics.

The remainder of the paper is organized as follows: In Sect. 2, we discuss related work about task scheduling and placement on FPGAs. In Sect. 3 we summarize our previous special 2D slot-based reconfiguration model and the proposed heuristics for slot creation and task assignment as well as for layout generation. The mathematical modeling for the novel optimal and heuristic approaches is then presented in Sect. 4. In Sect. 5 experimental results are detailed. Finally, Sect. 6 is devoted to the conclusions.

2 Related Work

The majority of related projects described hardware tasks as rectangular-shaped regions of reconfigurable logic and used an area model that can be categorized into 1D vs. 2D and free placement vs. slot-based placement. In the 1D area model, tasks can be allocated along one dimension only. This approach simplifies the placement problem and matches the partial reconfiguration abilities of earlier Xilinx devices. In the 2D area model, the rectangular tasks have to be allocated on the rectangular-shaped device. A free placement would allow allocating such a task on any feasible position on the device. In contrast, the slot-based model foresees a pre-partitioning of the device into rectangular regions that can accommodate tasks. The slot-based models allow for an easier, practical realization. In the following, we present selected related work in some of the areas related to task scheduling and placement.

In [8], the authors formulate an online real-time scheduling problem and present two heuristics for placing aperiodic hardware tasks. These heuristics are denoted as the horizon and stuffing techniques, considering both 1D and 2D area

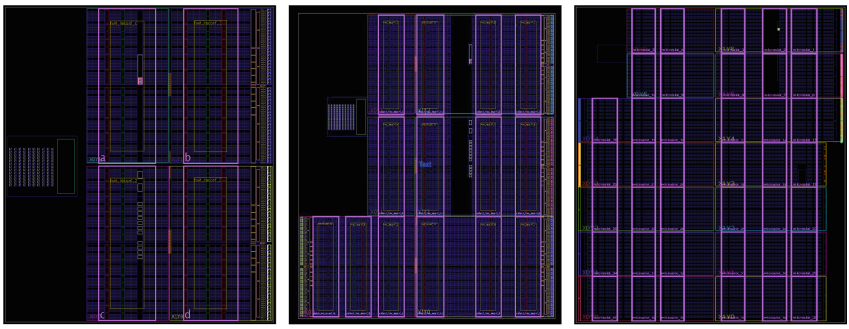
models Improved placement strategies that lead to reduced fragmentation and lower total execution times were presented in, e.g., [12]. Along the same line, [2] showed a level look-ahead approach with a non-preemptive EDF that delays the allocation of hardware tasks to reduce the fragmentation. In [3], the authors investigated two preemptive scheduling algorithms for periodic real-time tasks: EDF Next Fit (EDF-NF) and Merge-Server Distributed Load (MSDL). In [5] proposed the Finishing-Aware EDF (FAEDF) algorithm, which is EDF augmented with a look-ahead capability to locate future releases of adjacent areas. Another line of research dealt with the task placement or area allocation problem, respectively, and focused on online placement of relocatable, rectangular-shaped tasks that can be placed anywhere on the 2D surface of an FPGA device. The first work establishing the problem of online placement in the 2D area model was [1], where the authors proposed a fast online placement algorithm based on handling empty spaces. Later, [10] presented another placement method that relies on a partitioning of the reconfigurable resource and uses a hash matrix data structure to maintain the free space. While the works mentioned above consider homogeneous FPGA architectures, there are also algorithms for 2D placement on heterogeneous devices, e.g., [6]. The challenge of hardware task placement can also be seen from the perspective of d-dimensional orthogonal bin packing [7] and floorplanning, which is the first stage in physical VLSI design in the Electronic Design Automation (EDA) [11].

3 The Area Model

This section describes the area model used by the slot creation and task assignment (SCTA) as well as layout generation (LG) algorithms. The area model and the heuristics are results of our previous work, presented in [4], but require a summary for the introduction and comparison to optimal SCTA and LG algorithms.

The area model of this paper bases on two ideas: First, an FPGA is subdivided into rectangular reconfigurable regions, so-called **micro-slots**. The partition is done in such a way that all micro-slots have the same type and amount of resources. This simplifies the mapping of hardware tasks to hardware. Additionally, the borders of the micro-slots follow the boundaries of the partially reconfigurable frames imposed by a vendor's partial reconfiguration tool flow. This makes the area model practically useful. Second, a task executed on an FPGA is mapped to one or multiple micro-slots. Micro-slots allocated to serve a task are forming a so-called **slot**, which has to be a consolidated and a rectangular region on an FPGA. The size of a micro-slot is selected sufficiently large to be able to serve a small task. At the same time, the micro-slot size is chosen as small as possible to minimize fragmentation. Figure 1 shows for the Xilinx Zynq 7020, 7030, and 7045 devices the subdivision of their area. Each micro-slot contains 600 slices, providing in total 2400 LUTs, 4800 registers, 180 kB RAM, and 20 DSP blocks.

Once the area model is defined, the algorithmic challenges of hardware task scheduling and slot allocation can be presented in detail. Assuming a list of



(a) **(b)** **(c)**

Fig. 1. Partitioning into micro slots: The Xilinx Zynq 7010 (a), the Xilinx Zynq 7020 (b) and the Xilinx Zynq 7045 (c) devices.

periodic and independent hardware tasks is given as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where for each task τ_i the amount of required micro slots k_i, execution time c_i, and period p_i are given as $\tau_i = (k_i, c_i, p_i)$. To justify hardware task reconfiguration, the total amount of required resources $\sum_i k_i$ should be larger than the number of micro slots available on the FPGA. The first question is: Given that every hardware task can be hypothetically instantiated and released at any point in time, is there a schedule guaranteeing all tasks meeting their deadlines and, at the same time, respecting the upper bound of available micro slots? The second question is: Given a valid schedule of hardware tasks, can the hardware tasks be mapped to slots, i.e., non-overlapping rectangular regions of micro slots?

The procedure of hardware task assignment and layout generation starts with a set of periodic real-time tasks that have been synthesized to an FPGA device family such as the Xilinx Zynq. The procedure runs in two phases, with the first one creating the reconfigurable slots in a way that each task with all its instances is accommodated in exactly one such slot and all tasks assigned to one slot are schedulable. The result is a list of reconfigurable slots, characterized only by their sizes. In a second phase, a feasible layout is generated for a given FPGA device, i.e., a layout that provides slots with widths and heights. Heuristic algorithms for both phases are presented in the following sections.

4 Optimal Techniques for Slot and Layout Creation

In this section, we detail the mathematical modeling of the two problems (i) reconfigurable slot creation and task assignment and (ii) layout generation in the form of Quadratic Constraint Programs (QCP). The QCPs can then be solved to optimality.

4.1 Optimal Slot Creation and Task Assignment Approach

We start the formalization of the QCP with $n \cdot m$ binary decision variables x_{ij} that indicate, whether the i'th task is mapped into the j'th slot S_j with n as the number of tasks and m as the number of slots:

$$x_{ij} = \begin{cases} 1 & \text{if } \tau_i \in S_j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

As a first constraint, we have to enforce that a task is mapped to exactly one slot:

$$\forall i \in \{1, ..., m\} : \qquad \sum_{j=1}^{m} x_{ij} = 1. \tag{2}$$

For the next modeling steps, we need an upper bound for the number of required slots. We can determine such a bound based on the time utilization factors of the tasks, defined as defined as $u_i = \frac{c_i}{p_i}$, in the following way:

$$m = \left\lceil \sum_{i}^{n} u_i \right\rceil + 1. \tag{3}$$

Let A_j be the area of the largest task assigned to the slot S_j. Then the following constraints on the areas of the reconfigurable slots must hold:

$$\forall j \in \{1, ..., m\}, \quad \forall i \in \{1, ..., n\} : \qquad A_j \geq x_{i,j} \cdot k_i. \tag{4}$$

The objective is to minimize the total area required to map all reconfigurable slots. Therefore, the cost function of our QCP accumulates the total slot area, and the objective is to minimize this expression:

$$\min \sum_{j=1}^{m} A_j. \tag{5}$$

4.2 Optimal Layout Generation Approach

The intuitive challenge of packing boxes into a container is computationally surprisingly complex. The search space becomes even larger when allowing boxes to be "soft", i.e., be configured only by the area and an interval for the aspect ratio. While in Sect. 2, we have given an overview of related work on two-dimensional box packing, our method presented here is inspired by an approach developed for arranging modules on a chip die [9]. There, the authors have formalized the task of placing blocks within a rectangular chip area as a mixed ILP [9]. We have adopted this model with a few modifications. In particular, we have introduced soft blocks and preplaced blocks to the model. Soft blocks are only specified by their area requirement, which is exactly what we find when trying to place the reconfigurable slots. Preplaced blocks are important, since we can use them

to mask FPGA regions that do not contain any micro slots, e.g., regions that contain the processing system of an FPGA. The resulting model can, again, be cast as a QCP in the following way:

The geometrical position of a slot S_i is specified by its lower-left corner (x_i, y_i) and the width and height (w_i, h_i). Slots may not overlap and may not be placed outside the chip area with width W and height H. The first constraint can be enforced by introducing two binary variables p_{ij} and q_{ij} for each slot, ensuring that exactly one of the following inequalities is sharp, i.e., holds:

$$\begin{aligned} &x_i + w_i \leq x_j + W(p_{ij} + q_{ij}), && \text{slot i to the left of slot j} \\ &x_i - w_j \geq x_j - W(1 - p_{ij} + q_{ij}), && \text{slot i to the right of slot j} \\ &y_i + h_i \leq y_j + H(1 + p_{ij} - q_{ij}), && \text{slot i below slot j} \\ &y_i - h_j \geq x_j - H(2 - p_{ij} - q_{ij}), && \text{slot i above slot j} \end{aligned} \tag{6}$$

Placing boxes outside the area of an FPGA is avoided by:

$$\begin{aligned} x_i + w_i \leq W, \\ y_i + h_i \leq H. \end{aligned} \tag{7}$$

The objective of the QCP is to minimize the area xy of the rectangle enclosing all slots. However, to avoid a quadratic objective function, we minimize the rectangle's perimeter $2x + 2y$, which implicitly minimizes the area. The previously defined constraints are therefore sharpened to:

$$\begin{aligned} x_i + w_i \leq x, \\ y_i + h_i \leq y. \end{aligned} \tag{8}$$

For slots specified only by their area A_i, a quadratic constraint ensures that the width w_i and height h_i of a slot are sufficiently large:

$$w_i \cdot h_i \geq A_i \qquad \forall i \in \{1, ..., n\} \tag{9}$$

5 Evaluation

When comparing the optimal approaches presented in the previous section with the heuristics developped in [4], two main questions arise: By what margin are the heuristics behind the optimal approaches, and what are the computation times of the optimal algorithms? To answer these questions, we have designed two experiments. In the first experiment, the slot and layout creation algorithms are compared using task sets with increasing computational and resource requirements regarding generated slot and floorplan sizes as well as the computational times. In the second experiment, the algorithms are tested on how many of the task sets with FPGA time area product utilization factors between 0.1 and 1.0 can be successfully placed.

All optimal algorithms and the heuristic layout generator have been developed in C++ and use the Gurobi solver v.8.1.1. The slot generation and task assignment heuristic has been developed in Python.

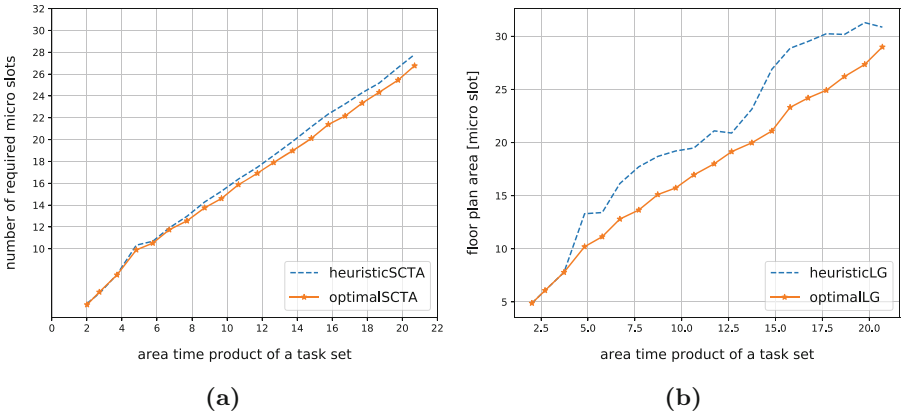


Fig. 2. Simulation results: (a) required number of micro slots and (b) the area of the floorplan depending on the application load of task sets given by the optimal and heuristic solutions. Each line point is an overage over 50 task sets.

5.1 Comparing Slot Set and Layout Sizes

To test the algorithms, we have generated 1000 task sets with the cumulated task set computation times in the range of 1 to 30 time units, resource requirements in the range of 1 to 6 micro slots, and time utilization factors in the range 0.10 to 0.50. Figure 2a shows for the optimal and heuristic SCTA algorithms the number of computed micro slots depending on the application load of a task set. The application load of a task set Γ is defined as $\sum_{\tau_i \in \Gamma} \frac{c_i}{p_i} \cdot k_i$, which represents the area-time product consumed by all tasks. The first observation is that the heuristic approach is very close to the optimal algorithm on average. Only for task sets with an accumulated application load beyond 7, a small difference starts to appear.

The computation times of optimal and heuristic SCTA algorithms differ, however, significantly. While the heuristic finishes within a few milliseconds, the QCP solver of the optimal approach often needs days on a large multi-core machine with a lot of main memory to be able to compute a result. This may be acceptable if the slot configuration is calculated once, at the design time of a system. Increasing the sizes of task sets further above 30 time units would render the optimal approach rather impractical.

The discrepancies between the optimal and heuristic LG algorithms are more prominent in Fig. 2b. Starting with an application load of 7, slots can be placed more compact by the optimal algorithm. The difference grows up to 5 slots for a time area utilization factor of 15 to 17.5. The results indicate that the naïve greedy (construction) heuristic approach we borrowed from strip-based bin packing has a lot of potential for improvement. We envision replacing the algorithm by a more computationally complex improvement heuristic, such as Simulated Annealing and Genetic Algorithms, to achieve better asymptotical results.

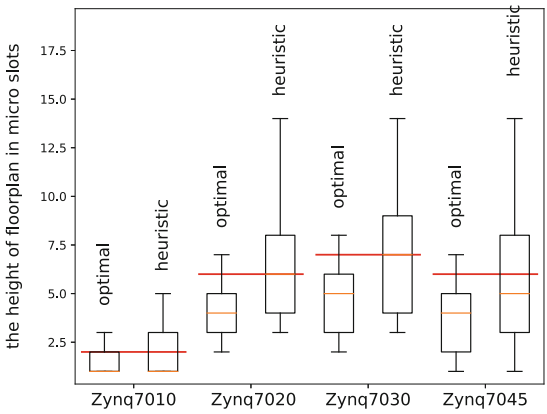


Fig. 3. Simulation results: Heights of the layout (floor plan) for different Zynq devices computed by optimal and heuristic approaches.

The computation time difference is significant but not as dramatic as for the SCTA algorithms. The heuristic LG approach can compute results within a few milliseconds, while the optimal algorithm takes up to a quarter of an hour on a large multi-core machine for a single slot set.

5.2 The Maximum Utilization Experiments

The goal of this experiment is to figure out to what extent the area utilization factor of an FPGA can be increased before the LG algorithms stop producing valid layouts. We would also like to understand better how pronounced the gap between the LG algorithms shown in Fig. 2b actually is. To this end, we have created slot sets with area utilization factors ranging from 0.1 to 1.0 with a step size of 0.1. Each slot set contains a series of slot areas randomly generated with respect to the area utilization. As target FPGAs, we have selected the Xilinx Zynq 7010, 7020, 7030, and 7045 devices. By fixing the FPGA's widths, we let the LG algorithms minimize the height of the surrounding box around the placed slots. Figure 3 shows boxplots for the achieved layout heights. The red lines in the figure indicate the maximal capacity, i.e., the height of the corresponding Zynq device. Hence, slot sets with a layout height below the red line can be executed on the according FPGA.

The first observation is that the optimal LG approach is very successful in mapping 75%, 87.5%, 87.5%, and 87.5% of the task sets to the target FPGAs. This indicates that the fragmentation inherent to the area model, which we accept in favor of efficient algorithmic task scheduling and slot mapping, lies around 25% for smaller task sets and reduces when FPGAs became larger. The heuristic LG lags behind, as already seen in the previous experiment. Only 50% to 62.5% of the task sets can be mapped. Compared to the optimal LG approach, 25%, 37.5%, 37.5%, and 25% fewer tasks can be mapped to the chosen FPGAs.

6 Conclusion

In this paper, we have presented the mathematical modeling of reconfigurable slot creation, including task assignment, and layout generation as QCPs. This allows us to solve these problems to optimality, where in previous work, we had developed and presented heuristics. We then have quantitatively compared the optimal approaches with the heuristics using a library of different randomly generated real-time task sets. We could show that the heuristics generally compute very good results. The results produced by the heuristic for reconfigurable slot creation, including task assignment, are very close to optimal; the results for the heuristic layout creation lag a bit behind. A further result is that the slot-based area model chosen in our approach adds roughly 25% fragmentation, which, however, reduces with larger task sets. The drawback of the optimal slot creation, task assignment, and layout generation approaches is their exhaustive computation times. The heuristic methods compute their outputs within a few milliseconds. While the optimal approaches are highly useful to evaluate the performance of our heuristics, applying them for larger tasks set is impractical. In future work, we aim at improving the heuristic for layout generation by taking inspiration from known floorplanning techniques in the electronic design automation domain.

Acknowledgment. This work has been partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre 901 "On-The-Fly Computing" under the project number 160364472.

References

1. Bazargan, K., Kastner, R., Sarrafzadeh, M.: Fast template placement for reconfigurable computing systems. IEEE Des. Test Comput. **17**(1), 68–83 (2000)
2. Cui, J., Gu, Z., Liu, W., Deng, Q.: An efficient algorithm for online soft real-time task placement on reconfigurable hardware devices. In: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2007, pp. 321–328. IEEE (2007)
3. Danne, K., Platzner, M.: Periodic real-time scheduling for FPGA computers. In: Third International Workshop on Intelligent Solutions in Embedded Systems, pp. 117–127. IEEE (2005)
4. Guettatfi, Z., Platzner, M., Kermia, O., Khouas, A.: An approach for mapping periodic real-time tasks to reconfigurable hardware. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 99–106, May 2019
5. Iturbe, X., Benkrid, K., Hong, C., Ebrahim, A., Arslan, T., Martinez, I.: Run-time scheduling, allocation, and execution of real-time hardware tasks onto Xilinx FPGAs subject to fault occurrence. Int. J. Reconfigurable Comput. **2013** (2013). 32 pages
6. Koester, M., Porrmann, M., Kalte, H.: Task placement for heterogeneous reconfigurable architectures. In: International Conference on Field-Programmable Technology, pp. 43–50. IEEE (2005)
7. Scheithauer, G.: Introduction to Cutting and Packing Optimization. ISORMS, vol. 263. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-64403-5

8. Steiger, C., Walder, H., Platzner, M.: Heuristics for online scheduling real-time tasks to partially reconfigurable devices. In: Y. K. Cheung, P., Constantinides, G.A. (eds.) FPL 2003. LNCS, vol. 2778, pp. 575–584. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45234-8_56
9. Sutanthavibul, S., Shragowitz, E., Rosen, J.B.: An analytical approach to floorplan design and optimization. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **10**(6), 761–769 (1991)
10. Walder, H., Steiger, C., Platzner, M.: Fast online task placement on FPGAs: free space partitioning and 2D-hashing. In: Proceedings International Parallel and Distributed Processing Symposium, IEEE (2003). 8 pp.
11. Wang, L.T., Chang, Y.W., Cheng, K.T.T.: Electronic Design Automation: Synthesis, Verification, and Test. Morgan Kaufmann Publishers Inc., San Francisco (2009)
12. Zhou, X.G., Wang, Y., Huang, X.Z., Peng, C.L.: On-line scheduling of real-time tasks for reconfigurable computing system. In: IEEE International Conference on Field Programmable Technology, FPT 2006, pp. 57–64. IEEE (2006)