Efficient Heterogeneous Architecture Floorplan Optimization using Analytical Methods

Asma Kahoul, Alastair M. Smith,
George A.Constantinides, and Peter Y.K. Cheung
Imperial College London

This paper argues the case for the use of analytical models in FPGA architecture exploration. We show that the problem, when simplified, is amenable to formal optimization techniques such as integer linear programming. However, the simplification process may lead to inaccurate models. To test the overall methodology, we feed the resulting architectures to VPR 5.0 and quantify their performance in comparison with traditional design methodologies. Our results show that the resulting architectures are better than those found using parameter sweep techniques. In addition, we show that these architectures can be further improved by combining the accuracy of VPR 5.0 with the efficiency of analytical techniques. This is achieved using a closed loop framework which iteratively refines the analytical model using the place and route outputs from VPR.

Categories and Subject Descriptors: D.2.7 [**Architecture Exploration**]: Reconfigurable Computing—*documentation*; H.4.0 [**Analytical Techniques**]: Optimization; I.7.2 [**Analytical Techniques vs. Empirical Tools**]: Architecture Exploration—*Analytical Model*

General Terms: ILP, VPR

Additional Key Words and Phrases: Architecture Exploration, Reconfigurable Computing

1. INTRODUCTION

The advances in field programmable gate arrays (FPGAs) over the past decade have made it possible to place significantly large circuits on a single FPGA chip [Compton and Hauck 2002]. The need for estimation and optimization techniques has therefore become crucial to divide, conquer and explore the large design space of potential architectures. While there exists a significant amount of work on homogeneous architecture design [Emmert and Bhatia 1999], [Hutton 2006], there is currently limited research on heterogeneous FPGA architectures consisting of a mix of coarse and fine grain components [Feng and Mehta 2006], [Smith et al. 2008].

Department of Electrical & Electronic Engineering,
Imperial College London, Exhibition Road,
London SW7 2BT, United Kingdom.

In this work we focus on the issue of device floorplanning. Floorplanning of heterogeneous reconfigurable devices is fundamentally different to classical floorplanning due to a number of reasons: First, the device may be reused for a number of purposes, meaning that the device layout will change the layout of a design mapped to it and that the architecture floorplan can have a significant effect on device performance. Second, the introduction of heterogeneous resources places restrictions on where architecture blocks may be layed-out, and subsequently circuit blocks of designs mapped onto the reusable fabric. To evaluate architecture floorplans, a framework for representing architectures and mapping designs on to them is needed. The Versatile Place and Route tool (VPR) [Betz et al. 2008] provides such a framework.

The VPR tool uses detailed knowledge of the placement and routing information from mapping a design onto the architecture, as well as knowledge about architectural information such as routing capacitance and resistance to extract accurate timing and information. This accuracy comes at a cost: that of run-time. Moreover, the tool must be run for each architecture layout specification. This makes the evaluation of all possible layouts a time consuming task. An alternative is to sample the design space and select the architecture that best suits one, or a combination of metrics such as area, delay, and power [Hutton 2006]. As a consequence of sampling the design space, optimal architecture layouts maybe not be found.

This article addresses the drawbacks of such parameter sweep techniques by using analytical modeling. Our framework targets FPGA architectures consisting of different resource types such as CLBs, RAMs, and multipliers arranged in columns. Our aim is to simultaneously optimize the floorplan of heterogeneous FPGA architectures *i.e* the quantity and location of architecture resource components, while floorplanning circuits onto the optimized architecture. We show in this article that our framework explores the design space efficiently by modeling heterogeneous architectures using mathematical programming in the form of Integer Linear Programming (ILP).

An ILP model has been used in our previous work to achieve optimal solutions to problems such as architecture generation [Smith et al. 2008]. The model allows the elimination of the dependencies on heuristic algorithms and provides more efficient ways to explore the design space. While the results have shown provably optimal bounds on the relative computational speed for various benchmarks, their efficiency is limited by the assumptions made and the accuracy of the model itself. Hence, if the architecture models are poor compared with empirical flows such as VPR, then the results may be meaningless. Moreover, if the optimization process is not scalable in execution time, this approach loses its attractiveness as compared to a typical parameter sweep approach. This paper shows that the results of analytical formulations can be fed into VPR to verify the quality of the designs. We demonstrate that the resulting architectures are an improvement over using VPR and parameter sweep for a fixed time budget [Kahoul et al. 2009].

The resulting architectures from [Kahoul et al. 2009]are further improved using a closed loop framework in which we combine the accuracy of VPR to iteratively refine the simplified model. While this methodology maintains the simplicity of the ILP model, it improves the quality of the formulation by modifying the objective

function coefficients and therefore guiding the optimal search to find better architectures.

The main contributions of this paper can be summarized as follows:

(1) An analytical model which simultaneously floorplans the layout of the FPGA device and a circuit that will be mapped onto the device.
(2) An enhanced formulation of the heterogeneous FPGA layout problem, leveraging advances in facility layout models from the operational research community.
(3) The quantification of analytical modeling efficiency over a parameter sweep approach in the design of new FPGA architecture layouts.
(4) Closing the loop between the efficiency of analytical techniques and the accuracy of empirical tools to improve the quality of FPGA architectures found using the analytical model.

2. RELATED WORK

The development of advanced architectures and more efficient Computer Aided Design (CAD) mapping algorithms have improved speed, area and power consumption over the past decade [Compton and Hauck 2002]. The introduction of fixed-functionality hard blocks in heterogeneous FPGAs, for example, made it possible to execute their functions more efficiently. A recent study shows that coarse grain components can reduce the area gap between FPGAs and ASICs from 40 to 20× while improving the gap of speed and static power consumption [Kuon and Rose 2006]. The main disadvantage of heterogeneous devices is that these coarse grain components are beneficial only when they are used and are a waste in silicon area and routing otherwise [He and Rose 1993]. Consequently, the exploration of the mix of these coarse grain components and different architecture layouts has become an interesting research subject [Hutton 2006].

The design of modern FPGA architecture layouts in particular is a challenging task, as the locations of the different blocks can significantly affect device performance due to the delay, area and power consumption used in routing to and from such blocks. In this paper we define the FPGA architecture exploration problem as finding an optimal layout of the FPGA chip, and positioning circuit blocks while minimizing an objective function comprising selected performance metrics.

There are many algorithms available in the literature for floorplanning circuits onto Application-Specific Integrated Circuits (ASICs) [Tsay et al. 1991],[Murata et al. 1996]. While these algorithms can be applied to homogeneous FPGAs consisting of only configurable logic blocks (CLBs), they must be modified significantly to be used in the context of modern heterogeneous FPGAs. ASIC floorplanning algorithms have the flexibility to place circuit blocks anywhere within the chip area making no distinction between the device architecture and the circuit floorplan. Such distinction is necessary in our work since our aim is to floorplan both the FPGA silicon area and the benchmark circuit.

Recently, several algorithms have been developed for heterogeneous flooplanning for FPGAs [Singhal and Bozorgzadeh 2007],[Banerjee 2009]. While these contribu-

tion target the floorplanning problem for heterogeneous FPGAs they do not cover the floorplanning of the FPGA device itself.

In the absence of efficient floorplanning algorithms for heterogeneous architectures, methodologies based on parameter sweeping are currently the only automated methods to design new FPGA architecture layouts. In comparison with such approaches, analytical tools offer the ability to explore a much wider design space within any given computational time budget. The analytical tool of [Smith et al. 2008], for instance, has been used in the past to produce optimal architectures within the accuracy of the formulation. However, the model suffers from its exponential time complexity with respect to the number of circuit blocks. Enhanced formulations of the problem could result in reduced solution time and therefore be applied to larger circuits. As a result, we have taken advantage of the advances in the facility layout problem [Sherali et al. 2003] in the aim of efficiently formulating the heterogeneous FPGA layout problem analytically.

ILP-based architecture exploration models suffer from their dependence on assumptions and simplifications which cause uncertainty about the accuracy of the results. On the other hand, empirical tools such VPR 5.0 offer much higher accuracy of the FPGA architectures models. The accuracy of VPR in comparison with our model is due to the level of detail in describing the FPGA architecture. While our tool is a highly simplified model of the FPGA architecture based on the Manhattan distance for routing delay, VPR models more accurately architectural features such as the routing architecture: single or bi-directional driver routing, path delays extracted from transistor level simulations, block pin locations, and switch-box models. The price one pays for this level of detail, however, is the inability to use analytical techniques to optimize over the design space, hence the need for a comparison. In addition to this technical point, the choice of VPR was further motivated by its capability of targeting a broad range of FPGA architectures, and its widespread use in academic architecture research.

In this paper, we propose a framework that combines the efficiency of the analytical tools with the accuracy of VPR 5.0. We initially present comparative results showing the quality of architectures generated with our enhanced analytical model by feeding them to VPR 5.0 and comparing them with those found using parameter sweep techniques [Kahoul et al. 2009]. In addition, our experiments explore an important factor in heterogeneous architecture design which is the impact of architecture layout on performance.

This ILP model if extended in this paper to a closed loop framework which refines the quality of the ILP model and therefore the quality of the resulting architectures. This framework iteratively calibrates the ILP model to tune it with VPR's place and route algorithm and therefore guide the search for more efficient architectures.

In addition, this article we present additional analysis of the analytical framework: First, the impact of VPR place and route noise caused by the heuristic nature of the placement algorithm on our results is analyzed. Second, the scalability and runtime of the ILP framework with respect to problem size is presented.

3. PROBLEM DEFINITION

The aim of this paper is the design of efficient heterogeneous FPGA layouts consisting of resources grouped in columns. The design process involves finding an architecture layout that optimizes a set of metrics for a specific benchmark set. Most research in the literature uses floorplanning techniques only to map circuits onto the FPGA chip. Our objective is to use advances in these floorplanning techniques to develop an architecture exploration model capable of finding both optimal reconfigurable architecture floorplans and optimal circuit layouts on the generated architecture, as a combined one stage problem.

The generic VLSI floorplanning problem can be described as a special case of the well-studied operational research facility layout problem. Indeed, both problems aim at finding a non-overlapping planar orthogonal arrangement of rectangular blocks within a rectangular facility, such that the cost of interaction between blocks is minimized. Most of the progress achieved in this area, with the exception of few Mixed Integer Programming (MIP) models [Sherali et al. 2003], use improvement heuristics to find good layouts. MIP models usually capture all the constraints of the layout problem and achieve optimal solutions for relatively small problems. However, they suffer from scalability limitations due to the exponential increase in solution time with respect to the number of binary variables present in the formulation. We use, in our work the advances achieved in this field to improve the floorplanning formulation and reduce the solution time.

4. HETEROGENEOUS ARCHITECTURE EXPLORATION USING AN ENHANCED ILP FORMULATION

The primary aim of this paper is to design efficient heterogeneous FPGA architectures. To achieve this we use analytical tools in combination with accurate models such as VPR. In Section 4.1 we initially describe the ILP model and illustrate an efficient bounding procedure to improve the solution time. A generic floorplanning model using an enhanced formulation from the advances in the facility layout research field [Sherali et al. 2003] is also described. Based on this model, we have developed a model to describe the column-based nature of modern FPGAs in Section 4.2. The resulting architectures are used in Section 6 to illustrate the efficiency of analytical techniques in exploring the design space in comparison with a parameter sweep approach.

4.1 Generic Formulation of the Floorplanning Problem

This section describes the generic linear programming formulation of the layout floorplanning problem and provides the key notations used in this paper. We denote a set of n rectangular circuit blocks as B. The width and height of each block $i \in B$ are represented by w_i, h_i respectively. In contrast to the formulation in [Sherali et al. 2003] which is for variable outline problem, we use a fixed-die (fixed outline) formulation in which the FPGA chip is modeled as a fixed rectangular shape of width W and height H. The locations of the blocks are determined by their centroid locations (x_i, y_i) in a two dimensional coordinate system aligned with the chip height, width, and its origin located at the south west corner of the chip.

4.1.1 *Objective Function.* Area minimization has been the main objective in traditional floorplanners [Feng and Mehta 2006]. However, due to the significant impact of interconnect on circuit delay caused by the rapidly increasing number of transistors and their switching speed, it has become necessary to design interconnect-based tools. In this paper we optimize the critical path of a design where delays account for both logic delay according to component type, and routing delay, which is linearly proportional to the Manhattan distance, where Manhattan distance is the sum of the horizontal and vertical separation between signal source and sink. The Manhattan distances are calculated between centroid locations of circuit blocks.

Additional architectural features such as rotation and reflection can be added to our model. This however would add extra complexities as the size of the optimization problem grows not only with the size of the FPGA architecture to be designed, but also with the size of each benchmark upon which it is designed. Moreover, modern FPGA logic blocks contain carry chains, and are designed in such a way that consistent orientation of blocks across the device is somewhat necessary.

The model minimizes an objective function comprising the critical path of the circuit. This objective function is tuned in the next sections with VPR delay model.

The delay optimization problem can be stated as follows:

Minimize
Critical Path
subject to
Fit in the chip constraints
Non-Overlap constraints
Logic delay constraints
Routing delay constraints

4.1.2 *Fit in the Chip Constraints.* To ensure that circuit blocks are contained within the die area, the origin (south west corner of the chip), the chip width, and its height are used as lower and upper bounds to the location of the blocks centroids as shown in inequalities (1).

$$\forall i \in B \begin{cases} \frac{1}{2}w_i \leq x_i \leq W - \frac{1}{2}w_i \\ \\ \frac{1}{2}h_i \leq y_i \leq H - \frac{1}{2}h_i \end{cases} \tag{1}$$

4.1.3 *Non-Overlap Constraints.* In order to constrain the block placements and to prevent them from overlapping, a set of separation constraints are added. These constraints force the blocks to be separated either on the *x-axis* or the *y-axis* as shown in Fig. 1.

The non-overlapping constraints in either axes can be described using the following mathematical disjunction:

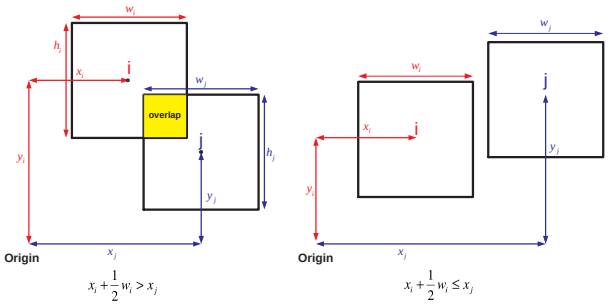


Fig. 1. Illustration of the separation constraints on the *x-axis*.

$$\left(x_i + \frac{1}{2}w_i \leq x_j - \frac{1}{2}w_j\right) \vee \left(y_i + \frac{1}{2}h_i \leq y_j - \frac{1}{2}h_j\right)$$
$$\vee \left(x_j + \frac{1}{2}w_j \leq x_i - \frac{1}{2}w_i\right) \vee \left(y_j + \frac{1}{2}h_j \leq y_i - \frac{1}{2}h_i\right)$$

This disjunction ensures separation by setting *at least* one of the inequalities to true. The difficulty in formulating these separation constraints in ILP is the result of introducing binary variables necessary to write the inequalities in a linear form. The most common approach to linearize a set of disjunctions is the so-called **Big-M** formulation illustrated in Equations (2).

$$Mz^x_{ij} + x_j - x_i - \frac{1}{2}w_i - \frac{1}{2}w_j \geq 0 \tag{2a}$$
$$Mz^x_{ji} + x_i - x_j - \frac{1}{2}w_j - \frac{1}{2}w_i \geq 0 \tag{2b}$$
$$Mz^y_{ij} + y_j - y_i - \frac{1}{2}h_i - \frac{1}{2}h_j \geq 0 \tag{2c}$$
$$Mz^y_{ji} + y_i - y_j - \frac{1}{2}h_j - \frac{1}{2}h_i \geq 0 \tag{2d}$$
$$z^x_{ij} + z^x_{ji} + z^y_{ij} + z^y_{ji} \leq 3 \tag{2e}$$
$$z^x_{ij}, z^y_{ij} \in \{1, 0\}, \forall i \neq j \tag{2f}$$

By forcing at least one of the binary variables to be zero using (2e) and (2f) we force the blocks to be separated in at least one direction. The **Big-M** formulation requires four binary variables for any pair of blocks which necessitates in total $4\binom{n}{2}$ variables. In the worst case, ILP run-time is exponentially dependent on the number of integer variables, which explains the limitation of this approach. Dropping the integrality constraints (LP relaxation) and solving the resulting LP problem is usually used to obtain global lower bounds on the optimal value of the problem [Balas 1998]. These are, in turn, used within a systematic solution technique such as the branch and bound scheme [Vecchietti et al. 2003]. However, this relaxation tends to produce trivial bounds causing this particular ILP to suffer from particularly

long computation times, which motivates a tighter formulation of the floorplanning problem as proposed in [Sherali et al. 2000]. This improved formulation produces tighter bounds by adding a set of valid inequalities and capturing the smallest set (convex hull) containing the feasible solutions of the disjunctions. Moreover, contrary to the big-M formulation, the convex hull formulation enforces separation in one direction by utilizing the c_{ij} variables which represent the separation distance. This allows minimum separation distances to be specified, providing better bounds as described in [Sherali et al. 2000].

Based on the model in [Sherali et al. 2000] we derive the corresponding convex hull representation for the architecture and circuit floorplanning problem using a set of continuous variables $\forall i < j : c^x_{ij}, c^y_{ij}, \Delta^x_{ij}, \Delta^y_{ij}$ as shown in Equations (3-9), and simplified from the original formulation in [Sherali et al. 2000]. The variables Δ^x_{ij} and Δ^y_{ij} are used to represent separation distance. Similarly, variables c^x_{ij}, c^y_{ij} also represent separation, but in conjunction with the binary variables z^x_{ij}, z^y_{ij} they are used to enforce a separation distance in at least one direction.

$$\frac{1}{2}(w_i + w_j)z^x_{ij} \leq c^x_{ij} \leq (W - \frac{1}{2}w_i - \frac{1}{2}w_j)z^x_{ij},\ \forall i \neq j \tag{3}$$

$$\frac{1}{2}(h_i + h_j)z^y_{ij} \leq c^y_{ij} \leq (H - \frac{1}{2}h_i - \frac{1}{2}h_j)z^y_{ij},\ \forall i \neq j \tag{4}$$

$$\begin{aligned} -(W - \frac{1}{2}w_i - \frac{1}{2}w_j)(1 - z^x_{ij} - z^x_{ji}) \\ \leq \Delta^x_{ij} \leq (W - \frac{1}{2}w_i - \frac{1}{2}w_j)(1 - z^x_{ij} - z^x_{ji}) \end{aligned} \tag{5}$$

$$\Delta^x_{ij} = x_j - x_i - c^x_{ij} + c^x_{ji} \tag{6}$$

$$\begin{aligned} -(H - \frac{1}{2}h_i - \frac{1}{2}h_j)(1 - z^y_{ij} - z^y_{ji}) \\ \leq \Delta^y_{ij} \leq (H - \frac{1}{2}h_i - \frac{1}{2}h_j)(1 - z^y_{ij} - z^y_{ji}) \end{aligned} \tag{7}$$

$$\Delta^y_{ij} = y_j - y_i - c^y_{ij} + c^y_{ji} \tag{8}$$

$$z^x_{ij} + z^x_{ji} + z^y_{ij} + z^y_{ji} = 1 \tag{9}$$

$$z^x_{ij}, z^y_{ij}, z^x_{ji}, z^y_{ji} \in \{1, 0\} \tag{10}$$

This convex hull representation is used to build an efficient heterogeneous architecture ILP model as shown in the following section.

4.2 Enhanced Heterogeneous FPGA Floorplanning Model Formulation

Floorplanning for column-restricted FPGAs requires the placement of circuit blocks of a particular resource type within the boundaries of the corresponding resource column. Constraints to map these nodes into their respective regions as well as setting the widths and locations of each column are added in this section. The convex hull relaxation based model discussed in the previous section is modified to include column restrictions. This will allow the exploration of different architecture floorplans of the FPGA chip. An example of the simplified FPGA architecture layout used in our formulation is shown in Fig. 2. The constraints required for the formulation are thus:

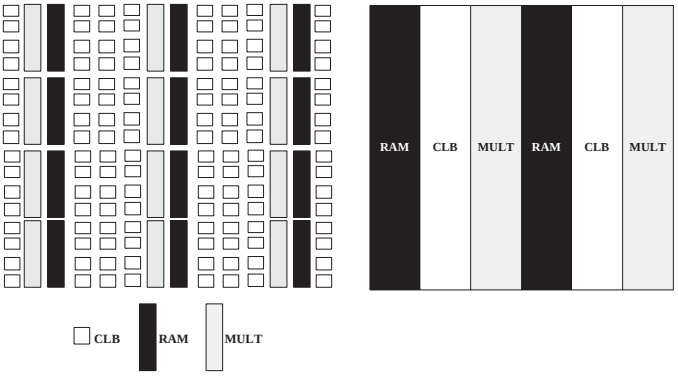


Fig. 2. A Simplified Heterogeneous Architecture Floorplan Consisting of Columns of CLBs, RAMs, and MULTs.

Minimize
Critical Path
subject to
Circuit Block Fit in the chip constraints
Resource Columns Fit in the chip constraints
Block Placement inside Resource Columns
Block Non-Overlap constraints
Resource Columns non-Overlap constraints
Logic delay constraints
Routing delay constraints

We introduce the following notations to model heterogenous FPGA architectures: In addition to their widths and heights, circuit blocks are constrained by their resource type denoted by $t_i \in T$ where $T = \{CLB, RAM, MULT\}$. We denote the set of resource columns available on the chip as R where each resource column $u \in R$ is a rectangular block of half-width w_u and half-height $h_u$which equals half the chip height, centroid locations (x_u, y_u), and resource type $t_u \in T$.

Our ILP model takes advantage of the similarities between circuit blocks and resource columns which are both rectangular blocks placed within the boundaries of the FPGA chip. Non-overlap constraints are applied between architecture blocks, in other words, columns of different resource types to obtain a consistent architecture floorplan. Non-overlap constraints are also applied between all circuit blocks in order to obtain a consistent placement of the circuit onto the architecture. Finally, non-overlap constraints are also applied between architecture blocks and circuit blocks of different resource types: this in effect maps circuit blocks onto the correct portion of the chip - for example multipliers in a circuit may not overlap with area reserved for RAM blocks.

Equations (3-9) are applied to all circuit block pairs $\forall i, j \in B$, and to all resource column pairs $\forall i, j \in R$. To formulate the separation between circuit blocks and resource columns of different resource types, we apply equations (3-9) to all pairs

that satisfy this condition: $\{\forall i, j$ where: $i \in B$ with resource type $t_i \in T$, $j \in R$ with resource type $t_j \in T$ and $t_i \neq t_j\}$. A circuit block i for instance, with resource type $t_i = CLB$ is allowed to overlap with any CLB column and is separated from the $MULTs$, $RAMs$ columns, and all other circuit blocks using the convex hull separation constraints.

Having successfully formulated the problem analytically we use this model in Section 6 to generate heterogeneous architectures and compare it with a parameter sweeping approach. This latter, is described in the following section.

5. ARCHITECTURE EXPLORATION USING A PARAMETER SWEEPING APPROACH

In a typical design framework, FPGA architectures are selected using an experimental methodology. This is conducted by mapping a set of benchmarks into potential architectures and comparing the results using selected performance metrics. In this work, we develop a parameter sweep methodology, which is used to compare to the ILP method of selecting architecture layouts. In order to do so, the space of potential architecture floorplans must be parameterized. Since VPR provides a framework to do this, and allows accurate determination of critical path delay of circuits mapped onto these devices, we use this as a starting point for our parameter sweep block. We have created a tool that is based on this methodology and which uses architecture parameter sweeping to generate a set of architectures and test them on VPR 5.0. This tool is used to vary the layout of the FPGA architecture by sweeping the positions and number of the resource columns within the chip area.

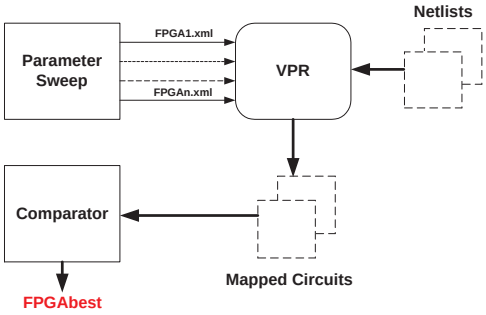


Fig. 3. Parameter-sweep approach.

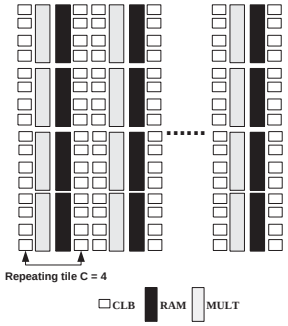


Fig. 4. An architecture constructed with a repeating tile of 4.

The parameter sweep framework consists of three main blocks and is interfaced with VPR 5.0 for placement and routing, as shown in Fig. 3.

Given a fixed chip area, the sweeping procedure targets the number r, and position p of each resource type that could be placed on the architecture. These parameters are varied using a structured approach in which the chip area is divided into subsets called repeating tiles. Each repeating tile comprises C resource columns as shown in Fig. 4. The parameter-sweep block is used to create architecture files for all possible combinations of resources that fit in these repeating tiles.

In other words, instead of exploring all possible architecture layouts given a set of resource columns and a fixed chip area, we fully explore the layout of a smaller portion of the chip and duplicate it along the chip area. This procedure allows the exploration of architectures with significantly different layouts within a fixed time budget, resulting in a structured sampling approach of the design space. This parameter sweep, is hence a structured, tile based approach used to limit the size of the search space and make the size parameterizable offering designer-controlled tradeoff between quality of results and execution time. In addition, the use of this repeating tiles fits well with the CAD flows and existing architectures.

The size of the repeating tiles is chosen based on the time frame for the architecture exploration procedure. Increasing the size of the repeating tile results in a larger number of possible permutations and therefore a larger set of explored architectures. For example, given the choice from the set of resources $\{CLB, MULT, RAM\}$ a repeating tile of size $C = 5$ results in 3^C combinations which is 243 potential architectures. The parameter sweep block output (set of combinations) are translated into VPR 5.0 architecture format, which in turn performs the placement and routing of test benchmarks on the sample architectures.

The comparator block collects the results of the placement and routing of the circuits on each architecture. The critical path is used as the comparison metric. Consequently, the architecture resulting in the lowest critical path is selected. The use of critical path as our performance metric provides information about the impact of architecture layout on circuit delays and inter-node delays. This is particulary important given the significant contribution of interconnect delays in the overall circuit delay. This information will be used in Section 7 in order to refine the ILP interconnect model.

This framework is used in the following section to compare the efficiency of the previously described ILP model in exploring the design space with a parameter sweep approach.

6. ILP-BASED ANALYTICAL APPROACH VS. PARAMETER-SWEEP APPROACH

6.1 Experimental setup

The main focus of this paper is to illustrate that combining analytical models with more accurate tools such as VPR, performs better than a typical parameter sweep approach. We have therefore conducted a comparative experiment on a set of test benchmarks as shown in Fig. 5. The time budget for the parameter sweep framework is tuned to match the time taken by the ILP model to obtain an optimal architecture for a fair comparison.

ASIC benchmarks were selected and modified to explore a more comprehensive design space. In particular, we have used MCNC netlists which do not by themselves have particular resource types that need to be mapped to certain locations, as would be the case in an FPGA flow. In order to use these benchmarks, we have therefore assigned a resource type to each block. This assignment was performed by randomly selecting a resource type from a distribution biased so that the ratio of the various resources types matches those ratios found in other heterogeneous logic studies [Smith et al. 2005].

This experimental approach does not only compare the efficiency of the two

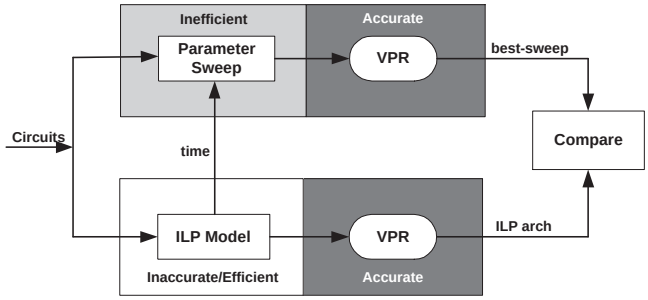


Fig. 5. Comparative experiment between ILP-based analytical model and parameter-sweep approach.

Benchmark	ILP runtime (sec)	C
poly-eval-7	12.34	3
xerox	6023.3	7
apte	210.8	5
hp	234.7	5
poly-eval-27	86400[1]	10
ami33	86400[1]	9

Table I. ILP solution time and the parameter sweep repeating tile.

frameworks for the same time budget, but also combines the advantages of analytical techniques and empirical models such as VPR. This is achieved by taking the results generated by the simplified ILP model and feeding it back to VPR for a more accurate architecture model.

The objective function of the ILP model is tuned accordingly with the routing model of VPR 5.0. This has been achieved using an experimental approach where a best fit model has been applied to the Manhattan distance between two circuit blocks and the corresponding routing delay. The coefficients of this best fit model are used to model the interconnect delay between connected blocks.

6.2 Parameter sweep vs. analytical framework results

The experiment described in Fig. 5 was conducted and the results are described in this section. For the ILP approach, optimal solutions were obtained for smaller benchmarks and the model has been left to run for 24 hours for larger benchmarks. The best known solution (upper bounds) are used for comparison. These ILP solutions were translated to the VPR 5.0 architecture format and used for the placement and routing of the test benchmarks.

Table 6.2 shows the ILP solution times and the size of the repeating tile used to generate the sample architectures for each test benchmark. The size of the repeating tile is chosen to generate the sample architectures where the time taken by the parameter sweep procedure to place and route these architectures matches the time taken by the ILP model to generate the architecture floorplan.

[1]ILP stopped at 24 hours.

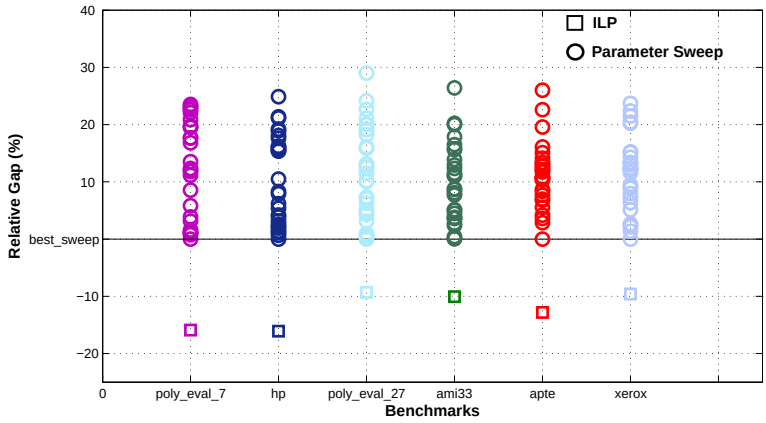


Fig. 6. Relative gap between parameter-sweeped architectures, ILP generated architecture, and the best parameter-sweeped architectures.

Fig. 6 shows the critical paths of all architectures explored relative to the best architecture generated with the parameter sweep framework and a subset of other architectures explored with the same framework. These gaps present an important aspect of heterogeneous FPGA design, which is the significant impact of architecture layout on performance. Changing the layout of the architecture can vary its performance by up to 40%.

Fig. 6 also shows the critical path of the optimal ILP generated architecture relative to the best parameter sweep architecture obtained within the same time frame. These results illustrate a significant improvement of up to 15% on the critical path using our analytical framework over architectures designed with the parameter sweep approach. This is mainly caused by limitations of the parameter sweep approach in exploring a large design space within a restricted time budget. These limitations are induced by the size of the repeating tiles which restricts the potential architecture layouts explored.

The efficiency of ILP model is further illustrated in Fig. 7 where we compare the architectures found by the ILP to those during the parameter sweep during the time-frame of exploration procedure. At each time interval the repeating tile size C is incremented accordingly to adjust it with the time budget. To further demonstrate the efficiency of the ILP, we have continued the parameter sweep method beyond the time required for the optimal ILP solution as shown in Fig. 7. It is interesting to see that throughout the architecture exploration, the ILP model produces better architecture layouts than the parameter sweep approach.

6.3 ILP and parameter sweep runtime scalability with problem size

In this section we illustrate the scalability of the ILP and the parameter sweep techniques with respect to the problem size. We have used a set of polynomial evaluator benchmarks with variable sizes to show how the ILP solution time scales

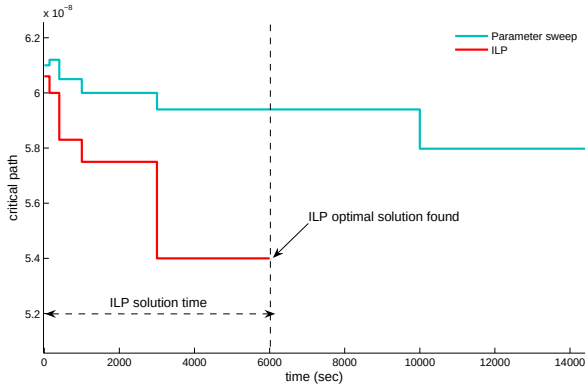


Fig. 7. ILP vs. parameter sweep over different intervals in the solution time.

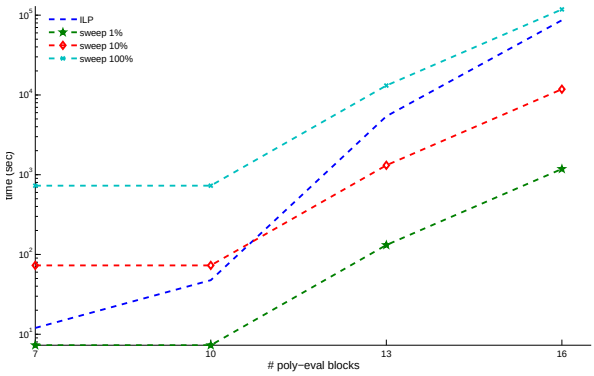


Fig. 8. ILP and parameter sweep scalability with problem size

in comparison with different coverage of the parameter sweep.

Fig. 8 also shows 1%, 10% and a full coverage, *i.e.* 100% of the design space by the parameter sweep procedure. In other words, our experiment examines the scalability of the parameter sweep when the problem and the underlying architecture size increases. The runtime of the parameter sweep methodology were estimated using the time taken by VPR place and route algorithm and the number of possible architectures. For an architecture consisting of 12 columns for example, the time taken by a full coverage parameter sweep is estimated by $t = T \times 3^{12}$ where T is the average place and route time for the specific benchmark.

The solution time is affected by both the benchmark size and the number of columns used in the architecture. The number of columns is selected to model the smallest architecture that fits the test benchmark. Consequently, the problem size increases with the number of blocks in the test circuit as shown in Fig. 8.

These results show that problem size has a significant effect on the solution time of both the ILP and the parameter sweep procedures. Our results, however, show

that the ILP model, even when not solved to optimality and when stopped after 24 hours, still finds better architecture than the parameter sweep.

Moreover, while a full coverage of the parameter sweep procedure guarantees the best architecture layout, it suffer from a significant increase in solution time with respect to the benchmark size and the number of architecture columns as shown in Fig. 8. This leads architects to use a smaller coverage making the parameter sweep methodology less efficient than our tool.

6.4 VPR placement and routing heuristic noise impact

VPR tool uses a simulated annealing algorithm [Betz et al. 2008] during the placement and routing of the benchmark circuit. This algorithm uses statistical information during this process to update a temperature parameter which decides on the algorithm's next move. Effectively, the annealing scheduler initially chooses a random placement of the circuit and then performs a set of moves to optimize the objective function. This heuristic nature of the VPR placer could have an impact on the results. We have therefore conducted a set of experiments to measure the impact of the heuristic placer on the results. This was achieved by varying the initial random seed used by the simulated annealing algorithm which determines the initial placement of the benchmark circuit. This resulted in $\pm 3\%$ average variation in the critical path due to changes in the placer seeds using the same architecture. This illustrates that the improvement obtained in the critical path is in effect the result of the ILP model.

6.5 Summary

The results presented in this section show that by simplifying the problem and applying formal optimization techniques in the form of ILP, better quality architectures are generated. In fact, while the ILP framework may not model heterogeneous architectures as accurately as VPR, it still is able to improve on the parameter sweep technique by exploring a wider range of designs.

The parameter sweep technique chosen in this paper was designed to be the most naive scalable strategy for a sound and reproducible comparison point. There are several ways to improve over an exhaustive search by modifying the parameter sweep technique. In the following section, we propose further improvement to the model by using a combined framework of analytical and empirical tools.

7. CLOSED LOOP MODEL

In the previous sections we have successfully shown the efficiency of optimization techniques in exploring the design space over a traditional parameter sweeping methodology. Our approach represented a fair comparison between the two techniques over the same time budget.

In this section we show that it is possible to further improve the quality of the architectures by combining the accuracy of empirical tools with the efficiency of analytical techniques. While VPR was used in the previous sections to verify the quality of the architectures, in this section we propose the use of VPR to further improve the quality of the architectures obtained by the ILP model. This can be achieved by modifying the ILP model and guiding the optimal search to find better architectures. This modification is motivated by considering the model of routing

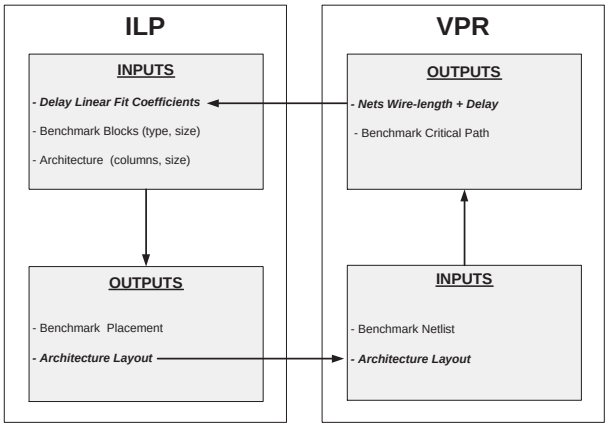


Fig. 9. Architecture Exploration Closed Loop Framework.

delay. In the cost function, routing delay between elements i and j is given by (11), where T_{ij} represents the delay between circuit elements, D_{ij} represents the Manhattan distance between circuit elements, and C and K represent constant coefficients that model routing delay linearly.

$$T_{ij} = C \times D_{ij} + K \tag{11}$$

In Section 4, the routing coefficients were evaluated through the use of VPR and an uncongested circuit. However, congestion is highly likely, since we are routing for minimum channel width. Moreover, it is likely that congestion is dependent on the device layout, as different blocks may connect to each other in different ways.

The VPR place and route model is based on the simulated annealing algorithm which optimizes a combination of the wire-lengths and the critical path of the corresponding circuit on the output architecture. VPR, being an accurate model also takes into consideration architecture parameters such as commonly occurring net-lengths and congestion. Thus to improve the routing delay model, delays from the circuit, which account for congestion and the VPR cost-function can be used.

We propose the closed loop framework illustrated in Fig. 9 which iteratively refines the routing model. Each time the ILP is solved, the resulting optimal architecture layout is fed to VPR. Routing delays are obtained from the placed and routed design, and the linear fit routing coefficients are re-evaluated based on the connections, the Manhattan distances between points and the experimental delay. Hence, the iterative model will account for the effects of congestion, leading to higher quality architecture layouts. Once the model converges, the final architecture layout is obtained.

8. CLOSED LOOP FRAMEWORK RESULTS

Using the framework shown in Fig. 9 we have performed the following experiments to verify the efficiency of this closed loop architecture exploration model:

8.1 ILP Model refinement

The refinement of the ILP cost function coefficients at different stages of the framework is analyzed in this section. These coefficients are obtained using a least-squares fit of wire-lengths and the corresponding delays for a specific benchmark and the architecture at the corresponding iteration. Fig. 10 shows the best fit model in the initial, second and last iterations of the closed loop framework. At each of these iterations we observe the changes in the best fit model which is induced by the new architecture and benchmark floorplans. The improvement in the architecture floorplan is explained by the increased clustering in each iteration. This is further illustrated by the increasing correlation factor denoted in each graph. The correlation factors determine the quality of the best fit model and therefore the accuracy of the linear representation of net-length and delay.

8.2 Timing Analysis

Fig. 11 illustrates the effect of the closed loop model on the ILP solution time at each iteration. The figure shows the results collected for the set of benchmarks for which optimal solutions were found. It is observed that the refinement induced by the closed loop model results in significant reduction in the convergence time between the initial and the final iteration. This is due to the iterative improvement of the accuracy of the ILP model and therefore an increase in the convergence time.

8.3 Architecture improvement

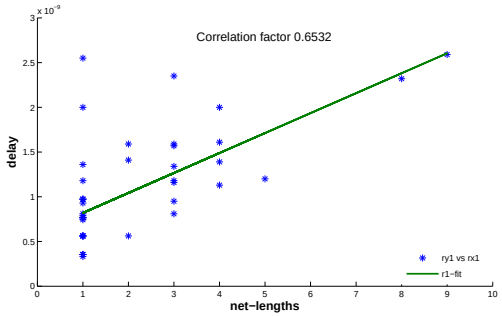
Fig. 12 shows the improvements achieved in the architecture using the closed loop model over a set of benchmarks. The results show that over the different circuits our framework has improved the output architecture with an average of 10% in comparison with the first iteration where congestion is not considered as (Fig. 12). In addition these results represent up to 25% total improvement in comparison with the best architectures found using the parameter sweep.

8.4 Summary

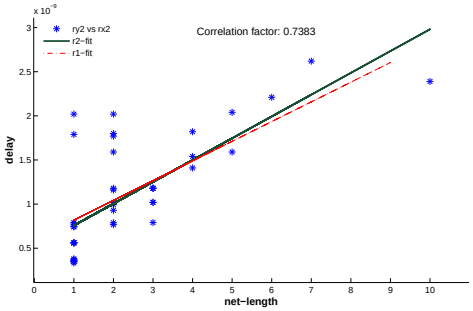
We have shown that while the ILP formulation is a simplified model of the FPGA architecture layout, the it produces better architectures than a traditional parameter sweep methodology. Moreover, this improvements is further increased using a closed loop methodology in which the accuracy of the VPR tool is used to refine the ILP model and therefore guide the optimal search to explore better quality architectures.

9. CONCLUSION AND FUTURE WORK

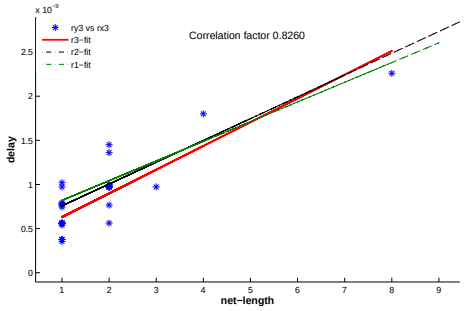
This paper has presented the benefits of using an analytical framework in the design of heterogeneous FPGA architectures over a typical parameter sweep approach. The framework uses mathematical modeling in the form of linear programming to model column-based architectures. An enhanced formulation motivated by the advances in the facility layout problem, has proved to successfully bound the design space and consequently reduce the solution time. Using this framework we have been able to simultaneously generate heterogeneous architecture layouts and reduce the critical path.



(a) First iteration



(b) Second Iteration



(c) Convergence Iteration

Fig. 10. Best fit model and correlations at first, second and convergence iterations.

The efficiency of this framework has been tested using a comparative experiment. For this purpose, a parameter sweep tool has been developed to sample the design space and test selected architectures on VPR 5.0. The experiments show an average improvement of up to 15% on the critical path induced by our analytical model in comparison with the parameter sweep approach. This shows that despite the assumptions that have been made to model the FPGA architectures in ILP, it still provides better architectures than a parameter sweep approach given the same time frame.

The framework has been extended to use the accuracy of VPR to refine the ILP

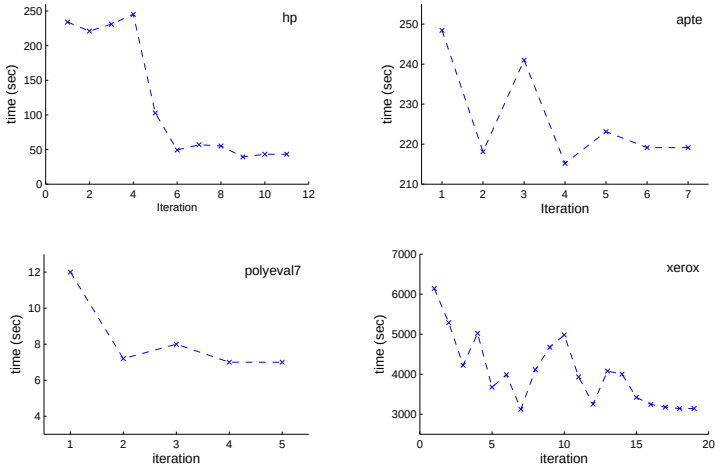


Fig. 11. Effect of closed loop model on ILP solution time.

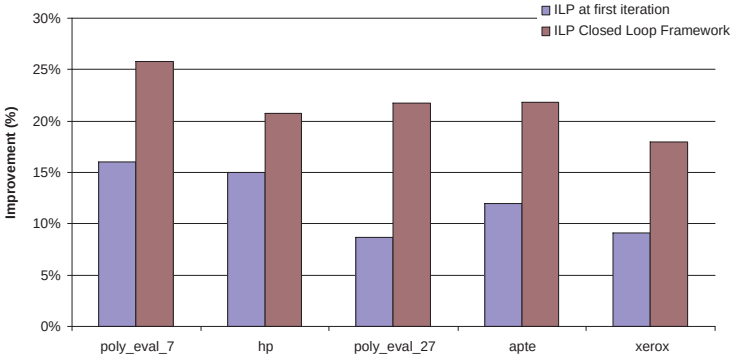


Fig. 12. Architecture improvement relative to best sweep architecture.

model and therefore improve the resulting FPGA architectures. This combined framework uses the efficiency of analytical tools and the accuracy of tools such VPR to obtain efficient architecture within a specific time budget. Our results showed a further average improvement of 10% and a total improvement of 25% in comparison with the parameter sweep methodology.

For future work we propose additional improvements to the model by introducing different coefficients for different connection types. This will result in an enhanced framework which will further improve the quality of the resulting architectures.

In this paper we initially aimed at showing the advantages of using analytical techniques over traditional ones, and measure the performance gained by combining

heuristics and analytical tools. While this model targeted delays, the model can also be further modified to account for other architectural aspects such as power.

REFERENCES

Balas, E. 1998. Disjunctive programming: properties of the convex hull of feasible points. *Discrete Applied Mathematics 89,* 1-3, 3–44.

Banerjee, P. Sur-Kolay, S. B. A. 2009. Fast Unified Floorplan Topology Generation and Sizing on Heterogeneous FPGAs. In *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*. Vol. 28. 651–661.

Betz, V., Campbell, W., Fang, P., Jamieson, P., Kuon, I., Luu, J., Marquardt, A., Rose, J., and Ye, A. 2008. Vpr 5.0 manual. In *http://www.eecg.toronto.edu/vpr/*.

Compton, K. and Hauck, S. 2002. Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv. 34,* 2, 171–210.

Emmert, J. and Bhatia, D. 1999. A methodology for fast fpga floorplanning. In *Proceedings. 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*. ACM New York, NY, USA, 47–56.

Feng, Y. and Mehta, D. 2006. Heterogeneous floorplanning for fpgas. In *Proceedings. IEEE International Conference on VLSI Design*. 257–262.

He, J. and Rose, J. 1993. Advantages of heterogeneous logic block architecture for fpgas. In *Proceedings. Custom Integrated Circuits Conference*. 7–4.

Hutton, M. Aug. 2006. Fpga architecture design methodology. In *Proceedings. International Conference on Field Programmable Logic and Applications*. 1.

Kahoul, A., Constantinides, G. A., Smith, A. M., and Cheung, P. Y. K. 2009. Heterogeneous architecture exploration: Analysis vs. parameter sweep. In *ARC*. 133–144.

Kuon, I. and Rose, J. 2006. Measuring the gap between fpgas and asics. In *Proceedings. 14th International Symposium on Field Programmable Gate Arrays*. ACM New York, NY, USA, 21–30.

Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y. 1996. VLSI module placement based on rectangle-packing by thesequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 15,* 12, 1518–1524.

Sherali, H., Fraticelli, B., and Meller, R. 2003. Enhanced model formulations for optimal facility layout. *Operations Research 51,* 4, 629.

Sherali, H., Smith, J., and Adams, W. 2000. Reduced first-level representations via the reformulation-linearization technique: results, counterexamples, and computations. *Discrete Applied Mathematics 101,* 1-3, 247–267.

Singhal, L. and Bozorgzadeh, E. 2007. Novel multi-layer floorplanning for Heterogeneous FPGAs. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*. 613–616.

Smith, A., Constantinides, G., and Cheung, P. 2005. An analytical approach to generation and exploration of reconfigurable architectures. In *Proceedings. International Conference on Field Programmable Logic and Applications*. 341–346.

Smith, A., Constantinides, G., and Cheung, P. 2008. Integrated floorplanning, module-selection, and architecture generation for reconfigurable devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16,* 6, 733–744.

Tsay, R., Kuh, E., Center, I., and Heights, Y. 1991. A unified approach to partitioning and placement [VLSI layout]. *IEEE Transactions on Circuits and Systems 38,* 5, 521–533.

Vecchietti, A., Lee, S., and Grossmann, I. 2003. Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Computers and Chemical Engineering 27,* 3, 433–448.