Improving Simulated Annealing-Based FPGA Placement With Directed Moves

Kristofer Vorwerk, *Student Member, IEEE*, Andrew Kennings, and Jonathan W. Greene, *Member, IEEE*

***Abstract*—Simulated annealing remains a widely used heuristic for field-programmable gate array placement due, in part, to its ability to produce high-quality placements while accommodating complex objective functions. This paper discusses enhancements to annealing-based placement which improve upon both quality and run-time. Specifically, intelligent strategies for selecting and placing cells are interspersed with traditional random moves during an anneal, allowing the annealer to converge more quickly and to attain better quality with less statistical variability. For the *same amount of computational effort*, the contributions discussed in this paper consistently improve both critical path delay and wire length compared to traditional annealing perturbations.**

***Index Terms*—Computer-aided design (CAD), directed moves, field-programmable gate array (FPGA), placement, simulated annealing.**

I. Introduction

MODERN field-programmable gate arrays (FPGAs) are displacing application-specified integrated circuits (ASICs) in many applications due to their ease of use, faster time to market, and lower nonrecurring engineering costs. Unlike ASICs, FPGAs employ predesigned routing fabrics that are specifically architected to possess good routability. Conversely, due to their prefabrication, FPGAs are not yet capable of achieving the high clock frequencies offered by ASICs. Thus, there is a need for better FPGA timing performance.

Placement is one of the most influential steps in the FPGA computer-aided design (CAD) flow—it is directly responsible for determining the relative locations of modules and (indirectly) for establishing the lengths of the routes between them. In a study of FPGA placement and routing heuristics [1], a finely tuned annealing-based placer was found to yield critical path delays which were up to three times smaller, on average, than those produced by a naive random scattering strategy. Given its influence over solution quality, FPGA placement is, therefore, a reasonable avenue in which to investigate ways for improving design performance and is the focal point for the remainder of this paper.

Typically, placement seeks to minimize wire length and critical path delay subject to the constraint that cells be placed in legal positions. Simulated annealing remains a widely used heuristic for FPGA placement due, in part, to the flexibility with which the annealing objective can be adapted to handle realistic architectural constraints. However, as FPGAs continue to grow in size, the large run-times incurred by simulated annealing are becoming prohibitive. To limit the search space, practical implementations perform random perturbations of logic within range-limited windows [2]. Such "simple moves" are inexpensive, but many such moves must be performed to achieve good quality.

This paper investigates the concept of "directed moves" during simulated annealing. This notion was inspired by previous research in the ASIC domain [3], [4] where intelligent deterministic strategies for selecting and replacing "poorly placed" cells were *interspersed* with random simple moves during annealing. Directed moves serve to reduce the size of the search space by focusing on cells and locations of interest; this allows the annealer to converge more quickly and to attain a better placement for the same amount of run-time as if random moves had been used alone.

In this paper, several concepts for directed moves are considered, such as median placement [5], cell rippling, graph coloring, optimal linear assignment [6], and the minimization of monotonic path deviation (MPD) [7]–[9]. These directed moves were implemented in a modern C++-based academic FPGA placement framework called `KPF`. The results confirm that, for the same amount of computational effort, wire length-driven and timing-driven directed moves—when used in combination with simple moves—routinely lead to improvement in both critical path delay and wire length, compared to having used simple moves alone. Moreover, directed moves are shown to reduce the statistical variability of the annealing-based placements. This paper furthers the work of [10] by additionally presenting a technique for measuring the effectiveness of moves, as well as by exploring the effects of moves in clustered FPGA architectures.

The rest of this paper is organized as follows. Section II presents an overview of simulated annealing, as well as a discussion of historical attempts to improve FPGA placement. Section III discusses the implementation of the directed moves. Finally, Section IV presents numerical results and Section V concludes this paper.

Manuscript received March 5, 2008; revised July 11, 2008 and September 15, 2008. Current version published January 21, 2009. This work was supported in part by a Grant from Actel Corporation. This paper was recommended by Associate Editor K. Bazargan.

K. Vorwerk is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada, and also with Actel Corporation, Mountain View, CA 94043-4655 USA (e-mail: kris@vorwerk.ca).

A. Kennings is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

J. W. Greene is with Actel Corporation, Mountain View, CA 94043-4655 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.2009167

II. Background

A. Overview of Simulated Annealing and Directed Moves

Simulated annealing is a search-based heuristic for minimizing an objective function $\mathbf{F}$ which takes real values over a set of states S. Given a perfect cooling schedule, simulated annealing will ultimately converge to an optimal solution [11]; however, the state space exploration must satisfy some conditions for optimality to be assured.

The function for generating a new state j given a current state i is given as $g(i, j)$. A matrix $\mathbf{P}$ can be used to represent the state transition probability of a traditional annealing process [12], and takes the form

$$P_{ij} = \begin{cases} 0, & \text{if } j \notin N(i) \text{ and } i \neq j \\ \min\left\{1, e^{\frac{-F(j)-F(i)}{T}}\right\}, & \text{if } j \in N(i) \\ 1 - \sum_{j \in N(i)} P_{ij}(T) & \text{if } i = j \end{cases} \quad (1)$$

where $N(i)$ represents the neighbors of the state i, $i \notin N(i)$, and T is the temperature. For a given T, the probability distribution of the annealing problem can be viewed as a stationary time-homogeneous Markov chain, represented as

$$\pi_i(T) = \frac{g(i)e^{-\frac{F(i)}{T}}}{G(T)} \quad (2)$$

where $g(i)$ is a normalizing function such that $\sum_{j \in N(i)}(g(i, j)/g(i)) = 1$, and $G(T)$ is a scaling factor such that $\|\pi(T)\| = \sum_{i=1} |S|\pi_i(T)$.

For this Markov model to converge to the optimal solution, the neighbor generation criterion must satisfy $g(i, j) = g(j, i)$ for all states i, j [11]. This leads to the stationary probability condition $\pi(T)\mathbf{P}(T) = \pi(T)$, which in turn, leads to the detailed balance criterion

$$\frac{\pi_i(T)}{\pi_j(T)} = \frac{g(i)}{g(j)} e^{\left(-\frac{F(j)-F(i)}{T}\right)} = \frac{P_{ji}(T)}{P_{ij}(T)}. \quad (3)$$

That is, the probability of changing from state i to state j must be the same as the probability of generating i when the system is in j times the probability of accepting it [13].

The motivation for this paper stems from the observation that an annealer may spend time revisiting previously explored states and may require significant time before it finds the lowest cost states. If $g(i, j)$ could be made to explore neighbor states that are more likely to yield improvement, the amount of time required for the anneal can be reduced. This preferential state exploration forms the basis for the "directed moves" described in this paper.

If directed moves are implemented without consideration for the detailed balance criterion, the risk of oscillation and of converging to a local minimum is raised. It is worth noting that it is possible to implement directed moves which do not harm detailed balance by ensuring that the probability of accepting the "reverse" move (i.e., P_{ji}) compensates for the earlier, directed state exploration. However, as will be shown in this paper, this compensation is not necessary, in practice, to achieve high-quality FPGA placements provided that a sufficient portion of attempted moves are still of the traditional, nondirected kind.[1] Moreover, directed moves do not preclude the attainment of high-quality results in a practical annealer since numerous heuristics (such as clustering, windowing, noninfinite starting temperatures, and so forth) are already employed to reduce runtime at the expense of producing suboptimal solutions. Thus, the goal, in this paper, is for directed moves to serve as a means of "shifting" the cost function curve so that better quality placements can be produced more quickly. This concept is shown in Fig. 1, where the application of directed moves improves the quality of result (QoR) within a given run-time range.

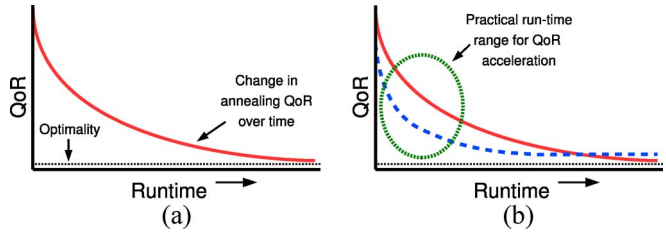

Fig. 1. QoR achieved by an annealer is typically a function of its run-time, as shown in (a). QoR offered through the use of directed moves, as shown by the dotted line in (b), comes as a result of accelerating the QoR improvement within a range of practical run-times.

Only two prior works, [3] and [4], are known to have discussed the concept of directed moves in the context of cell placement. In [4], wire length-reducing moves were coupled with simple, random moves to reduce the search space and improve the packing of the `Parquet` ASIC floorplanner. In [3], a move based on weighted centroids was interspersed with random moves to improve the quality of the `Timberwolf` [14]–[16] standard cell placer.

This paper presents five additional contributions for directed moves compared to the previous body of literature:

1) many new types of directed moves are considered;
2) a new technique (based on cell rippling) to retain placement feasibility is described;
3) the use of multiple directed moves to optimize separate annealing objectives (such as both wire length and critical path delay) is explored;
4) a heuristic for automatically ranking moves and for determining when to terminate an anneal based on the moves' effectiveness is presented;
5) a comprehensive quality versus run-time analysis of the directed moves is discussed for both clustered and nonclustered FPGA architectures.

While some of the directed moves described in this paper have been employed in postplacement optimization strategies (e.g., [8]), the integration of these methods into the annealing loop (1) unifies and simplifies the implementation and (2) allows the methods to consider a wider search space which can lead to potentially better results.

B. Historical Techniques for Improving FPGA Placement

`VPR` is, perhaps, the best known annealing-based academic FPGA placer. Through `VPR`, Betz and Rose [2] introduced

[1]The move selection strategy described in Section III-D typically ensures that this is the case.

TABLE I
COMBINATIONS OF SOURCE AND TARGET CELL MOVES CONSIDERED IN THIS PAPER, AS WELL AS THE METHODS CONSIDERED FOR RESOLVING INFEASIBILITIES (VIA ASSIGNMENT OR CELL RIPPLING). FOR COMPARISON PURPOSES, USE A "**WL**" OR "**CP**" IS USED TO INDICATE WHETHER THE MOVE TARGETS PRIMARILY WIRE LENGTH OR CRITICAL PATH DELAY. "♡" IS USED TO INDICATE MOVES THAT LEAD TO HIGHER QUALITY AND "♄" TO DESIGNATE HIGHER RUN-TIME COMPLEXITY

Target Selection ⇒ **Source Selection** ⇓	**Random**	**Domino** (WL)	**Median** (WL)	**MPD** (CP)	**Weighted Median** (CP)	**Centroid** (CP)	**Priority List** (CP)
Random	Assign (♡♡♄)	n/a	Assign (♡♡♡♄♄) Ripple (♡♡♡♡♄♄♄)	Assign (♡♡♡♡♄♄) Ripple (♡♡♄♄♄)	Assign (♡♡♄♄♄) Ripple (♡♡♡♄♄♄♄)	Assign (♄)	Assign (♡♄)
Coloring	n/a	Assign (♡♡♡♡♄♄♄♄)	n/a	n/a	n/a	n/a	n/a
Priority List	n/a	n/a	Ripple (♡♡♄♄♄)	Ripple (♄♄)	n/a	n/a	n/a

several key improvements to FPGA placement, including the concept of path-based weighting for timing-driven optimization, timing-driven clustering, fast incremental bounding box computation, and a well-described experimentally tuned annealing schedule.

Since the inception of VPR, numerous attempts have been made to improve upon placement quality and run-time. Most techniques have fallen into one of the following categories: 1) better initial placement (e.g., by coupling annealing with another placement strategy); 2) better clustering (for clustered FPGA architectures); 3) modifications to the circuit netlist (e.g., through logic duplication or basic logic element (BLE)-level placement); 4) more accurate objective functions (e.g., path-based timing weights, or the incorporation of congestion metrics); or 5) changes to the annealing schedule (temperature, acceptance criteria, and so on).[2]

Alternative strategies have been investigated to reduce the amount of run-time spent in high-temperature annealing regimes. For instance, faster placement heuristics, such as recursive min-cut partitioning, have been employed [18], [19] to quickly produce better initial placements, thereby requiring the annealer to run only in a lower temperature regime.

Better packing algorithms have also been shown to improve the quality of annealing-based placements for clustered architectures. A new method for packing logic was proposed in [20] where physical cell distances were incorporated into the packing cost function. A routability-driven packing algorithm was described in [21] where careful attention was paid to the selection of the seed BLE, and the packing was controlled by the Rent parameter of the architecture. Depth-optimal and depth-relaxed packing methods [22]–[24] have also been described in which timing-critical logic was duplicated during clustering to obtain a set of clusters with optimal depth.

Recent approaches for improving quality during annealing-based placement have focused on modifying the circuit netlist prior to, during, or after placement. One method has been to perform logic duplication and path straightening [7]–[9], [25]. Alternatively, a clustered [cluster-based logic block (CLB)] netlist itself can be reclustered during placement by moving BLEs, as performed in SCPlace [25].

[2]Some of these schemes are only applicable to clustered architectures, although fine-grained nonclustered architectures also exist; cf., [17].

The design of better annealing objective functions has also been considered. By changing timing weights [26] and deriving more accurate models for net costs, better placement results have been reported. For instance, [27] describes a method for computing timing criticalities based on path counting and the use of a discount function. Numerous works have also discussed the integration of global routing and placement to improve the fidelity of the placer's objective function [28].

Run-time improvements have also been discussed in the recent literature. For instance, parallel techniques have been described to accelerate annealing [29], [30]. Shorter temperature schedules have additionally been proposed [31].

III. IMPLEMENTATION

An academic annealing-based placer, called KPF, was developed as part of this paper. It targets the same type of architecture and CAD flow as VPR.

In KPF, a "move" is the fundamental operation of perturbing a placement (i.e., generating a neighbor state)—picking a set of source ("from") cells S choosing a set of target ("to") locations T and then assigning S to T. Like most annealers, feasibility is maintained throughout the anneal. Thus, if a cell S_0 is assigned to an occupied location T_1 (i.e., occupied by cell S_1), it will either assign S_1 to T_0 (as in VPR) or *ripple* a set of cells along a path from T_1 to T_0 to retain legality. The concept of rippling will be discussed in more detail in the following sections.

In KPF, multiple cells can be assigned to multiple locations in a single perturbation—the delta cost for moving all cells is computed at once, and the *entire move* is either accepted or rejected. It is possible that significant improvement could be wrought from such a move, but at the same time, if the move is rejected, it could be costly in terms of run-time.

VPR uses a random strategy to choose *source* cells and a random, shrinking window method to choose *target* locations; these moves are referred to as "simple moves." KPF implements such moves as well as other strategies for picking source cells and target locations. In addition, both assignment and rippling have been investigated as means of retaining feasibility, as summarized in Table I. As discussed in Section IV-A, not all possible combinations of strategies in this table were exhaustively tested, and of the moves which were tested, not all proved to be useful. This paper focuses primarily on the subset of moves which did yield improvement in early testing.

A. Heuristics for Determining Source Cells

Three heuristics were implemented for determining source cells. The first heuristic was a traditional **random** cell selection, as implemented in VPR.

The second heuristic was based on the concept of **graph color**ing in which the netlist hypergraph is colored *prior* to placement. During placement, the coloring is used to randomly choose independent nonconnected cells (i.e., cells which do not share a net) in subsets of up to 15 cells at a time. This graph coloring permits the later application of the Domino technique (described in the next section) to compute an assignment (and therefore, a placement) for the cells.

The final selection heuristic was based on **priority lists**, where a cell is chosen, at random, from a list containing the 25% *worst-placed* cells in the design. At each annealing temperature update, this priority list is recreated by scoring the cells based on: 1) their distance away from their optimal half-perimeter wire length (HPWL) positions as well as 2) the maximum timing cost of paths through the cell. The goal of this source selection strategy is to "focus" the efforts of the annealer so that it chooses cells that are more likely to yield improvements in quality (and less time is wasted in unproductive moves). The distance from the optimal HPWL positions is measured using the concept of median placement (which is described in the following section). The timing cost ranks are computed as a function of pin criticality multiplied by delay for each driver-sink pair.

B. Heuristics for Determining Target Locations

Several heuristics for determining the target locations for cells were implemented. These methods generally incorporate aspects of randomness within a "focused" area and are usually geared toward optimizing wire length or timing.

The first heuristic was a standard **random** target location within a shrinking window, as implemented in VPR.

Another heuristic, primarily geared toward wire length, was based on **Domino** [6]. Given a set of cells, KPF solves a minimum cost linear assignment problem to assign the cells to sites where the sites are located at the positions of the original cells. When setting up the assignment problem, it is important that the costs on each arc closely model the *actual* cost of assigning a cell to a location. By selecting the source cells via graph coloring, it is assured that none of the cells in the transportation problem are interconnected (i.e., no cells share a common net); consequently, the wire length costs on the arcs can be determined perfectly, and the placement will be optimal for wire length [6]. One caveat to this approach is that the assignment ignores timing, and this can result in the move being rejected. (In this paper, the Goldberg–Tarjan push-relabel method [32] was used to compute this assignment.)

Another target location heuristic was developed based on the concept of optimal wire length positioning via **median placement** [5], [33], This technique determines the optimal range of (x, y) positions which minimize HPWL for a given cell i, and works as follows. First, the x and y minimums and maximums of the bounding boxes for all nets (excluding the

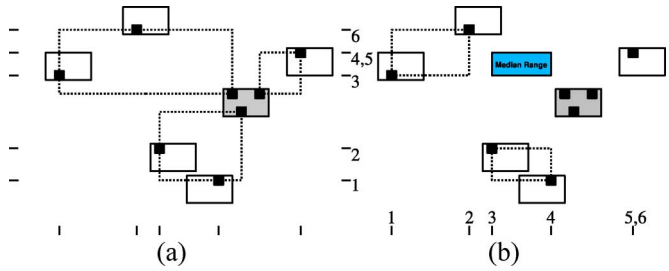

Fig. 2. Median calculation for a cell C connected to three nets: the original placement of cells is shown in (a), while the optimal range of (x, y) values for cell C is shown in (b).

points contributed by cell i) are inserted into two vectors—one for each direction. The vectors are then sorted by increasing value. The median $(\lfloor n/2 \rfloor)$ and median plus one$^{\text{th}}(\lfloor n/2 \rfloor + 1)$ entries in this vector yield the range of positions into which cell i can be moved to optimally improve its HPWL (assuming that all other cells are fixed). A target is chosen *at random* from within this range, as shown in Fig. 2.

To minimize critical path delay, a target heuristic was implemented based on the concept of minimizing MPD. Given a source cell S_0, the *feasible region* [9], which would reduce the MPD, is computed. A location is picked (at random) from within this region as the target location into which S_0 will be moved. This implementation computes feasible regions exactly as in [9]. (Superfeasible regions are not computed.)

Another timing-driven technique for target selection was implemented based, again, on the concept of **priority lists**. In this approach, a list was used to determine target cell locations by tracking the top 25% of cells in the netlist containing the most timing slack. One of the cells in this target list is chosen at random, and the source cell and this newly chosen cell are swapped. (This move is conceptually similar to one employed in Parquet [4].)

Another modification to the *median placement* strategy was also considered where, instead of deriving a region from the median positions of connected cells, a *weighted median* computation is employed to improve timing. In this approach, the median values of the nets are *weighted* by the timing cost of their respective pins. This approach skews the median region toward pins with higher timing costs. The target location into which to move a cell is chosen at random from within the weighted median range. The intent of this heuristic was to produce a move that could simultaneously reduce wire length as well as timing.

Lastly, a *weighted centroid* approach, akin to [3], was investigated. Given a source cell, the target location for the cell is computed as the weighted centroid position of its drivers and receivers, where the weights are based on the timing costs of the pins.

C. Resolving Infeasibilities

Early on, it was observed that median placement—despite being optimal for wire length (for a cell moving into the computed median range)—often resulted in numerous rejects when "swapping" with a cell. The median placement would yield good improvement for the first cell which was being moved

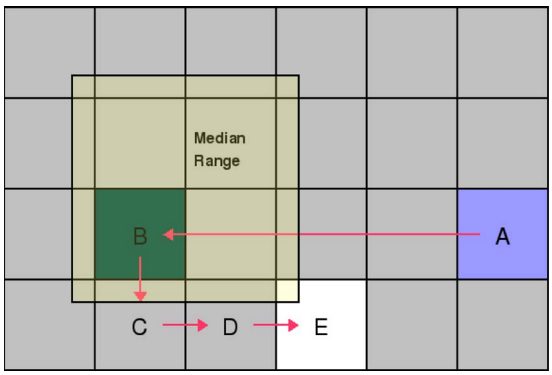

Fig. 3. Example of cell rippling when used in conjunction with the median placement strategy.

into its optimal range, but would often result in an increase in wire length for the second cell as it was swapped back into the first cell's original position. To lessen this impact, a new method—cell rippling—was developed to maintain legality during placement.

Cell rippling works as follows. Given a source cell S_0 (at location T_0), some technique (e.g., median placement) is first used to calculate a target location, T_n, for that cell. If T_n is unoccupied, S_0 will simply be assigned to that location; however, if T_n is occupied, S_0 is moved to T_n, and the previous contents are *rippled* from T_n by one grid unit toward the nearest empty location (which will be bounded within the distance of T_0 to T_n). This rippling is computed for all cells along the path toward the empty site, creating a set of rippled cells and "spreading" the perturbation across several cells instead of just one cell.

An example of cell rippling is shown in Fig. 3. In this diagram, cell A was chosen as the source cell, and cell B was chosen at random (from within a median region) as its target location. The cell rippling first discovers the nearest empty location within four units of B; B is subsequently rippled *toward* the empty location and placed at C. The cell formerly at C is then rippled to D, and the cell at D is rippled into the empty space at E. Four cells are assigned at one time in this "move," and the entire rearrangement is either accepted or rejected by the annealer.

Although the nearest empty location is always used, the rippling directions are chosen randomly so that in the event of multiple calls with the same source cell and target locations, the chosen rippling path may be different each time. The cell rippling does not attempt to choose a path which minimizes the effect on timing; it was felt that doing so would incur too much of a run-time penalty for the move and might render it too deterministic.

D. Move Selection and Effectiveness

Typically, a given directed move will impart different changes in the cost function at different temperatures. Ideally, moves will be employed when they are most likely to improve the cost function. To accomplish this goal, the move selection probabilities are adapted dynamically based on changes in the cost function.

In [34], the effectiveness (or, "quality factor") of moves was computed as part of this dynamic move probability selection. The quality factor for a move type m was computed as follows:

$$\mathcal{Q}(m) = \frac{\sum_{\text{accepted moves } i \text{ of type } m} |\Delta c_i|}{\sum_{\text{attempted moves } i \text{ of type } m} t_i} \tag{4}$$

where Δc_i is the computed delta cost and t_i is the time taken to compute and perform a move i.

In this paper, the effectiveness is computed in a similar fashion, except that the equation is modified as follows:

$$\mathcal{E}(m) = \frac{\sum_{\text{attempted moves } i \text{ of type } m} |p_i \cdot \Delta c_i|}{\sum_{\text{attempted moves } i \text{ of type } m} t_i} \tag{5}$$

where p_i is the probability of accepting the move.[3] This reduces the sampling noise.

At the beginning of each temperature change, the annealer uses the relative effectiveness of each move to determine the probability of selecting that move in the forthcoming pass of the annealer. The more effective that a move was in the previous annealing pass, the more likely that it will be chosen in the next pass of the annealer. Specifically, the probability $\mathcal{P}$ of selecting a move i is computed as

$$\mathcal{P}(i) = \frac{\mathcal{E}(i)}{\sum_{\forall \text{ moves } j} \mathcal{E}(j)}. \tag{6}$$

The probability of selecting a move remains constant until the next temperature change.

To prevent rapid fluctuation in the probabilities from one temperature change to another, the move selection probabilities are smoothed using the function

$$\mathcal{P}_{\text{smoothed}}(i) = \lambda \cdot \mathcal{P}_{\text{prev}}(i) + (1 - \lambda) \cdot \mathcal{P}(i) \tag{7}$$

where $\mathcal{P}_{\text{prev}}(i)$ is the probability of selecting move i from the previous annealing temperature pass, λ controls the effect that historical probabilities impart on the selection of current move probabilities, and $\mathcal{P}(i)$ is the probability computed in (6). The probability of selecting a move is constrained so that it never falls below a predefined threshold—this ensures that poorly performing moves are occasionally attempted (and not completely eliminated in the event that they experience a "string of bad luck").

Fig. 4 shows the relative effectiveness of different moves during the placement of a design. In this diagram, three types of moves were employed—median placement (with rippling), MPD (without rippling), and random moves—with the move probabilities of each initially set to 30%, 20%, and 50%.

[3]Note that in Metropolis rejection schemes

$$p_i = \begin{cases} 1, & \text{if } \Delta c_i \leq 0 \\ e^{-\frac{\Delta c_i}{T}}, & \text{if } \Delta c_i > 0 \end{cases}$$

where T is the current temperature.

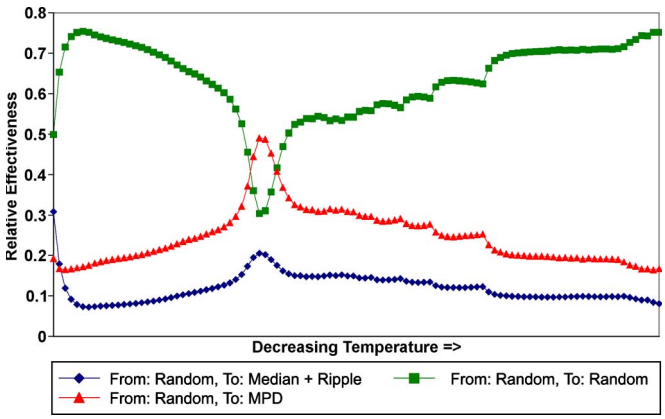

Fig. 4. Illustration of the probabilities of selecting median placement, MPD, and random moves over the course of an anneal.

As the temperature cooled, the directed moves became more effective than the random moves. At low temperatures, most cells were placed in their optimal median ranges, and timing paths were relatively straight; thus, the directed moves become less effective (for the amount of run-time that was required for their computation) than the simpler random perturbations.

The concept of move effectiveness can also be used to determine when to terminate the anneal. If the effectiveness of all moves drops below a threshold, the annealer can stop. VPR's criterion [2], on the other hand, is tuned to specific parameters, and is set to stop at an iteration n if $T(n) < (\|F(n)\|/200 \times |E|)$ where $T(n)$ is the temperature at n, $\|F(n)\|$ is the normalized objective function value for the placement, and $|E|$ represents the number of nets in the netlist. This stopping criterion does not work if $T(n)$ is zero, and may not be suitable for all cost functions, whereas move effectiveness is independent of temperature and design parameters.

IV. Numerical Results

Directed moves have the potential to produce high-quality results but can require more run-time than simple moves. In this section, evidence is presented to support the claim that directed moves (when interspersed with simple moves) can consistently produce *better quality* placements *for the same amount of work* than if simple moves had been used alone.

For comparison purposes, KPF was used to produce placements and VPR was used for routing. KPF employs criticality history costs [26] and path counting [27] to improve timing-driven results, and can optionally perform BLE-level operations akin to [25]. With these enhancements, KPF produces wire length, critical path delay, and run-time results which are on-par with leading academic placers (cf., [25], [26]). It is felt that KPF is of sufficiently high quality that the conclusions drawn in this paper regarding directed moves are valid.

The 20 largest designs from the Microelectronics Center of North Carolina (MCNC) testsuite [35] were employed for testing. Several architectures were examined—specifically, one BLE/CLB, four BLE/CLB, and eight BLE/CLB architectures. Low-stress routing [2] was employed in all cases as this paper aimed to quantify the benefits of different types of annealing perturbations on the quality of placement results and not to measure the minimum architectural channel widths. (It is noted that low-stress routing architectures also exist commercially; cf., [17].) Each circuit from the testsuite was run with five different seeds with the average over the five seeds used for each design.

The quality of KPF with directed moves was compared to KPF with simple moves alone. The probabilities of selecting a given directed move (versus a simple move) were determined as described in Section III-D. To produce the quality versus run-time curves, up to 16 different combinations of starting temperatures and moves per temperature (inner_num) were tested. The average of five seeds for each of the 20 benchmark circuits (for the various combinations of directed moves) was computed.

To aid in visualizing the QoR trends, the graphs in the following sections present the normalized *geometric means* of the QoR for *the entire suite* (not for individual circuits), plotted against the normalized run-times for the suite. The QoR and run-time values were normalized against the longest running simple-move anneal—thus, a value of "1.0" on the y- or x-axis roughly corresponds to the QoR or run-time achievable by an anneal with an inner_num of ten and an initial starting temperature of 20 times the measured standard deviation.[4]

Care was taken to ensure that the annealing parameters were properly tuned for each of the run-time comparison points presented in this paper. It is noted that the baseline's nominal (1.0x) run-times may, subjectively, be considered large by industrial standards. Consequently, it is worthwhile to consider the dominance of directed moves across a range of run-times. It is expected that the improvement offered by directed moves at the nominal run-time will typically be smaller than the improvement offered at faster run-times based on the justification presented in Section II-A and Fig. 1.

A. Commentary on Successful Moves

Very few of the directed moves that were implemented produced meaningful improvement in terms of quality versus run-time compared to using simple moves alone. Moves which did not show early promise were quickly eliminated from further testing. Some moves yielded good improvement but the run-time costs usually outweighed the benefits—more often than not, it would have been better to anneal longer using simple moves than to have employed certain directed moves.

Given the large number of combinations of moves and ways of retaining legality, benchmark sweeps were conducted on one BLE/CLB high-utilization architectures to prune combinations of moves which did not appear to be worth pursuing. Some leniency and engineering judgment were applied when choosing the most successful moves for later analysis—success was qualitatively weighed based on run-time and quality relative to the other move combinations, as well as the effectiveness of the move throughout the anneal. It is worth noting that, when plotted, successful moves produced an effectiveness pattern

[4]In other words, each trend-line in each graph in the following sections was produced from 1600 individual placements—20 designs times five seeds per design times 16 different combinations of starting temperature and moves per temperature.

TABLE II
SUMMARY OF THE RESULTS FOR UNSUCCESSFUL DIRECTED MOVES RELATIVE TO A BASELINE ANNEAL

Source Strategy	Target Strategy	Legality	`timing_tradeoff`	WL Ratio	CP Ratio	CPU Ratio
Domino	Graph Coloring	Assign	0	0.97	0.98	$4.7\times$
Domino	Graph Coloring	Assign	0.5	0.97	0.99	$2.9\times$
Priority List	Median	Ripple	0	1.00	1.01	$1.9\times$
Priority List	Median	Ripple	0.5	0.97	1.03	$1.6\times$
Priority List	MPD	Ripple	0.5	1.05	1.05	$1.5\times$
Random	Centroid	Assign	0.5	1.03	1.06	$1.6\times$
Random	Priority List	Assign	0.5	1.04	1.06	$1.4\times$
Random	Weighted Median	Assign	0.5	0.89	1.05	$1.5\times$
Random	Weighted Median	Ripple	0.5	0.90	1.04	$2.1\times$

similar to Fig. 4 whereas unsuccessful moves approached near-zero effectiveness after a couple of temperature passes.

Examples of move combinations which were attempted and deemed to be unsuccessful are summarized in Table II. Note that this table excludes the results produced by the weighted median and MPD moves because these were among the most successful combinations, and are treated in more detail in the subsequent sections. For each row in this table, the numerical results were produced for the specified directed move in combination with a traditional simple move. The table highlights the ratios of quality and run-time relative to an anneal without directed moves.

The least successful moves fell into one of two categories:

1) moves that yielded improvement, but were otherwise too computationally expensive;
2) moves which did not yield improvement.

The Domino move offered some improvement but its run-time excluded it from further consideration. The timing-weighted median placement move offered improvement in wire length but not in timing, but because of the run-time overhead, it was not adequately compelling to use versus a nonweighted median move. The priority list and weighted centroid strategies, on the other hand, tended to produce both worse results and worse run-time. It may be reasonable to place cells at centroids to minimize wire length, but it is not necessarily the case that the centroid is the right place to put a cell to minimize critical path delay. Similarly, swapping critical cells with those that are noncritical prevents the annealer from exploring potentially empty grid sites and does not guarantee a progression to better solutions.

By and large, the most successful directed moves were those which coupled a sufficient amount of randomness with a quick easy-to-compute high-quality placement heuristic. The three best examples, from this paper, are based on a *random* source selection and, for target selection, either the use of *median placement*, *median placement with cell rippling*, or the minimization of *MPD*. It is this set of moves which form the basis of the analysis for the remainder of this paper.

It is noted that the move selection algorithm (cf., Section III-D) decreases the probabilities of ineffective moves until they approach near-zero values. Consequently, all moves could be turned "on" initially and the probabilities of selecting ineffective moves will be reduced automatically. However, this requires several temperature passes which can slow down the placement. As a result, in the remainder of this paper, only the effective moves were enabled during placement.

B. Nonclustered Architectures

The one BLE/CLB architecture was used for investigating directed moves under a variety of scenarios.

1) Device Utilization Tests: In the first experiment, combinations of directed moves were tested on devices with both high-utilization ($\approx$100% utilized) and medium-utilization ($\approx$60% utilized) architectures with the VPR-equivalent cost function parameter `timing_tradeoff` $= 0.5$. While the literature has traditionally presented placement results in the context of high-utilization architectures, medium-utilization devices were also considered in this paper since these tend to be more representative of what is seen in industry.

Results are presented for a set of the more successful directed moves including median placement ("median"), median placement followed by cell rippling ("median + rippling"), weighted median placement (to account for timing criticalities), and MPD. Note that, in these figures, as long as the data points for directed moves fall below the baseline's trend-line, the directed moves offer *improvement* in QoR.

The success of the directed moves on a quality versus run-time basis for high-utilization architectures is shown in Fig. 5. This diagram shows the results from having used different algorithms for source ("from") cell selection and target ("to") location determination. Infeasibilities are resolved by swapping or rippling, as indicated in the diagram. In these architectures, the use of median placement with rippling resulted in an improvement of 5% in wire length (on average) at the nominal run-time and at the cost of 2% worsening (on average) in critical path delay. The use of occasional MPD moves resulted in 3% improvement in critical path delay (on average) at the cost of 2% in wire length (on average) at the nominal run-time.

The success of directed moves is much more apparent in medium-utilization architectures, as shown in Fig. 6. These curves show that, for the same amount of run-time, median plus rippling moves yielded 9% better wire length (on average) and 2% better critical path delay (on average) at the nominal run-time. The use MPD moves yielded 4% better critical path delay (on average) and 2% better wire length (on average), at the nominal run-time.

It is believed that the greater success of directed moves in the medium-utilization architectures stems from two sources. First, the larger the device, the larger the "search space" of available locations that can be considered by the annealer. The use of directed moves allows this search space to be pruned and focuses the annealer on regions most likely to yield improvement. Second, directed moves are not constrained by shrinking

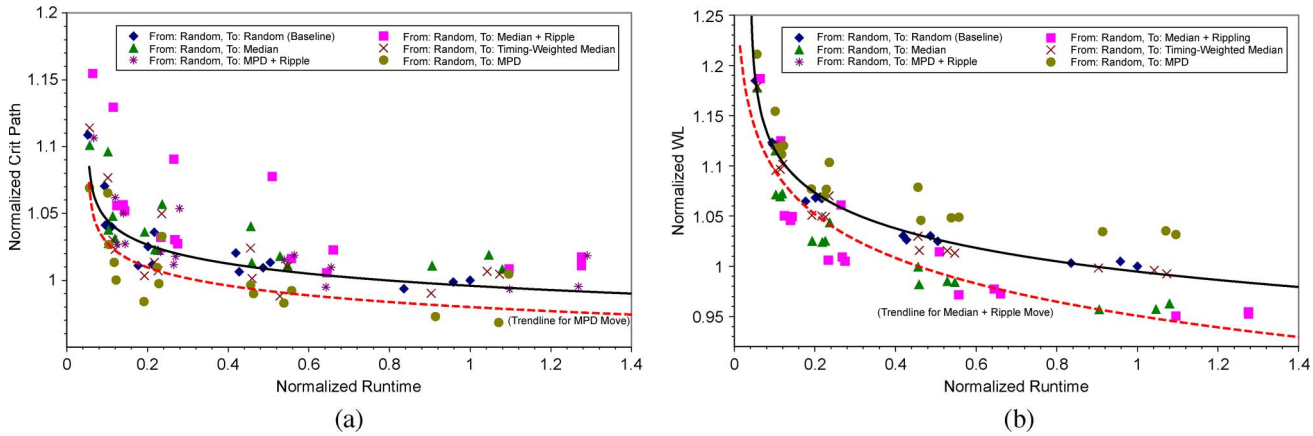

Fig. 5. (a) Critical path and (b) wire length quality curves for one BLE/CLB high-utilization architectures.

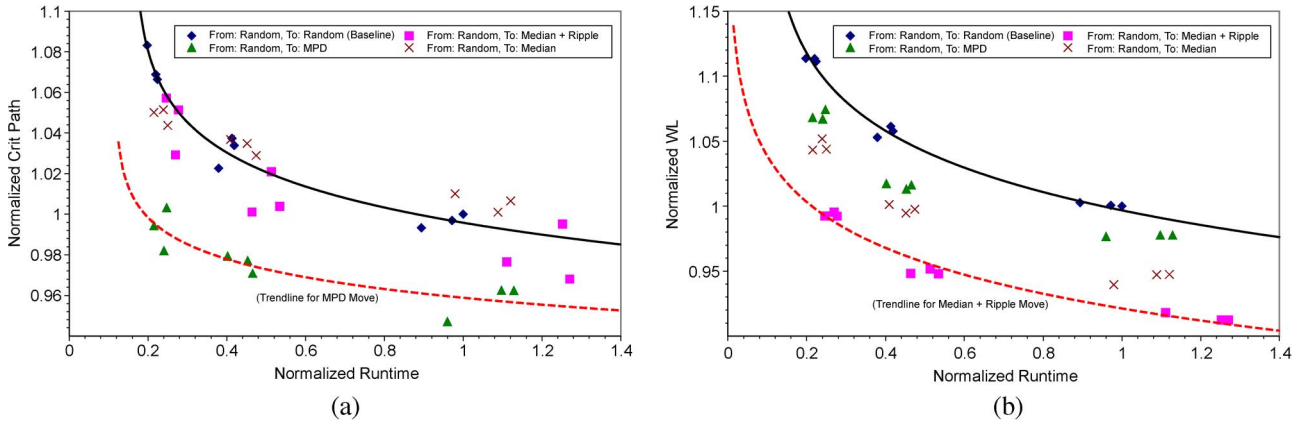

Fig. 6. (a) Critical path and (b) wire length quality curves for one BLE/CLB medium-utilization architectures.

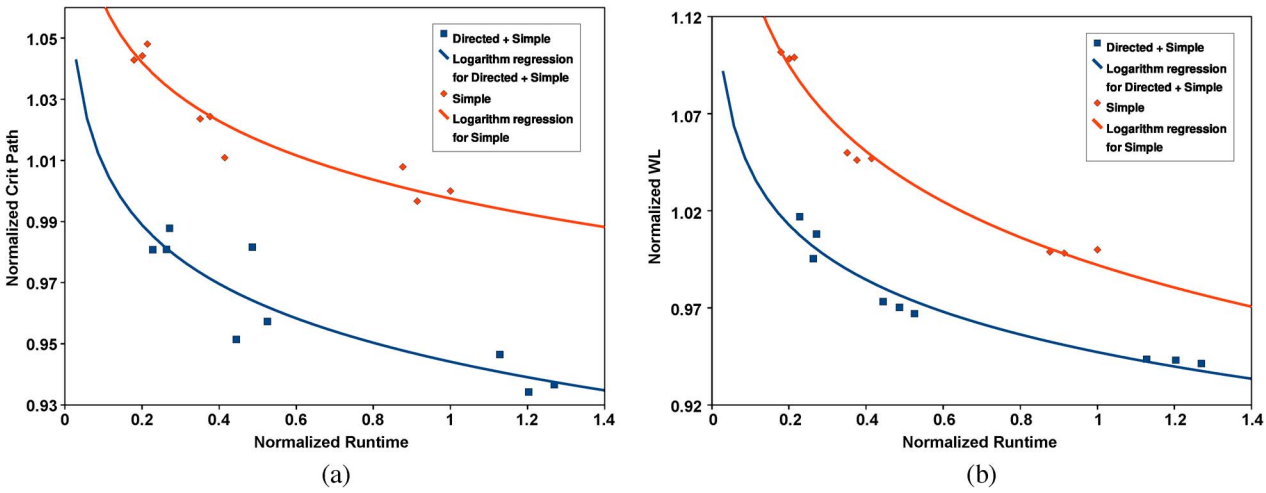

Fig. 7. (a) Critical path and (b) wire length quality curves for `timing_tradeoff` $= 0.1$, illustrating the dominance of directed moves versus simple moves alone.

windows, so they can move ill-placed cells further distances at cooler temperatures than random moves. It is important to stress that the improvements achieved by directed moves in the larger architectures are *not* a result of decreased routing congestion as large channel widths were purposefully used to ensure low-stress routing.

In general, cell rippling was found to improve wire length quality when coupled with the median placement moves, but it typically resulted in a 1% worsening of critical path delay. Similarly, cell rippling was found to reduce the negative impact to wire length in the MPD moves, but the critical path delay improvement was not as good. This is attributable to the fact that the rippling occasionally perturbs timing-critical cells. Due to its success in wire length minimization, the effect of cell rippling was examined primarily with the median placement move in the remainder of this paper.

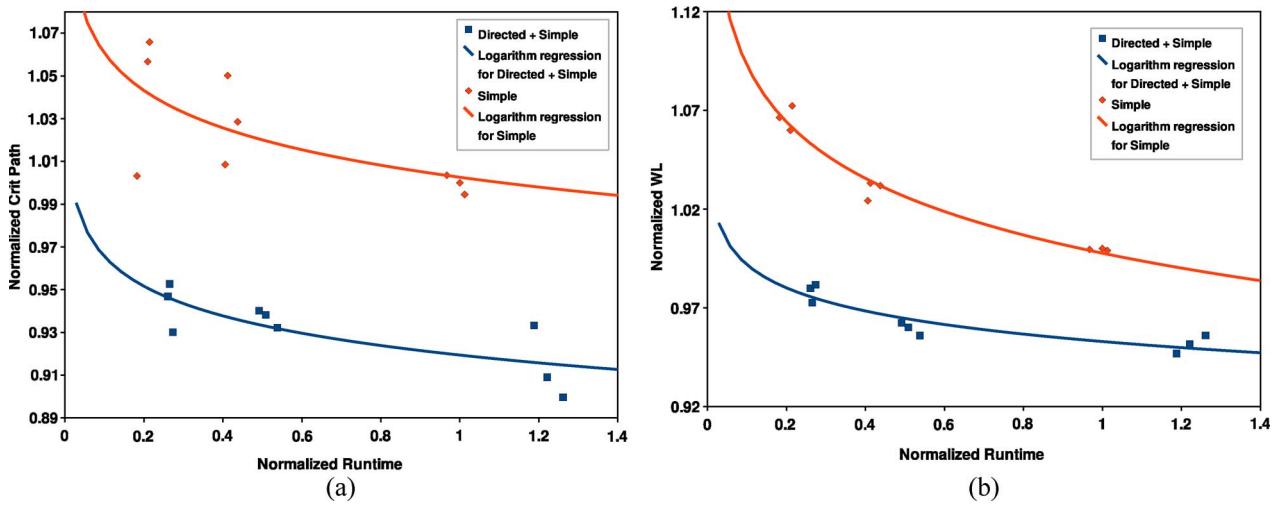

Fig. 8. (a) Critical path and (b) wire length quality curves for various `timing_tradeoff` = 0.9, illustrating the dominance of directed moves versus simple moves alone.

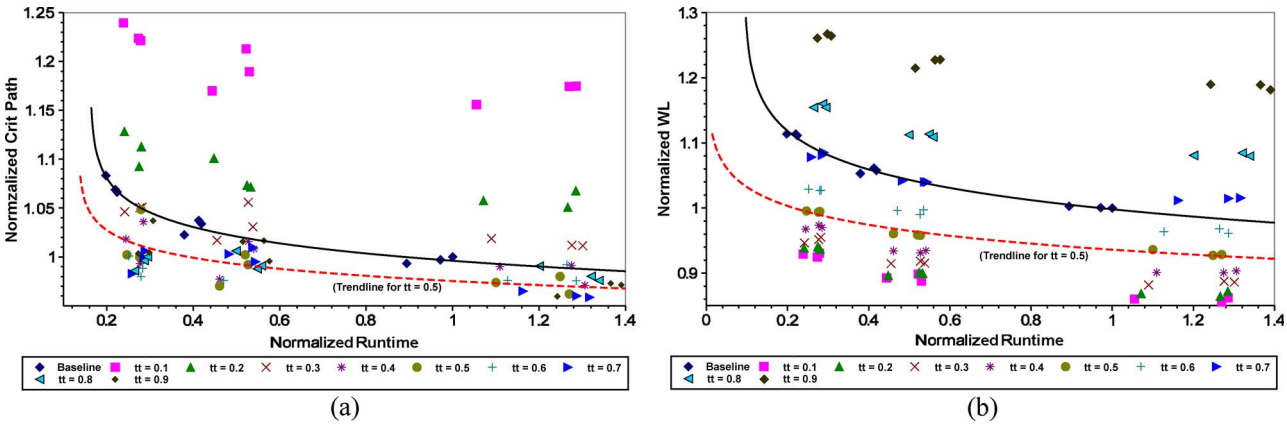

Fig. 9. (a) Critical path and (b) wire length quality curves for various `timing_tradeoff` parameters on a one BLE/CLB, medium-utilization architecture. The red-dotted trend-line is shown for the typical `timing_tradeoff` of 0.5, while the blue trend-line is shown for the baseline anneal (with a `timing_tradeoff` of 0.5) without directed moves.

2) Comparison of Timing Trade-Offs: Given the improvements to wire length and critical path delay, one might be inclined to conclude that, just by changing the `timing_tradeoff` parameter, one could achieve similar results with simple moves alone. This is **not** the case. The success of directed moves was found to be orthogonal to the design of the cost function—no VPR-like parameters can be changed to achieve the same improvement offered by directed moves.

To validate this assertion, the `timing_tradeoff` parameter was swept (from 0.1 to 0.9) for placements generated by both simple moves and simple moves with directed moves interspersed. The purpose of this test was to confirm that, for the same amount of run-time, directed moves consistently dominated in both wire length and critical path delays versus simple moves alone. For this test, the median (with rippling) move, the MPD move, and random moves were employed at the same time.

In all cases, directed moves were found to dominate in terms of both critical path and wire length. Figs. 7 and 8 provide two examples of the domination offered by directed moves at different ends of the `timing_tradeoff` spectrum. The use of directed moves achieved, on average, better results over the entire benchmark suite in terms of wire length and critical path delays, *irrespective of the* `timing_tradeoff` *parameter in the cost function*.

TABLE III
COMPARISON OF AVERAGE STATISTICAL VARIABILITY IN WIRE LENGTH AND CRITICAL PATH OVER THE BENCHMARK SUITE

	Simple (Baseline)		Simple & Directed		Ratio (vs Baseline)	
	WL	CP	WL	CP	WL	CP
Variance	1.34e05	1.16e-13	1.31e05	9.91e-14	0.98	0.85
Std Dev	3.29e02	3.08e-07	3.17e02	2.85e-07	0.96	0.92

The values of `timing_tradeoff` were also swept (from 0.1 to 0.9) to determine the best overall quality versus run-time tradeoff. The normalized results, acquired on a one BLE/CLB, medium-utilization architecture (and computed with respect to the baseline annealer using `timing_tradeoff` = 0.5, `inner_num` = 5) are shown in Fig. 9. In general, the best balance between wire length and critical path delay was offered by `timing_tradeoff` = 0.5. Using this value (shown by the trend-line), the combination of directed moves achieved a 2% improvement in critical path delay and a 5% improvement in wire length, at the baseline algorithm's nominal (1.0x) run-time.

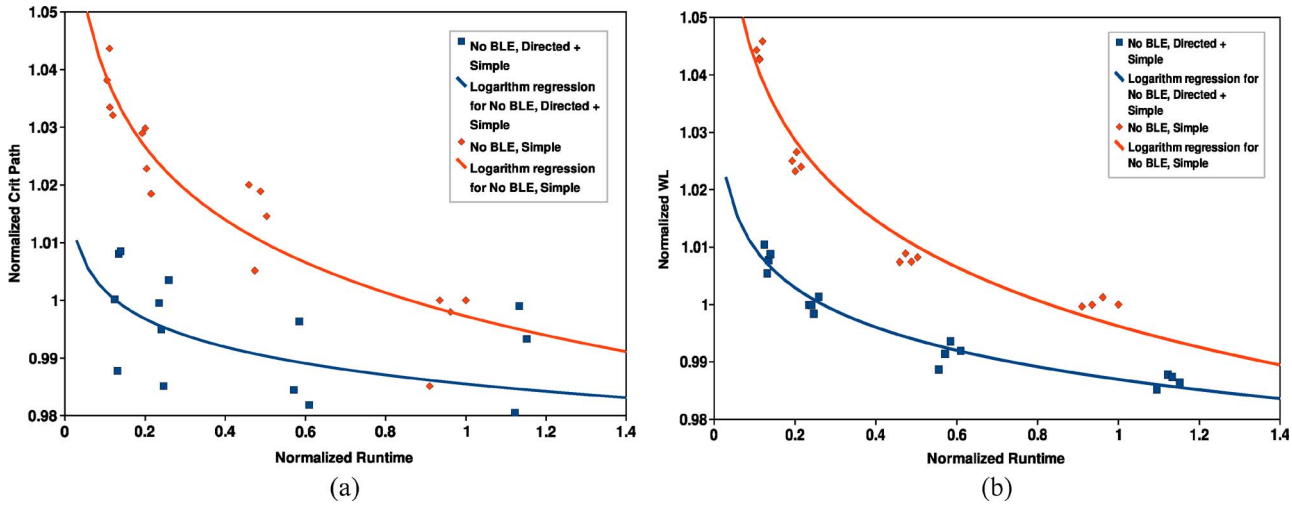

Fig. 10. (a) Critical path and (b) wire length quality curves for four BLE/CLB medium-utilization architectures with directed moves applied on the CLB-level netlist.

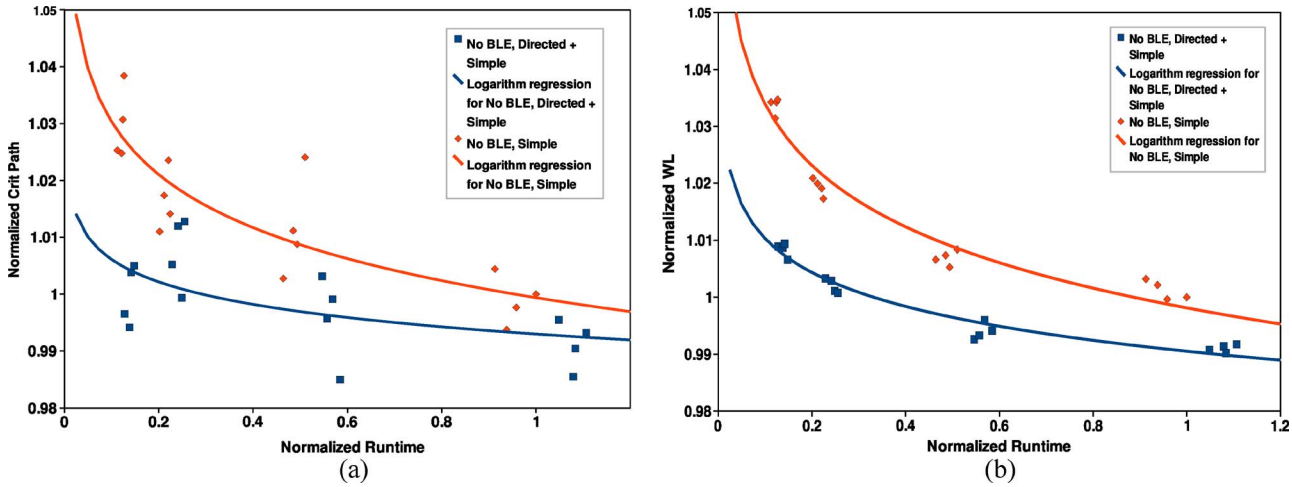

Fig. 11. (a) Critical path and (b) wire length quality curves for eight BLE/CLB medium-utilization architectures with directed moves applied on the CLB-level netlist.

3) Statistical Variance Measures: To measure statistical variability, each design in the suite was placed using 39 different seeds, once with simple moves and again, with both simple and directed moves enabled (using a mixture of simple, median placement with rippling, and MPD minimizing moves). The averages of the variance and standard deviation of the timing costs and bounding box costs are presented in Table III. In general, directed moves decreased the variance of the placements and tended to make results more repeatable.

TABLE IV
SUMMARY OF THE RESULTS FOR HIGH-UTILIZATION CLUSTERED ARCHITECTURES WITH DIRECTED MOVES APPLIED ON THE CLB-LEVEL NETLIST (NEGATIVES INDICATE IMPROVEMENT)

CPU Ratio	Architecture	WL	CP
$\approx 0.25\times$	4 BLE/CLB	-1.5%	-1.5%
$\approx 0.50\times$	4 BLE/CLB	-1%	-1%
$\approx 1.0\times$	4 BLE/CLB	-0.5%	-0.5%
$\approx 0.25\times$	8 BLE/CLB	-1%	-1%
$\approx 0.50\times$	8 BLE/CLB	-0.5%	-0.5%
$\approx 1.0\times$	8 BLE/CLB	$\approx 0\%$	$\approx 0\%$

C. Clustered Architectures

Directed moves have also been tested on architectures with more than one BLE/CLB. (Clustered architectures were not considered in [10].) For these tests, the directed moves were chosen as a combination of simple, median placement with rippling, and MPD minimizing moves, as these had shown the greatest promise in earlier testing.

1) CLB-Level Moves: Directed moves were first tested on clustered designs (that is, using CLB-level netlists) without moving BLEs. Figs. 10 and 11 show that directed moves work well on CLB-level netlists on medium-utilization architectures. This is a straightforward application of the technique from one BLE/CLB architectures, and the dominance trends are similar. The results for high-utilization architectures are summarized in Table IV. (The amount of improvement quoted in these tables represents an approximate reading of the differences in the trend-lines of the dominance charts at the stated runtime points.) Overall, the quality improvement offered by CLB-oriented directed moves for the high-utilization clustered circuits was small. This is due to the small size of the clustered

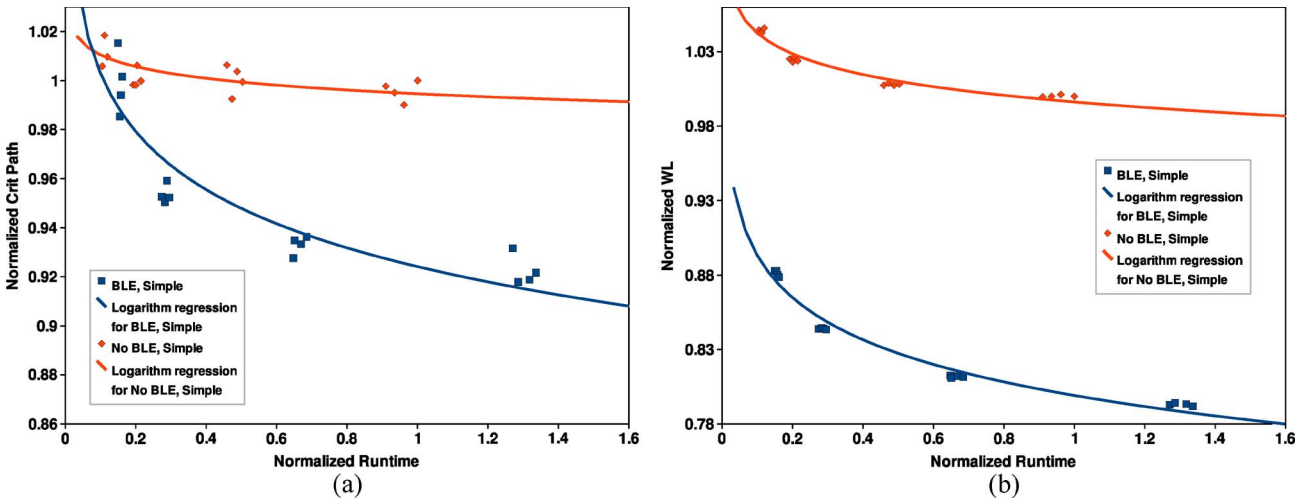

Fig. 12. (a) Critical path and (b) wire length quality curves for a four BLE/CLB medium-utilization architecture without directed moves but with BLE-level moves.

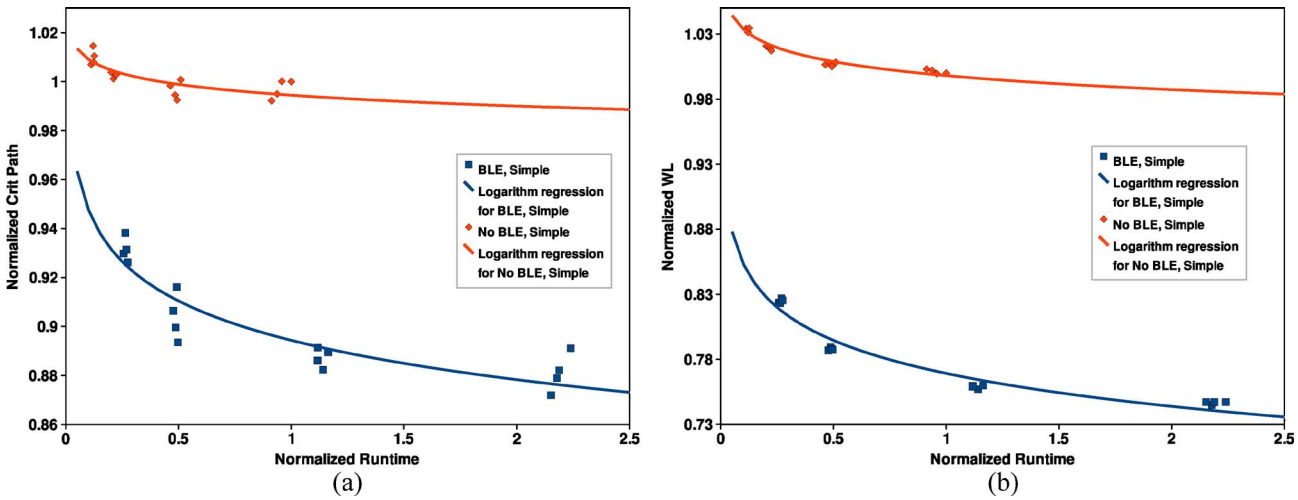

Fig. 13. (a) Critical path and (b) wire length quality curves for an eight BLE/CLB medium-utilization architecture without directed moves but with BLE-level moves.

designs and the fact that the baseline annealer does a good job of exploring the search space for such small circuits, so there is less opportunity for improvement.

2) BLE-Level Moves: In testing clustered architectures, BLE moves were required to achieve the quality expected from a modern placer [25]; such quality could not be achieved using the traditional pack-then-place flow from VPR. BLE moves are conceptually similar to the CLB-level operations performed by annealers like VPR except that they place BLEs instead of CLBs. One consequence of performing BLE-level operations is that additional design-rule constraint (DRC) checking and bookkeeping are required; these can be run-time intensive. Thus, to maintain reasonable run-times, fewer BLE moves are typically performed than CLB moves. At the same time, the acceptance rate of BLE moves tends to be smaller because they may not only be rejected by the change in cost function, but also by the DRC checking.

In KPF, the BLE moves were implemented after the anneal in a greedy zero-temperature pass using move effectiveness as the termination criterion. While this differs from the implementation in SCPlace, it was the intent of this paper to establish a reasonable baseline against which the success of directed moves could be measured and not to address the question of where to perform BLE-level operations.

TABLE V
SUMMARY OF THE RESULTS FOR HIGH-UTILIZATION CLUSTERED ARCHITECTURES WITHOUT DIRECTED MOVES BUT WITH BLE-LEVEL MOVES (NEGATIVES INDICATE IMPROVEMENT)

CPU Ratio	Architecture	WL	CP
$\approx 0.25\times$	4 BLE/CLB	-17%	-2%
$\approx 0.75\times$	4 BLE/CLB	-19%	-4%
$\approx 1.0\times$	4 BLE/CLB	-20%	-5%
$\approx 0.25\times$	8 BLE/CLB	-20%	-5%
$\approx 1.0\times$	8 BLE/CLB	-25%	-7%
$\approx 2.0\times$	8 BLE/CLB	-26%	-9%

To justify the use of a zero-temperature BLE-level phase, tests were performed on both four and eight BLE/CLB architectures *without directed moves*. Figs. 12 and 13 summarize the results with and without BLE-level moves for medium-utilization architectures, while the results from high-utilization architectures are summarized in Table V. (The amount of improvement quoted in this table represents an approximate

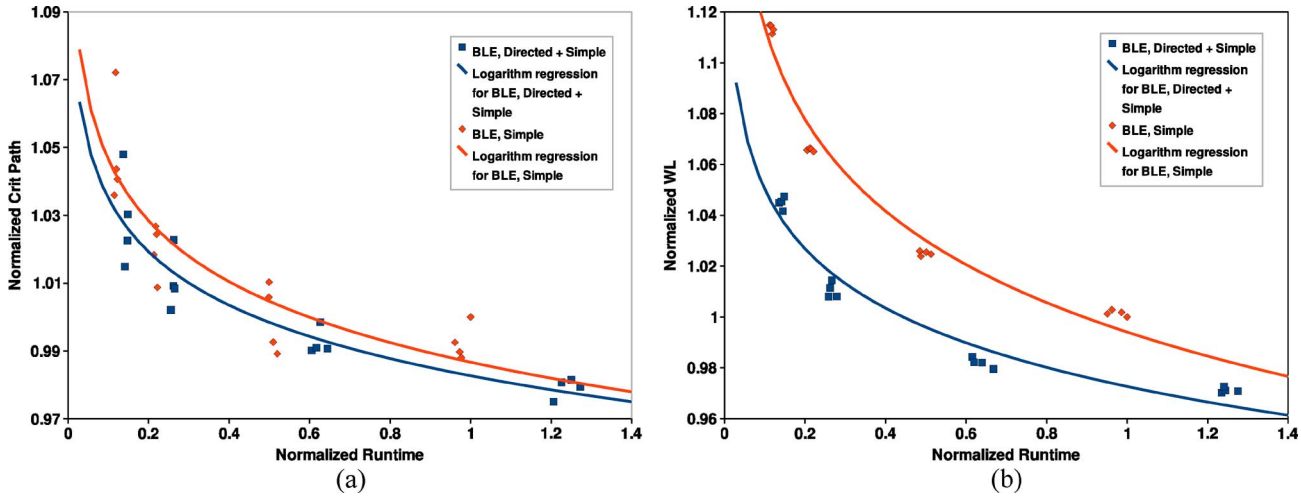

Fig. 14. (a) Critical path and (b) wire length quality curves for a four BLE/CLB architecture with BLE-level moves and directed moves applied on both the CLB- and BLE-level netlists.

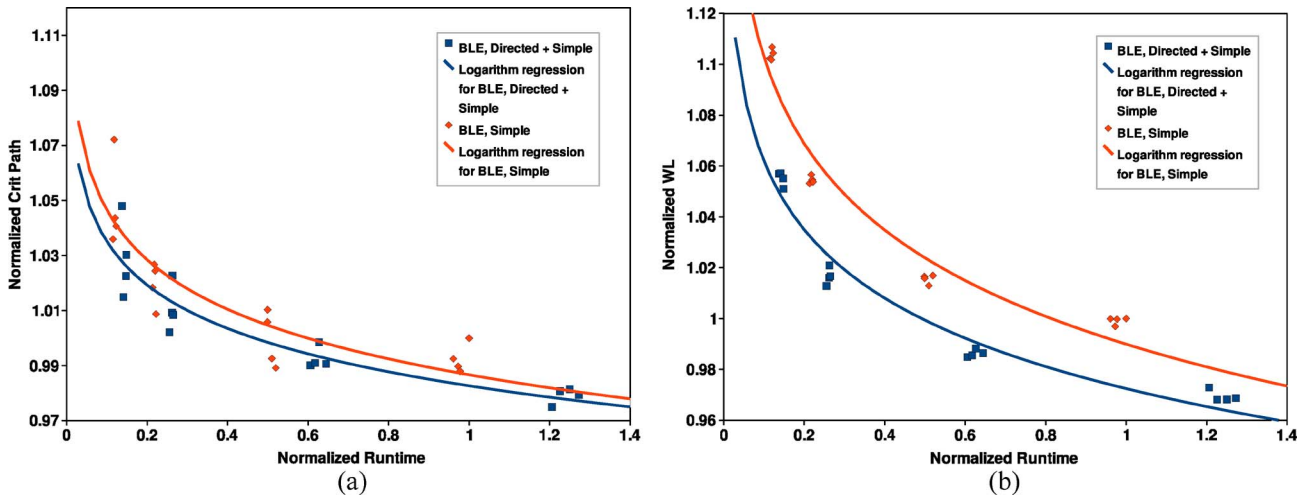

Fig. 15. (a) Critical path and (b) wire length quality curves for an eight BLE/CLB architecture with BLE-level moves and directed moves applied on both the CLB- and BLE-level netlists.

reading of the differences in the trend-lines of the dominance charts at the stated run-time points.)

As expected, the wire length and critical path delays improved with more clustered architectures. The overall improvement (in the eight BLE/CLB medium-utilization case) was a reduction of 25% in wire length and 11% in critical path delay at the nominal run-time, and this is in-line with the amount of improvement reported in [25] for the contribution due to BLE operations during annealing. Hence, it was felt that this implementation offered a reasonable basis for investigating directed moves in clustered architectures.

3) BLE-Level Moves With Directed Moves: The ability for directed moves to work in conjunction with BLE-level moves was also tested. For this test, directed moves were used during both the CLB- and BLE-level phases of the placement and compared to the baseline (where CLB- and BLE-level moves were employed *without* directed moves). The results are shown in Figs. 14 and 15. In general, the directed moves maintained dominance, but the amount of improvement was smaller than without BLE-level moves. Directed moves offered improvement of 4% in wire length and 1% in critical path delay at the 0.2x run-time point, but the improvement at the nominal run-time point was only 2% in wire length and little change in critical path delay. It is believed that this difference may be attributable, in part, to the size of the MCNC benchmarks: in architectures of four or eight BLEs per CLB, the designs are small by modern standards and may not have as much room for QoR improvement once BLE moves are applied, since most of the search space may be explored by the default annealing schedule. The results for high-utilization architectures, summarized in Table VI dominance trends, albeit with less overall improvement.

V. Conclusion

This paper described a means of augmenting a traditional, VPR-like FPGA annealer using the concept of directed moves. Multiple types of moves were described, and results were presented that showed significant improvements over simple, random moves for the same amount of work.

TABLE VI
SUMMARY OF THE RESULTS FOR HIGH-UTILIZATION CLUSTERED ARCHITECTURES WITH BLE-LEVEL MOVES AND DIRECTED MOVES (NEGATIVES INDICATE IMPROVEMENT)

CPU Ratio	Architecture	WL	CP
$\approx 0.25\times$	4 BLE/CLB	-3%	-2%
$\approx 0.5\times$	4 BLE/CLB	-2%	-1%
$\approx 1.0\times$	4 BLE/CLB	-1%	-0.5%
$\approx 0.25\times$	8 BLE/CLB	-2%	-0.5%
$\approx 0.5\times$	8 BLE/CLB	-1%	-0.5%
$\approx 1.0\times$	8 BLE/CLB	-0.5%	$\approx 0\%$

Moreover, it was shown that directed moves are useful in (realistic) devices with lower utilization. This paper described how directed moves can reduce the statistical variability in placements, which can lead to more repeatable results. Furthermore, it was established that the benefits of directed moves cannot be achieved by changing the annealer's cost function. A new approach for BLE operations was described, and a technique for measuring move effectiveness was also proposed.

In the future, the authors intend to investigate the usefulness of directed moves on more realistic architectures with RAMs and IP blocks, as these were not considered in this paper. Furthermore, there may be value in investigating additional types of moves for power reduction or other placement objectives such as congestion minimization.

REFERENCES

[1] C. Mulpuri and S. Hauck, "Runtime and quality tradeoffs in FPGA placement and routing," in *Proc. FPGA*, 2001, pp. 29–36.

[2] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Field-Programmable Logic and Applications*, W. Luk, P. Y. Cheung, and M. Glesner, Eds. Berlin, Germany: Springer-Verlag, 1997, pp. 213–222.

[3] R. F. Hentschke and R. A. d. L. Reis, "Improving simulated annealing placement by applying random and greedy mixed perturbations," in *Proc. Integr. Circuits Syst. Des.*, 2003, pp. 267–272.

[4] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003.

[5] K. Vorwerk, A. Kennings, and A. Vannelli, "Engineering details of a stable analytic placer," in *Proc. ICCAD*, 2004, pp. 573–580.

[6] K. Doll, F. M. Johannes, and K. J. Antreich, "Iterative placement improvement by network flow methods," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 10, pp. 1189–1200, Oct. 1994.

[7] M. Hrkic, J. Lillis, and G. Beraudo, "An approach to placement-coupled logic replication," in *Proc. DAC*, 2004, pp. 711–716.

[8] G. Beraudo and J. Lillis, "Timing optimization of FPGA placements by logic replication," in *Proc. DAC*, 2003, pp. 196–201.

[9] G. Chen and J. Cong, "Simultaneous timing-driven placement and duplication," in *Proc. FPGA*, 2005, pp. 51–59.

[10] K. Vorwerk, A. Kennings, J. Greene, and D. T. Chen, "Improving annealing via directed moves," in *Proc. FPL*, 2007, pp. 363–370.

[11] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," in *Proc. Conf. Decision Control*, 1985, pp. 761–767.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[13] E. C. Segura, "Evolutionary computation with simulated annealing: Conditions for optimal equilibrium distribution," *J. Comput. Sci. Technol.*, vol. 5, no. 4, pp. 178–182, Dec. 2005.

[14] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," *IEEE J. Solid-State Circuits*, vol. SSC-20, no. 2, pp. 510–522, Apr. 1985.

[15] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 3, pp. 349–359, Mar. 1995.

[16] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proc. DAC*, 1995, pp. 211–215.

[17] Actel Corporation, *ProASIC3 flash family FPGAs,* 2007. [Online]. Available: http://www.actel.com/documents/PA3_DS.pdf

[18] R. Tessier, "Fast placement approaches for FPGAs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 7, no. 2, pp. 284–305, Apr. 2002.

[19] P. Maidee, C. Ababei, and K. Bazargan, "Fast timing-driven partitioning-based placement for island style FPGAs," in *Proc. DAC*, 2003, pp. 598–603.

[20] D. T. Chen, K. Vorwerk, and A. Kennings, "Improving timing-driven FPGA packing with physical information," in *Proc. FPL*, 2007, pp. 117–123.

[21] A. Singh and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction in FPGAs," in *Proc. FPGA*, 2002, pp. 59–66.

[22] J. Cong and M. Romesis, "Performance-driven multi-level clustering with application to hierarchical FPGA mapping," in *Proc. DAC*, 2001, pp. 389–394.

[23] C. Sze, T.-C. Wang, and L.-C. Wang, "Multilevel circuit clustering for delay minimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 7, pp. 1073–1085, Jul. 2004.

[24] M. Dehkordi and S. Brown, "Performance-driven recursive multi-level clustering," in *Proc. FPL*, 2003, pp. 262–269.

[25] G. Chen and J. Cong, "Simultaneous timing driven clustering and placement for FPGAs," in *Proc. FPL*, 2004, pp. 158–167.

[26] Y. Zhuo, H. Li, Q. Zhou, Y. Cai, and X. Hong, "New timing and routability driven placement algorithms for FPGA synthesis," in *Proc. GLSVLSI*, 2007, pp. 570–575.

[27] T. Kong, "A novel net weighting algorithm for timing-driven placement," in *Proc. ICCAD*, 2002, pp. 172–176.

[28] K. Shahookar and P. Mazumder, "VLSI cell placement techniques," *ACM Comput. Surv.*, vol. 23, no. 2, pp. 143–220, Jun. 1991.

[29] A. Ludwin, V. Betz, and K. Padalia, "High-quality, deterministic parallel placement for FPGAs on commodity hardware," in *Proc. FPGA*, 2008, pp. 14–23.

[30] P. K. Chan and M. D. F. Schlag, "Parallel placement for field-programmable gate arrays," in *Proc. FPGA*, 2003, pp. 43–50.

[31] J. D. Vicente, J. Lanchares, and R. Hermida, "Annealing placement by thermodynamic combinatorial optimization," *ACM Trans. Des. Autom. Electron. Syst. (TODAES)*, vol. 9, no. 3, pp. 310–332, Jul. 2004.

[32] B. V. Cherkassky and A. V. Goldberg, "On implementing the push-relabel method for the maximum flow problem," *Algorithmica*, vol. 19, no. 4, pp. 390–410, 1997.

[33] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. Circuits Syst.*, vol. CAS-28, no. 1, pp. 12–18, Jan. 1981.

[34] H. S. Tim, "A new standard cell placement program based on the simulated annealing algorithm," M.S. thesis, Dept. Elect. Eng. and Comput. Sci., Univ. California, Berkeley, Mar. 1988.

[35] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectron. Center North Carolina, Research Triangle Park, NC, 1991. Tech. Rep.

Kristofer Vorwerk (S'02–M'04–S'04) received the B.A.Sc. and M.A.Sc. degrees in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002 and 2004, respectively, where he is currently working toward the Ph.D. degree under the supervision of Dr. Andrew Kennings.

He is currently a Staff Software Engineer with Actel Corporation, Mountain View, CA. His research interests include combinatorial optimization and very large scale integration physical design, with particular emphasis on placement and synthesis.

Andrew Kennings received the B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1992, 1994, and 1997, respectively.

He is an Associate Professor of electrical and computer engineering with the University of Waterloo. His research interests are in the general area of very large scale integration computer-aided design and programmable devices.

Jonathan W. Greene (S'80–M'84) received the B.Sc. degree in biology from Brown University, Providence, RI, in 1979 and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1984.

He was with the LSI Logic Systems Research Laboratory in 1984, and Actel Corporation in 1986. In 1989, he entered the field of computer-aided pharmaceutical design, first as a Visiting Scholar with Columbia University, New York, NY, then with BioCAD, CombiChem Inc., and Cambios Computing. He rejoined Actel in 2003, where he is currently a Fellow working in the area of architectures and design algorithms for field-programmable logic circuits. He is a coauthor of over 40 papers and 17 patents.