# An Integer Programming Placement Approach to FPGA Clock Power Reduction

Alireza Rakhshanfar
Dept. of ECE, University of Toronto
Toronto, ON Canada
e-mail: ali.rakhshanfar@utoronto.ca

Jason H. Anderson
Dept. of ECE, University of Toronto
Toronto, ON Canada
e-mail: janders@eecg.toronto.edu

**Abstract— Clock signals are responsible for a significant portion of dynamic power in FPGAs owing to their high toggle frequency and capacitance. Clock signals are distributed to loads through a programmable routing tree network, designed to provide low delay and low skew. The placement step of the FPGA CAD flow plays a key role in influencing clock power, as clock tree branches are connected based solely on the placement of the clock loads. In this paper, we present a placement-based approach to clock power reduction based on an integer linear programming (ILP) formulation. Our technique is intended to be used as an optimization post-pass executed after traditional placement, and it offers fine-grained control of the amount by which clock power is optimized versus other placement criteria. Results show that the proposed technique reduces clock network capacitance by over 50% with minimal deleterious impact on post-routed wirelength and circuit speed.**

## I. INTRODUCTION

Society today is in the midst of a mobile electronics revolution, with an enthusiastic public embracing devices such as the Blackberry, iPhone, netbook, and the iPad. The power consumption of field-programmable gate arrays (FPGAs) excludes them from this revolution, and in fact, power stands as a key barrier to PLD market growth. The reason for this is an FPGA's inherent programmability. The overhead associated with programmability comes at a high energy cost and recent work suggests that dynamic power is 7-14× higher in FPGAs compared with custom standard cell ASICs for implementing a given logic function [1]. Significant reductions are needed in FPGA power to close the gap with ASICs, and to enable their use in the types of mobile devices that are transforming modern society.

FPGA power reduction has been an active research area in recent years, with a variety of techniques being proposed to reduce power at the CAD, architecture and circuit levels. Dynamic power dissipation in CMOS circuits can be modeled as $P = \sum_{i \in nets} f_i \cdot C_i \cdot V^2$, where $f_i$ represents the toggle rate of signal $i$ (i.e. $i$'s switching activity), $C_i$ represents the capacitance of signal $i$, and $V$ represents supply voltage. Clock power is a major source of power dissipation in FPGAs. Naturally, clocks have very high fanout (high capacitance) and toggle at a high frequency (high switching activity). Prior work has shown that clocks comprise from 20-39% of power consumption in commercial FPGAs [2, 3]. In this paper, we offer a CAD technique to reduce dynamic power dissipated in the FPGA clock network.

Clock signals in FPGAs are distributed to loads (i.e. flip-flops and sequential hard-IP blocks) through a pre-fabricated programmable routing tree network. This is in constrast to custom ASICs where the clock tree is "custom-built" by the router based on the placement of clock loads. In FPGAs, programmable switches within the tree are turned on as needed to feed the clock signal to portions of the clock tree. Meaning that clocks are *only* distributed to portions of the tree where placed loads require them. The power consumption of the clock network can therefore be reduced through a careful placement of clock loads, thereby lowering the number of clock tree wire segments that are needed to route the clock signals.

In this paper, a placement-based optimization technique is proposed to reduce routing capacitance of the FPGA clock network. The clock power reduction problem is formulated as an integer linear program (ILP), executed as an optimization pass *after* traditional core logic placement is complete. To our knowledge, ours is the first work to apply ILP to FPGA power reduction.

We believe our work is particularly suitable when clock power cannot be directly incorporated into the placer's objective function, as is the case when analytical (e.g. non-iterative) placement techniques are used for the core placement algorithm, such as FastPlace [4]. Analytical placers formulate the placement problem mathematically as a system of equations to be solved, and require a continuous and differentiable objective function. Measuring FPGA clock resource usage is a discrete problem, and therefore difficult to optimize within an analytical placement algorithm. While simulated annealing has been used for FPGA placement in the past, it scales poorly with design size, and analytical approaches exhibit superior scalability [5]. There is a need, then, for clock power optimization approaches such as ours that can operate outside of the annealing context. Notably, the Xilinx commercial FPGA placer is based on analytical techniques [6], and we therefore expect the proposed techniques will be of interest to the FPGA vendors.

The remainder of this paper is organized as follows: Section II describes related work on clock power reduction in FPGAs and reviews integer linear programming. Section III introduces the proposed clock power reduction scheme. An experimental study is presented in Section IV. Conclusions and suggestions for future work appear in Section V.

## II. BACKGROUND

### A. Clock Network Model

We use the clock distribution network model shown in Fig. 1, inspired by both Xilinx and Altera commercial FPGAs [7, 8]. The FPGA chip is divided into clock regions for the purposes of clock signal distribution. We assume that each quadrant of the chip corresponds to a clock region. Clock routing resources are organized as a tree structure to minimize clock skew. At the top of the tree are vertical root spines that feed the clock signals to the horizontal spines in each region. The horizontal spines drive vertical half-spines within the regions, which ultimately deliver clock signals to logic blocks.

In the commercial FPGA context, the Xilinx Virtex-5 65 $n$m family of FPGAs [8] has 32 top-level root spines that deliver clock signals to regions. Virtex-5 regions are 20 logic block tiles in height and span half of the device width. At most 10 of the 32 clock signals can be driven into each region and any of these 10 signals can be routed to a logic block in the region. Altera Stratix-III FPGAs [7] have quadrant-oriented regions, similar to the model assumed here. Note that while we use the model of Fig. 1 in this paper, our approach applies equally well to other clock routing architectures.

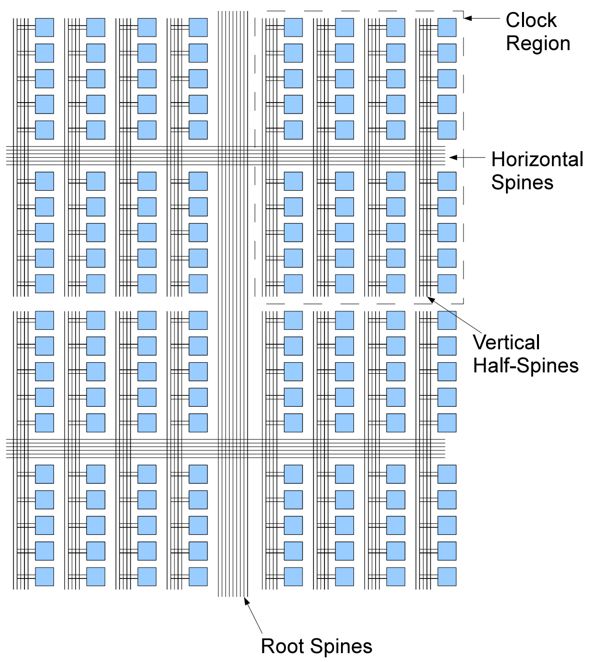

Fig. 1. Clock distribution network architecture model.

### B. Clock Power Reduction in FPGA Placement

Several recent works have considered reducing clock power during the placement phase of the CAD flow. Lamoureux and Wilton incorporated clock power into simulated annealing-based placement. They designed a cost function that aimed to reduce the clock capacitance within a region and also reduce the number of regions spanned by a clock signal [9]. Actel and Xilinx considered clock power in the industrial context in two recent papers and showed promising results [10, 11]. Both industrial works reduce the number of used clock spines through a cost function in an iterative annealing-style placer framework.

In our recent related work, we proposed incorporating clock gating capabilities at different points within the clock distribution network [13]. Enable pins were added to the programmable switches within the clock tree, permitting the clock signal to be temporarily shut off to certain logic blocks on the device. A placer cost function was designed to locate logic blocks with shared clock enables in a manner favourable to the underlying clock gating architecture.

The approach proposed in this paper has several differences relative to the prior approaches. In prior works, clock power was incorporated as a term in a composite placer objective function and then optimized within an iterative placement framework (e.g. simulated annealing). In such approaches, it is problematic to attach the correct "weight" to the clock power term, and effectively manage the impact of clock power on other placement criteria. In this paper, our approach uses formal techniques to reduce clock power as a post-pass placer optimization. We incorporate flexibility into the formulation permitting one to directly control the amount by which clock power reduction can affect other placer metrics. Moreover, unlike simulated annealing-based placement, which is heuristic and non-deterministic, our approach can be applied to find the minimal (optimal) number of clock resources for a region. Our approach bears some similarity to a very recent work by Chakraborty and Pan who attacked clock power in a structured ASIC context [12].

### C. Integer Linear Programming

Linear programming (LP) is a formal optimization technique with broad application in CAD and other areas (such as operations research). In LP, the objective function is a linear function of the variables. Linear equality and inequality constraints on the variables may also be specified in the formulation. LP programs with real-valued variables can be solved efficiently in polynomial time. However, if the LP variables are required to be integers, then the LP problem is called an integer linear program (ILP). ILP problems generally fall into the NP-hard complexity class. Despite its worst-case complexity, ILP can be applied successfully to many practical problems, with a recent example being the constrained placement of I/Os in FPGAs with multiple signaling standards [14].

In this work, we cast the clock power reduction problem as an ILP problem and use a publicly-available LP solver to solve the formulation [15].

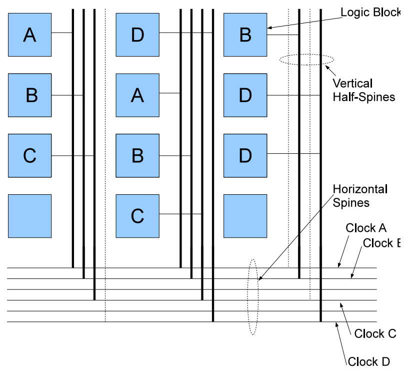

Fig. 2. Initial placement of logic blocks requiring 4 clock signals.

## III. Clock Power Reduction in Placement

In the clock network model used in this paper (Fig. 1), vertical half-spines are the final branches in the clock network that deliver the clock signals to logic blocks. A clock signal is *only* routed from a horizontal spine to a vertical half-spine if there exists at least one logic block requiring that signal in the column adjacent to the vertical half-spine. Clock signals of any reasonable fanout will have loads spanning across many (possibly even most) columns, and thereby require many vertical half-spines. In such cases, the clock network capacitance is mainly determined by the number of vertical half-spines needed to route to each of the clock network loads.

The key idea in our approach (as in [11, 10]) is to place the load blocks of clock signals across as few columns as possible, thereby reducing the number of vertical half-spines required in the clock routing. To illustrate, consider the example placement shown in Fig. 2. In the example, there are 4 clock signals: $A$, $B$, $C$ and $D$. Logic blocks are labelled with the clock signal they must receive, for example, a block labeled $A$ must receive clock signal $A$. Observe that there are 3 logic blocks requiring clock signals $A$, $B$ and $C$, and two blocks requiring clock signal $D$. In the placement of Fig. 2, three vertical spines must be used to route the necessary clock signals in the left-most column. In the second and third columns, 4 and 2 vertical half-spines are needed to route the required clock signals, respectively. A total of 9 vertical spines are needed for the placement.

By adjusting the placement of logic blocks, we can reduce the number of vertical half-spines required to route clock signals. Fig. 3 shows an optimized placement of the blocks in Fig. 2. In the optimized placement, all of the blocks using $A$ and $C$ have been moved to the first column; those blocks using clocks $B$ and $D$ have been moved to columns 2 and 3, respectively. The optimized placement uses 4 vertical half-spines, whereas the original placement required 9 – a significant reduction in clock capacitance.

The input to our approach is a placement that has presumably been optimized according to the traditional metrics of wirelength and tim-

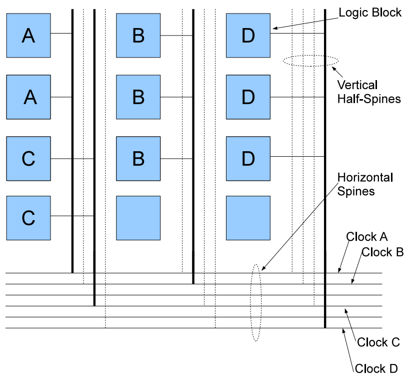

Fig. 3. Clock power-optimized placement of logic blocks.

ing. It is desirable to minimize the disruption to wirelength and timing when altering the placement for the purpose of clock power minimization. With this in mind, our approach incorporates a maximum allowable manhattan move distance for each logic block relative to its original location. Clock power is optimized subject to such move distance restrictions. In Fig. 3, each logic block was permitted to move a distance of at most one from its original placement in Fig. 2. One can expect that with increasing maximum move distances, more clock power reduction will be possible, at the expense of increased wirelength and critical path delay.

## *A. ILP Formulation*

Given a completely placed design (for example, as produced by VPR [16] or any other placement algorithm), we optimize clock power on a region-by-region basis. Specifically, we formulate and solve a separate ILP problem for each clock region, computing a new placement for the blocks placed within that region. The ILP problems for each region are independent from one another and can be solved in parallel, making our approach well-suited to implementation on a multi-core processor. We begin by defining the key variables used in our ILP formulation:

- $clks$: the set of all clock signals that are required in the clock region.
- $cols$: the set of all columns in the clock region.
- $CS_{i,j}$: takes on a value of 1 if a vertical half-spine is required in column $j$ to route clock signal $i$; 0 otherwise.

The objective function of our formulation is the number of vertical half-spines spanned by clocks in the region. Each time a clock signal is used in a column of the region, a vertical half-spine is required to deliver the signal to that column. Consequently, we seek to minimize the number of columns spanned by the loads of each clock signal in the region:

$$\text{minimize } \phi = \sum_{i \in clks,\ j \in cols} CS_{i,j} \tag{1}$$

To compute the $CS$ value in (1) for each clock signal and column, we introduce several additional variables:

- $locs$: the set of all locations in the clock region that logic blocks can be placed in.
- $blk_{m,n}$: takes on a value of 1 if block $m$ is placed in location $n$; 0 otherwise.
- $mov\ loc_m$: the set of all locations that block $m$ can be placed in, subject to the allowable manhattan movement distance.
- $blks_{clk_i}$: the set of all blocks that require clock signal $i$ in the clock region.
- $\alpha$: a large integer (1000 was used in our experiments).
- $col_j$: the set of all locations in column $j$.
- $orig_i$: the original location of block $i$ in the input placement.

Using the variables above, we solve to find the minimum value for $\phi$ in (1) subject to the following constraints (which are described in detail below):

$$\forall \{m \in blks\} : \left( \sum_{n \in mov\ loc_m} blk_{m,n} \right) = 1 \text{ (placement)} \tag{2}$$

$$\forall \{n \in locs\} : \left( \sum_{m \in blks} blk_{m,n} \right) \leq 1 \text{ (exclusivity)} \tag{3}$$

$$\forall \{j \in cols,\ i \in clks\} : \left( \sum_{m \in blks_{clk_i}} \sum_{n \in col_j} blk_{m,n} \right) \leq \alpha \cdot CS_{i,j} \text{ (spine count)} \tag{4}$$

$$\forall \{i \in clks, j \in cols\} : CS_{i,j} \geq 0 \tag{5}$$

$$\forall \{m \in blks, n \in locs\} :\ 0 \leq blk_{m,n} \leq 1 \tag{6}$$

Constraint (2) (placement) ensures that each logic block is placed in exactly one location within its permissible movement window. Constraint (3) (exclusivity) is required to produce a feasible non-overlapping placement: each location within the clock region can accommodate at most a single logic block.

Constraint (4) deserves a detailed elaboration, as this constraint is responsible for setting the values of the $CS_{i,j}$ variables that are directly used in the objective function (1). The idea is that we want $CS_{i,j}$ to be set to 1 when clock signal $i$ is needed by a logic block placed in column $j$. Constraint (4) is established for each clock signal ($i$) and column ($j$) in the region. For each such clock signal and column pair combination, the double summation counts the number of blocks placed on column $j$ that require signal $i$. If the computed summation is at least one, then constraint (4) dictates that $CS_{i,j}$ must be at least 1; and, signal $i$ is needed on column $j$ and a vertical half-spine will be required in the clock routing of $i$. The right-hand side of the inequality in (4) multiplies $CS_{i,j}$ with $\alpha$, where $\alpha$ is a large positive quantity. The objective function (1) seeks to minimize the sum of $CS$ values, and consequently, constraint (4) will be met in a way that minimizes the $CS_{i,j}$ values. In particular, $CS_{i,j}$ will be set to *exactly* 1 only for those cases where a load of clock $i$ is used in column $j$; otherwise, $CS_{i,j}$ will take on value 0.

Finally, constraints (5) and (6) ensure that the variables are binary.

## *B. Maintaining Placement Quality*

We initially experimented with the above formulation and found that while it was successful at minimizing clock capacitance, it also moved logic blocks unnecessarily from their original placement locations. In particular, we observed there to be multiple solutions to the ILP problem. The above formulation encodes no preference for any one solution over another. In general, across the different solutions, it is preferable to choose a solution that minimizes the damage to traditional placement critera – wirelength and timing. To achieve this, we consider an augmented objective function, where we introduce an *anchoring term* that encourages logic blocks to remain in their original placement positions, where possible:

$$\begin{aligned} \text{minimize } \phi = \sum_{i \in clks,\ j \in cols} CS_{i,j} + \\ 0.1 \cdot \sum_{m \in blks,\ n \in mov\ loc_m,\ n \neq orig_i} blk_{m,n} \end{aligned} \quad (7)$$

The anchoring term (the right-hand operand of the + in (7)) counts the number of logic blocks placed in locations other than their original position. The term is weighted by a small scalar constant (0.1), reflecting its tie-breaker role in (7). In our experimental study below, we consider the formulation both with and without the anchoring term.

## C. Managing Run-Time

We control run-time in two different ways, the first being what we refer to as the *optimization window size*. Recall that the target FPGA has four regions (see Fig. 1) and that our approach works on a region-by-region basis. A key observation is that we do not need to optimize the entire region at once with a single ILP problem instance. Instead, we can consider a *window* of columns within the region and formulate the (smaller) ILP problem instance for the window. After optimizing cell placement within a window, we slide the window by one column to a new position, and formulate a new ILP problem instance to be solved. The windowed optimization continues until the entire region is optimized. Since our optimization involves moving blocks between columns, the minimum sensible window size to consider is 2. Large windows will produce larger ILP problem instances, and will likely yield improved quality-of-result. Note that as we slide the window and formulate new ILP problem instances, we ensure that the maximum permissible move distance constraint for each block is honored.

The second way we manage run-time is through a time-out parameter supplied to the ILP solver. One can specify a limit (in seconds) on the solver run-time and if the limit is exceeded, the solver returns the best solution found so far. We experimented with various caps on the total number of seconds permitted for optimization of an entire circuit. We allocate time as follows: Say, for example, we wish to allow 100 seconds to optimize the circuit, we compute the total number of ILP problem instances to be solved, $t$, which depends on the optimization window size. We then specify a time limit to the solver of $100/t$ seconds for solving each ILP problem instance.

## IV. Experimental Study

We implemented the ILP-based clock power optimization within the VPR place and route framework [16] and use the VPR placement as input to our algorithm. We target an FPGA architecture with 10 4-LUT/flip-flop pairs per logic block and length-4 wire segments. To our knowledge, there are no publicly available benchmark circuits that contain multiple clock domains. Consequently, for this investigation, we altered the 20 combinational and sequential benchmark circuits that are most commonly used in FPGA research (i.e. [17]) to contain multiple clock signals. The logic blocks in each circuit were arbitrarily divided into four groups – each corresponding to a different clock domain. Although we synthetically transform the circuits into multi-clock domain circuits, we believe the approach is reasonable enough to demonstrate the utility of our approach and illustrate trade-offs between clock power and other criteria.

For each circuit $cct$, we determined the minimum number of tracks per channel, $W_{min,cct}$, needed to route the circuit in the unmodified VPR (clock unaware). Then, for all experiments conducted, the circuit was routed in an architecture with $1.3 \times W_{min,cct}$ tracks per channel[1]. In this way, we held the routing architecture invariant, allowing us to see the impact of our placement changes alone. The comparative baseline for all experiments is unmodified VPR (which does not optimize clock power).

[1]This is the generally accepted methodology in FPGA architecture research to reflect a medium-stress routing scenario.

## A. Alternative Approach for Comparison

To help us to understand whether the results produced by the proposed ILP approach are reasonable, we implemented our own version of the annealing-based placement approach described in [18][2]. The annealing-based placer in VPR uses a composite cost function that optimizes both wirelength and timing: $Cost = \alpha \cdot WireLength + \beta \cdot Timing$. We extended the annealing cost function to include an additional term which counts the number of used clock spines for the current placement:

$$Cost_{new} = \alpha \cdot WireLength + \beta \cdot Timing + \gamma \cdot SpineCost \quad (8)$$

We implemented the *tilt factor* described in [18] that attempts to steer the annealer towards good solutions as follows: 1) when no logic blocks are using a spine, the cost for the spine is zero; and, 2) when logic blocks are initially added to a spine, the tilt factor cost ramps up quickly; and, 3) when the spine is 1/2 full, the tilt factor cost increases slowly, up to a maximum when the spine is full. A specific tilt factor function was not given in [18]. For a given spine, $s$, we found that the tilt function below works well. Let $m$ represent the number of logic blocks using spine $s$, and $S$ the number of locations available on the spine. The tilt for $s$ is defined to be:

$$\begin{aligned} tilt_s &= 0, \text{ for } m = 0 \\ tilt_s &= 1 + 1.5 \times (m/S), \text{ for } 0 < m \leq 0.5 \times S \\ tilt_s &= 1.5 + 0.5 \times (m/S), \text{ for } m > 0.5 \times S \end{aligned} \quad (9)$$

That is, the tilt cost increases with slope 1.5 as blocks are initially added to the spine, and then increases more gradually (with slope 0.5) when the spine is more than half full. The $SpineCost$ in (8) is set equal to the sum of the tilt cost for each spine in each region.

Note that the comparison with an annealing approach is only to ensure that our method produces reasonable quality results. The main value of our approach is that it can be applied in non-annealing-based placers.

## B. Results

Fig. 4 gives results for our ILP approach for an optimization window size of 2. Part a) of the figure shows the percentage reduction in clock spine count relative to baseline VPR; part b) of the figure shows the percentage increase in post-routed wirelength. The percentages are based on the geometric mean across 20 benchmark circuits. The horizontal axis represents the maximum permissible manhattan move distance for each block. The five bars for each move distance represent five different run-time limits for optimizing the entire circuit: 50, 100, 200, 300 and 400 seconds, respectively. These results correspond to the objective function containing the anchoring term (7).

In Fig. 4 a), we observe that our approach is quite effective in reducing vertical clock spine count. When the permissible move distance is 1, the vertical clock spine count is reduced by 37%, on average. Increasing the permissible move distance to 2 yields a 45% reduction in clock spine count. No significant improvements are observed when the permissible distance is increased beyond 2. Fig. 4 b) shows the increase to the post-routed wirelength of logic signals owing to the clock power optimization (i.e. the increase in the # of wire segments used in the routing). The 37% decrease in clock spine usage (distance 1) is associated with a 3% increase in logic signal wirelength; the 45%

[2]The work in [18] was done at Actel Corp. and as such, the program code is proprietary and not released publicly.

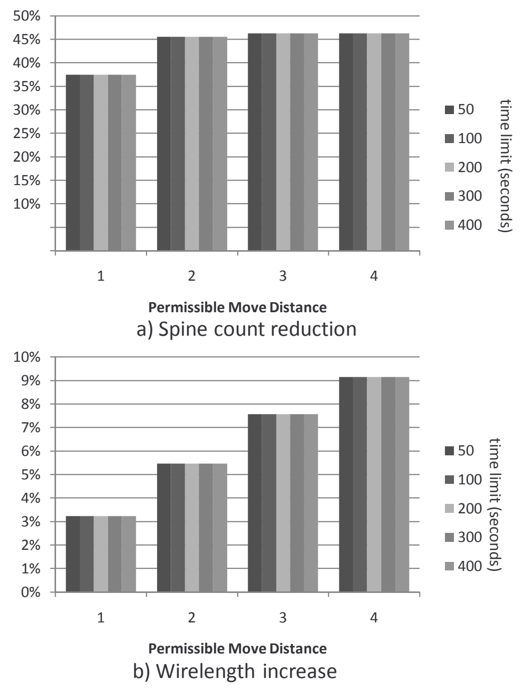

Fig. 4. Spine count and wirelength results for window size 2.

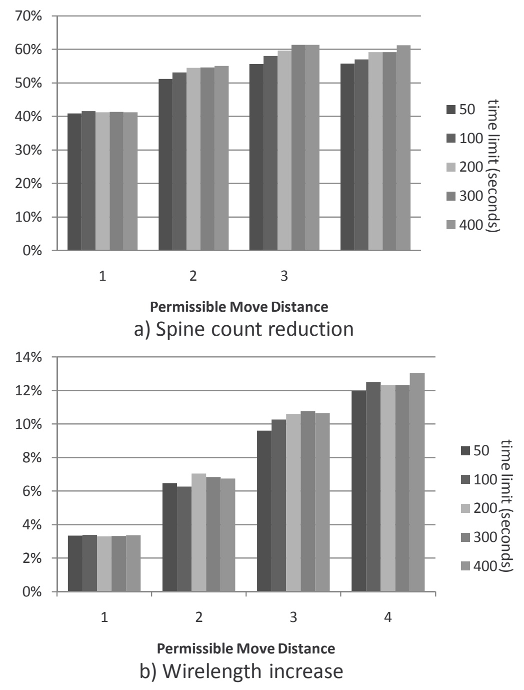

Fig. 5. Spine count and wirelength results for window size 4.

| | Baseline VPR | | VPR + clock power optimization | |
|---|---|---|---|---|
| **Benchmark** | **Clk Spine Count** | **# Wire Segs** | **Clk Spine Count** | **# Wire Segs** |
| alu4 | 73 | 8148 | 34 | 8664 |
| apex2 | 92 | 13021 | 39 | 13758 |
| apex4 | 74 | 9740 | 32 | 10410 |
| bigkey | 66 | 13870 | 32 | 14306 |
| clma | 217 | 64526 | 109 | 68509 |
| des | 78 | 17913 | 42 | 18927 |
| diffeq | 65 | 5578 | 31 | 6340 |
| dsip | 67 | 13056 | 30 | 13733 |
| elliptic | 113 | 18857 | 46 | 20433 |
| ex1010 | 154 | 36901 | 71 | 39430 |
| ex5p | 60 | 9404 | 27 | 10185 |
| frisc | 118 | 22549 | 48 | 24023 |
| misex3 | 73 | 8537 | 34 | 8848 |
| pdc | 163 | 42788 | 70 | 44606 |
| s298 | 97 | 7716 | 45 | 8123 |
| s38417 | 193 | 26605 | 88 | 29296 |
| s38584.1 | 199 | 30842 | 92 | 33685 |
| seq | 86 | 11830 | 40 | 12884 |
| spla | 147 | 28673 | 65 | 30117 |
| tseng | 49 | 4981 | 22 | 5599 |
| **Geomean:** | 99.0 | 15563.9 | 45.1 | 16660.7 |
| **Ratio:** | | | 0.46 | 1.07 |

TABLE I
CIRCUIT-BY-CIRCUIT RESULTS FOR OPTIMIZATION WINDOW SIZE 4, MOVE DISTANCE 2, 200 SECOND TIME LIMIT.

decrease in clock spine usage (distance 2) is associated with a 5.5% increase in logic signal wirelength. Increased logic signal wirelength will lead to higher capacitance and power. The increases in wirelength are, in essence, the "cost" of the reduction in clock spine usage. Below, we attempt to estimate the net power benefit of our approach. Fig. 4 also shows that identical results are achieved for all five run-time limits considered (50, 100, 200, 300 and 400 seconds). With a window size of 2, 50 seconds is more than enough time to optimize the entire placement, as the same results are seen when more time is allowed.

Fig. 5 gives results for an optimization window size of 4. Parts a) and b) of Fig. 5 are analogous to parts a) and b) of Fig. 4. Part a) of Fig. 5 shows that larger spine count reductions are achieved versus when the optimization window size is 2. Spine count reductions are 40%, 52-54%, and 58-61% for permissible move distances of 1, 2 and 3, respectively, relative to baseline VPR. The larger optimization window size (4) allows more freedom for cell movement and a broader view of the optimization space, yielding better clock power results. In Fig. 5 b) we see that coupled with the decreases in clock spine count, are increases in post-routed wirelength. Post-routed wirelength is increased by 3%, 6-7%, and 10-11%, for permissible move distances of 1, 2, and 3, respectively. For a permissible move distance of 4, the clock spine count reductions are no better than with a move distance of 3, and the wirelength is considerably worse. For completeness, Table I gives detailed results for each circuit for a permissible move distance of 2, corresponding to an average spine count reduction of 54% at a cost of 6.8% wirelength increase. The spine count reductions are quite consistent across all circuits, as is the increase in routing wire segment usage. We also ran experiments without the anchoring term in the formulation (i.e. using objective function (1)). Without the anchoring term, wirelength was increased by 9.3% for a permissible move distance of 2 (versus 6.8% with the anchoring term).

In Fig. 5, we observe that the five different run-time limits affect the results when the optimization window size is 4. With a small run-time limit (50 or 100 seconds), and a move distance of 2 or 3, the ILP solver terminates early for some benchmark circuits, prior to finding the optimal solution. This is apparent with move distance 3, where clock spine count is reduced by several additional percent when additional solver run-time is allowed.

We now consider the quality of our solutions relative to those produced by an alternative approach – clock power reduction in simulated annealing-based placement [18]. The wirelength and timing terms in (8) are weighted automatically by VPR (parameters $\alpha$ and $\beta$); however, it is not straightforward to weight the clock power reduction term in the annealing cost function, i.e. how to set weight parameter $\gamma$ in (8). We therefore placed and routed the benchmark circuits multiple times, each with a different weight ($\gamma$). For each weight, we computed the geometric mean spine count and wirelength across all circuits. We then computed the percentage change versus the baseline VPR (which does not consider clock power).

The results for the annealing approach are shown in Fig. 6. The vertical axis shows the average % increase in wirelength; the horizontal axis shows the % spine reduction. Each data point in Fig. 6 represents the mean result across all benchmarks for one weight. Weight values increase from left to right in the figure. Weight is very effective in controlling the trade-off between clock power and other criteria. When the clock power weight is very low, we actually see slightly improved wirelength vs. baseline VPR, which is counter-intuitive and we believe is due to the additional variability in the cost function affecting the annealing schedule (which is dynamic/adaptive). As weight is increased to a large value, clock power dominates wirelength in the annealing cost function: large spine count reductions are achieved at a very high wirelength cost. The "knee" of the curve corresponds to an average spine count reduction of 60% at a cost of 3-4% in wirelength, which is fairly close to the results achieved by our approach, giving us confidence that our method produces reasonably good results. We reinforce that our approach is *not* intended to beat prior techniques in quality. Rather, we offer an new formulation for clock power reduction that can be applied in scenarios where prior techniques cannot work, such as analytical placement.

Just as the proposed optimization impacts logic signal wirelength, it also affects post-routed critical path delay (as reported by VPR). We found that with an optimization window size of 2 and with a permissible move distance of 2, critical path delay was increased by 3-4%, on average[3]. With an optimization window size of 4, with a permissible move distance of 2 or 3, critical path delay was increased by 5-9%, on average. Many applications do not need to operate at the highest possible speed, and therefore, while the technique does reduce performance, we believe it will be useful for power-sensitive applications. Note that in general, the speed reduction does *not* imply that the non-power-aware baseline can be run at a lower speed to achieve the *same* power as our approach: Comparing the baseline implementation of a circuit with our implementation of the same circuit, both running at a specific clock speed, our implementation will provide better power characteristics (estimated below). Full performance data is excluded due to page limitations. Note that it is easy for our approach to disallow moves for blocks on the critical path – such blocks can be given a single (fixed) location in the ILP formulation.

Regarding the scalability of our approach to modern industrial designs, as noted earlier, the proposed approach is suitable for a multi-core parallel implementation, with different cores optimizing clock regions and/or windows concurrently. We also stress that in commercial FPGAs such as Xilinx Virtex-5, the clock regions are just 20 logic blocks high, making the windowed optimization approach viable.

### C. Projected Power Benefit

While commercial power estimation tools from FPGA vendors can measure clock power, there is currently no publicly available power estimation framework for FPGA clock power that can handle the clock architectures considered in this paper. Consequently, we make a rough back-of-the-envelope projection of the total dynamic power benefit as follows: Based on the results above, we estimate that the proposed method reduces clock capacitance by $\sim$50% and that clock power comprises $\sim$25% of total FPGA dynamic power. The clock power reductions come at a cost of 6% increase in the routed wirelength of logic signals and a corresponding increase in the capacitance of those signals. Toggles on logic signals are assumed to comprise $\sim$50% of total FPGA power [3, 2]. Hence, the optimizations *reduce* total power by $0.5 \times 25\% = 12.5\%$, but *increase* total power by $0.06 \times 50\% = 3.0\%$, yielding a projected net benefit of $\sim$9%. While the power improvements are modest, the observed improvements are for clock power only and the technique can be a useful part of a power-aware CAD flow, in particular, one that includes power optimization of the logic (non-clock) signals. We expect the approach may particularly benefit designs that contain very high frequency clock signals.

[3]VPR reports critical path delay as though the designs have a single clock domain.

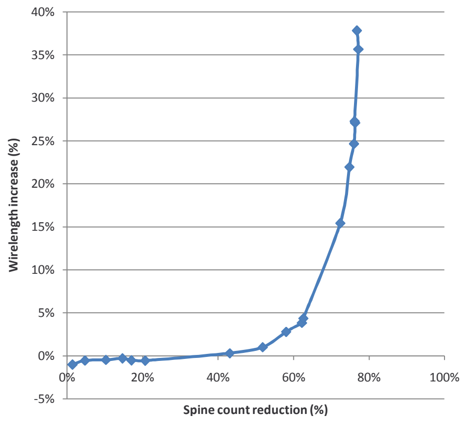

Fig. 6. Spine count reduction vs. wirelength increase for annealing-based placement for different cost function weights.

## V. Conclusions and Future Work

Clock power is a significant component of overall power in FPGA circuits. In this paper, we presented a placement-based technique for reducing FPGA clock power. The clock power reduction problem is formulated as an integer linear program, executed as a post-pass after traditional core placement is complete. Results show our method can reduce the number of used clock spine resources by over 50%, with minimal damage to traditional placer metrics. To our knowledge, no prior work has applied formal mathematical techniques to power reduction in FPGA CAD. The proposed method is useful when the core placement algorithm is non-iterative, e.g. analytical placement.

Future work involves evaluating the proposed ILP-based technique within a comprehensive power-aware FPGA CAD system that optimizes power throughout the flow, including in synthesis, packing, placement and routing.

## References

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. On Computer Aided Design*, vol. 26, no. 2, pp. 203–215, Feb. 2007.
[2] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in the Virtex-II FPGA family," in *ACM Int'l Symp. on FPGAs*, 2002, pp. 157–164.
[3] V. Degalahal, and T. Tuan, "Methodology for high level estimation of FPGA power consumption," in *IEEE Asia and South Pacific Design Automation Conf.*, 2005, pp. 657–660.
[4] N. Viswanathan and C. C.-N. Chu, "FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model," in *ACM/IEEE Int'l Symp. on Physical Design*, 2004, pp. 26–33.
[5] H. Bian, A. Ling, A. Choong, and J. Zhu, "Towards scalable placement for FPGAs," in *ACM Int'l Symp. on Field Programmable Gate Arrays*, 2010, pp. 147–156.
[6] S. Gupta, J. Anderson, L. Farragher, and Q. Wang, "CAD techniques for power optimization in Virtex-5 FPGAs," in *IEEE Custom Integrated Circuits Conference*, San Jose, CA, 2007, pp. 85–88.
[7] *Stratix-III FPGA Family Data Sheet*, Altera, Corp., San Jose, CA, 2008.
[8] *Virtex-5 FPGA Data Sheet*, Xilinx, Inc., San Jose, CA, 2007.
[9] J. Lamoureux and S. Wilton, "Clock-aware placement for FPGAs," in *IEEE Int'l Conf. on Field Programmable Logic and Applications*, 2007, pp. 124–131.
[10] K. Vorwerk, M. Rahman, J. Dunoyer, Y.-C. Hsu, A. Kundu, and A. Kennings, "A technique for minimizing power during FPGA placement," in *IEEE Int'l Conf. on Field Programmable Logic and Applications*, 2008, pp. 233–238.
[11] Q. Wang, S. Gupta, and J. Anderson, "Clock power reduction for Virtex-5 FPGAs," in *ACM Int'l Symp. on Field Programmable Gate Arrays*, 2009, pp. 13–22.
[12] A. Chakraborty, and D. Pan, "PASAP: Power aware structured ASIC placement," in *IEEE Int'l Symp. on Low Power Electronics Design*, 2010, pp. 395–400.
[13] S. Huda, M. Mallick, J. Anderson, "Clock gating architectures for FPGA power reduction," in *IEEE Int'l Conf. on Field Programmable Logic and Applications*, 2009, pp. 112–118.
[14] W.-K. Mak, "I/O placement for FPGAs with multiple I/O standards," in *ACM Int'l Symp. on Field Programmable Gate Arrays*, 2003, pp. 51–57.
[15] http://lpsolve.sourceforge.net/5.5/, "The lp_solve mixed integer linear programming (MILP) solver," 2010.
[16] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, M. Fang, and J. Rose, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *ACM Int'l Symp. on FPGAs*, 2009, pp. 133–142.
[17] http://www.eecg.toronto.edu/∼vaughn/challenge/challenge.html, "The FPGA place-and-route challenge benchmark circuits," 2010.
[18] K. Vorwerk, A. Kennings, V. Pevzner, A. Kundu, M. Raman, J. Dunoyer, and Y.-C. Hsu, "Power minimisation during field programmable gate array placement," *IEE Computers and Digital Techniques*, vol. 4, no. 3, pp. 170–183, 2010.