Clock Power Reduction for Virtex-5 FPGAs

Qiang Wang
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
qiangw@xilinx.com

Subodh Gupta
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
subodhg@xilinx.com

Jason Anderson
ECE Dept., Univ. of Toronto
10 King's College Road
Toronto, ON M5S 3G4
janders@eecg.toronto.edu

ABSTRACT

Clock network power in field-programmable gate arrays (FPGAs) is considered and two complementary approaches for clock power reduction in the Xilinx®Virtex™-5 FPGA are presented. The approaches are unique in that they leverage specific architectural aspects of Virtex-5 to achieve reductions in dynamic power consumed by the clock network. The first approach comprises a placement-based technique to reduce interconnect resource usage on the clock network, thereby reducing capacitance and power (up to 12%). The second approach borrows the "clock gating" notion from the ASIC domain and applies it to FPGAs. Clock enable signals on flip-flops are selectively migrated to use the dedicated clock enable available on the FPGA's built-in clock network, leading to reduced toggling on the clock interconnect and lower power (up to 28%). Power reductions are achieved without any performance penalty, on average.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids

General Terms

Design, Algorithms

Keywords

Field-programmable gate arrays, FPGAs, power, optimization, low-power design, clocking

1. INTRODUCTION

While field-programmable gate arrays (FPGAs) are a widely-used platform for many digital circuit applications, one area where they are virtually absent is within the low-power domain. The programmability of FPGAs is achieved through large numbers of static RAM (SRAM) configuration cells, programmable logic and interconnect fabrics. Naturally, these structures represent an "overhead" in comparison to a fixed-function ASIC. Such overhead contributes to longer metal wirelengths and higher capacitance, as well as higher leakage currents. To be sure, a recent work by Kuon and Rose compared 90*n*m FPGAs to ASICs and found FPGAs consume 7-14 times more dynamic power than ASICs, and 5-87 times more leakage power [15].

Bridging the power gap between FPGAs and ASICs requires a broad research thrust across the spectrum from architectures (e.g. [20, 22, 21, 7, 13, 16]), to circuits (e.g. [23, 11, 29, 4]), to CAD algorithms. Commercially, we also see the major vendors increasing their emphasis on power: the Actel IGLOO, Altera Stratix-III and Xilinx Virtex-5 FPGAs all incorporate special features and circuits for reduced power [1, 3, 32].

Our work falls in the CAD domain, where prior studies report power reductions at various stages of the flow. The research described in [28, 9, 27, 10] considers power optimization in front-end synthesis. Power-aware technology mapping and the interactions between power-aware mapping, packing, placement and routing algorithms have been studied [17]. In the back-end of the flow, power has been the objective of clustering [8, 26] placement and routing [14, 24], with the general approach being to reduce the capacitance of the design signals having high toggle rates. Post-routing optimizations for dynamic and leakage power minimization have also been demonstrated [5, 14].

FPGAs have dedicated programamble clock networks specially designed to distribute clock signals to all logic and IP blocks with low skew and latency. Prior work considered the breakdown of dynamic power in the Xilinx Virtex-2 FPGA and reported that up to 22% of power consumption is due to clocking [25]. In this paper, we propose two approaches to clock power reduction. We first describe a placement-based technique that accounts for the underlying architecture of the clock network to achieve a reduction in clock routing capacitance. We next describe an approach for *clock gating* in FPGAs, which is a known power reduction technique in the ASIC domain. We selectively migrate clock enable signals on flip-flops to use the clock enable pins available on the clock network buffers. While this approach does not target clock network capacitance, it does reduce the switching activity on the clock network, reducing power. Board-level current measurements show the efficacy of both techniques for power reduction.

To our knowledge, no prior research considers clock gating in the FPGA context. However, the related topic of dynamic clock management for FPGAs was considered in an early paper [6]. Rather than completely disabling the clock signal

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
FPGA'09, February 22–24, 2009, Monterey, California, USA.
Copyright 2009 ACM 978-1-60558-410-2/09/02 ...$5.00.

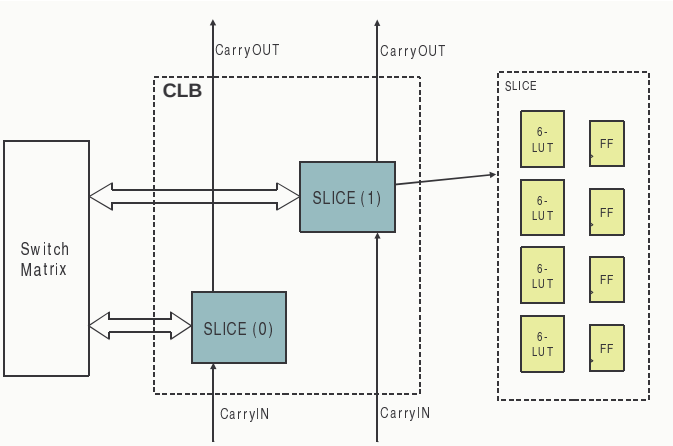


Figure 1: Virtex-5 FPGA CLB and SLICE.

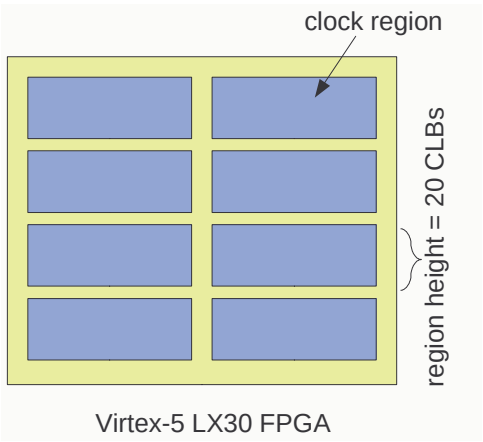


Figure 2: Clock regions in Virtex-5.

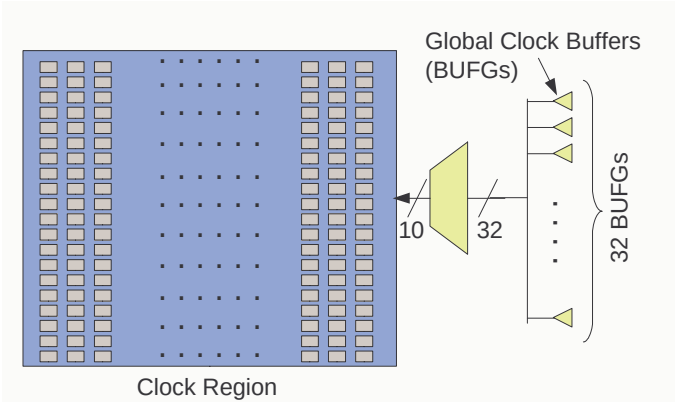


Figure 3: A clock region in Virtex-5.

to circuit blocks when their output does not affect overall circuit outputs (clock gating), dynamic clock management seeks to *slow down* the clock supplied certain circuit blocks when high throughput is not required of those blocks. The prior work dealt primarily with circuit-level techniques, not CAD issues.

For clock-aware FPGA placement, the only works to consider CAD and architecture for reducing FPGA clock network power are those of Lamoureux and Wilton [18, 19], and the very recent paper by Vorwerk et al. at Actel [30]. Modern FPGAs are partitioned into regions, with limitations on the number of clock signals that may be routed into any given region. The placer in [19] minimizes the number of regions spanned by a clock network and also minimizes the number of resources used within regions, thereby reducing clock routing capacitance. Unlike [19], in this paper we do not consider which clocks should be routed on global versus regional clock resources – our placer assumes such decisions have already been made. However, like [19], our placer does consider minimizing the clock network resources used within a region, albeit using what can be viewed as an extension of the placement algorithm in [19]. Furthermore, we target a 65nm commercial Xilinx FPGA. With respect to Vorwerk [30], we target a different commercial FPGA, Virtex-5, and consequently use a somewhat different approach to clock capacitance reduction.

It is worth reinforcing that an important distinction of our work versus all prior publications on FPGA clock power is that our power results are based on board-level current measurements of a real commercial FPGA, and not based on estimates of interconnect capacitance and net activity.

The rest of this paper is organized as follows: Section 2 provides relevant background on the clock network architecture of the Virtex-5 FPGA, which is the target of our optimization. Our placement-based approach to clock capacitance reduction is described in Section 3. Section 4 introduces the clock gating optimization. A discussion of results appears in Section 5. Conclusions and suggestions for future work follow in Section 6.

2. BACKGROUND

In this section, we describe the architectural features of Virtex-5 that are relevant to the two power optimization methods proposed in the paper. We first give an overview of the clock network architecture of Virtex-5, and then move on to describe the available hardware resources for implementing clock enables.

2.1 Virtex-5 Clock Network Architecture

The basic element for implementing combinational logic in Virtex-5 is called a *configurable logic block* (CLB), which contains two SLICEs, each of which contain four 6-input look-up-tables (LUTs) and four registers. A Virtex-5 SLICE and CLB are shown in Fig. 1. Each CLB's inputs and outputs connect to a programmable interconnection network that permits CLBs to be connected to one another, as needed for the design implemented in the FPGA. In addition to CLBs, the Virtex-5 fabric contains large hard-IP blocks, such as block RAMs and DSPs, as well as tiles for I/O, clock management and varied other tiles.

Today's digital designs may contain many different clock signals, and the Virtex-5 FPGA is designed to accomodate this. *Global clock buffers* (BUFGs) within Virtex-5 receive clock signals, from either external or internal sources, and feed such signals into the dedicated global clock interconnection network. The global clock interconnection network distrubutes clock signals throughout the FPGA with low-skew, low-jitter and low-power.

Each Virtex-5 chip contains 32 global clock buffers and therefore, can support the presense of 32 global clock signals

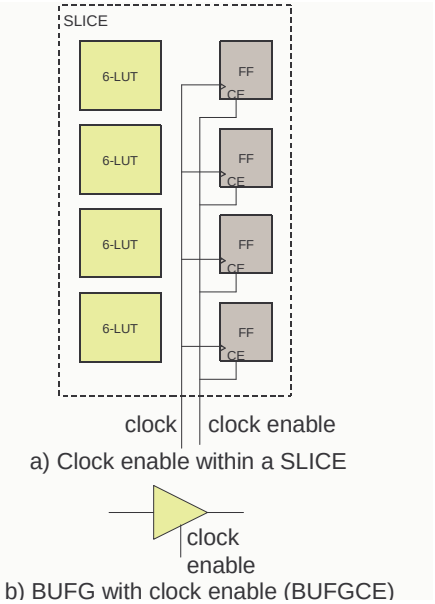


Figure 4: Clock enable options in Virtex-5 FPGA.

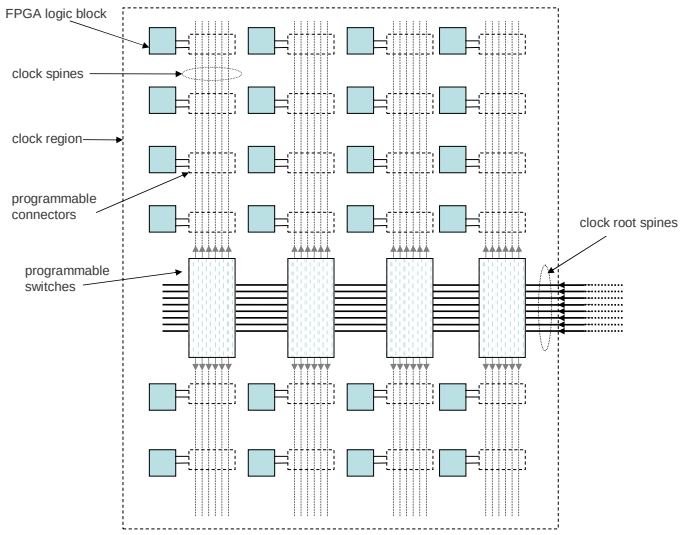


Figure 5: Clock region interconnect details.

in the design. Architectural constraints limit the possibility of having 32 separate global clock signals reach each and every flip-flop on the device[1]. Instead, the Virtex-5 fabric is partitioned into *clock regions* for the purposes of clock distribution. Clock regions in Virtex-5 are 20 CLBs tall and span half of the Virtex-5 die horizontally. Up to 10 global clock signals may be fed into any given region. The 10 global clocks within a region may be selected from any of the (up to) 32 global clock signals present in the design. Fig. 2 depicts the eight regions in the Virtex-5 LX30 FPGA; larger members of the Virtex-5 family will contain more regions. Fig. 3 gives a blow-up view of a clock region. Observe that up to 10 clock signals, driven by 32 BUFGs, are selected to be driven into the clock region.

2.2 Clock Enables in Virtex-5

As will be discussed in Section 4, clock gating involves selectively disabling the clock signal from reaching sequential elements in the design. Here, we introduce two clock enable mechanisms built-in to Virtex-5 that may be used for realizing clock gating. Fig. 4(a) shows that the registers within a SLICE have a clock enable pin, permitting clock enables to be implemented at a fine level of granularity. Observe that all registers in the SLICE must receive the same clock enable signal.

Fig. 4(b) illustrates that the global clock buffer (BUFG) that drives clock signals into the dedicated clock interconnection network has an enable pin. When the BUFG is used with the enable, it is referred to as a BUFGCE. The enable pin can be driven by a signal from an I/O or an internal signal. When the enable is deasserted, the clock network driven by the BUFG becomes silent and no dynamic power is dissipated.

A key observation is that using the clock enable pins on SLICEs, as in Fig. 4(a), does not eliminate toggling on the global clock network and therefore does not save dynamic power on the global clock network. However, using the clock enable pin on the BUFG, as in Fig. 4(b), ceases all toggling on the global clock network driven by the BUFG.

3. CLOCK SPINE REDUCTION

The clock resources in Virtex-5 FPGAs provide considerable flexibility for varied clocking needs. The Xilinx physical design tools must appropriately allocate the clock resources to the clock networks in customer designs. As we will demonstrate, the power consumption of the clock network is strongly influenced by decisions taken by the design tools.

3.1 Clocks and Power Consumption

As described in Section 2, clock resources in Virtex-5 FPGAs are organized into clock regions. Fig. 5 shows a detailed view of the interconnect structure within a clock region. There are ten *root spines* (shown as dark lines in the figure) horizontally crossing the center of the region[2]. At the intersection of each column, each horizontal root spine can be programmably connected to two vertical spines: one traveling north for the top-half of the region, and one traveling south for the bottom-half of the region. The vertical spines are shown as vertical dashed lines in Fig. 5. Since there are 10 horizontal root spines in a region, there are 20 vertical spines for each column of the region. The logic blocks in the region receive their clock signals from the vertical spines. Each upper or lower vertical spine corresponds to a single horizontal root spine. Every horizontal root spine can be driven by one of 32 global clock buffers (BUFGs), as shown in Fig. 3.

The power consumption of a clock signal within a clock region is mainly determined by the number of vertical spines connected to the root spine. A root spine is *only* connected to a vertical clock spine if the placement necessitates it. If a

[1]In addition to flip-flops, DSP blocks and block RAMs and all other synchronous blocks must capable of receiving global clock signals.

[2]For clarity, fewer than ten root spines are shown in the Fig. 5.

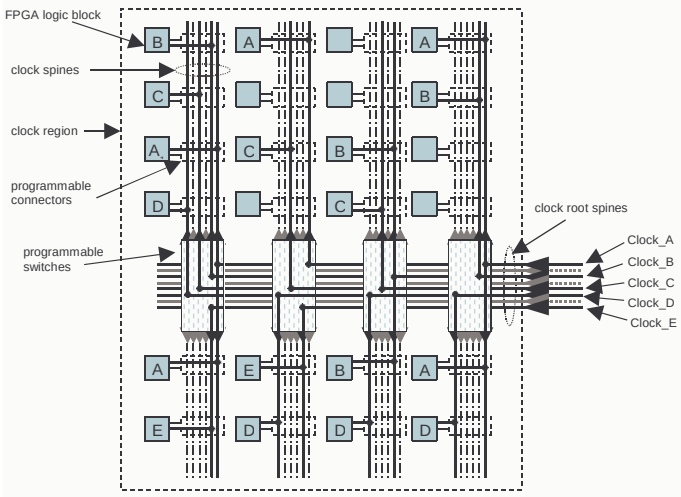


Figure 6: Initial placement of clock loads.

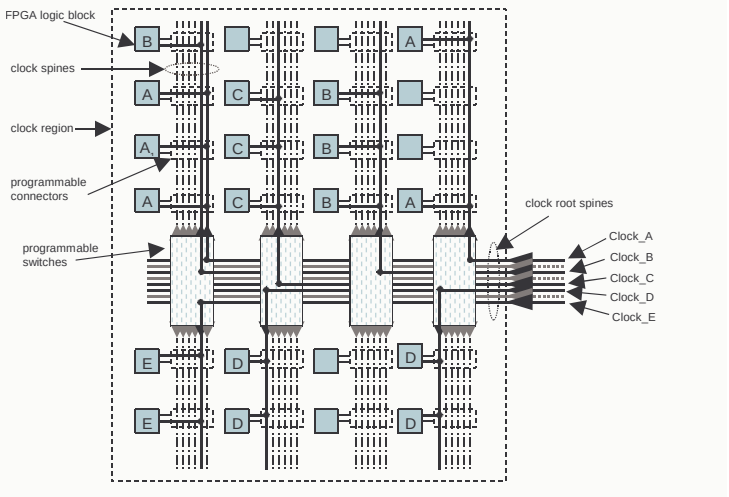


Figure 7: Placement with reduced clock spine usage.

horizontal root spine carries a clock signal, $clkA$, connections to a vertical spine will only happen for those half-columns containing a load of $clkA$. Vertical clock spines contribute significantly to the capacitance of the clock signal, thereby affecting clock power. Indeed, the total power consumption for a clock signal is strongly tied to the sum of the number of vertical spines used in all the clock regions within which the load components of the clock are placed. Roughly 90% of clock network capacitance is contributed by vertical spines in typical designs.

3.2 Reducing Clock Spine Usage

Traditionally, the primary goal of placement is to minimize the total wirelength of routing connections among all connected components. Consequently, the loads of a clock signal are generally placed such that the wirelengths of the non-clock signal connections between such loads are minimized. Fig. 6 shows an example of placement results for components that are related to five different clock signals. Tiles marked with letter A indicate that load components of $Clock\_A$ are placed on them. Similarly, $Clock\_B$, $Clock\_C$, $Clock\_D$, and $Clock\_E$ have load components placed on tiles that are marked with letters B, C, D, and E, respectively. The load components of each clock signal are spread across many logic block columns. Within this clock region, *eighteen* vertical clock spines are used, based on the placement of all the load components of the five clock signals.

To reduce power consumption on clock networks, we can alter the placement to prefer that loads of the same clock share the same vertical spines. This can be accomplished by prefering to place loads of a given clock signal on the same half-column. Fig. 7 shows the results after changing the placement for the load components. The adjusted placement uses *eight* vertical spines, as opposed to the eighteen vertical spines used in Fig. 6. Certainly, the clock power consumption for the placement in Fig. 7 will be lower than that for the placement of Fig. 6. The example of Figs. 6 and 7 show the core idea of our placement optimization: aim to minimize vertical clock spine utilization by preferring to place loads of the same clock signal on the same half-columns within a clock region.

In any logic circuit, the speed performance is generally limited by the delays of logic signals, and not clock signals. As such, if we are too aggressive in our minimization of vertical clock spine usage, we may end up increasing the delays of the logic signals and increasing the critical path delay. Therefore, placement decisions regarding clock spine reduction must be made judiciously, considering the design's speed performance. As described in the next section, our placement optimization only aims to co-locate the loads of a clock signal on the same half-column, if the loads are already close to one another in the wirelength and performance-driven placement. Therefore, after our clock spine-aware placement, there may still remain some loads of the same clock using multiple vertical spines that are distant from one another.

3.3 Implementation

The Xilinx placer is based on a multi-step flow beginning with I/O placement and clock region assignment, followed by analytical (mathematical) placement, and ending with swap-based optimization. A more detailed overview of the placer appears in [2]. Here, we focus on the optimization step, which is used to improve the analytical placement results.

The optimization is conducted by attempting a series of pair-wise component swaps between two locations on the FPGA. For an arbitrary component placed on a location, called its source site, the optimizer will consider a number of alternative locations, called target sites, to which to move the component. For each pair of source and target sites, the optimizer will evaluate the impact caused by the component swap between the two sites. Only the favorable swap attempts will be accepted. The overall placement is improved by iteratively choosing all of the placed components, one after another, as source components, and finding the best target site to move to. The decision on whether or not each attempted swap between any pair of components leads to a better quality placement is guided by a cost function. The cost function that is used in optimizer, described in [14], is:

$$Cost = a \cdot W + b \cdot T + c \cdot P_{avg} \quad (1)$$

where W represents wirelength cost, T represents timing cost, and P_{avg} represents the estimated dynamic power, which is computed as in [14], and a, b, and c are scalar constants.

Clock spine optimization is incorporated into the optimization step through a cost function extended with an additional term:

$$Cost = a \cdot W + b \cdot T + c \cdot P_{avg} + C_{ClockSpine} \qquad (2)$$

where $C_{ClockSpine}$ reflects vertical clock spine usage. The purpose of the term, $C_{ClockSpine}$, is to direct the optimizer to reduce total vertical clock spine usage and reduce clock power.

$C_{ClockSpine}$ in (2) encourages clock load components to share common vertical spines by moving them to a common column, or columns, of sites. To achieve this, we define two types of forces, called *anchoring forces* and *attracting forces*, and use these forces within $C_{ClockSpine}$.

The anchoring notion decides whether or not a vertical spine should be used for a clock to attract and hold loads of the clock to the corresponding column. More loads placed on the column naturally form a stronger force to attract other loads. The attracting force can be viewed as a needle pointing in the direction towards which a load component should be moved. The cost function component encourages the load components of the same clock to move to a smaller number of columns, thereby increasing the sharing of the clock spines. Our cost function also identifies and selects certain logic block columns as anchor columns (described below), which can be thought of as the preferred destination columns that load components should move to.

For each clock load component, our cost function evaluates whether or not its current site can share a vertical clock spine with other load components of the same clock. The more components that share the same spines, the lower the cost value is. At the same time, the shorter the distance between the current component and the closest anchor column, the lower the cost value. Before we introduce the cost function term, we define several parameters:

- $N_{i,j}$: Number of loads of clock j placed on a half-column, i.
- D_k: Distance (in columns) from a clock load k to the closest anchor spine column for the clock signal containing load k. For example, if load k is on clock signal j, D_k is the distance from the column where k is placed to the closest anchor column for signal j.
- $Clks_i$: Set of clock signals used on vertical half-column i. Each of these will require a separate vertical spine.

The cost function for clock spine minimization is:

$$C_{ClockSpine} = \sum_{i=1}^{n} \sum_{j \ \epsilon \ Clks_i} \frac{S}{N_{i,j}} + \sum_{k=1}^{m} A \cdot D_k \qquad (3)$$

where n is the number of vertical half-columns, S and A are constants, and m is the number of clock loads in the design. Essentially, the first operand of + in (3) walks over all used vertical clock spines and tallies their costs, where the cost of a spine is S divided by the number of loads routed through it. This term reflects *anchoring forces*. Parameter S in (3) is set to 0 for the case of anchor clock spines.

The second operand of + in (3) reflects *attracting forces*. The intent is to draw clock loads towards anchor spines. Specifically, this term represents the total cost of clock loads that are not placed on an anchor clock spine. D_k equals zero if load k is already placed on an anchor spine. By properly choosing the values of constants S and A, we can tune the balance between the attraction and anchoring components of the clock spine reduction cost function. Likewise, the magnitudes of S and A can be used to trade-off the importance of reducing clock spines with other traditional placement criteria, such as wirelength and speed performance.

The selection of anchoring spines can be determined naturally by the optimizer through iterative swaps between pairs of placed components. To improve the predictability of the spine reduction results, before the optimizer starts, we selectively promote a set of vertical spines as anchors. Consider a vertical half-column i with a load of clock j placed on it. We wish to determine if i should be designated as an anchor column for clock j. Define:

- $D_{i,j}$: Distance from column i to the closest anchor column for clock j in the region.
- $Common_i$: The set of clock signals used on column i having the most loads.

Using these definitions, column i is qualified as an anchor for clock j if:

$$\begin{aligned} Qual_{i,j} \quad = \quad & [(D_{i,j} \geq width) \bigvee (N_{i,j} \geq thresClk)] \bigwedge \\ & [(|Clks_i| \leq thresCol) \bigvee (j \ \epsilon \ Common_i)] \end{aligned} \qquad (4)$$

where $width$, $thresClk$, and $thresCol$ are empirically-derived tuning parameters.

The first clause in (4) qualifies i as an anchor for clock j if there is no other anchor column nearby. The second clause qualifies i if the number of loads of j already placed on i exceeds a given threshold. The last two clauses consider the competition between clock signals for designation as anchors on a given column. Concretely, the third clause checks whether the number of different clocks competing for column i exceeds a threshold. If the threshold is exceeded, column i is only designated as an anchor for clock j, if j happens to be one of the clocks with the *most* loads placed on column i (fourth clause).

The pre-selection process walks from left-most column to the right-most column in each clock region, and for each column that holds at least one clock load, it determines whether or not the column should be selected as an anchor using (4). In the event that $Qual_{i,j}$ is false, a nearby column is selected as an anchor for clock j. Recognize that the pre-selection process is conducted once for every clock signal.

3.4 Results

The power reduction technique described above has been incorporated into the Xilinx commercial tools and will be released to customers in the forthcoming ISE™11.1i software release. We follow the experimental methodology in [14], where power is evaluated on a set of designs collected from Xilinx customers. The designs have been augmented with a built-in random vector generator attached to their data inputs. A linear feedback shift register-based (LFSR-based) pseudo-random vector generator is used for input vector stimulus. This permited board-level current measurements

Table 1: Clock spine reduction results.

Circuit	LUTs	FFs	Spine Reduction	Power Reduction
industry1	22927	3021	46.7%	3.0%
industry2	23488	6641	26.9%	1.0%
industry3	15814	1378	11.5%	1.4%
industry4	23119	25967	15.5%	1.0%
industry5	8273	3747	42.0%	9.9%
industry6	14351	4620	8.6%	3.9%
industry7	26754	2163	47.6%	4.0%
industry8	11877	21311	21.4%	1.5%
industry9	19340	8311	22.0%	0.4%
industry10	14011	10022	16.7%	12.5%
industry11	16253	16612	5.4%	1.5%
industry12	3022	2736	18.3%	5.4%
Average:			**23.5%**	**3.8%**

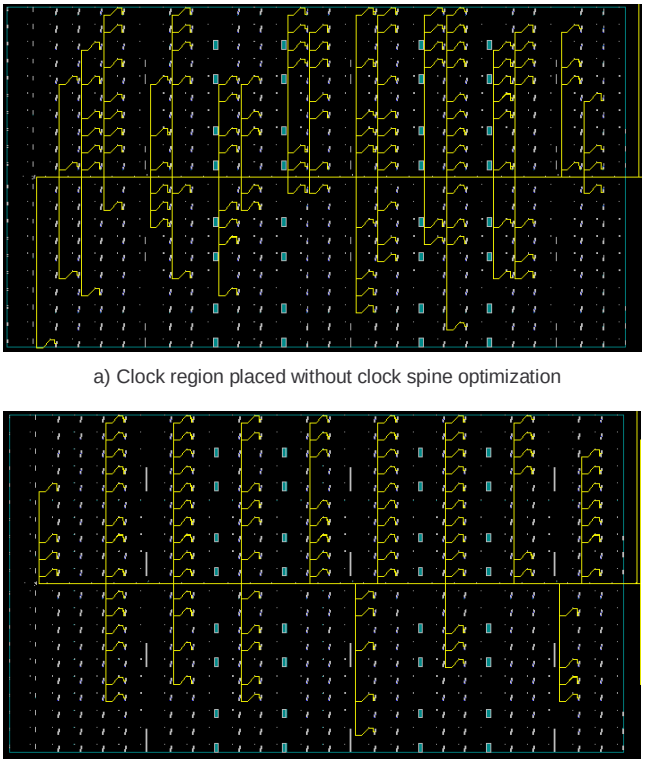
of power, without requiring externally supplied input vectors. Power results are based on measured current drawn from the V_{ccint} power supply while circuit inputs are excited with random vectors.

To evaluate the power savings of clock spine optimization, each benchmark circuit was placed and routed twice: with and without clock spine optimization. In both cases, the same aggressive speed performance constraint was supplied to the place and route tool. We compare the baseline power-aware flow described in [14], with the same flow augmented with clock spine reduction. Table 1 shows the power benefits of clock spine reduction.

We first examine the effectiveness of our approach in reducing vertical clock spine usage. Columns 1, 2 and 3 of Table 1 list a set of benchmark circuits, and the number of LUTs and flip-flops (FFs) in each circuit. Column 4 of Table 1 shows the percentage reduction in clock spine usage for each circuit when clock spine optimization is turned on. Some circuits exhibit a modest reduction in spine usage (5%); however, for other circuits, spine usage is reduced by nearly half. The data demonstrates that (3) is indeed very effective in co-locating loads of clocks on common spines, thereby reducing spine count.

Column 5 of the table shows the percentage reduction in power. Some designs show virtually no power benefit, while the maximum benefit seen was about 12.5%. The power benefits in column 5 do not track well with the spine reductions in column 4. We believe the reason for this is that although preferring to place loads on common vertical spines will certainly reduce clock power, it may end up elongating non-clock logic signals, resulting in higher power on such logic signals, canceling out some clock power benefits.

Nevertheless, on average, our approach produces nearly 4% power reduction, on average, and can be combined with the already-existing optimizations in the Xilinx power-aware flow (described in [14]). Since clock spine optimization is integrated into the placer's cost function, it is conceivable that it may negatively affect the maximum achievable performance for a design. To investigate this, we used an iterative approach to find the maximum achievable frequency for each benchmark. We performed this analysis for implementations with and without the clock spine optimization



a) Clock region placed without clock spine optimization

b) Clock region placed with clock spine optimization

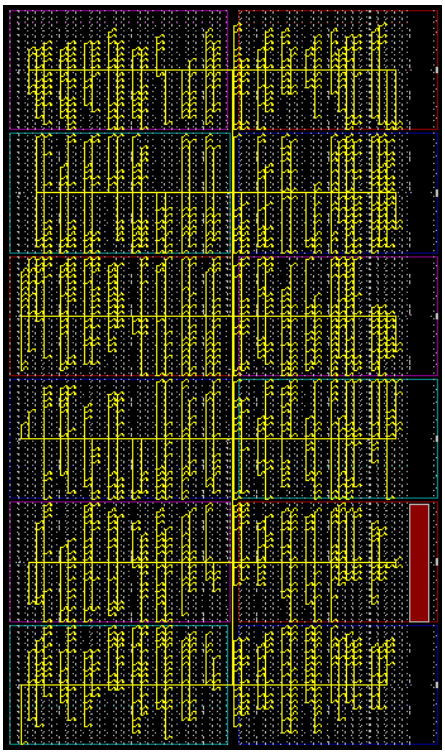
Figure 8: Clock network routing in a single region without and with clock spine reduction for a benchmark circuit.

turned on. The results were noisy with some circuits exhibiting improved performance and some degrading. On average, across all circuits, we observed a degradation of 0.3% in performance due to our clock spine reduction techniques. Thus, on average, the power benefits shown in Table 1 do not come with a performance hit.

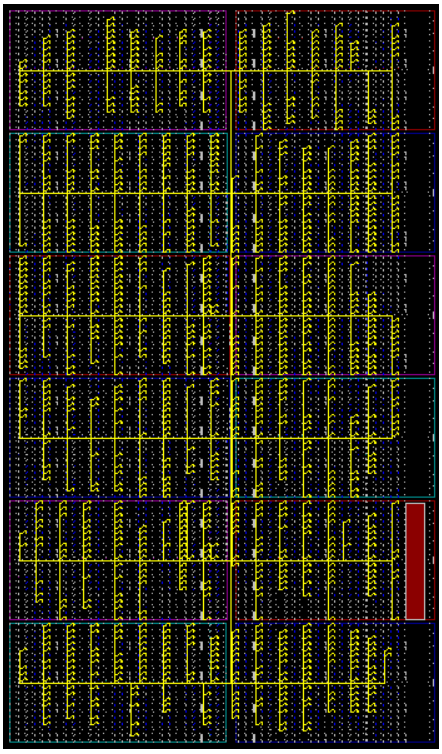
Figs. 8 and 9 qualitatively illustrate the effects of clock spine optimization for one benchmark. Fig. 8 shows the actual clock signal routing in a single region of the Virtex-5 chip. Part (a) of the figure shows the clock signal routing without clock spine optimization; part (b) shows the clock signal routing in the same region, with clock spine optimization turned on. Observe the reduction in the number of used veritcal clock spines in part (b) of the figure. Fig. 9 shows the clock signal routing across the entire Virtex-5 FPGA for the same benchmark circuit. Again, part (a) of the figure shows the results without clock spine optmization; part (b) shows the results with clock spine optimization. Though it is difficult to discern precise details from the figure, a reduction in clock spine usage is apparent when clock spine optimization is applied, as is an improved organization and regularity in the clock signal routing.

4. CLOCK GATING

Clock gating is a simple and effective method for reducing dynamic power consumption. Clock gating decreases dynamic power by eliminating unnecessary toggling on the outputs of flip-flops of a circuit, gates in the fanout of the



a) Without clock spine optimization



b) With clock spine optimization

Figure 9: Clock network routing without and with clock spine reduction for a benchmark circuit.

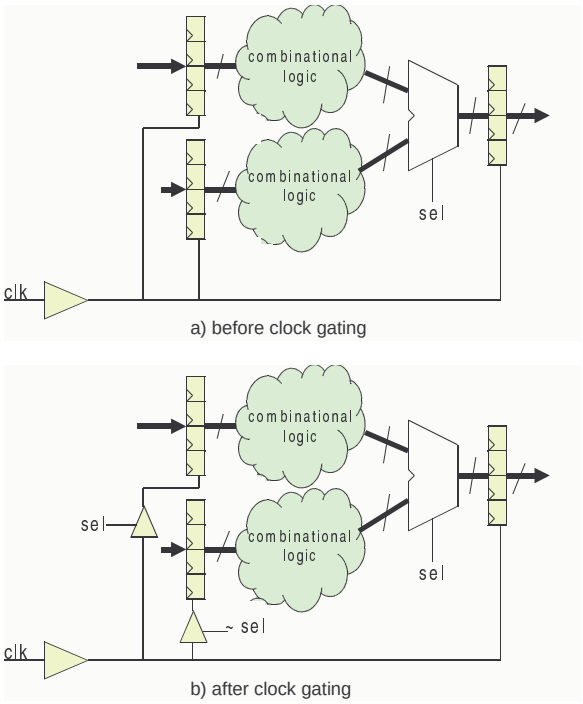


Figure 10: An example of "classic" clock gating.

flip-flops, and clock signals. Fig. 10 depicts a classic example of clock gating. In part (a) of the figure, we see two register files on the left, each feeding a combinational logic circuit. The outputs of the two combinational logic circuits feed the inputs of a multiplexer. The multiplexer's select signal, *sel*, selects which combinational circuit's output is passed to the input of a destination register file. Part (b) of the figure shows the circuit after clock gating has been applied. The signal *sel* is used to derive a clock enable signal on clocks feeding the input register files. Power is reduced through several mechanisms. First, the capacitive loading on the clock network itself is reduced. Second, unnecessary toggling within the combinational circuits is eliminated. In particular, in the optimized circuit, toggling only occurs in the combinational circuit whose output is selected by the multiplexer to be passed to the destination register file.

Clock gating has been used extensively in ASICs for power optimization, e.g. [12]. However, clock gating has not been explored in depth for FPGAs. In this section, we present a simple clock gating technique for Virtex-5 FPGAs. Our approach is related to that in described in [31], which proposed ASIC clock gating at the root of the clock tree, as opposed to clock gating at the level of the clock signal's load registers.

Clock gating in FPGAs is different from ASICs because of the fixed pre-fabricated clock interconnection network. Our approach to clock gating is best understood by an example. In Fig. 11, a clock is driving a number of flip-flops. The top two rows of flip-flops are connected to a clock enable signal, *clkEnable*, whereas the bottom row of flip-flops is not connected to any clock enable signal. Observe that the clock is driven by global clock buffer, BUFG, as described in Section 2. In order to gate the clock, we instantiate a new global clock buffer and use the clock enable on the global

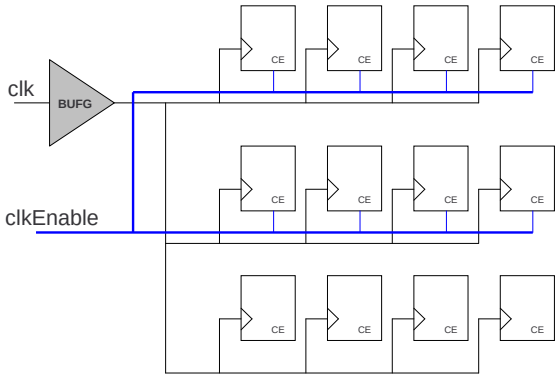


Figure 11: Clock network prior to clock gating.

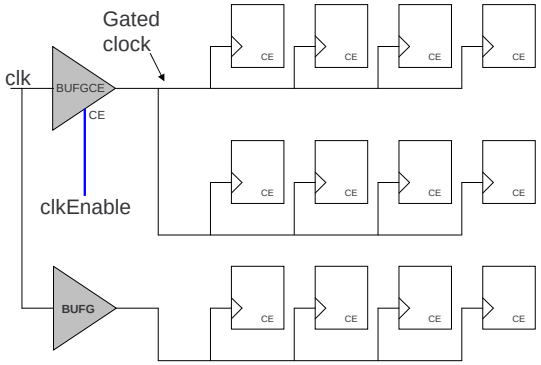


Figure 12: Clock network after clock gating transformation.

clock buffer, shown previously in Fig. 4. The new global clock buffer, called BUFGCE, is shown in Fig. 12. The input to this buffer is also *clk*, however, the clock enable of this buffer is connected to flip-flop's enable signal *clkEnable*, and the *clkEnable* signal is disconnected from the flip-flops it was previously feeding. The newly gated clock now feeds the flip-flops in the top two rows. In order to feed other flip-flops, we keep the original buffer and the ungated clock will feed the remaining flip-flops in the bottom row. Note that BUFGCE is synchronous with respect to the original clock, *clk*, hence any glitch in the enable signal will not affect the gated clock.

At its core, our optimization consists of migrating a pre-existing clock enable signal on load flip-flops to use a clock enable pin on a global clock buffer. Although we introduce a new global clock buffer, driving a separate clock network, power is saved overall because toggling is reduced on the newly introduced clock network, and the capacitance of the original network is reduced. Note that the classic example in Fig. 10 saves power both in the combinational logic and on the clock signal; however, our approach only saves power on the clock signal. We are not dynamically identifying new opportunities for clock gating; rather, we are shifting existing clock enable signals to use the best FPGA resources.

The algorithm for clock gating is:

1. Identify the clock enable signal, CE, on load flip-flops. We select a clock enable signal with a high number of flip-flops connected to it, also considering the fraction of time the clock enable spends in the logic-0 state. We disallow the selection of clock enables driving flip-flops that are connected to any synchronous set/reset signals, as such clock enables could not be migrated without affecting the circuit's logic functionality.

2. Identify the clock signal, CLK, corresponding to flip-flops driven by the selected clock enable signal, CE.

3. Instantiate a new BUFGCE, driven by CLK with a new output net $gatedCLK$, which drives the clock pins of all flip-flops connected to CE.

4. The CE signal is then attached to the clock enable pin of the BUFGCE and the CE signal is disconnected from the flip-flops it was previously driving.

The algorithm above is for a migrating a single clock enable signal to use the clock enable on a global clock buffer. The optimization can be applied iteratively for multiple clock enable signals within a single design. Certainly, the optimization is most beneficial for those designs where a CE signal drives a large number of fanouts. Diminishing returns arise for cases of low-fanout clock enables. Since the optimization involves instantiating additional global clock buffers, we must also be careful not to exceed the number of available clock buffers (32). Note also that we handle the special case of *all* loads of a clock signal having the *same* clock enable signal: introducing an additional BUFGCE is not necessary in such a case.

4.1 Results

As with the clock spine optimization described in Section 3, clock gating has been implemented in the Xilinx commercial tools and will be released to customers in the ISE 11.1i software. It is invoked after technology mapping is complete, prior to placement. We use the same methodology in 3.4 to measure the power benefits of clock gating. In this case, however, we use a different set of benchmark circuits, selected based on their containing high fanout clock enable signals. Section 5 discusses issues relating to combining both gating and clock spine reduction.

Table 2 shows the power reduction results for clock gating. The first four circuits, `industry-a,b,c,d`, are DSP circuits, while the remaining circuits are collected from customers and are of unknown function. As shown in the table, very large power reductions of over 20% are observed for the DSP circuits. These circuits contain high fanout clock enable signals driving many flip-flops. When such signals are migrated to use the enables on global clock buffers, there are two consequences. First, as expected, total switched-capacitance on the clock network is reduced, lowering clock power. Second, a high fanout signal is essentially eliminated from the design, lowering overall routing congestion and capacitance, potentially leading to improved routing for non-clock logic signals.

We also measured the performance impact of clock gating and found that for the designs in Table 2, performance improved by 1.1%, on average. However, due to the relatively small number of circuits, we believe the performance benefit falls in the noise range.

Table 2: Clock gating results.

Circuit	LUTs	FFs	Power Reduction
industry-a	677	893	18.1%
industry-b	1325	1773	18.0%
industry-c	2621	3473	21.7%
industry-d	10397	13193	27.9%
industry-e	19774	10564	1.8%
industry-f	16253	16612	2.5%
industry-g	19259	9972	4.4%
Average:			**13.5%**

5. DISCUSSION

While we have access to many customers circuits in Xilinx, our ability to use a given circuit as a benchmark for power optimization research is limited by whether or not we can adapt it to contain built-in input vector generation, as is needed for board-level power measurements. Most circuits received from customers have unknown functionality, and as such, for many circuits, we cannot apply random input vectors in a meaningful way to exercise internal signals sufficiently. Therefore, we have relatively few benchmarks we can use for power research.

The clock gating optimization described above targets a specific class of circuits, namely, those containing high fanout clock enables. So far, we have not found benchmark circuits that benefit from *both* clock gating and clock spine reduction cumulatively. That is, while a circuit may benefit from gating or spine reduction, we have yet to find circuits where we conclusively observe additive gains. However, as we continue to develop and receive more power benchmarks and apply our optimizations, we are optimistic that we will discover circuits that benefit from both optimizations.

6. CONCLUSIONS AND FUTURE WORK

Low-power is an important goal for FPGA vendors and their customers. Clock power comprises a significant fraction of dynamic power in FPGAs. In this paper, we presented two approaches for reducing clock power in Virtex-5 FPGAs. Our first approach is placement-based, and consists of aligning the loads of a clock signal to reduce clock spine usage and total clock network capacitance. The second approach is a simple clock gating transformation: clock signals on load flip-flops are migrated to use the clock enable built into the global clock buffers in Virtex-5. Clock spine reduction and clock gating reduce power by up to 12 and 28%, respectively.

There are several directions for clock gating that we plan to explore in the future. In particular, we plan to build clock gating into the synthesis steps of the CAD flow, to automatically identify opportunities for gating and data shielding, and synthesize the appropriate control signals. For clock spine reduction, we plan to combine our optimization with a commercial implementation of [19], which attempts to minimize the number of regions spanned by a clock. The two approaches combine to form a comprehensive clocking solution wherein *both* the number of regions is minimized, as well as the number of used spines within each region.

7. REFERENCES

[1] Actel, Corp., Mountain View, CA. *IGLOO Low-Power Flash FPGAs General Description*, 2007.

[2] T. Ahmed, P. Kundarewich, J. Anderson, B. Taylor, and R Aggarwal. Architecture-specific packing for Virtex-5 FPGAs. In *ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays*, pages 5–13, Monterey, CA, 2008.

[3] Altera, Corp., San Jose, CA. *Stratix-III FPGA Family Data Sheet*, 2008.

[4] J.H. Anderson and F.N. Najm. Low-power programmable FPGA routing circuitry. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 602–609, San Jose, CA, 2004.

[5] J.H. Anderson, F.N. Najm, and T. Tuan. Active leakage power optimization for FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 33–41, Monterey, CA, 2004.

[6] I. Brynjolfson and Z. Zilic. Dynamic clock management for low power applications in FPGAs. In *IEEE Custom Integrated Circuits Conference*, pages 139–142, Orlando, FL, 2000.

[7] B.H. Calhoun, F.A. Honore, and A. Chandrakasan. Design methodology for fine-grained leakage control in MTCMOS. In *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pages 104–109, Seoul, Korea, 2003.

[8] D. Chen and J. Cong. Delay optimal low-power circuit clustering for FPGAs with dual supply voltages. In *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pages 70–73, Newport Beach, CA, 2004.

[9] D. Chen, J. Cong, and Y. Fan. Low-power high-level synthesis for FPGA architectures. In *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pages 134–139, Seoul, Korea, 2003.

[10] D. Chen, J. Cong, Y Fan, and Z. Zhang. High-level power estimation and low-power design space exploration for FPGAs. In *IEEE/ACM Asia and South Pacific Design Automation Conference*, pages 529–534, Yokohama, Japan, 2007.

[11] L. Ciccarelli, A. Lodi, and R. Canegallo. Low leakage circuit design for FPGAs. In *IEEE Custom Integrated Circuits Conference*, pages 715–718, Orlando, FL, 2004.

[12] M. Donno, A. Ivaldi, L. Benini, and E. Macii. Clock-tree power optimization based on RTL clock-gating. In *IEEE/ACM Design Automation Conference*, pages 622–627, Anaheim, CA, 2003.

[13] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and T. Tuan. Reducing leakage energy in FPGAs using region-constrained placement. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 51–58, Monterey, CA, 2004.

[14] S. Gupta, J. Anderson, L. Farragher, and Q. Wang. CAD techniques for power optimization in Virtex-5 FPGAs. In *IEEE Custom Integrated Circuits Conference*, pages 85–88, San Jose, CA, 2007.

[15] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Trans. On CAD of*

Integrated Circuits and Systems, 26(2):203–215, February 2007.

[16] J. Lamoureux, G. Lemieux, and S. Wilton. GlitchLess: an active glitch minimization technique in FPGAs. In *ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays*, pages 156–165, Monterey, CA, 2007.

[17] J. Lamoureux and S.J.E. Wilton. On the interaction between power-aware FPGA CAD algorithms. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 701–708, San Jose, CA, 2003.

[18] J. Lamoureux and S.J.E. Wilton. FPGA clock network architecture: flexibility vs. area and power. In *ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays*, pages 101–108, Monterey, CA, 2006.

[19] J. Lamoureux and S.J.E. Wilton. Clock-aware placement for FPGAs. In *IEEE International Conference on Field-Programmable Logic and Applications*, pages 124–131, Amsterdam, The Netherlands, 2007.

[20] F. Li, D. Chen, L. He, and J. Cong. Architecture evaluation for power-efficient FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 175–184, Monterey, CA, 2003.

[21] F. Li and L.He. Vdd programmability to reduce FPGA interconnect power. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 760–765, San Jose, CA, 2004.

[22] F. Li, Y. Lin, and L. He. FPGA power reduction using configurable dual-Vdd. In *ACM/IEEE Design Automation Conference*, pages 735–740, San Diego, CA, 2004.

[23] A. Rahman and V. Polavarapuv. Evaluation of low-leakage design techniques for field-programmable gate arrays. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 23–30, Monterey, CA, 2004.

[24] K. Roy. Power-dissipation driven FPGA place and route under timing constraints. *IEEE Transactions On Circuits and Systems*, 46(5):634–637, May 1999.

[25] L. Shang, A. Kaviani, and K. Bathala. Dynamic power consumption in the Virtex-II FPGA family. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 157–164, Monterey, CA, 2002.

[26] A. Singh and M. Marek-Sadowska. Efficient circuit clustering for area and power reduction in FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 59–66, Monterey, CA, February 2002.

[27] R. Tessier, V. Betz, D. Neto, and T. Gopalsamy. Power-aware RAM mapping for FPGA embedded memory blocks. In *ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays*, pages 189–198, Monterey, CA, 2006.

[28] K.O. Tinmaung, D. Howland, and R. Tessier. Power-aware FPGA logic synthesis using binary decision diagrams. In *ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays*, pages 148–155, Monterey, CA, 2007.

[29] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger. A 90nm low-power FPGA for battery-powered applications. *IEEE Trans. On CAD of Integrated Circuits and Systems*, 26(2):296–300, February 2007.

[30] K. Vorwerk, M. Rahman, J. Dunoyer, Y.-C. Hsu, A. Kundu, and A. Kennings. A technique for minimizing power during FPGA placement. In *IEEE International Conference on Field Programmable Logic and Applications*, pages 233–238, Heidelberg, Germany, 2008.

[31] Q. Wang and S. Roy. Power minimization by clock root gating. In *IEEE/ACM Asia and South Pacific Design Automation Conference*, pages 249–254, Kitakyushu, Japan, 2003.

[32] Xilinx, Inc., San Jose, CA. *Virtex-5 FPGA Data Sheet*, 2007.