CLOCK GATING ARCHITECTURES FOR FPGA POWER REDUCTION

Safeen Huda, Muntasir Mallick, Jason H. Anderson

Dept. of ECE, Univ. of Toronto
Toronto, ON Canada
e-mail: {safeen.huda, muntasir.mallick, jason.anderson}@utoronto.ca

ABSTRACT

Clock gating is a power reduction technique that has been used successfully in the custom ASIC domain. Clock and logic signal power are saved by temporarily disabling the clock signal on registers whose outputs do not affect circuit outputs. We consider and evaluate FPGA clock network architectures with built-in clock gating capability and describe a flexible placement algorithm that can operate with various gating granularities (various sizes of device regions containing clock loads that can be gated together). Results show that depending on the clock gating architecture and the fraction of time clock signals are enabled, clock power can be reduced by over 50%, and results suggest that a fine granularity gating architecture yields significant power benefits.

1. INTRODUCTION

The Achilles' Heel of field-programmable gate arrays (FPGAs) in mobile electronics applications is their high power consumption. The circuit overhead associated with logic flexibility and re-programmability leads to longer wirelengths and more dynamic power vs. custom chips. A recent work suggests that dynamic power in FPGAs is 7-14 X higher than in custom ASICs for implementing a given circuit [1]. Large strides are needed on the power front to enable deployment of programmable logic in the handheld battery-powered devices pervasive in society today.

We focus on reducing dynamic power consumption, which is power consumed due to transitions on circuit signals, defined by $P_{avg} = \frac{1}{2}\sum_{i \in nets} f_i \cdot C_i \cdot V^2$, where f_i is the average toggle rate on net i (net i's *switching activity*), C_i is the capacitance of net i and V is the supply voltage. Specifically, we focus on reducing the power consumed by the FPGA's clock network, which prior work has shown to comprise up to 22% of power in an FPGA tile [2]. Reducing the other components of dynamic power, e.g. within the logic cells and general interconnect, has been considered recently by other works (e.g. [3, 4]).

Clock gating is a way of reducing switching activity on circuit signals, and it entails temporarily disabling the clocking of specific registers when the outputs of those registers are inconsequential to circuit outputs [5]. Long used successfully in the ASIC domain (e.g. [6, 7]), it was applied recently for FPGAs, albeit in a limited way [8]. We consider architectures for clock gating in FPGAs and propose incorporating gating capabilities at points within the clock distribution network. The clock network has a tree structure, comprising wire segments and switches that distribute clock signals with low skew. Closing a switch amounts to driving a clock signal into a particular region of the device. We propose making such switches capable of receiving an enable signal that when deasserted, disables clocking within the corresponding region. To our knowledge, this is the first work to consider a range of architectures for clock gating in FPGAs. The potential for clock gating as an FPGA power reduction strategy is underscored by the recently announced Xilinx Virtex-6 40 nm FPGA, which permits gating at some clock network switches [9].

Research on FPGA architecture goes hand-in-hand with research on computer-aided design (CAD) tools, as proper architectural evaluation demands CAD tools that leverage architectural enhancements. We present a placement approach whose aim is two-fold: 1) reducing routing capacitance on the clock network, and 2) placing design objects in a manner that takes advantage of the proposed clock gating architectures. Results show the effectiveness of our placement approach and demonstrate that considerable power reductions are possible with gating, depending on the granularity at which it is applied. The remainder of the paper is organized as follows: Section 2 describes related work on clock power reduction in FPGAs and clock gating. Section 3 introduces the proposed gating architectures and describes our placement approach. An experimental study is presented in Section 4. Conclusions and suggestions for future work appear in Section 5.

2. BACKGROUND AND RELATED WORK

2.1. Clock Architecture and Placement

Clock signals in FPGAs are distributed through a programmable clock tree. Fig. 1(a) shows a clock tree resembling that in the Xilinx Virtex-5 FPGA [10]. The device is partitioned into clock regions (4 regions are shown) such that a limited number of clock signals can be delivered into each region. As shown in Fig. 1, at the top level of the tree are the root spines, which deliver clock signals into regions. Root

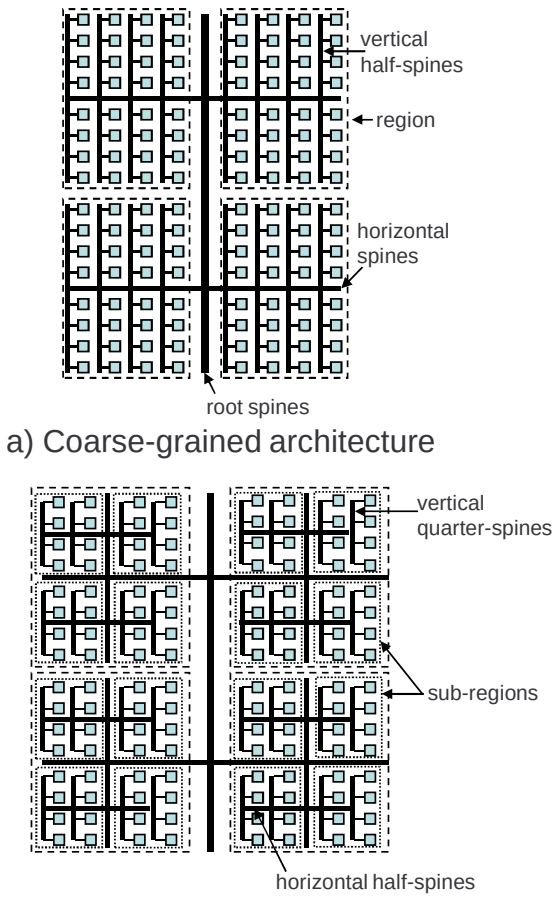

Fig. 1. Clock network architectures.

spines can connect to "horizontal spines" that span each region. Horizontal spines can connect to vertical "half-spines" that span half a region vertically. It is the vertical half-spines that ultimately route the clock signal to clock pins on logic blocks. Routing switches are present where the different spines intersect with one another and such switches are *only* closed on an as-needed basis. A clock signal would not be routed into a region containing no loads requiring the signal, as doing so would needlessly consume power.

Fig. 1(b) shows an alternative "fine-grained" clock architecture resembling that in the Altera Stratix-III FPGA [11]. In this case, the tree is deeper and there are 16 sub-regions, with 4 sub-regions in each of the quadrant regions. The sub-regions have the same structure as the quadrant regions in part (a) of the figure. Within sub-regions, horizontal "half-spines" can connect to vertical "quarter-spines" that route clocks to logic blocks. For both architectures in Fig. 1 the number of spines is monotonically decreasing with each level. For example, in the Virtex-5 FPGA, there are 32 root spines at the top level, however, there are only 10 horizontal spines in each region.

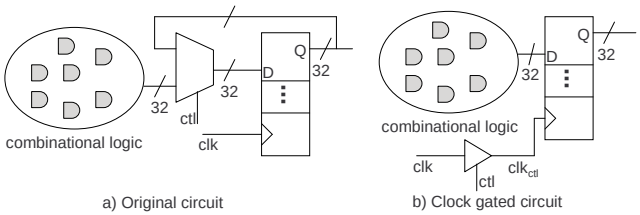

Fig. 2. Clock gating example.

The structured nature of the clock network and the switches between clock tree levels implies that placement of clock loads has a considerable impact on clock wire usage. Clock load placement can be done such that clock capacitance is minimized, reducing dynamic power. Lamoureux and Wilton were the first to propose clock-aware placement techniques that reduce the number of regions spanned by clock signals and reduce the resources used within regions [12, 13]. Since then, both Actel and Xilinx have published placement approaches for reducing clock spine usage for their respective architectures [8, 14]. Our placer borrows ideas from these prior works and builds upon them to explore clock gating architectures.

2.2. Clock Gating

Fig. 2 gives a straightforward example of clock gating. Fig. 2(a) shows a combinational logic circuit feeding a 32-bit register file through a multiplexer. The values in the register file are *only* updated with the combinational output when the multiplexer select signal ctl is asserted; otherwise, the register file content remains unchanged. Fig. 2(b) shows the circuit after clock gating is applied. The multiplexer is removed and the clock signal itself is gated with the ctl signal, producing a gated clock signal, clk_{ctl}. Let α_{ctl} represent the percentage of time that ctl is asserted on average, ranging from 0 to 1. The frequency of clk_{ctl} is $\alpha_{ctl} \cdot f_{clk}$, where f_{clk} is the frequency of the original clock signal. Power savings are realized through reducing switching activity on a portion of the clock network. With ctl deasserted, the clock signal no longer drives the capacitive load of the 32 register file clock pins. Relative to the case in Fig. 2, clock gating strategies can be considerably more sophisticated in real circuits and generally involve analysis of when registers may be kept idle and subsequent synthesis of clock gating (enable) signals. In this paper, we use the term *enable domain* to refer to the loads of a clock network that share a common clock enable signal.

Modern commercial FPGA architectures have some capability for clock gating. Stratix-III and Virtex-5 FPGAs have clock enable pins on the flip-flops in logic blocks; however, use of such pins does not provide switching activity reduction on the clock signal, and gives no power savings on the clock network [11, 10]. Aside from clock enable pins on flip-flops and other blocks, FPGAs provide gating at the top-level of the clock tree (as exploited in [8]). The top-level

gating shuts down the *entire* clock network and therefore, cannot be applied when some clock loads are used in ungated form and some are used in gated form, or when there exist multiple *enable domains* for a single clock signal. The recently introduced Xilinx Virtex-6 FPGA has clock gating capability on a regional basis [9] and Xilinx suggests that gating can save 30-80% of the clock tree power in some designs [15]. It is worth noting that use of clock gating is not limited to general LUT-based logic blocks; it also applies to the large IP blocks present in modern FPGAs, such as DSP blocks and multipliers.

Our goal is not to synthesize enable gating signals for clocks. Rather, in this work, such enable signals are already present in designs. Our goal is to study the power implications of different hardware architectures for implementing clock gating.

3. GATING ARCHITECTURES AND PLACEMENT

We envision that clock gating can be implemented naturally and with low-cost using the existing FPGA clock tree architecture. Traditionally, switches in the clock tree network are controlled by SRAM configuration cells. We propose to implement clock gating through a minor hardware modification whereby a *subset* of clock network switches can also be controlled by an enable signal. The enable signal would be received from general purpose interconnect. We do not believe adding an enable pin onto a fraction of clock tree switches will present an onerous routing demand. In fact, we believe that implementing clock gating through the clock network will lower routing demand vs. implementing gating through clock enable pins on flip-flops in logic blocks, as is often done today.

Table 1 shows five different clock gating architectures considered in this paper. The first gating option, `NONE`, is to have no gating at all, in which case, clock enables must be implemented using clock enable pins on logic blocks. The next two options, beginning with `CG`, pertain to the coarse-grained architecture in Fig. 1(a). `CG_REGION` permits gating when clock signals enter a region. That is, the switches in the clock network that connect from the root spine to the horizontal region spines have enable pins on them. `CG_COLUMN` represents a finer granularity of clock gating capability, wherein gating is possible *within* regions where the horizontal spines connect onto the vertical half-spines. The next two options, beginning with `FG`, pertain to the fine-grained architecture in Fig. 1(b). The gating options are analogous: `FG_REGION` permits gating where clock signals enter sub-regions; `FG_COLUMN` permits gating where the horizontal half-spines in sub-regions connect to vertical quarter-spines. The architectures are illustrated in Fig. 3. Fig. 3(a) shows the `REGION` architecture where enables are present on switches entering a region. Fig. 3(b) shows the more flexible `COLUMN` architecture where enables are also present on switches driving vertical spines in logic block columns. Thus, we consider a broad range of clock gating architectures with various levels of granularity, within clock distribution frameworks that resemble those in commercial chips.

Table 1. Clock gating architectures considered.

Gating Style	Description
NONE	No gating
CG_REGION	Gating on a region basis
CG_COLUMN	Gating on half-columns in regions
FG_REGION	Gating on a sub-region basis
FG_COLUMN	Gating on quarter-columns in sub-regions

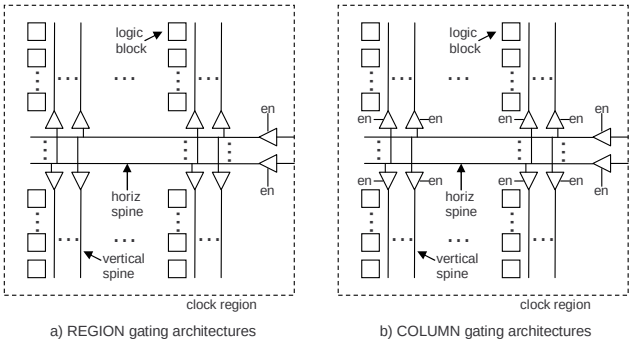

Fig. 3. Clock gating architectures.

For this work, we assume that each level of wire on the clock network tree can be driven by all wires in the preceding level, as shown in Fig. 4(a). This allows us to create multiple gated forms of a single clock at the next lower level of the tree, as illustrated in Fig. 4(b). This property is useful when there exist multiple enable domains for a single clock signal. In practice, we believe the same results can be achieved with substantially depopulated switches. Clock networks in commercial chips likely have partially depopulated switches. In the Xilinx Virtex-5, since any 10 of 32 clocks can be driven into any region, there must exist considerable routing flexibility between the top and regional levels. On the other hand, we expect there is less flexibility in the switches within a region, from the horizontal spines to the vertical spines. Note that our flexibility assumptions are not equivalent to a full crossbar, as the clock tree is unidirectional from root to leaves.

3.1. Placement

We use a modified version of the VPR placer [16] to explore clock gating architectures. VPR's placer algorithm uses simulated annealing to optimize a composite cost function comprising wirelength and timing performance. We augmented the cost function to incorporate clock network power:

$$Cost = \rho \cdot WL + \sigma \cdot T + \upsilon \cdot CP \qquad (1)$$

where WL, T and CP represent wirelength, timing and clock power, respectively, and the other parameters are scalar

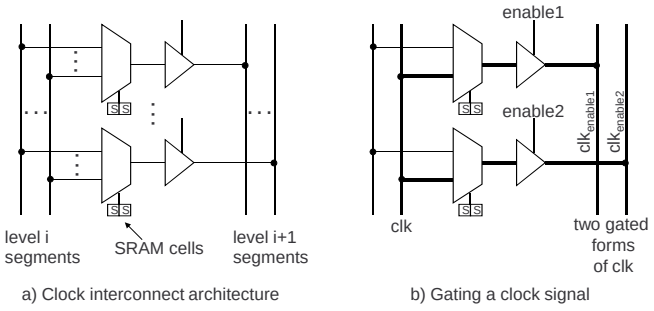

Fig. 4. Clock routing architecture.

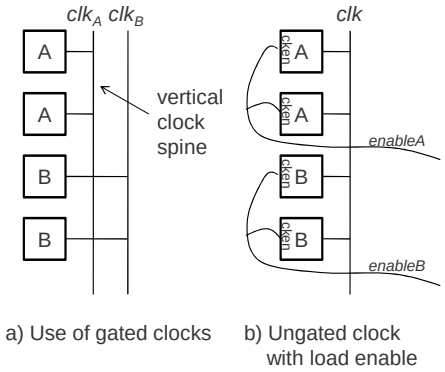

Fig. 5. Enable implementation choices.

tuning weights reflecting the importance of each term. For a given placement, we can compute the clock power term in (1) as the sum of the power consumed by each clock signal in the design: $CP = \sum_{clk \in clks} CP_{clk}$. The power for a particular clock signal clk, CP_{clk}, can be computed based on the current placement and the properties of the FPGA clock network architecture being evaluated. The structured nature of the clock network allows us to rapidly compute a clock routing during placement, and thereby estimate clock capacitance and power.

Computing CP_{clk} is more involved than is readily apparent. For clock loads within an enable domain of clk, there is a choice: 1) use gated clock signals created via clock enables on switches within the clock distribution network, or 2) use an ungated clock signal along with clock enable pins on the load blocks themselves. It is not always beneficial to use the enables within the clock network (option #1). The decision taken is the one that yields lower power consumption. To see this, consider the column placement shown in Fig. 5(a) comprising four logic blocks requiring two enable domains of clk, A and B. Option #1 uses gated clocks and is shown in Fig. 5(a). Option #2 uses an ungated clock and enable pins on the blocks themselves and is shown in Fig. 5(b). If C_v is the capacitance of a vertical spine, the clock power consumption of Fig 5(a) is $f_{clk} \cdot \alpha_A \cdot C_v \cdot V^2 + f_{clk} \cdot \alpha_B \cdot C_v \cdot V^2$ Watts, whereas the clock power consumption of Fig. 5(b) is $f_{clk} \cdot C_v \cdot V^2$ Watts. Comparing these power values, option #1 should be chosen only if $\alpha_A + \alpha_B < 1$; otherwise, we should use an ungated clock. By extension, if a column contains a placed load that uses the ungated form of clk, then all loads in the column, including those in the gated domains, should use the ungated clock to minimize overall power.

Fig. 6 gives a generalized core procedure we apply in the process of computing CP_{clk}. For a clock signal clk in region R, the procedure computes a set of columns U in R that will be fed by clk in ungated form, using a single vertical spine per column [as in Fig. 5(b)]. Parameter COL represents the set of columns in R that have a load of clk placed on them (including both gated and ungated loads). Line 2 initializes U to contain the columns of R where ungated loads of clk are placed. These columns *must* receive clk in ungated form.

```
1: set COL: all columns in R that use clk
2: set U: all columns in R with ungated loads
          of clk
3: for each column c in COL − U
4:   set E: all enable signals used in c
5:   T = Σ_{e ∈ E} α_e
6:   if T ≥ 1 then
7:     add c to U
```

Fig. 6. Column procedure for clk in region R.

Lines $3-7$ walk through the columns of R where gated loads of clk are placed, and for each such column, sums the α's of the enable signals needed in the column. If the sum exceeds 1, then the column is added to U and clock power is minimized by using clk ungated in the column, similar to Fig. 5(b).

The approach for determining the clock routing and computing CP_{clk} differs depending on where clock enables are available in the clock distribution network – region-based enables (`CG_REGION`, `FG_REGION`) or column-based clock enables (`CG_COLUMN`, `FG_COLUMN`).

3.1.1. Region-Based Clock Enable Architectures

We execute the procedure in Fig. 6 for each clock region, after which we have a set of columns U receiving ungated clk, and a set of columns, $COL - U$, receiving gated forms of clk. For the region-based clock enable architecture, gating is only available at the entry points of regions and therefore, the gated clocks used on columns of a region must also be routed on horizontal spines in the region. We have not yet accounted for the power consumed on the horizontal spines, and when accounted for, it may become advantageous to move columns from $COL - U$ into U and thereby use ungated clocks in such columns.

Consider the case of an enable signal e used by a column in $COL-U$ in region R. If C_h is capacitance of a horizontal spine in the region, then routing clk_e will consume $f_{clk} \cdot \alpha_e \cdot C_h \cdot V^2$ Watts of power on a horizontal spine. This horizontal "cost" must be less than the vertical "benefit" of using the gated clock clk_e in the region. So, what is the benefit of

```
1: set B: columns in COL − U that use e
2: Cost = C_h · α_e
3: Benefit = 0
4: for each column c in B
5:   set E: all enable signals used in c
6:   Benefit += (C_v − Σ_{e ∈ E} C_v · α_e )
7: If Benefit < Cost then
8:   U = U ∪ B
```

Fig. 7. Procedure for computing the cost and benefit of gated clock clk_e in region R.

the gated clock clk_e? The benefit of clk_e is power reduction on the vertical columns that use clk_e vs. using an ungated clock on those columns. If the cost exceeds the benefit, all columns using clk_e are moved into U and gated clock clk_e is not used in the region: the loads of the enable domain in the region will be driven by clk in ungated form and signal e is supplied directly to the loads [akin to Fig. 5(b)].

Fig. 7 shows the procedure in detail. We walk through the gated clock domains used in the region and for each such enable domain e, we call the procedure in Fig. 7, which may move columns fed by clk_e into U based on cost/benefit analysis. This is a heuristic approach as it depends on the order in which we consider the enable domains. We may decide to leave a domain as gated, and then columns fed by that domain are later moved into U, thereby reducing a benefit observed earlier. We deal with this by walking through the enable domains multiple times, and found two walks to be sufficient in practice.

There is an edge case worth mentioning which is that of having no ungated load within a region. In this scenario, it is not mandatory for a horizontal spine carrying the ungated clock to be present. So, we compute the clock power in two ways: with and without the presence of a horizontal spine carrying the ungated clock. We choose the implementation with the lower power.

3.1.2. Column-Based Clock Enable Architectures

The added flexibility provided by gating capability at column entry simplifies the clock power computation. For each region, we execute the column procedure in Fig. 6. If U is non-empty, then there exists one column in the region that requires clk ungated, in which case, only a single horizontal spine, carrying ungated clk is used in the region and all gated clocks required on columns are generated at the horizontal to vertical spine switches. On the other hand, if U is empty, we sum the α's of all enable domains in the region, and if the sum is ≥ 1, we route a single ungated clock horizontally in the region; otherwise, we route all gated forms required for the region on horizontal spines. We recursively repeat this decision process, traversing the clock tree upwards towards the root.

3.1.3. Other Placement Considerations

There are a few additional points that warrant discussion. First, regarding the power consumed by clock enable signals themselves, we believe enables toggle relatively infrequently and are therefore not usually a significant power drain. However, enables can have high-fanout and migrating such signals from logic blocks to enable pins on routing switches in the clock distribution network may reduce routing congestion and improve performance. Second, we reinforce that even for the `NONE` architecture, our placer is clock power-aware, and will still attempt to minimize CP in (1) by way of reducing overall spine usage, as is done in prior works [14, 8, 13].

To gauge clock network power in placement, we need to know the clock interconnect capacitances, e.g. C_v and C_h. Precise interconnect capacitance values are considered proprietary by the commercial vendors. Therefore, in this study, we assume a square FPGA die and that C_v and C_h are proportional to their lengths. For example, in the coarse-grained architecture of Fig. 1(a), the horizontal spines span half the die width and the vertical spines in regions span a quarter of the die width, making $C_h = 2 \cdot C_v$. The same approach is applied recursively to compute capacitance values for the smaller spines in the fine-grained architecture shown in Fig. 1(b). Through this modeling approach, even without precise capacitance values for a given wire segment on the clock network, we are able to compute and compare clock power consumption *relative* to a baseline architecture. Aside from the modeling used here, it should be apparent to the reader that the proposed placement approach is a general one, and can be used with arbitrary clock capacitance values.

4. EXPERIMENTAL STUDY

To our knowledge, there are no public benchmark circuits that contain clock enables. Consequently, in this study, we use the circuits from [13], which are large sequential circuits created by stitching together multiple MCNC circuits. More details on the circuits can be found in [13]. We synthetically altered the circuits to contain clock enables. For each clock domain in each benchmark circuit, we partitioned the flip-flop loads into four equal-sized groups. The first group of loads were designated to receive the clock signal in ungated form. The three remaining groups form three separate enable domains for the clock. We believe a small number of enable domains per clock is reasonably representative of actual circuits with clock gating.

Regarding the α values for the clock enables, we generated two sets of α's corresponding to *limited* and *substantial* clock gating. The limited set contains high α values where flip-flops are enabled most of the time, and we expect smaller power benefits. For the three enable domains, we use α values of 0.5, 0.8 and 0.6 for the limited gating runs. The substantial gating α's correspond to lower clock activity. We use α values of 0.2, 0.1 and 0.1 for the three enable domains in the substantial gating runs. In a commercial tool, the α values could be extracted from a simulation

of the design with realistic vectors, or could alternatively be provided by the user based on their intuition of the fraction of time circuit blocks are disabled.

We target an FPGA architecture with 10 4-LUT/flip-flop pairs per logic block and length-4 wire segments. Packing was enhanced to ensure no more than two enable domains were packed into a logic block (similar to the Xilinx Virtex-5 FPGA [10]). For each circuit, we determined the minimum number of tracks per channel, W_{min}, needed to route the circuit in the unmodified VPR. For each clock gating architecture studied, each circuit was routed into an architecture with $1.3 \times W_{min}$ tracks per channel. In this way, we held the routing fabric invariant for each circuit across all architectures, thereby allowing us to see the impact of our placement changes.

4.1. Results

Table 2 gives the results for all clock gating architectures considered. The first column lists the benchmark circuits. The remaining columns give power and wirelength results for each clock gating architecture. For each architecture, there are two columns of results: one column showing the percentage reduction of clock power relative to the `NONE` architecture, and one column showing the impact of gating-aware placement on the total bounding box wirelength of logic signals (non-clock signals). Values in the table are percentage changes relative to baseline results for a given architecture. Recall that the baseline has no clock gating capability within the clock distribution network, however, clock power is optimized by reducing clock spine usage. Consequently, the values in the table can be interpreted as the amount of benefit/cost afforded by a particular clock gating architecture versus a baseline architecture with no built-in gating capability in the clock distribution network.

The left-side (unshaded columns) of Table 2 give results for the runs with limited clock gating (high α values). Considering first the results for the coarse-grained architectures (`CG` columns), observe that average clock power reductions for `CG_REGION` and `CG_COLUMN` are -0.2 and 12%, respectively. Meaning, power was essentially unchanged for the `CG_REGION` architecture vs. the baseline of implementing clock gating with enable pins on logic blocks. The benefit of added flexibility afforded by the `COLUMN` architecture is apparent in the considerable (12%) power reduction achieved. Coupled with the power changes, the wirelength of logic signals increased by 1.3 and 2.6%, respectively, likely leading to longer wirelengths and more power for logic signals (discussed below). Reducing power on the clock network in placement aligns clock loads on spines, damaging bounding box wirelength. For the fine-grained (`FG`) architectures, a similar trend is apparent. When gating is only possible on a region basis, power actually increases vs. the baseline architecture. When gating is possible on columns driving logic blocks in sub-regions, clock power is reduced by 16%.

The right-side of Table 2 (shaded columns) gives results for runs with substantial clock gating (low α values). Observe that clock power reductions are close to 50% across all 4 gating architectures. For `CG_REGION`, `CG_COLUMN`, `FG_REGION` and `FG_COLUMN`, clock power reductions are 54, 58, 47 and 51%, respectively. We conclude that clock activity has a significant impact on the power benefits of clock gating and techniques for gating should be used judiciously, based on the fraction of time clocks are enabled. One can see that higher power reductions are achieved with the `COLUMN` architectures, which is intuitive as these architectures have have finer-grained gating flexibility. Moreover, the `CG` architectures provide larger power benefits vs. the `FG` architectures. This is due to the increased depth of the clock tree in the `FG` architectures, leading to more clock metal capacitance overall in both baseline and experimental.

To estimate the overall dynamic power benefit in an FPGA tile, we apply the results in [2] which demonstrate clock power to be $\sim$20% of total power in an FPGA tile, and logic signal power to be $\sim$60% of total power. Using this breakdown, combined with the data in Table 2, we can estimate the overall power benefit of clock gating. Note that we presume logic signal power will rise in proportion to routed wirelength increases reported in Table 2. Results of our analysis appear in Table 3. Observe that there is little-to-negative power benefit when clock activity is high (limited gating runs). On the other hand, total estimated power reductions range from 6.2-7.7% when clock activities are low (substantial gating runs). Such power benefits, while modest, are for clock gating alone and we believe that gating is a useful tool in the arsenal of circuit and CAD techniques that can be brought to bear in building a low-power FPGA solution. It is also important to realize that in this study, we arbitrarily assigned clock loads to different enable domains. We expect that enable domains in real circuits will exhibit more locality, thereby mitigating the increases to bounding box wirelength of logic signals.

Regarding impact on critical path delay, we also analyzed the estimated critical path delay reported by VPR post-routing. Considering all architectures and clock activity scenarios, critical path delay increased by 0-2%, on average, when the CP term in (1) was incorporated (i.e. relative to VPR as described in [16]). This is to be expected as the goal of locating cells to minimize clock power may collide with the goal of locating cells to minimize critical path delay. Many FPGA applications do not require maximum device performance. We therefore expect the power reductions afforded by clock gating will prove desirable in power-sensitive applications.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we considered clock gating for power reduction in FPGAs and propose that clock gating capability be

Table 2. Clock gating power and wirelength results for limited and substantial (shaded) clock gating.

Architecture	CG_REGION		CG_COLUMN		FG_REGION		FG_COLUMN		CG_REGION		CG_COLUMN		FG_REGION		FG_COLUMN	
Circuit	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase	Clock Power Reduction	Wirelength Increase
10lrg	-2.58	1.90	10.41	2.26	-2.48	2.59	16.18	4.24	52.12	6.34	57.75	6.51	48.38	4.86	49.14	6.45
10lrg_reordered	-0.79	2.60	9.02	2.14	-2.36	2.82	14.34	2.60	53.63	7.72	57.13	5.95	47.01	5.86	52.42	7.11
2lrg4med80sml	-1.10	1.53	6.65	1.39	-1.10	1.88	15.22	3.34	50.14	6.04	55.72	7.17	46.38	6.23	50.11	6.95
30med	2.90	4.81	21.03	6.03	-5.42	6.26	14.58	5.38	55.57	10.16	61.26	9.53	50.35	7.57	49.58	6.03
30mixedsize	-1.85	-0.34	11.53	1.76	0.00	0.11	13.51	1.70	49.69	5.83	54.56	4.91	41.84	2.92	47.43	4.57
30seq20comb	-1.16	0.33	15.25	0.89	-10.11	-0.10	17.21	2.45	60.56	3.58	62.96	2.64	54.18	3.24	55.50	3.96
3lrg50sml	-0.72	1.40	9.20	4.12	-2.48	3.05	12.35	5.63	51.24	8.32	56.23	7.69	41.98	6.95	48.71	7.47
40mixedsize	-1.04	-0.04	15.24	1.83	2.65	0.44	19.94	3.05	57.21	6.49	61.96	5.43	48.98	3.73	54.64	4.61
4lrg4med16sml	-2.08	0.50	8.98	2.57	0.00	0.14	16.92	2.49	53.78	6.92	57.68	6.93	47.37	4.94	51.35	5.46
4lrg4med4sml	-2.68	-0.65	10.31	1.91	-1.03	0.21	14.07	2.64	55.13	6.13	57.83	5.77	46.87	3.99	49.92	3.68
50mixedsize	-2.39	-2.54	8.47	-0.20	-0.73	-0.49	18.19	1.32	53.68	3.35	55.75	4.56	48.29	2.90	51.71	3.90
5lrg35sml	-0.16	2.08	14.32	2.72	1.97	1.98	17.16	3.70	56.21	7.53	59.11	7.73	45.79	5.24	50.03	6.14
60mixedsize	-0.92	-2.74	9.23	-2.07	-4.93	-1.66	16.04	-0.55	47.50	1.31	53.55	1.51	42.44	1.64	48.79	2.35
6lrg60sml	1.77	2.08	9.21	3.76	1.77	2.00	17.75	4.64	53.63	8.38	57.78	8.00	48.67	7.30	51.87	7.37
70sml	3.11	3.54	15.97	4.23	-4.16	3.28	15.46	4.24	53.96	9.64	59.90	8.22	49.41	6.11	51.72	5.83
8otsaffs	1.19	5.92	17.28	8.30	2.41	6.64	19.72	8.82	56.88	11.31	62.32	13.12	49.33	10.88	52.80	10.03
8otsaffs2	4.54	1.69	12.31	3.21	-0.63	0.86	14.33	4.85	54.09	7.68	56.77	6.71	46.19	6.52	51.27	6.77
Average	**-0.23**	**1.30**	**12.02**	**2.64**	**-1.57**	**1.76**	**16.06**	**3.56**	**53.83**	**6.87**	**58.13**	**6.61**	**47.26**	**5.35**	**51.00**	**5.80**

Table 3. Estimated overall FPGA power reduction.

Architecture	Power Reduction % (Limited Gating)	Power Reduction % (Substantial Gating)
CG_REGION	-0.8	6.6
CG_COLUMN	0.8	7.7
FG_REGION	-1.4	6.2
FG_COLUMN	1.1	6.7

incorporated at various points in the clock distribution network. We presented a placement algorithm that incorporates clock power and locates clock loads accordingly to take advantage of clock gating architectural enhancements. To our knowledge, this work represents the first study of clock gating architectures in FPGAs. Results show that depending on the architecture and clock activity, clock power can be reduced by over 50%.

Future work will involve incorporating the clock gating-aware placement algorithm into a complete power-aware FPGA CAD flow, allowing trade-offs between other power-aware tool phases to be explored. Another future direction involves modeling clock gating within the VPR router to permit extraction of accurate post-routing critical path delay and interconnect capacitance. It will also be important to understand the impact of gating on timing analysis to ensure functional correctness is maintained when enable signals are migrated from flip-flops to enable pins on clock network switches.

6. REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. On CAD*, 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/SIGDA Int'l Symp. on FPGAs*, 2002, pp. 157–164.

[3] D. Lewis, E. Ahmed, D. Cashman, T. Vanderhoek, C. Lane, A. Lee, and P. Pan, "Architectural enhancements in Stratix-III and Stratix-IV," in *ACM/SIGDA Int'l Symp. on FPGAs*, 2009, pp. 33–42.

[4] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90nm low-power FPGA for battery-powered applications," *IEEE Trans. On CAD*, vol. 26, no. 2, pp. 296–300, Feb. 2007.

[5] G. Yeap, *Practical Low Power Digital VLSI Design*. Boston, MA: Kluwer Academic Publishers, 1998.

[6] Q. Wang and S. Roy, "Power minimization by clock root gating," in *IEEE/ACM ASP-DAC*, 2003, pp. 249–254.

[7] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *IEEE/ACM DAC*, 2003, pp. 622–627.

[8] Q. Wang, S. Gupta, and J. H. Anderson, "Clock power reduction for Virtex-5 FPGAs," in *ACM/SIGDA Int'l Symp. on FPGAs*, 2009, pp. 13–22.

[9] *Virtex-6 Family Overview*, Xilinx, Inc., San Jose, CA, 2009.

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

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

[12] J. Lamoureux and S. Wilton, "FPGA clock network architecture: flexibility vs. area and power," in *ACM/SIGDA Int'l Symposium on FPGAs*, 2006, pp. 101–108.

[13] ——, "Clock-aware placement for FPGAs," in *IEEE Int'l Conf. on Field-Programmable Logic and Applications*, 2007, pp. 124–131.

[14] 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 FPL*, 2008, pp. 233–238.

[15] *White paper: Power consumption at 40 and 45 nm*, Xilinx, Inc., San Jose, CA, 2009.

[16] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Int'l Workshop on FPL*, 1997, pp. 213–222.