Power-Aware Placement

Yongseok Cheon*, Pei-Hsin Ho*, Andrew B. Kahng†, Sherief Reda†, Qinke Wang†

*Advanced Technology Group, Synopsys, Inc.

†CSE Department, University of California at San Diego

{cheon,pho}@synopsys.com, {abk,sreda,qiwang}@cs.ucsd.edu

ABSTRACT

Lowering power is one of the greatest challenges facing the IC industry today. We present a power-aware placement method that simultaneously performs (1) *activity-based register clustering* that reduces clock power by placing registers in the same leaf cluster of the clock trees in a smaller area and (2) *activity-based net weighting* that reduces net switching power by assigning a combination of activity and timing weights to the nets with higher switching rates or more critical timing. The method applies to designs with multiple clocks and gated clocks. We implemented the method and obtained experimental results on 8 real-world designs after placement, routing, extraction and analysis. The power-aware placement method achieved on average 25.3% and 11.4% reduction in net switching power and total power respectively, with 2.0% timing, 1.2% cell area and 11.5% runtime impact. This method has been incorporated into a commercial physical design tool.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS—*Design Aids*; J.6 [**Computer Applications**]: COMPUTER-AIDED ENGINEERING

General Terms

Algorithms, Design, Performance

Keywords

Net Switching Power, Clock Tree, Dynamic Power

1. INTRODUCTION

Temperature profile and battery life requirements for tethered and un-tethered systems have made power consumption a primary optimization target for IC designs. IC power consumption consists of three basic components: short circuit power, leakage power and net switching power [12]. *Short circuit power* is the power dissipation that happens briefly during the switching of a logic gate and *leakage power* is the power dissipation due to spurious currents in the non-conducting state of the transistor.

Net switching power is often the largest source of the total power dissipation. Net switching power dissipation of a net can be modeled as $kCV^2\alpha$, where k is a constant, C is the total capacitance that is to be charged and discharged (including both wire capacitance and gate input pin capacitance), V is the supply voltage and α is the *switching rate*; i.e., the number of switching events per unit time. Therefore, net switching power is proportional to the product $C\alpha$ of the total capacitance and the switching rate. The power-aware placement method in this paper aims at reducing net switching power by reducing the product $C\alpha$.

Clocks switch much more frequently and drive much larger capacitances then most signal nets. Hence it is not surprising that the clock networks typically consume up to 40% of the total power across a variety of design types [6, 8, 13, 18].

Despite the fact that the placement of the registers directly impacts the overall clock-tree power, virtually all conventional placement methodologies treat registers no differently than combinational cells. We believe that this leads to sub-optimal placements in terms of power. To our knowledge there is no work in the literature that proposes a register placement technique for the purpose of low power.

In this paper, we present a power-aware placement method that performs both *activity-based register clustering* and *activity-based net weighting* to simultaneously reduce the clock and signal net switching power.

Activity-Based Register Clustering The goal of activity-based register clustering is to reduce the capacitance of clock nets. Assigning larger net weights to some nets in the placer is a well-known method for reducing the lengths of these nets. However, simply assigning a very large weight to a clock net (to combat the total weight of all nets connecting to all driven registers) during placement is not a good idea for reducing clock-tree capacitance since placing all the registers close to the clock source may introduce hot spots and highly congested areas on the chip.

Figure 1 shows the distribution of clock-tree capacitance on an industrial design. The *level* of a segment of the clock tree is the minimum number of buffers between the segment of the clock tree and a clock sink. For example, level 0 includes all the clock sinks and the wires connecting them and the driving buffers. The figure shows the total wire capacitance and pin capacitance for each level of the clock tree. Since most of the clock tree capacitance (about 80% for this specific design) is at the leaf level, an effective way of reducing clock tree capacitance is to reduce the capacitance

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

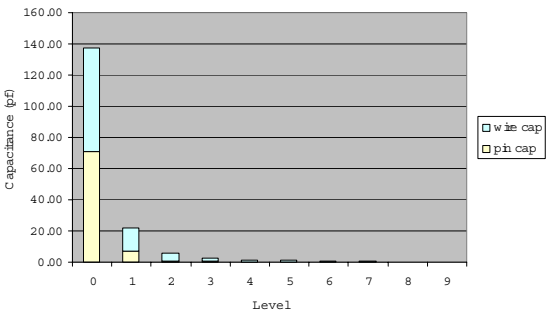


Figure 1: Distribution of clock tree capacitance on a customer design.

at the leaf level. The register clustering technique clumps the registers within the same leaf cluster of the clock tree into a smaller area, which reduces the leaf-level wire capacitance and potentially the skew. Note that clock skew may introduce extra clock buffers and thus extra capacitance.

We have observed that register clustering can effectively reduce the capacitance of the leaf-level clock tree, but it often increases the length of some signal nets and thus the net switching power of the signal nets. This may cancel out the power reduction attained by register clustering. Figure 2(a) shows a conceptual layout of a clock tree and the nets incidental to the two registers driven by a clock buffer. In Figure 2(b), we apply the technique of register clustering and placed the two registers closer to each other. As a result, the two nets incidental to the registers become longer.

Activity-Based Net Weighting Assigning a large weight to the signal nets with higher switching rates reduces the total net switching power. As shown in Figure 2(c), we assign a larger weight to the net incidental to the left register than the one incidental to the right register based on their switching rates. As a result, the register cluster is placed closer to left and thus the total net switching power is reduced. We have found that combining activity-based register clustering and activity-based net weighting further reduces the total net switching power.

We implemented the power-aware placement method on the framework of Synopsys IC Compiler and tested the implementation on eight real-world designs in terms of power, timing, cell area and runtime. The experimental data are obtained after running the designs through a complete physical design flow including physical synthesis, clock-tree synthesis, global route, detailed route, extraction and timing and power analysis. We have observed that the power-aware placement method on average achieved 25.3% reduction in total net-switching power and 11.4% reduction in total power, with 2.0% timing, 1.2% cell area and 11.5% runtime degradation.

The organization of this paper is as follows. Section 2 summarizes known approaches to net switching power minimization. Sections 3 and 4 describe the techniques of activity-based register clustering and activity-based net weighting. Experimental results are reported in Section 5 using a complete industrial flow. The paper concludes in Section 6.

2. PREVIOUS WORK

Assigning net weights according to switching activities is commonly applied to reduce total net switching power [3, 14, 17, 19, 21]. For example, in a recent paper on temperature-aware placement [14], net weights are assigned proportional to the product of switching rate and pin count, in order to reduce net switching power.

Traditional clock-tree construction methods [4, 10, 11] focus on minimizing clock-tree wirelength or clock skew. On the other hand, more recent studies [6, 13, 8, 18] agree that clock trees are the largest consumers of power in microprocessors, and a number of techniques have been proposed to reduce clock-tree power including *clock gating* [2, 6, 7, 15], *buffer sizing* [1, 20], and *multiple-supply voltage* [9, 16].

Clock power can be saved by disabling clock signals from inactive flip-flops in idle circuit parts. Through the insertion of control gates and control signals in the clock tree, one can shut down the clock in selected subtrees, and save a substantial amount of power. Work in this area focuses on calculating the active/idle periods of different flip-flops and inserting the gating logic into the netlist [2, 6]. Since flip-flops that should be gated together may be placed far apart, it is possible that gating control signals will end up increasing routing and power demands of the clock tree. To overcome this, Farrahi *et al.* [7] suggest tying commonly-gated flip-flops together by fake nets; this biases the placer to place such flip-flops closer together, hence reducing the wiring overhead of gating logic. However, the possible increase of signal wire length (not counting the fake nets) is not mentioned in [7], and no empirical data on realistic benchmarks are given. Another line of research seeks to save clock power through several simultaneous optimizations, e.g., wire and buffer sizing [5], as well as simultaneous clock tree construction and buffer insertion [20].

Our work aims at reducing the net switching power of not only clock nets but also signal nets at the placement stage without modifying the netlist or supply voltages in any way.

3. REGISTER CLUSTERING

In this section, we present a register placement method that reduces the capacitance of the leaf level of the clock tree by clumping the registers in each leaf cluster of the clock tree closer.

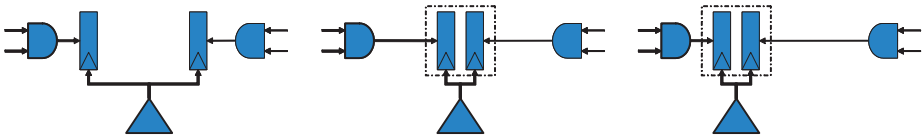
3.1 The Quick Clock-Tree Synthesis Algorithm

Given an existing coarse placement of the design, the first step of register clustering is to group registers into clusters such that each cluster can become a leaf cluster of the actual clock tree. We designed the Quick CTS algorithm in Figure 3 for this purpose.

First of all, the Quick CTS algorithm decides a scope of target cluster size heuristically according to the size of the clock net, the DRCs (Design Rule Constraints) such as max fanout and max load constraints, and user configuration. The clustering algorithm is performed for each clustering direction (as explained in the following paragraph); among all the clustering results returned, the one with the best CTS objective (minimum clock skew, at default) is selected.

Basically, the clustering algorithm starts with the leftmost (rightmost, highest or lowest) clock pin and regards it as the current cluster. The algorithm adds to the current cluster the clock pin with the shortest Manhattan distance to the capacitance weighted centroid of the current cluster.[1] The

[1]Capacitance weighted centroid is the geometric centroid of the clock pins in a cluster, weighted by pin capacitance.



(a) Before register clustering (b) After register clustering (c) After register clustering and net weighting

Figure 2: An example layout (a) before register clustering, (b) after register clustering, and (c) after register clustering and net weighting.

Quick CTS

Input:
A set of N clock pins $\{s_i\}$ with pin capacitance $\{p_i\}$
Placement of clock pins $\{(x_i,\ y_i)\}$
DRC constraints: max load L and max fanout M
User-specified cluster size K_u

Output:
A set of clusters $\{C_j\}$

Algorithm:
01. Decide a target cluster size K according to N, M and K_u
02. **For** each clustering direction d in $\{Leftmost, Rightmost, Highest, Lowest\}$
03. Un-clustered pins $U = \{s_i\}$
04. **While** $U \neq \emptyset$
05. Find the outermost pin $s \in U$ along direction d
06. The current cluster $C = \{s\}$ and $U = U - \{s\}$
07. **While** $|C| < K$ and $total\_pin\_cap(C) < L$
08. Find pin $s \in U$ to min. $distance(s,\ C)$
09. $C = C + \{s\}$ and $U = U - \{s\}$
10. Generate a cluster C
11. Generate a clustering result $\{C_j\}$
12. Compute the skew of $\{C_j\}$
13. Keep the best clustering result till now
14. Output the best clustering result $\{C_j\}$

Figure 3: Quick CTS

current cluster grows until it reaches the target cluster size or the max load limit. Then the next cluster starts with the leftmost (rightmost, highest or lowest) clock pin among all un-clustered ones and grows. The algorithm repeats growing clusters until all registers are clustered.

Note that the clustering result of the Quick CTS algorithm may be different from the result of any particular CTS algorithm. The idea is that after register clustering, since the registers in each cluster would be placed close to each other, most CTS algorithms would identify similar leaf clusters.

3.2 Group Bounds

After registers are grouped into clusters, the second step of register clustering is to place registers of the same cluster closer to each other. A naive method is to add a pseudo net to connect the registers and assign it a large weight. However, this does not reduce the wire length effectively for large nets like the clock nets

All industrial placers and some academic placers have the capability to constrain the placement of specified groups of objects within specified bounding boxes. These *group bounds* control the bounding box of the cluster and reduce it as much as desired while still fitting the registers. In our approach, we define a group bound for each cluster and transfer it to the actual placer, so that it constrains the registers inside the specified bounding box. We first determine each group's bounding box based on registers' current locations. Then we shrink the bounding box proportionally by a factor of p and use it as the group bound.

A design may have multiple clock nets. Not all of them have the same switching rate. It is also possible that part of the clock net is gated and has a lower switching rate. The shrink ratio of a group of registers should be decided based on the switching rate of the clock net: if the switching rate SR is relatively small compared to the Maximum Switching Rate MSR, the bounding box of the cluster should shrink less or not shrink at all. We determine the actual shrink ratio p according to the following formula:

$$p = \begin{cases} 1 - (1 - p_0)(SR/MSR) & if\ SR > 0.3\ MSR \\ 1 & otherwise \end{cases} \quad (1)$$

where p_0 is a user specified shrink ratio or a heuristically determined parameter based on the size of the design. According to the formula, a clock net with the maximum clock switching rate has the minimum shrink ratio p_0, and a clock net with a switching rate less than 30% of the maximum clock switching rate does not shrink ($p = 1$).

When the shrink ratio p is close to 1, the new bounding box should have an aspect ratio near to the original one, so that the wire length of signal nets are not affected seriously without much reduction of the clock wire length. However, with a small shrink ratio, it does not matter so much for the new bounding box to keep the original aspect ratio. We would like the bounding box to have an aspect ratio close to 1 in order to reduce the clock skew. Therefore, we use a linear function to decide the aspect ratio AR_{new} of the new bounding box based on the original aspect ratio AR_{old} and the shrink ratio p.

$$AR_{new} = 1 - p + p \cdot AR_{old} \quad (2)$$

We implement the activity-based register clustering method on the framework of Synopsys IC Compiler. IC Compiler placer encompasses multiple passes. During each pass, the clustering algorithm is performed to group registers into clusters based on the current layout, and then group bounds are generated for each cluster. The placer adjusts the layout of registers accordingly by clumping registers more closely. After each pass, a new placement is obtained with a further reduced clock tree and the old group bounds are discarded.

4. ACTIVITY-BASED NET WEIGHTING

Register clustering can be applied to effectively reduce clock capacitance and thus clock power dissipation, with a sacrifice of signal net wire length and switching power. In this section, we assign a combination of activity and timing based weights to signal nets to reduce the capacitance of nets with higher switching rates or more critical timing, so that the impact of register clustering on signal net switching power and design performance is alleviated and switching power of the design is further reduced.

We assign power weights to nets as shown in Figure 4. For a signal net with switching rate SR, the power weight

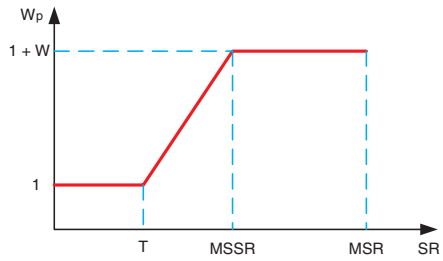


Figure 4: Power weight as function of net switching rate.

w_p is

$$w_p = \begin{cases} 1 + W \cdot \frac{SR-T}{MSSR-T} & if\ MSSR > SR > T \\ 1 & if\ SR \leq T \\ 1 + W & SR \geq MSSR \end{cases} \quad (3)$$

where T is a switching rate threshold for selecting high activity nets, $MSSR$ is the maximum signal net switching rate of the design, and W controls the scope of power weights: the maximum power weight is $1 + W$. The clock net has the largest switching rate, much larger than the switching rates of most signal nets. Since register clustering is applied to reduce clock power and here our focus is common signal nets, we assign the maximum power weight to the nets with the highest switching activities among signal nets, as well as clock nets with higher switching activities.

Besides activity-based power weights, nets are also assigned weights according to their timing criticality in commercial placers. A linear combination of power weight w_p and timing weight w_t is applied and the final weight w is:

$$w = \alpha \cdot w_p + (1 - \alpha) \cdot w_t \quad (4)$$

where the power ratio α is value between 0 and 1. It controls the ratio of power weight to the final net weight, and provides a knob to trade-off between timing and power objectives for the placer.

We analyze power consumption using IC Compiler that approximates the switching rate for each net from the timing constraints for the design (that include the specification of all clocks behaviors) plus any one of the following three sources: (1) a VCD file that records the waveform of the design during the simulation of the gate-level design, (2) a SAIF file that profiles the switching rates of the nets during the simulation of the gate-level design or (3) the switching rates of all primary input signals of the design. Given the simulation models of the technology libraries, during the simulation-based functional verification, all commercial Verilog simulators can create the VCD file and some can produce the SAIF file. If the source is the specification of the input switching rates, IC Compiler performs a probabilistic simulation internally to estimate the switching rates for all nets based on the clock definitions and the switching rates of the primary inputs.

5. EXPERIMENTS

In this section, we empirically test our approach on eight real designs within a complete IC implementation flow and measure its impacts on power, timing, cell area and runtime.

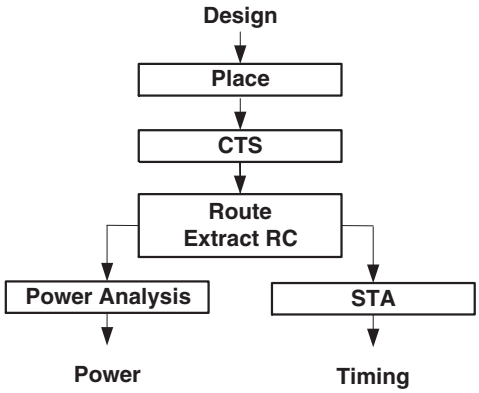


Figure 5: Flow for evaluating power-aware placement techniques.

5.1 Experimental Setup

We use eight industry circuits as our test cases, with the number of cells (including registers) ranging from 20k to 186k and the number of registers ranging from 2.3k to 44.2k. Clock power occupies 31.9% (ranging from 22.0% to 75.0%) of total power on average; and net switching power occupies 38.9% (ranging from 28.9% to 96.0%) of total power on average.

The experimental flow is shown in Figure 5. The inputs for each design include the synthesized netlist, technology libraries, timing constraints (including all clock definitions) and floorplan. To enable IC Compiler to estimate the switching rates of all nets (see the previous section), we specify the switching rate for each primary input of the design.

For each design, the placer is applied to perform two placement runs: with and without power-aware placement. For the power-aware placement run, register clustering is performed with a shrink ratio, and activity-based net weighting is applied with a power ratio (Both the shrink ratio and the power ratio were tuned around 0.8). After each placement, we perform clock tree synthesis, timing and congestion driven (global and detailed) routing, RC extraction and power analysis using IC Compiler and finally timing analysis using PrimeTime.

5.2 Results

Table 1 summarizes the low power results after completing the physical design flow for eight industrial test cases, and compares them with the results from normal timing-driven placement. The number of cells (including registers) and registers of each design are shown in the second and third columns. For each design, Table 1 shows the results of reference run and power-aware placement run, and the percentage improvements in a variety of metrics. Note that the improvements in clock skew and WNS are computed relative to the correspondent clock periods.

Clock net switching power, net switching power and total power are shown in the fifth to seventh columns. The low power flow achieves an average improvement of 11.3% (ranging from 1.6% to 34.5%) in clock switching power, an average improvement of 25.3% (ranging from 10.5% to 47.1%) in total switching power, and an average improvement of 11.4% (ranging from 6.5% to 18.8%) in total power.

Clock wire length is reduced by 10.1% on average, mainly

Design	Cells	Regs	Method	Clock Switching Power (mw)	Total Net Switching Power (mw)	Total Power (mw)	Clock WL	Clock Skew (ns)	WNS (ns)	Cell Area	CPU (%)
D1	186K	44244	Reference	153.29	319.86	908.51	879529	0.156	0.34	4619435	
			Low Power	100.43	182.59	737.71	843626	0.110	0.13	5041092	
			Imp (%)	34.48%	42.92%	18.80%	4.08%	1.38%	6.31%	-9.13%	-25.22%
D2	49K	5621	Reference	542.03	710.63	739.97	85192	0.029	2.18	1114383	
			Low Power	493.14	636.25	664.63	77569	0.032	2.61	1161019	
			Imp (%)	9.02%	10.47%	10.18%	8.95%	-0.10%	-14.33%	-4.18%	-18.22%
D3	134K	43528	Reference	168.61	302.57	1127.23	1492789	1.180	4.61	42612408	
			Low Power	150.87	224.95	1054.08	1266024	0.427	3.78	43121730	
			Imp (%)	10.52%	25.65%	6.49%	15.19%	5.02%	5.53%	-1.20%	-51.29%
D4	172K	23372	Reference	102.32	258.53	789.27	484661	0.095	0.46	4871915	
			Low Power	100.65	218.94	717.80	482264	0.088	0.54	4646738	
			Imp (%)	1.63%	15.31%	9.06%	0.49%	0.18%	-2.00%	4.62%	21.32%
D5	116K	9071	Reference	20.74	37.74	127.27	173554	0.169	3.73	2444381	
			Low Power	18.49	32.42	117.92	143063	0.174	4.10	2433401	
			Imp (%)	10.85%	14.10%	7.35%	17.57%	-0.03%	-2.47%	0.45%	6.96%
D6	20K	2315	Reference	1.64	3.00	10.64	46130	0.031	0.00	535949	
			Low Power	1.54	2.44	9.58	39254	0.030	0.00	447993	
			Imp (%)	6.10%	18.67%	10.03%	14.91%	0.00%	0.00%	16.41%	-32.57%
D7	126K	12864	Reference	21.54	46.87	133.28	260509	0.222	0.15	3136603	
			Low Power	19.31	33.52	113.58	252242	0.249	0.48	3471139	
			Imp (%)	10.35%	28.48%	14.78%	3.17%	-0.68%	-8.25%	-10.67%	-9.02%
D8	138K	8727	Reference	3.18	6.35	21.97	114542	0.178	3.26	1603950	
			Low Power	2.94	3.36	18.84	95760	0.285	3.38	1701496	
			Imp (%)	7.55%	47.09%	14.24%	16.40%	-0.54%	-0.60%	-6.08%	16.23%
			AVG Imp (%)	**11.31%**	**25.34%**	**11.37%**	**10.09%**	**0.65%**	**-1.98%**	**-1.22%**	**-11.48%**

Table 1: Comparison of low power flow against traditional timing-driven flow for eight real designs. Note that the improvements in clock skew and WNS are computed relative to the design clock period.

because of register clustering. The clock skew of the synthesized clock tree is shown in the ninth column. Although register clustering reduces the wire capacitance in the clock tree, its impact on the clock-tree skew after clock tree synthesis is random in our experiments.

The impact of power optimization on design performance in terms of WNS (worst negative slack) is shown in the tenth column. Comparing to the timing-driven flow, the low-power flow increases the WNS by 2.0% on average. Out of the eight test cases, the low power flow made significantly negative impact to WNS (14.3% to 8.3%) on two of them (designs D2 and D7). In Section 5.3 and 5.4 we show how we can trade off power for timing and vice versa by tuning the power ratio and shrink ratio.

The last two columns in Table 1 show total cell areas and increases in total runtime of the power-aware placement run. Upsizing and buffering are usually performed to improve design performance. According to the results, total cell areas are increased to improve performance for some designs. Notice that the total cell area increase here makes no impact to the final chip area (or cost) since the floorplan did not change and both placement and route completed in spite of the total cell area increase. Runtime is also increased due to more timing optimization after power-aware placement. The average increase in total cell area and runtime are 1.2% and 11.5%, respectively.

5.3 Power-Timing Trade-Off with Power Ratio

The power weighting ratio (α) can be used to trade-off power dissipation and design performance. In this experiment, the power-aware placement flow is run with varying power ratios (α's, from 0.2 to 1.0) for circuit D2 (for which meeting performance constraints is difficult). The results are summarized in Table 2. Figure 6 shows the curves of total switching power and WNS as functions of power weight ratio. We see that total switching power generally decreases with the power ratio increasing; routed wire length, WNS, TNS (total negative slack) and total cell area generally increase with the power ratio.

α	Total Switching Power (mw)	Total WL	WNS (ns)	TNS (ns)	Cell Area
0.2	707.12	3729885	2.02	5415	1154655
0.4	693.76	3632845	2.33	5345	1152080
0.7	667.29	3769301	2.35	5976	1163974
0.75	654.78	3860067	2.68	6029	1169672
0.8	652.23	3808004	2.31	5823	1168550
0.85	636.25	3813674	2.61	5911	1161019
0.9	650.55	4168253	2.39	6112	1187515
0.95	655.20	4197984	2.56	6226	1189243
1	635.35	4244201	2.85	6691	1186044

Table 2: Results with varying power weighting ratios (α's) for circuit D2.

5.4 Power-Timing Trade-Off with Shrink Ratio

p_0	Clock WL	Clock Switching Power (mw)	WNS (ns)	TNS (ns)	Cell Area
0.6	79555	513.68	2.51	9087	1181792
0.7	78229	503.37	2.67	14780	1160205
0.75	81958	522.87	2.52	11021	1190133
0.8	83790	534.74	2.16	5428	1161576
0.85	83386	531.16	2.27	6123	1154991
0.9	84987	544.01	2.11	5767	1152562
0.95	84768	545.65	2.15	5438	1152437

Table 3: Results with varying cluster shrink ratios (p_0's) for circuit D2.

Shrink ratio (p_0) controls how much the clock tree shrinks. A smaller shrink ratio often leads to a smaller clock switching power, but a worse design performance. In this experiment, the low power flow is run with varying cluster shrink ratios (p_0's, from 0.6 to 0.95) for circuit D2. The results are

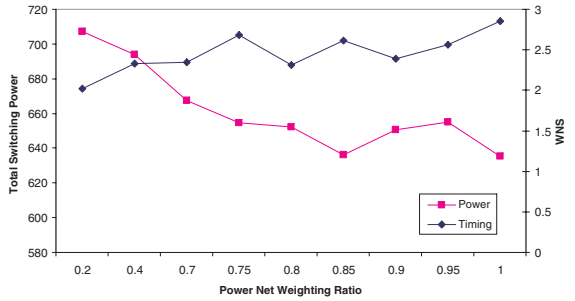


Figure 6: Total switching power and WNS as functions of power ratio (α) for circuit D2.

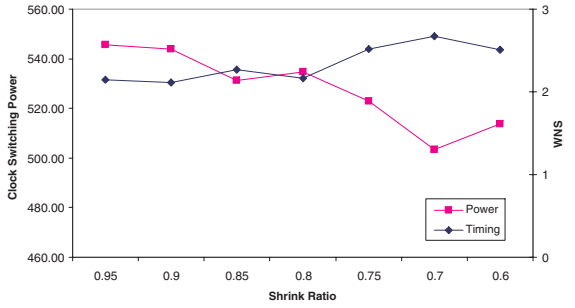


Figure 7: Clock switching power and WNS as functions of shrink ratio (p_0) for Circuit D2.

summarized in Table 3. Figure 7 shows the curves of clock switching power and WNS as functions of register clustering shrink ratio. We see that clock wire length and switching power generally decrease with the shrink ratio decreasing. However, WNS, TNS, and total cell area generally increase with the shrink ratio decreasing.

6. CONCLUSIONS

We have presented a power-aware placement method that performs activity-based net weighting and register clustering to reduce the capacitance of signal and clock nets that switch more frequently during placement without modifying the netlist. We have experimented the method on eight real designs through a complete industrial physical design flow. Our approach achieved average 25.3% and 11.4% reduction in net switching and total power, with 2.0% timing, 1.2% total cell area and 11.5% runtime degradation.

We have also demonstrated that the power-aware placement method can be applied to trade-off between timing and power, which may be useful for designing chips with both low-power and performance versions targeting both tethered and un-tethered systems.

7. REFERENCES

[1] V. Adler and E. G. Friedman, "Repeater Insertion to Reduce Delay and Power in RC Tree Structures," *IEEE Asilomar COnf. on Signals, Systems and Computers*, 1997, pp. 749-752.

[2] L. Benini, P. Siefel, and G. D. Micheli, "Automatic Synthesis of Gated Clocks for Power Reduction in Sequential Circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(6) (1996), pp. 630-643.

[3] K.-Y. Chao and D.-F. Wong, "Floorplanning for Low Power Designs," *Proc. IEEE Int. Symp. Circuits and Systems*, 1(28) (1995), pp. 45-48.

[4] T. Chao, Y. Hsu, J. Ho, K. Boese, and A. Kahng, "Zero Skew Clock Routing with Minimum Wirelength," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 39(11) (1992), pp. 799-814.

[5] J. Cong, C. K. Koh, and K. S. Leung, "Simultaneous Buffer and Wire Sizing for Performance and Power Optimization," *ACM/IEEE Int. Symp. Low-Power Electronics and Design*, 1996, pp. 271-276.

[6] M. Donno, E. Macci, and L. Mazzoni, "Power-Aware Clock Tree Planning," *Proc. ACM/IEEE Int. Symp. Physical Design*, 2004, pp. 138-147.

[7] A. Farrahi, C. Chen, A. Srivastava, G. Tellez, and M. Sarrafzadeh, "Activity-Driven Clock Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(6) (2001), pp. 705-714.

[8] P. E. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-Performance Microproccesor Design," *IEEE Journal of Solid-State Circuits*, 33(5) (1998), pp. 676-686.

[9] M. Igarashi *et al.*, "A Low-Power Design Method Using Multiple Supply Voltages," *ACM/IEEE Int. Symp. Low-Power Electronics and Design*, 1999, pp. 145-150.

[10] M. Jackson, A. Srinivason, and E. Kuh, "Clock Routing for High-Perfomance ICs," *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 573-579.

[11] A. Kahng, J. Cong, and G. Robins, "High-Perfomance Clock Routing Based on Recursive Geometric Matching," *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 322-327.

[12] A. Krishnamoorthy, "Minimize IC Power without Sacrificing Performance," *EEdesign*, July 15, 2004. Available at `http://www.eedesign.com/article/showArticle.jhtml?articleId=23901143`.

[13] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power Dissipation in a Microprocessor," *Proc. Workshop on System Level Interconnect Prediction*, 2004, pp. 7-13.

[14] B. Obermeier and F. M. Johannes, "Temperature-Aware Global Placement," *Proc. Asia and South Pacific Design Automation Conf.*, 2004, pp. 143-148.

[15] J. Oh and M. Pedram, "Gated Clock Routing for Low-Power Microprocessor Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(6) (2001), pp. 715-714.

[16] J. Pangjun and S. Sapatnekar, "Clock Distribution Using Multiple Voltages," *Proc. ACM/IEEE Int. Symp. Low-Power Electronics and Design*, 1999, pp. 145-150.

[17] S. M. Sait, M. R. Minhas and J. A. Khan, "Performance and Low Power Driven VLSI Standard Cell Placement Using Tabu Search," *Proc. Congress on Evolutionary Computation*, 2002, pp. 372-377.

[18] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing Power in High-Performance Microprocessors," *Proc. ACM/IEEE Design Automation Conf.*, 1998, pp. 732-737.

[19] N. Togawa, K. Ukai, M. Yanagisawa and T. Ohtsuki, "A Simultaneous Placement and Global Routing Algorithm for FPGAs with Power Optimization," *Proc. IEEE Asia-Pacific Conf. Circuits and Systems*, 1998, pp. 125-128.

[20] A. Vittal and M. Marek-Sadowska, "Low-Power Buffered Clock Tree Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(9) (1997), pp. 965-975.

[21] H. Vaishnav and M. Pedram, "PCUBE: A Performance Driven Placement Algorithm for Low Power Designs", *Proc. Design Automation Conf. with EURO-VHDL*, 1993, pp. 72-77.