Enabling Concurrent Clock and Power Gating in an Industrial Design Flow

Leticia Bolzani Andrea Calimera Alberto Macii Enrico Macii Massimo Poncino
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, ITALY

Abstract— **Clock-gating and power-gating have proven to be very effective solutions for reducing dynamic and static power, respectively. The two techniques may be coupled in such a way that the clock-gating information can be used to drive the control signal of the power-gating circuitry, thus providing additional leakage minimization conditions w.r.t. those manually inserted by the designer. This conceptual integration, however, poses several challenges when moved to industrial design flows. Although both clock and power-gating are supported by most commercial synthesis tools, their combined implementation requires some flexibility in the back-end tools that is not currently available.**

This paper presents a layout-oriented synthesis flow which integrates the two techniques and that relies on leading-edge, commercial EDA tools. Starting from a gated-clock netlist, we partition the circuit in a number of clusters that are implicitly determined by the groups of cells that are clock-gated by the same register. Using a row-based granularity, we achieve run-time leakage reduction by inserting dedicated sleep transistors for each cluster. The entire flow has been benchmarked on a industrial design mapped onto a commercial, 65nm CMOS technology library.

I. Introduction

Clock-gating [1] is, by far, the most widely adopted technique for reducing dynamic power in digital CMOS circuits. The reasons for this are rooted, on the one hand, into the capability of clock-gating of significantly reducing power consumption with a limited penalty in area and, most important, in timing. On the other hand, clock-gating is very suitable to automatic application [2], [3], [4], thus all design frameworks by the major EDA vendors do support clock-gating as their primary power optimization feature.

It is well know that the advent of sub-100nm technologies has made the power optimization problem more difficult to address, as dynamic consumption is now paired by a non-negligible amount of static consumption, mainly due to the sub-threshold currents in the off state. Among the several solution for reducing the sub-threshold leakage currents, *power-gating* [5] is the one that, in the near past, has gained the largest momentum. As for the case of clock-gating, also power-gating can be effectively supported from the CAD stand-point [6], [7], [8], and it is now featured by most commercial design flows.

Clock-gating and power-gating hold a number of similarities. In particular, both techniques require the introduction into the original circuit of some control logic, which is in charge of generating the signals that control the clock-gating and the power-gating mechanisms. Ideally, the same control signal could be used for managing clock-gating and power-gating, thus suggesting the opportunity of combining the application of the two techniques. In practice, however, the two signals are usually characterized by different timing behaviors; while clock-gating is applicable on a cycle-by-cycle basis, power-gating is not. Combining clock-gating and power-gating, although very desirable from the point of view of the achievable energy savings, may thus be quite difficult; in some cases, it may result even infeasible, as the overlapping of the potential gating conditions may tend to become null.

In order to check the effective feasibility of the simultaneous application of the two techniques, in [9] we presented an analysis tool which is able to evaluate an RTL design in order to determine whether clock-gating/power-gating integration is convenient or not. Assuming a circuit featuring clock-gating, the tool is able to verify whether the insertion of the sleep transistors necessary to actuate the power-gating strategy may lead to an overall reduction of the circuit power (dynamic + leakage). The analysis is based on an inspection of the circuits topology, as well as on the knowledge of the timing and the energy information regarding the circuit.

The results of the application of the tool are very promising, indicating that, in principle, clock-gating and power-gating should be used simultaneously to achieve the best total power minimization. Unfortunately, however, the combined application of the two techniques presents major difficulties in real-life design environments and for realistic circuits, due the inadequacy and lack of integration of the commercially-available synthesis and back-end tools.

Purpose of this work is to establish *a layout-oriented synthesis flow which integrates clock-gating and power-gating*. The input of the flow is a placed gated-clock netlist, as generated by any commercial low-power synthesis tool. The gated-clock regions constitute the starting point of a clustering algorithm which determines the groups of cells to which power-gating is applied. Clearly, both functional (i.e., idle conditions) and physical (i.e., position in the placement) information of the cells are used to determine the most appropriate clusters to be connected to the sleep transistors. The flow is validated on an industrial design mapped onto a commercial, 65nm CMOS standard cell library provided by STMicroelectronics.

Although the idea of combining clock-gating and power-gating has been the subject of recent investigation [10], to the best of our knowledge this is the first time that the problem

978-3-9810801-5-5/DATE09 © 2009 EDAA

is addressed with industrial strength, thus offering a solution which may easily find its way in commercial CAD flows and for the implementation of industrial circuits featuring leading-edge nano-CMOS technologies. In fact, the work in [10] determines the clusters of cells to be power-gated and inserts the power-gating circuitry cell-by-cell; P&R are then executed using standard tools. Clearly, the main short-coming of this approach is that, timing-driven placement may scatter the cells of a given cluster all over the die area, thus originating a placed netlist for which the application of power-gating to the selected clusters is infeasible.

II. Clock-Gating and Power-Gating

A. Clock-Gating Basics

Clock-gating (CG) is based on the idea of disabling (i.e., gating) the clock signal to a specific register that feeds a portion of combination logic that is not performing useful computations during some clock cycles.

A generic gated-clock architecture is shown in Figure 1; a signal, called *activation function* (F_a), is defined in order to selectively stop (when $F_a = 1$) the clocking of the circuit when the latter does not result in state or output transitions. The activation signal is filtered by a latch that is transparent when the global clock is low. The purpose of the latch is to filter potential glitches of the activation signal that should not propagate when the global clock is high.

The activation function is a combinational block that extracts idle conditions from the knowledge of primary and state inputs of the circuit. These conditions might be derived from a state-based description of the netlist of the circuit and can be purely topological or include functional information.

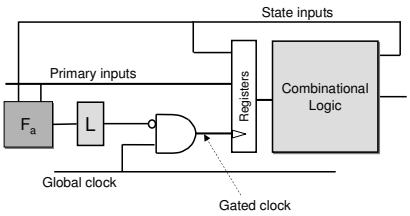

Fig. 1. Conceptual CG Architecture.

The successful application of clock-gating entails the solution of three main issues:

- *Timing closure*: There is a possible performance decrease due to the fact that the logic of the F_a may be on the critical path of the circuit.
- *Formal verification*: Because of the idle cycles, it is not trivial to perform the equivalence checking of the gated and non-gated circuits.
- *Testability*: The CG circuitry is functionally redundant. In fact, it is not needed for correct functionality, but at the same time its failure can have disastrous effects. In the presence of faults, some errors in the gating circuitry may be masked by the controlled sub-system (or vice-versa), with serious consequences on testability.

B. Power-Gating Basics

Power-Gating (PG) is a coarse-grained generalization of the so-called MTCMOS technique, in which a header and/or footer transistor is inserted on the pull-up and/or pull-down network of a CMOS gate, respectively; the transistors are turned off when the gate is in stand-by mode, thus reducing the leakage current that flows in the supply-ground path (Figure 2).

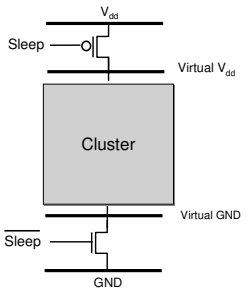

Fig. 2. Conceptual PG Architecture.

These transistors are usually called *sleep transistors* (ST) because they are driven by the same sleep signal (yet with opposite phase). In practice, one transistor suffices; thanks to its lower on-resistance, the nMOS footer is typically used.

While a cell-level gating technique introduces a large area and timing overhead, applying this idea to larger blocks of logic has proven quite successful. In a power-gated design, switch transistors control clusters of gates, instead of individual gates.

Although PG may appear a straight-forward technique, its adoption is very challenging. In fact, its effective implementation requires solving three main problems:

- *Clustering*: The definition of the granularity of the blocks to which gating is applied is an open issue and encompasses a trade-off between effectiveness and efficiency: Fine-grain gating (e.g., MTCMOS) has maximum overhead and largest optimization potential; on the other extreme, coarse-grain gating has smaller overhead but also has smaller optimization potential.
- *Sizing of the ST*: The size of the ST affects the performance of the gates connected to it. A small transistor slows down the circuit in active mode due to its high resistance, whereas a large one implies a large area overhead and a significant energy cost during ON/OFF transitions. Moreover, the ST size is also constrained by the maximum current injected by the gated cluster.
- *Physical design of the gating circuitry*: The size of the ST is far larger than that of any cell and its placement is a non trivial task. Moreover, since it is connected to all the cells in the cluster, it must be placed in such a way that excessive routing overhead is avoided. Thus, ST placement and clustering become strictly inter-dependent. Furthermore, the presence of a virtual ground rail implies the addition of an extra grid to the already existing power/ground distribution grids.

C. Integrated Clock and Power-Gating

From the discussion above, CG and PG may look as unrelated techniques. A more careful analysis, however, reveals that they are two different ways of exploiting the same property of a design, namely, its *idleness*. While CG stops the clock to a logic block during idle cycles, PG disconnects a logic block from the ground line during idle periods. Their fundamental difference is *how* idleness is determined. CG extracts (structurally or functionally) the cycle-by-cycle idle conditions and suppresses the clock in those cycles; conversely, PG is activated by an external signal. In principle, nothing prevents us from using the CG conditions extracted from the circuit and implemented by F_a also for controlling PG, so that both dynamic and static power is saved during the idle intervals.

Figure 3 shows the conceptual integration of the two solutions. The CG conditions (represented by the logic denoted by F_a) are used as additional PG conditions (logically OR-ed) to those externally provided as sleep signals. Notice that, under this scheme, clustering is implicitly determined by CG: A group of cells that is clock-gated by the same register automatically defines a cluster.

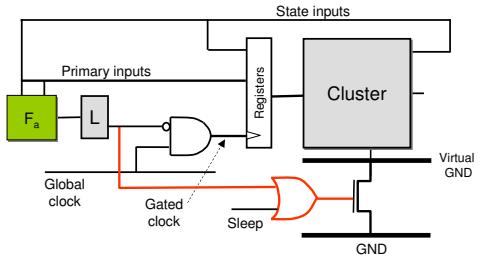

Fig. 3. Conceptual Integration of CG and PG.

Unfortunately, the integration of the two techniques is far from being simple, several issues must be considered in order to make it feasible.

1) Clustering: The extraction of the idle conditions usually originates different subsets of registers driven by different activation functions and, therefore, multiple clusters. As long as such clusters do not overlap, the scheme of Figure 3 is applicable as is.

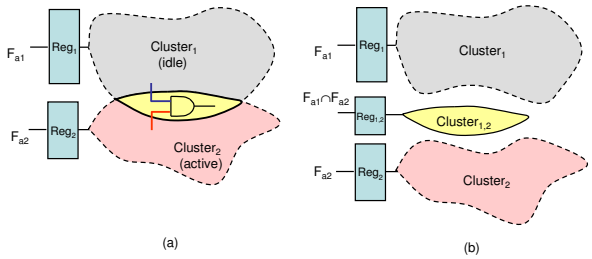

Fig. 4. Overlapping Clusters.

In most cases, however, clusters do overlap and some gates are shared among several clusters. This is not an issue for the application of CG alone; in fact, register-gating freezes the inputs of the cluster, but still allows normal operations of the gates in that cluster (Figure 4-(a)). Conversely, if the cluster is power-gated, *all* its gates are detached from the ground and, therefore, are not usable by other clusters. The only way to solve this problem while keeping the overhead acceptable is to generate a new cluster for every cluster intersection, whose activation function is the AND of the activation functions of the overlapping clusters (Figure 4-(b)). Obviously, based on the size of the overlapping region and of the gating conditions, it may be convenient not to power-gate some clusters.

2) Timing Granularity: The timing granularity of CG and PG may differ substantially; while the former can be applied on a cycle-by-cycle basis (as there is no reactivation delay involved), the latter can only be activated (i.e., the STs can be turned off) only if the expected sleep interval is longer than the ST reactivation time. Furthermore, ST reactivation has a power cost; too frequent switching due to short idle periods may offset the power savings resulting from the turn-off. This implies that not all CG conditions can be used for PG and some of them will be wasted, thus resulting in a reduction of potential dynamic power savings.

3) Physical Design: The implicit clustering resulting from CG may not be compatible with the ST insertion strategy of the PG. STs cannot be placed anywhere in the design, and some preferential ST placement strategies are used. Whatever the adopted strategy, it must be aware of the placement of the cells that constitute the cluster.

Conversely, the clustering implied by CG may include cells physically placed far away, thus making ST insertion less effective, if not infeasible. To solve this problem, the designer must force the placement in such a way that (i) the cells are physically close, and (ii) they are placed in a way that it is compatible with the placement of the ST. As an example, an effective ST insertion strategy relies on the idea that clusters to be gated contain entire rows of the layout (*row-based* PG [8]). Under this strategy, which is the one adopted in this work, clusters induced by CG should somehow be packed into the smallest possible number of rows.

III. Design Flow

Figure 5 shows the synthesis flow proposed in this paper. Starting from an RTL netlist, the flow consists of five steps:

- *Step 1 (Synthesis):* This step performs the synthesis of the RTL design. The tool synthesizes the RTL netlist according to a set of constraints and with CG insertion. The outputs of this stage are (i) a clock-gated netlist, (ii) a set of files necessary to the following steps (e.g., .DEF files), (iii) a VCD file with the switching information for every net of the design , and (iv) power and timing reports.
- *Step 2 (*OpenAccess *Database):* This step consists of the generation of an OpenAccess database of the synthesized design using, as inputs, the clock-gated netlist and the .DEF file previously obtained.
- *Step 3 (Clustering):* At this stage, the analysis tool of [9] evaluates the feasibility of integrating CG and PG by using the information available in the VCD file, as well as the design's OpenAccess database generated in Step

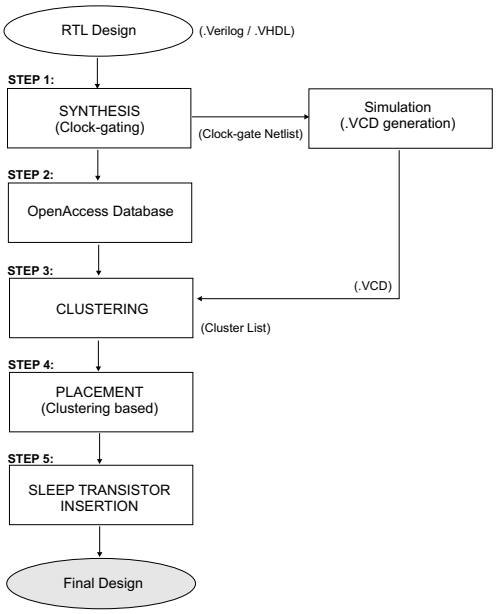

Fig. 5. Layout-oriented Synthesis Flow.

2. The analysis tool is able to identify the clock-tree, the clock-gating registers with their activation functions, and the group of cells associated to them.Moreover, it defines the *PG clusters* that represent groups of cells associated to the same clock-gating register and to a specific activation function. PG clusters represent groups of cells that have to be associated to a certain sleep transistor.
The clustering process amount thus to the association of every cell to a group of cells that will be controlled by the same sleep transistor. The tool classifies the PG clusters in two different classes:

- *CG-Register Cluster:* This clusters consist of cells that are present in only one fanout (i.e., the combinational logic between the register bank and the design's output ports or another register). These cells are thus associated to a single clock-gating register.
- *CG-Register Intersection Cluster:* This type of cluster is composed of cells that belong to more than one fanout. These cells are associated to more than one clock-gating register.

For each generated PG cluster, the tool estimates the total area needed to place it. Figure 6 shows an example of PG cluster generation. The schematic of a synthesized design contains 2 clock-gating registers (CGR1 and CGR2), their register banks and the group of cells that are part of their fanout. The diagonal lines represent the fanout associated to CGR1 and CGR2. The design is divided into three PG clusters: Clusters 1 and 2 are classified as CG-Register Clusters, while Cluster 1/2 is the intersection region between the fanout of CGR1 and CGR2. The activation function of Cluster 1/2 is the AND between the activation functions of CGR1 and CGR2. Thus, Cluster 1/2 will be shut-down only if Clusters 1 and 2 are turned-off.

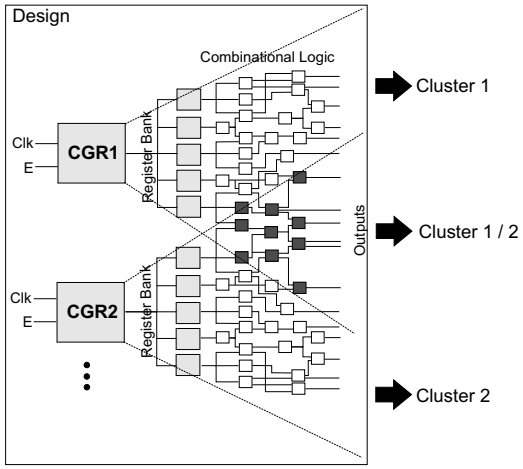

Fig. 6. PG Clusters in a Design.

- *Step 4 (Placement):* In this step, the placement of the design is performed, following the placement constraints resulting from the generation of the PG clusters. Figure 7 shows a possible layout of the design of Figure 6.

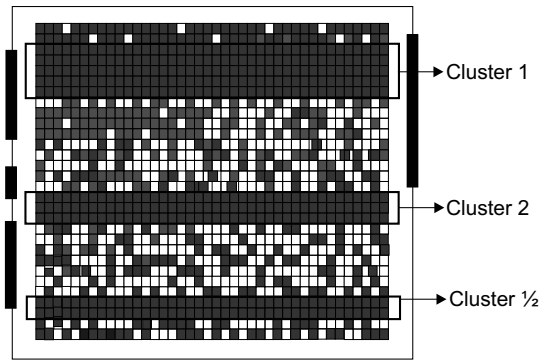

Fig. 7. Layout Design View.

It is important to highlight that the PG clusters will contain different numbers of cells, and thus have quite different areas. If the area necessary for placing a given PG cluster does not match the available area in a row of the layout, the cluster is not power-gated since the overhead (in terms of wire-length, congestion and dynamic power) caused by having cells of the same cluster in different rows, may become too high. Therefore, finding the proper trade-off between the potential leakage energy saving and the placement overheads is essential.

- *Step 5 (Sleep Transistor Insertion)*: In the last step, the ST size is defined based on the size of the PG cluster. The sleep transistor is inserted in one (or more, if needed) dedicated row , as described in [8]. It is important to emphasize that *the number of concurrent PG clusters present in a given row determines the number of inserted virtual grounds that must be inserted*.

IV. Experimental Results

We have validated the proposed design flow, on a unit of a network-on-chip design, namely, a 6x6 switch. This design analyzes the incoming data packets, determines the source and the destination device of the packets, and forwards them appropriately. It consists of around 8 Kgates, and it was synthesized onto a 65nm CMOS technology library by STMicroelectronics. We used Synopsys PhysicalCompiler to perform the synthesis process, which was done exploiting both area and timing optimization, and with the clock gating feature enabled. Using Mentor Graphics ModelSim, we ran functional simulations in order to obtain a complete and detailed temporal description of all the internal signals; dedicated testbenches were used to emulate workloads compliant with the specifications. Using in-house parsers, the VCD file obtained by simulation was used to compute both static probability and the number of toggles for the internal nodes; these values were annotated in Synopsys PrimeTime during power estimation. Thanks to the physical information contained in the technology library, we were able to evaluate the effect of parasitic elements on the interconnects, thus obtaining more accurate power and timing estimations.

Based on the topological structure of the circuit and the temporal waveforms listed in the VCD file, the clustering algorithm provides a complete list of logic clusters (18, in this design). For each cluster we know the number and the type of the standard cells contained in it, and the duration of the idleness intervals of the clock-gated registers, which also represent the stand-by periods of the sleep transistors. Using this information, we performed a constrained placement. More precisely, we created a *bounded layout region* for each cluster. Using an automated scripting flow, the placement tool tries to place the cells of each cluster in the reserved region (i.e., intra-cluster placement), and performs a coarse-grained placement of each cluster (i.e., inter-cluster placement). The final result is a standard layout where the cells of a cluster are placed in a dedicated area (i.e., a number of adjacent layout rows), such that each row contains standard-cells belonging to the same logic cluster. This is mandatory for guaranteeing a minimal area overhead for ST insertions. However, since standard placement tools are wire-length oriented, the final layout may not respect our physical constraints. In fact it may happen that a layout region dedicated to a given cluster contains gates of different clusters. As we will show later, this has deleterious effects for the integration of clock-gating and power-gating.

Figure 8 reports the leakage savings we obtained by combining clock-gating and power-gating. The plot shows, for each of the 18 clusters, a pair of bars. Dark bars denotes the amount of leakage energy spent by the circuit in the idle state (and thus it the energy that can be saved by applying power-gating). Light bars show the reactivation energy, namely, the energy spent for idle-to-active transitions. The percentage shown on top of each bar pair represent the total leakage savings during the stand-by period.

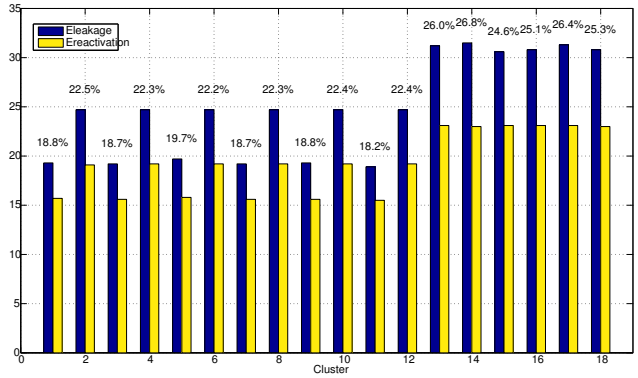

Fig. 8. Per-Cluster Leakage saving.

Combining CG and PG is therefore convenient when the energy necessary to reactivate the circuit is smaller than the leakage energy consumed by the circuit in the idle state. Figure 8 shows that this margin is sizable and does exist for all the clusters. On average, a 22.3% leakage energy is saved.

It is worth emphasizing that *we are considering the worst-case saving*, that is that we are evaluating the energy saved *during the shortest idle interval of a cluster*. Each cluster has in fact several idle intervals of different lengths. Clearly, longer intervals imply increased leakage savings. Therefore, the values of Figure 8 should be interpreted as lower bounds of the leakage savings.

Another important issue related to the reactivation time is the time required to turn-on the power-gated cluster. Experimental results shown that, for every cluster, the reactivation time is smaller than 50% of the clock-period. This assures the correct functionality of the clusters, which can be reactivated before the edge of the next active clock-period.

A. Physical Design Issues

As pointed out at the end of Section III, each layout row should contain as many cells as possible that belong to the same PG cluster. Ideally, no two cells of the same layout row should belong to different clusters; this would allow to use a single virtual-ground line to power-gate the entire row, with minimal area and routing overheads. However, commercial placement tools are not PG-aware, and it may happen that a row contains cells belonging to different clusters. Since each cluster needs its dedicated virtual-ground rail, we should route multiple metal lines for each row, thus increasing the area overhead.

Table I reports data about the distribution of PG-cluster cells over the layout rows. Columns *Avg*, *Min*, and *Max* show the average, minimum, and maximum number of cells belonging to different PG-clusters in a layout row.

Row *Standard* shows the cluster distribution for a standard, wirelength-driven placement. The average number of clusters per row is 14.6, implying that we would have to route, on average, around 14 metal lines, (8 in the best case, and 19 in the worst case) below each row. Needless to say, this solution has unacceptable area overhead.

Placement Style	*Avg*	*Min*	*Max*
Standard	14.6	8	19
PG-Aware - Low Effort	11.9	5	16
PG-Aware - Mid Effort	2.2	1	6
PG-Aware - High Effort	1.3	1	2

TABLE I DISTRIBUTION OF CLUSTERS: STANDARD VS. POWER GATING-AWARE PLACEMENT STYLES.

The other three rows of the table refer to our PG-aware placement, which tends to place gates belonging to the same cluster in the same row. How much this constraint is enforced corresponds to different area/delay tradeoff points, which we associate to a measure of *placement effort*. In particular, we select three different effort levels (*Low*, *Mid* and *High*); low-effort placement will likely achieve results that are only better than the standard placement , whereas the high-effort one will tend to achieve the ideal placement (rows having cells that belong to one cluster).

The second row of the table shows that using a *Low* effort we do not improve much over the standard placement; the average number of clusters per row goes down to 11.9, which is still too large for a realistic implementation. Using *Mid* and *High* efforts, however, the number of clusters per row becomes 2.2 and 1.3, respectively, thus making the integration of CG and PG feasible. Notice in particular that using the *High* effort, most of the rows contain cells of the same cluster, and only a few of them contain cells of two different clusters.

While a highly constrained placement may give benefits in terms of ST and virtual-ground insertion, the same could not be said for power (leakage and dynamic), timing, and area. Figure 9 shows the above mentioned design metrics as a function of the placement effort; values are normalized to the standard placement.

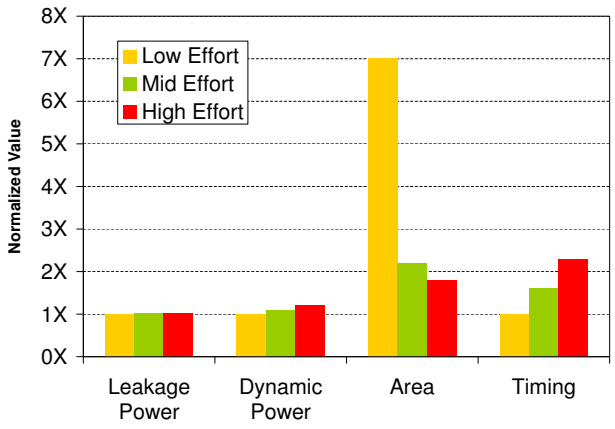

Fig. 9. Design Metrics Tradeoff.

We can see that leakage power is insensitive to the placement effort, since the number and the size of the cells remains unchanged during the placement process. The dynamic power slightly increases due to an increase of the wire-length; in fact, in order to meet our cluster-based constraints, the placer relaxes the constraint on the wire-length, thus yielding circuits with larger interconnect parasitics. Conversely, timing suffers a sizable increase. With a low-effort placement, the worst-critical path grows up to 2.3X with respect to the standard case. Concerning the area overhead, a highly constrained placement reduces the clusters' dispersion across the layout rows, thus reducing the average number of virtual-grounds for each row. As reported in the graph, using a low-effort placement, the layout area becomes 7x larger than the standard case, while with a high-effort placement, the area becomes is 1.8x.

It is worth emphasizing that the plot just reports *the overhead* of the various metrics, and it does not show the missed opportunities of a standard placement; although the latter yields the best figures for all the metrics, it does not allow to apply power gating due to the excessive dispersion of clusters over the rows.

V. CONCLUSIONS AND FUTURE WORK

The joint application of clock-gating and power-gating, although appealing from the theoretical stand-point, is quite problematic, mainly because of the lack of support by the existing tools and flows. In this paper, we have presented a physical synthesis methodology which enables this integration. The proposed flow relies on commercial synthesis and back-end tools, therefore it is usable in industry-strength EDA environments; the experimental results we have collected on some benchmarks demonstrate the practical effectiveness of our methodology.

REFERENCES

[1] L. Benini, P. Siegel, G. De Micheli, "Automatic Synthesis of Gated Clocks for Power Reduction in Sequential Circuits," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 32-40, 1994.

[2] L. Benini, G. De Micheli, "Transformation and Synthesis of FSMs for Low Power Gated Clock Implementation," *IEEE Transactions on CAD*, Vol. 15, No. 6, pp. 630-643, 1996.

[3] L. Benini, G. De Micheli, E. Macii, M. Poncino, R. Scarsi, "Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Synchronous Controllers," *ACM Transactions on Design Automation*, Vol. 4, No. 4, pp. 351-375, 1999.

[4] P. Babighian, L. Benini, E. Macii, "A Scalable Algorithm for RTL Insertion of Gated Clocks based on Observability Don't Cares Computation," *IEEE Transactions on CAD*, Vol. 24, No. 1, pp. 29-42, 2005.

[5] M. Anis, S. Areibi, M. Elmasry, "Design and Optimization of Multi-threshold CMOS Circuits," *IEEE Transactions on CAD*, Vol. 22, No. 10, pp. 1324-1342, 2003.

[6] C. Long, L. He, "Distributed Sleep Transistor Network for Power Reduction," *DAC-41: ACM/IEEE Design Automation Conference*, pp. 181-186, June 2003.

[7] P. Babighian, L. Benini, E. Macii, A. Remollino, "Post-Layout Leakage Power Minimization Based on Distributed Sleep Transistor Insertion," *ISLPED-04: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 138-143, August 2004.

[8] A. Sathanur, A. Pullini, L. Benini, A. Macii, E. Macii, M. Poncino, "Timing-Driven Row-Based Power Gating," *ISLPED-07: ACM/IEEE International Symposium on Low Power Electronics and Design,* pp. 104-109, August 2007.

[9] L. Bolzani, A. Calimera, A. Macii, E. Macii, M. Poncino, "Integrating Clock Gating and Power Gating for Combined Dynamic and Leakage Power Optimization in Digital CMOS Circuits", *DSD08: IEEE 11th Euromicro Conference on Digital System Design*, September 2008, pp. 298-303.

[10] K. Usami, N. Ohkubo, "A Design Approach for Fine-grained Run-Time Power Gating using Locally Extracted Sleep Signals", *ICCD-06: IEEE International Conference on Computer Design*, pp. 155-161, October 2006.