Clock-Tree Power Optimization based on RTL Clock-Gating

Monica Donno[◇]
monica.donno@bulldast.com

Alessandro Ivaldi[‡]
alessandro.ivaldi@polito.it

Luca Benini[*]
lbenini@deis.unibo.it

Enrico Macii[‡]
enrico.macii@polito.it

[◇]BullDAST s.r.l.
Torino, ITALY 10121

[‡]Politecnico di Torino
Torino, ITALY 10129

[*]Università di Bologna
Bologna, ITALY 40136

ABSTRACT

As power consumption of the clock tree in modern VLSI designs tends to dominate, measures must be taken to keep it under control. This paper introduces an approach for reducing clock power based on clock gating. We present a methodology that, starting from an RTL description, automatically generates a set of constraints for driving the construction of the clock tree by the clock synthesis tool. The methodology has been fully integrated into an industry-strength design flow, based on Synopsys DesignCompiler (front-end) and Cadence Silicon Ensemble (back-end). The power savings achieved on some industrial examples show that, when the size of the circuits is significant, savings on the power consumption of the clock tree are up to 75% larger than those achieved by applying traditional clock gating at the clock inputs of the RTL modules of the designs.

Categories and Subject Descriptors

B.5 [**Hardware**]: Register-Transfer-Level Implementation; B.6 [**Hardware**]: Logic Design; B.7 [**Hardware**]: Integrated Circuits

General Terms

Design

Keywords

Low-power design, clock-tree synthsis

1. INTRODUCTION

The clock distribution network normally accounts for more than 40% of the total power budget of a CMOS circuit, as the clock nets operate at the highest switching frequency of any other signal and they drive a large fanout. Designing the clock tree is thus critical not only for performance, but also for power.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

Early work on clock tree synthesis focused on the generation of zero-skew trees [10] or minimum wire-length [4] clock distribution networks. More recently the area of low power clock tree synthesis has been investigated.

In [5], the authors proposed an approach based on reduced voltage swings, while in [3] power savings on the clock network were obtained by taking advantage of interconnection parasitic inductance. Also, solutions based on power-constrained buffer insertion and simultaneous buffer and clock wire sizing can provide significant savings [11, 2, 1].

Although these techniques are effective, none of them considers that clock signals are not always needed, and thus power can be saved by masking off (i.e., *gating*) the clock when circuits are idle.

Clock gating can significantly reduce the switching activity in a circuit and on clock nets; thus, it has been viewed as one of the most effective logic, RTL and architectural approaches to power minimization. Unfortunately, if applied in a uncontrolled fashion, gating can adversely impact clock power. In fact, in order to amortize its power and area overhead, clock-gating logic should be shared among several flip-flops. If the flip-flops that share a common gated clock (i.e., *a gated-clock domain*) are widely dispersed across the chip, a significant wiring overhead is induced in the clock distribution network, as each domain must be independently distributed on dedicated wires. As a result, clock drivers in each domain are loaded with a much larger capacitance and power may increase even if switching activity is decreased [6, 9]. We then conclude that clock gating and clock tree construction should not be seen as two independent steps and a synergistic strategy is needed.

Several authors have focused on this problem in the recent past. In the sequel, we briefly summarize two contributions that have some common roots with the approach we propose in this paper.

In [7], Farrahi *et al.* defined a methodology based on behavioral synthesis to build an *activity-driven clock tree*. Given a pre-placement description of the design, the set of active and idle times, representing the *activity pattern* for each module, is extracted from the module's scheduling table. An activity pattern is a string of 0s and 1s, indicating idle and active control steps, respectively. The tree construction algorithm is heuristic, bottom-up, based on recursive weighted matching, where the cost function is the activity of the resulting subtree. The objective is to cluster into the same subtree modules with similar activity patterns, so that the clock

tree can be gated with high probability as close as possible to the root. The clusters of modules created by the recursive matching algorithm are translated into proximity constraints for module placement. Then, the clock tree is routed as an H-tree. Dynamic programming is finally used to determine where the gating logic must be inserted.
In [6], Oh *et al.* present a zero-skew gated clock routing technique for VLSI circuits that improves upon [7] in two ways. First, it starts from a placed netlist of modules. Second, it accurately accounts for the power consumption of control signals, jointly addressing the routing problem for both the clock tree and the gated clock control signals. The algorithm is applicable to a class of processors where activation signals are obtained from instructions and where the generation of all activation signals is centralized in a single module placed close to the center of the die. Clock tree building is done in two steps. First, possible locations of the internal nodes are calculated according to [4]. Then, the exact position is found by a greedy method that merges minimum switched capacitance nodes; by delaying the merging of high activity nodes the global activity in the tree is reduced.
Further work on gated clock tree construction can be found in [9, 8]. The first paper reports on an exploration of the impact of clock gating on traditional clock tree construction in the case of realistic benchmarks. The second contribution extends the work of [7] in the directions indicated by [6].
Experimental data of previous work have shown that the gated clock technique can significantly reduce the power dissipation in the clock distribution network. Also, it has been demonstrated the effectiveness of exploiting information on the clock activation functions during clock tree generation. However, the described approaches give little attention to integration issues with existing design flow and they have not been validated on real-life benchmarks. In this paper, we propose a novel methodology and algorithms for gated clock tree construction that are specifically geared towards integration with existing design flows, both in the front-end (i.e., RTL and logic-level clock activation function extraction and manipulation) and in the back-end (i.e., industrial-strength clock tree synthesis tools). By taking this approach we can construct power-optimal gated clock trees for real-life designs and provide a detailed assessment of our methodology using data extracted from fully placed and routed netlists, for the first time in the literature.
Results on a set of real-life circuits show that, when the size of the designs (and thus the structure of the clock tree is complex), the savings on the power consumption of the clock tree are up to 75% larger than those achieved by applying traditional clock gating at the clock inputs of the RTL modules of the designs.

2. METHODOLOGY OVERVIEW

This section briefly describes the methodology that we are proposing to build a low power clock tree and clarifies its rationale. We consider a hierarchical design methodology in which synchronous modules with a single clock can be identified; for these leaf modules we build a gated clock tree. The objective is to achieve a reduction of the power consumption on the clock tree without re-writing the tool for clock tree synthesis. The main reason for this choice is to leverage the capability of current industrial-strength clock construction tools to take into account a number of constraints of different nature (e.g., geometry, reliability, performance). The output of our clock-power optimization tool is *not* a routed clock tree, but it can be viewed by a set of constraints for a clock tree synthesis tool that lead to a low-power gated clock tree, while still accounting for all non-power-related requirements (e.g., controlled skew, low crosstalk-induced noise, etc.).
The steps in our approach can be summarized as follows. We start from two inputs: (i) A hierarchical RTL structural description of a synchronous circuit, containing a set of modules for each of which a single clock and a clock activation signal is identified. (ii) A placement of the modules, with specified positions of clock and activation function pins. This information can be obtained by available commercial clock gating tools (e.g., Synopsys' PowerCompiler) and by floorplanners or by RTL-to-placed-netlist synthesis tools (e.g., Synopsys' ChipArchitect and PhysicalCompiler), respectively.
As a first step, the RTL design is simulated to extract the waveforms of the activation functions. The placement information and the activation function's waveforms are then elaborated by our *LPclock algorithms*. The algorithm is split in two phases. In the first phase it builds a clock tree topology balancing the reduction in clock switching against clock and activation function capacitive loading estimates. In the second phase it inserts clock gating logic in the tree, balancing its power consumption against the power on the gated clock sub-tree. The output of *LPclock* is *not* a layout of the clock tree, but a *clock netlist*, which is then fed to an industrial-strength clock tree construction tool. After clock construction, the fully placed and routed gated clock tree (including the clock nets, buffers, clock-gating gates, and activation signal nets) is extracted from the layout and accurate analysis can then be performed to assess quality-of-results.

3. PROBLEM DEFINITION

Given a synchronous digital system $S(M, N)$, where $M = \{m_1, .., m_K\}$ denotes the hierarchical modules inside the design and $N = \{n_1, .., n_L\}$ is the set of the nets of the circuit, if for each $m_i \in M \mid i = 1, ..., K$ s_i represents the only clock entry point, the clock net is given by the set of sinks plus the source $N_{clock} = \{s_i \mid i = 1, ..., K\} \cup S_0$. We assume that the clock network N_{clock} can be represented by a complete binary tree $T(V, E)$ whose root is S_0 and whose leaves are the K sinks. Let the area $A = x_{layout} \cdot y_{layout}$ be a suitable placement for the considered design, then we denote by (x_{s_i}, y_{s_i}) the position of the clock sinks in the placement area and by (x_{S_0}, y_{S_0}) the source coordinates. All modules are characterized by a capacitance $C_i = N_{S_i} \cdot C_{clock}$ where N_{S_i} represents the number sequential elements inside the block and by an *Activation Function* $ACTF_i$ which is a boolean function whose value is "1" when the module does not need the clock and "0" otherwise.
For each possible module pair m_i, m_j with $i \neq j \mid i, j = 1, ..., K$ we define a *physical distance* as:

$$D(m_i, m_j) = |x_{s_i} - x_{s_j}| + |y_{s_i} - y_{s_j}|$$

The physical distance is calculated with the Manhattan metric, which is a good estimator of the wiring length between clock sinks, as horizontal and vertical directions are the only one allowed to the routing tools. Physical closeness means shorter interconnections, hence reduced congestion, less interconnection delay and a smaller parasitic capacitance.

Besides the physical distance, a *logical distance* is defined as:

$$L(m_i, m_j) = (C_i + C_j) \cdot p(i,j)$$

where

$$p(i,j) = P(ACTF_i = 1, ACTF_j = 1)$$

is the probability for modules i and j to be idle.
If $ACTF_i$ and $ACTF_j$ are completely independent $p(i,j) = P(ACTF_i = 1) \cdot P(ACTF_j = 1)$. Since the independence is not always verified, this probability is computed exactly through simulation waveform analysis: The values of both $ACTF$ are collected over N consecutive clock cycles and the number of times in which the logic AND of the two Activation Function takes on the value "1" is calculated:

$$p(i,j) = \frac{N_{AND}}{N}.$$

The logical distance measures the similarity of the module activities: By merging close activities, the resulting tree needs the clock signal for a percentage of time comparable to that of its leaves leading to a reduction of the overall activity in the tree.
Based on the previous definitions of distance, we formulate the gated clock tree construction problem as follow: Given the set of clock sinks K, build a gated clock network represented by $T(V,E)$ whose cost function is:

$$dist(i,j) = \alpha f(D(i,j)) + \beta g(L(i,j))$$

Parameters α and β allow the tuning of the weight of switching activity vs. wire-length. On the other hand, f and g express possible transformations over the distance metric (to account for uncertainty on wire-length estimation).
The detailed description of the algorithms that are used to build the clock tree requires further details about the calculation of the Activation Function and the definition of a power model.

3.1 Obtaining the Activation Function

The clock gating technique exploits high level information to decide when the clock signal can be shut down. For each design module an $ACFT$ must then be evaluated. Finding the set of conditions that allow to turn off the clock is a well known problem for which many solutions do exist.
For example, in Synopsys PowerCompiler, a simple clock gating scheme is applied to register banks with an available `enable` input. The method is based on the idea that when the `enable` input is "0" the clock is not needed since the register bank maintains previous stored data: The inverted *enable* signal itself is thus the $ACTF$ for the registers. Similarly, another example is given by *operand isolation* [12], which prevents the switching activity propagation in a module performing a redundant operation. Again, identifying redundant operations requires the computation of an activation function that is based on a structural analysis of the transitive fanout of the module.
In our flow, the identification of $ACTF$s is performed in a preliminary step using PowerCompiler; however, it could be done through any other technique the designer may have access to.

3.2 Power Model

Since our ultimate objective is to reduce clock power consumption, we need a power model to drive the gating logic insertion. Gated clock schemes obtain power savings by reducing the amount of capacitance that is switched when logic transitions take place.
While evaluating clock net power consumption, four contributions are considered: The modules and the clock gating port input capacitance, plus the capacitance switched by the interconnection in the clock tree and by the interconnection that feeds the control signal to the gating logic. Consider the example in Figure 1; let c_0 be the unit wire capacitance, l_i, l_g the interconnection length of the clock tree and of the control gating logic signal, respectively, C_i and C_g the input capacitance for the module and the gate logic. Power dissipation is then modeled as:

$$2(c_o l_i + C_i)p(i) + (c_0 l_g + C_g)p_{tr}$$

where $p(i)$ represents the probability for the module to be active ($p(i) = P(ACTF_i = 0)$) and p_{tr} is the probability to have a transition on the control signal net ($p_{tr} = N_{tr}/N-1$), where N_{tr} is the number of transitions in the Activation Function evaluated over N consecutive clock cycles.

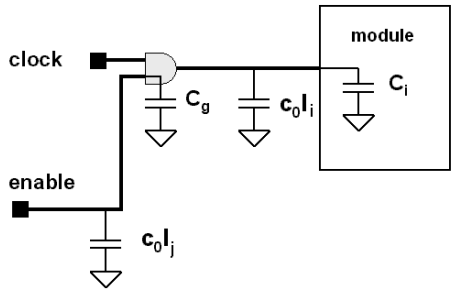

Figure 1: Power Model.

4. LPCLOCK METHODOLOGY

The *LPclock* methodology, described in this section, consists of three main components, as shown in Figure 2.

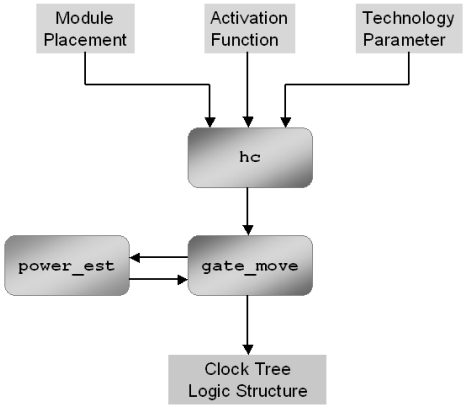

Figure 2: LPclock Methodology.

The `hierarchical clustering` (`hc`) component builds the clock tree structure; the `gate_move` component performs the gating logic insertion; finally, the `power_est` component takes care of the estimation of the power consumption, as this information is needed by the `gate_move` step.
Figure 2 clearly shows that the input parameters to LPclock are both physical (module placement, technology data) and functional information (Activation Function).

4.1 Hierarchical Clustering

Given the clock sink set for the considered design, `hc` builds a clock net structure based on a fully binary tree subject to the previously defined cost function:

$$dist(i,j) = \alpha f(D(i,j)) + \beta g(L(i,j))$$

where α and β are numerical values settled by an experimental sensitivity analysis of the cost function (see Section 6), while f and g are used as normalization functions for the physical and logical distances.
By defining f as the maximum layout dimension:

$$dim_{max} = max(x_{layout}, y_{layout})$$

and g as the module total capacitance:

$$C_{tot} = \sum_{i=1}^{K} C_i$$

the cost function becomes:

$$dist(i,j) = \alpha \left[\frac{1}{dim_{max}} (|x_{s_i} - x_{s_j}| + |y_{s_i} - y_{s_j}|) \right] +$$

$$\beta \left[1 - \frac{1}{C_{tot}} (C_i + C_j) p(i,j) \right].$$

The function `hc` builds the tree following a bottom-up clustering strategy, level by level. Each level requires the evaluation of the *Distance Matrix*, which is the matrix that contains the $dist(i,j)$ values for all possible node pairs at the considered level. If the current level has N nodes/sinks, then $\lfloor N/2 \rfloor$ clusters should be constructed to complete the next level, and each created pair needs the calculation of its position in the placement, its capacitance and its Activation Function. Consider, for example, the merging of two generic nodes n_i and n_j; the resulting cluster position is given by:

$$x_{cluster} = \frac{C_i x_{ni} + C_j x_{nj}}{C_i + C_j}, y_{cluster} = \frac{C_i y_{ni} + C_j y_{nj}}{C_i + C_j}$$

where this formulation balances the wire length so that the higher capacitance corresponds to the shorter wire in order to control the skew. The capacitance and the Activation Function are calculated as:

$$C_{cluster} = C_i + C_j$$

$$ACTF_{cluster} = ACTF_{ni} \wedge ACTF_{nj}.$$

We implemented two different merging schemes. The first one is a greedy method that pairs the two nearest nodes, i.e., the two nodes with the minimum $dist(i,j)$. The second is based on maximum weighted matching, where the objective function is the sum of all $dist(i,j)$ for the current level.

4.2 Gate Moving

Once the clock network is built, we have a complete binary tree in which both leaves and internal nodes are characterized with an Activation Function, but no gating elements are inserted yet. At this stage, it is possible to think that at each module input the proper gating logic is introduced by simply translating in hardware the corresponding Activation Function. The outlined situation, that we call *gated modules*, allows power savings inside the modules and, if the logic is added as close as possible to the parent node of the tree, it allows a reduction of the dissipation just in the last portion of the clock net.
`Gate_move` explores the opportunities for moving gating logic from the leaves towards the upper level inside the clock tree, so that the dissipation is reduced also in the clock network. The implemented algorithm is heuristic; it performs a postorder visit on the clock tree and for each node it tries to find a local minimum for dissipated power, thus the dissipation is the cost function for the `gate_move` procedure. The heuristic works as shown in Figure 3.

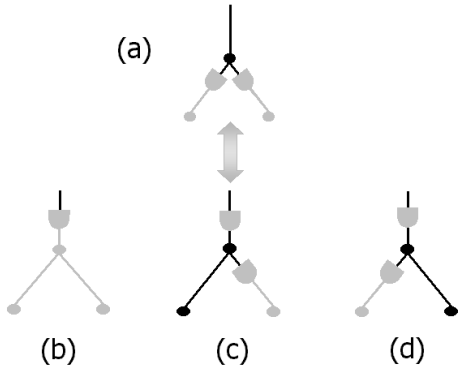

Figure 3: Heuristics Implemented by `Gate_move`.

When a node is visited three situations are evaluated. In principle, the best solution is represented in Figure 3(b); here, instead of using two gating logic blocks, a unique gating element is inserted in the upper level to control the whole subtree. But, if in the considered node the idle state for one child is considerably longer than for the other, moving up only the control logic of the child active for the longest time is more effective. Therefore, also situations like those shown in Figure 3(c) and 3(d) must be considered. Power consumption related to each of these three possibilities is evaluated through the function `power_est` and the optimal local solution is chosen.
Procedure `Power_est`, which is based on the power model previously discussed, performs the power estimation by a combined preorder/postorder visit of the clock tree. The preorder step is necessary to annotate, for non-gated nodes, the parent Activation Function, while during the postorder visit the switched capacitance is calculated. The last step performed by LPclock is that of translating the planned clock structure with the inserted gating logic into a Verilog description in which each internal node in the tree corresponds to an AND gate. If there is no gating logic in the considered node, one of the AND gate inputs is at the logic value "1", otherwise the input is connected to the gating logic block synthesized from the Activation Function by Synopsys DesignCompiler.

5. THE PROPOSED FLOW

This section describes how the LPclock methodology is implemented on top of and integrated into a common industrial design flow. We have used the Synopsys tools as front-end and the Cadence environment as back-end. Yet, we would like to stress that the LPclock methodology can be mapped onto other RTL-to-Layout flows with only little effort, as no conceptual changes are needed; LPclock simply provides, as output, adequate constraints that can be easily taken into account by EDA tools that provide clock tree synthesis capabilities.

We also observe that the initial assumption on the hierarchical structure of the input RTL description can be easily relaxed and the LPclock flow applied to non-hierarchical designs, since strategies do exist to cluster together registers with some mutual affinity.

The LPclock tool flow is shown in Figure 4.

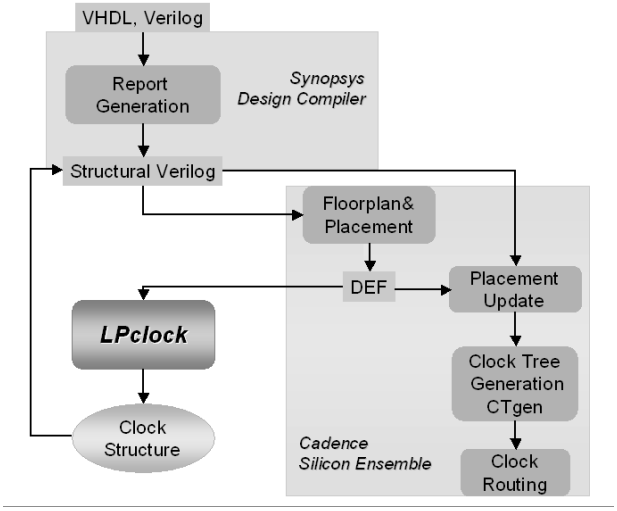

Figure 4: The LPclock Tool Flow.

Starting from an high-level design description (VHDL or Verilog), the circuit is first elaborated by DesignCompiler to obtain a RTL structural representation from which clocked modules and all nets, included the clock, can be extracted. The design at the RTL is then simulated to obtain the Activation Function and then translated into structural Verilog. A floorplan and a placement are then initialized by Silicon Ensemble. The placed DEF description, as well as the Activation Function, are the LPclock inputs. Introducing the clock structure elaborated by LPclock in the design requires a double update: First, in the already generated placement the `+PLACED` attribute is changed to `+FIXED`, in order to avoid module position changing. Second, the Verilog description is updated to import the new clock network structure.

Design changes are transfered to Silicon Ensemble and the clock structure and the gating logic are included in the design database via incremental placement. The flow then continues in the traditional way, that is, CTgen is invoked to perform buffer insertion and to check for timing closure. It should be pointed out that the insertion of the AND gate for each internal node in the clock tree prevents any change on the clock net by CTgen, forcing the tool to preserve the clock branching structure built by LPclock.

6. EXPERIMENTAL RESULTS

The experiments we carried out with LPclock had a twofold objective. First, to analyze the behavior of the cost function w.r.t. parameters α and β. Second, to assess the amount of power savings our method was able to achieve. The salient details of the industrial benchmarks we used are shown in Table 1.

Benchmark	*# of FF*	*# of Modules*	*# of Clk Sinks*
GCD	52	6	4
Dual	35	7	5
Fir	309	43	36
i8237	625	51	31
i8051	1341	78	59

Table 1: Characteristics of the Benchmark Circuits.

The cost function is a weighted linear combination of the physical distance (coefficient α) and the logical distance (coefficient β). To choose the best value for the weight parameters, we studied the trend of the α/β and β/α ratios by varying their values between 1.00 and 20.00, with a step of 0.01. For all considered α and β ratios, we simulated the benchmark circuits to estimate the capacitance switched by the clock tree. The results of the analysis, shown in Figure 5, indicate that the two distance metrics should have equal weight in the cost function, as $\alpha/\beta = 1.00$ seems the choice of the parameters that minimizes the cost function.

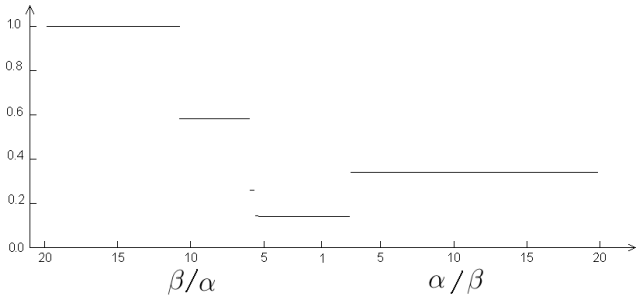

Figure 5: Sensitivity Analysis on α and β.

After choosing the values of α and β, as discussed above, we have performed some experiments in order to evaluate the effectiveness of LPclock in reducing power consumption.

To this purpose, we first generated the placed and routed netlists (including the clock network) of the original benchmark circuits, as well as the netlists of the designs with gated modules (i.e., gating logic inserted at the clock input of the modules) and with the clock tree structure created by LPclock. Layout extraction was then performed for all the designs, and the gate-level netlists back-annotated using the extracted parameters. Finally, gate-level power estimation was performed using Synopsys PowerCompiler. The whole synthesis process was timing driven, and mapping was done on the $0.18\mu m$ HCMOS8 technology library by STMicroelectronics. Clock tree synthesis with CTGen was performed using a *very tight maximum skew constraint* (less than 0.2% of the clock cycle, which was ranging between 10 and 50 nanoseconds, depending on the benchmark), and *no skew violation* was observed for all the benchmarks. This result clearly indicates the goodness of the constraints on the clock tree structure provided by LPclock.

Table 2 collects clock tree power results. In particular, column *Gated Module* shows the power savings w.r.t. the original circuit implementation achieved by inserting the gating logic only at the input of the RTL modules. On the other hand, column *LPclock* shows the power savings against the original design obtained by inserting the gating logic as suggested by *LPclock*. Finally, column Δ shows the difference in the savings between the two approaches.

Benchmark	*Gated Module*	*LPclock*	Δ
Dual	17.24%	19.72%	+3,02%
Fir	63.86%	65.37%	+4.18%
GCD	26.70%	27.74%	+1.42%
i8237	13.94%	53.54%	+46.03%
i8051	62.11%	90.26%	+74.31%

Table 2: Experimental Results.

Data clearly show that the clock trees generated using *LPclock* as preprocessor to CTGen are much superior (in terms of power) to those generated by CTGen alone for circuits of significant size, while they are negligible on smaller benchmarks. This is obvious, as for small circuits the clock tree tends to have a very simple structure, and thus the advantages that the LPclock approach can provide are very limited. Instead, for larger benchmarks, savings w.r.t. the use of traditional clock gating are as high as almost 75%.

7. CONCLUSIONS

Power consumption due to the clock tree is becoming more and more relevant in modern VLSI designs, due to the fact that interconnect capacitance tends to dominate over gate capacitance and clock frequencies tend to grow at a very fast pace in new technologies.

The problem of minimizing power consumption of the clock tree has been addressed in the past, and techniques have been proposed to drive physical design of the clock tree starting from a high-level of abstraction. However, most of the attempts made so far to attack this problem have not found a direct validation in industry-strength design flows.

In this paper, we have introduced a methodology for reducing clock power dissipation based on clock gating. The proposed solution allows us to automatically generate clock tree routing constraints to be fed to the physical design tool starting from an RTL description.

Distinguishing feature of the methodology is its capability of exploiting both logical and physical information of the given RTL design to optimize the clock tree structure.

Experimental results showed that clock power savings can be up to 75% higher than those achieved by applying traditional clock gating at the inputs of the RTL modules.

Future work on the subject of low-power clock tree synthesis will go in three directions. First, the LPClock flow will be fully automated and integrated into a RTL-to-layout design environment offering both power optimization and estimation capabilities. Second, the methodology will be extended to allow the handling of non-hierarchical descriptions, that is, designs in which registers are not necessarily confined into RTL components with well-defined boundaries. This would be an essential point to allow the application of the LPClock tool in industrial design settings. Third, we will investigate solutions that will allow us to raise the entry-level of the LPClock methodology beyond RTL, i.e., starting from more abstract views such as behavioral descriptions.

Acknowledgements

This work was supported, in part, by the European Commission, under grant IST-2001-30125 "POET", by Motorola SPS, EWDC, Geneva, Switzerland, and by STMicroelectronics, AST, Agrate Brianza, Italy.

8. REFERENCES

[1] J. Cong, C.-K. Koh; K.-S. Leung, "Simultaneous Buffer and Wire Sizing for Performance and Power Optimization," *ISLPED-96: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 271-276, Monterey, CA, August 1996.

[2] V. Adler, E. G. Friedman, "Repeater Insertion to Reduce Delay and Power in RC Tree Structures," *31st IEEE Asilomar Conference*, pp. 749-752, Pacific Grove, CA, November 1997.

[3] Y. I. Ismail, E. G. Friedman, J. L. Neves, "Exploiting On-Chip Inductance in High Speed Clock Distribution Networks," *43rd IEEE Midwest Symposium on Circuits and Systems*, pp. 1236-1239, Lansing, MI, August 2000.

[4] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, A. B. Khang, "Zero Skew Clock Routing with Minimum Wirelength," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 39, No. 11, pp. 799-814, November 1992.

[5] J. Pangjun, S. S. Sapatnekar, "Clock Distribution using Multiple Voltages," *ISLPED-99: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 145-150, San Diego, CA, August 1999.

[6] J. Oh, M. Pedram, "Gated Clock Routing for Low-Power Microprocessor Design," *IEEE Transactions on CAD/ICAS*, Vol. 20, No. 6, pp. 715-722, June 2001.

[7] A. Farrahi, C. Chen, A. Srivastava, G. Tellez, M. Sarrafzadeh, "Activity-Driven Clock Design," *IEEE Transactions on CAD/ICAS*, Vol. 20, No. 6, pp. 705-714, June 2001.

[8] C. Chen, C. Kang, M. Sarrafzadeh, "Activity-Sensitive Clock Tree Construction for Low Power," *ISLPED-02: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 279-282, Monterey, CA, August 2002.

[9] D. Garrett, M. Stan, A. Dean, "Challenges in Clock Gating for a Low Power ASIC Methodology," *ISLPED-99: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 176-181, San Diego, CA, August 1999.

[10] R.-S. Tsay, "An Exact Zero Skew Clock Routing Algorithm," *IEEE Transactions on CAD/ICAS*, Vol. 12, No. 2, pp. 242-249, February 1993.

[11] A. Vittal, M. Marek-Sadowska, "Low-Power Buffered Clock Tree Design," *IEEE Transactions on CAD/ICAS*, Vol. 16, No. 9, pp. 965-975, September 1997.

[12] M. Munch, B. Wurth, R. Mehra, J. Sproch, N. Wehn, "Automating RT-Level Operand Isolation to Minimize Power Consumption in Datapaths," *DATE-00: IEEE Design Automation and Test in Europe*, pp. 624-631, Paris, France, March 2000.