Power-Aware Clock Tree Planning

Monica Donno
BullDAST s.r.l.
R&D Division
10121 Torino, ITALY
monica.donno@bulldast.com

Enrico Macii
Politecnico di Torino
Dip. di Automatica e Informatica
10129 Torino, ITALY
enrico.macii@polito.it

Luca Mazzoni
Accent s.r.l.
R&D
20059 Vimercate, ITALY
luca.mazzoni@accent.it

ABSTRACT

Modern processors and SoCs require the adoption of power-oriented design styles, due to the implications that power consumption may have on reliability, cost and manufacturability of integrated circuits featuring nanometric technologies. And the power problem is further exacerbated by the increasing demand of devices for mobile, battery-operated systems, for which reduced power dissipation is mandatory. A large fraction of the power consumed by a synchronous circuit is due to the clock distribution network. This is for two reasons: First, the clock nets are long and heavily loaded. Second, they are subject to a high switching activity.
The problem of automatically synthesizing a power efficient clock tree has been addressed recently in a few research contributions. In this paper, we introduce a methodology in which low-power clock trees are obtained through aggressive exploitation of the clock-gating technology. Distinguishing features of the methodology are: (i) The capability of calculating powerful clock-gating conditions that go beyond the simple topological search of the RTL source code. (ii) The capability of determining the clock tree logical structure starting from an RTL description. (iii) The capability of including in the cost function that drives the generation of the clock tree structure both functional (i.e., clock activation conditions) and physical (i.e., floorplanning) information. (iv) The capability of generating a clock tree structure that can be synthesized and routed using standard, commercially-available back-end tools.
We illustrate the methodology for power-aware RTL clock tree planning, we provide details on the fundamental algorithms that support it and information on how such a methodology can be integrated into an industrial design flow. The results achieved on several benchmarks, as well as on a real design case demonstrate the feasibility and the potential of the proposed approach.

ISPD'04, April 18–21, 2004, Phoenix, Arizona, USA.

Categories and Subject Descriptors

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

General Terms

Digital design

Keywords

Low-power design, physical design and optimization, clock tree synthesis and routing

1. INTRODUCTION

The clock distribution network is responsible for an increasing fraction of the dynamic power consumed by modern processors and SoCs [1]. For example, Figure 1 shows the breakdown of power consumption for a recent high-performance microprocessor [2].

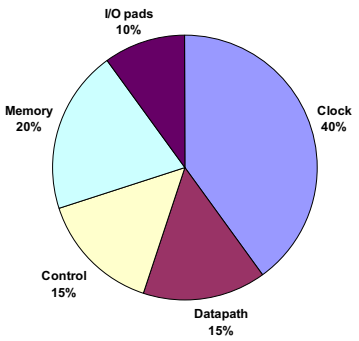

Figure 1: Processor Power Breakdown.

This result is common to many real designs: For the DEC Alpha 21164, 40% of the chip power (i.e., around 20W out of 50W) is consumed by the clock distribution network when the processor runs at its maximum speed [3]. Similarly, in the Motorola MCORE micro-RISC processor, the clock trees account for 36% of the total processor power [4]. And the picture is predicted to get worse as the complexity and the operating frequency of the circuits keep growing as a result of technology scaling [5].

Designing the clock tree has thus become critical not only for performance, but also for power, and the development of new modeling capabilities [6] and synthesis techniques that help in controlling the clock tree power effectively is one of the challenges that EDA engineers currently have to face.
Different solutions for minimizing the power consumed by the clock tree have been investigated in the recent past. In this paper, we focus our attention to an approach (named LPClock in the sequel) that relies on clock-gating, a well-established concept for power optimization at the gate and RT levels. The basis for the LPClock methodology can be found in [7]. That work introduced new algorithms for gated-clock tree construction that are specifically geared towards integration with existing design flows, both in the front-end (i.e., extraction and manipulation of RTL and logic-level clock activation functions) and in the back-end (i.e., interfacing to industry-strength clock tree synthesis tools). We show how such algorithms can be combined with innovative techniques for detecting clock-gating conditions [8] that go beyond the pure topological analysis of the RTL source code, to generate a power-efficient clock tree structure. We provide and discuss validation data obtained on a set of benchmark circuits, as well as on an industrial design.
We emphasize that the objective of the LPClock methodology is not that of replacing the back-end step of clock tree synthesis and routing; instead, the goal is that of generating a set of design constraints early enough in the design process (i.e., *planning* the clock tree structure at the RTL) that can then be exploited by traditional physical design tools during clock tree routing.
The rest of this paper is organized as follows. In Section 2 we briefly review previous work on clock tree power minimization, discussing techniques ranging from buffer insertion to adoption of multiple supply voltages, from reduced-swing clock signalling to different solutions for clock-gating. Section 3 provides an overview of the LPClock methodology, including the details on how clock-gating conditions are extracted based on the concept of observability don't care (ODC) and on the algorithms for planning the clock tree structure. Extensions to the methodology for handling non-hierarchical (i.e., flat) designs are also sketched. Section 4 discusses tool flow issues, thus addressing the problem of embedding and integrating LPClock into an industrial design framework. In Section 5 we report on the experimental results we have obtained on some meaningful design examples. Finally, Section 6 concludes the manuscript with some final remarks.

2. PREVIOUS WORK

The problem of synthesizing low-power clock distribution networks has been addressed recently and from different angles. Initially, the attention has focussed on techniques based on power-driven buffer insertion. In [9], buffers are added to the clock tree and sized as a post-processing operation, when the tree structure is already determined. Improved methods for buffer sizing [10] and simultaneous buffer and clock wire sizing [11] that target a minimum-power clock tree implementation have been proposed at a later time, while Vittal and Marek-Sadowska made a step forward in this domain by introducing a heuristic algorithm that performs concurrent design of the clock tree topology and buffer insertion [12].
A different approach to the problem of designing a minimum-power clock tree network was taken by Igarashi *et al.*; in [13], they proposed the use of multiple supply voltages to reduce clock tree power. The incoming, high-voltage clock signal is down-scaled by means of a low-voltage buffer stage. The low-V_{dd} signal is then propagated throughout the circuit, and regenerating elements (e.g., buffers) are inserted into the tree structure to ensure the appropriate speed and slew rate of the transitions. Finally, the original high-voltage is restored through level-shifters before the clock signals feed the flip-flops.
Although the method of [13] did target minimization of the power consumed by the clock network, it did not factor into the power balance the cost of buffering and voltage converters. The approach presented by Pangjung and Sapatnekar in [14] addresses this limitation by providing a more sophisticated algorithm for introducing buffers into the clock tree and for placing the low-to-high voltage shifters, which are now not necessarily located right in front of the flip-flops. The algorithm is a modification of the Deferred-Merged Embedding (DME) method [15, 16, 17] that considers the possibility of buffer insertion after every step of bottom-up subtree merging. In the interest of keeping the skew very close to zero, the algorithm guarantees that the number of regenerating elements is equalized along any root-to-sink paths of the tree. However, in spite of the solid theoretical basis of this solution, experimental results showed very small differences with the clock trees generated by the original approach by Igarashi *et al.*, witnessing the goodness of the greedy approach of [13].
An alternative to multi-voltage clock distribution networks, proposed first by Zhang and Rabaey in [18], is based on the idea of adopting a reduced-swing clock signalling scheme. The paper provides general guidelines for the design of driver and receiver circuits with reduced voltage swing, while [14] focuses on intermediate driver circuits, whose usage is suggested instead of traditional buffers and repeaters for guaranteeing the required level of performance. An efficient architecture of a low-swing receiver circuit that improves over the one in [18] is also proposed. Compared to the multi-voltage solution, reduced-swing clock trees are less power efficient, as the number of intermediate receivers that are needed to achieve the same speed of the multi-voltage implementation is substantially larger.
Although most of the techniques mentioned above are effective, none of them considers the fact that clock signals may not be always needed, and thus power can be saved by masking off (i.e., *gating*) the clock when a circuit (or part of it) is idle, that is, it is not performing any useful computation for one or more clock cycles.
Clock gating can significantly reduce the switching activity in a circuit and on the clock nets; thus, it has been viewed as one of the most effective logic, RTL and architectural approaches to dynamic power minimization [19]. Complex algorithms have been devised for calculating the idle conditions of a circuit and for automatically inserting the clock-gating logic into the netlist [20, 21, 22, 23]. Side effects of the clock-gating paradigm, such as its impact on circuit testability, have been explored in details [24], making this technology very mature also from the industrial stand-point. As of today, most commercial EDA tools for power-driven synthesis feature automatic clock-gating capabilities at different levels of design abstraction.

Unfortunately, if applied in a uncontrolled fashion, clock-gating can adversely impact the clock power. In fact, to amortize its power and area overhead, the 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 routed 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 [25, 26]. We then conclude that clock-gating and clock tree construction should not be seen as two independent steps and a combined strategy is needed.

Several authors have focused on the problem of minimizing clock tree power through exploitation of gated-clocks. In the sequel, we summarize two contributions that have some common roots with the approach we discuss in this paper.

In [27], 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 for the module the pattern refers to. The clock tree construction algorithm is heuristic, it works bottom-up and it is based on recursive weighted matching, where the cost function is the activity of the resulting sub-tree. The objective is to cluster into the same sub-tree 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 [26], Oh *et al.* present a zero-skew gated-clock routing technique for VLSI circuits that improves upon the work of [27] 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 [28]. Then, the exact position is found by a greedy method that merges minimum switched capacitance nodes; delaying the merging of high activity nodes reduces the global activity in the tree. Further work on gated-clock tree construction can be found in [25, 29]. 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 [27] in the directions indicated by [26]. 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 flows and they have not been validated on real-life benchmarks.

3. LPCLOCK OVERVIEW

The objective of LPClock is to build a power-optimal gated-clock tree structure, and use state-of-the-art physical design tools to perform detailed clock routing and buffering. As a consequence, the output of LPClock is not a completely routed clock tree; instead, it is a clock netlist (including clock-gating cells and related control logic) and constraints that, provided as input to commercial clock tree synthesis tools, 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).

LPClock requires two inputs: (i) A RTL structural description of a synchronous circuit, that can be obtained by any RTL synthesis tool; (ii) A placement of the RTL modules, that can be obtained by any RTL floorplanner.

The methodology consists of three steps, as shown in the flow diagram of Figure 2.

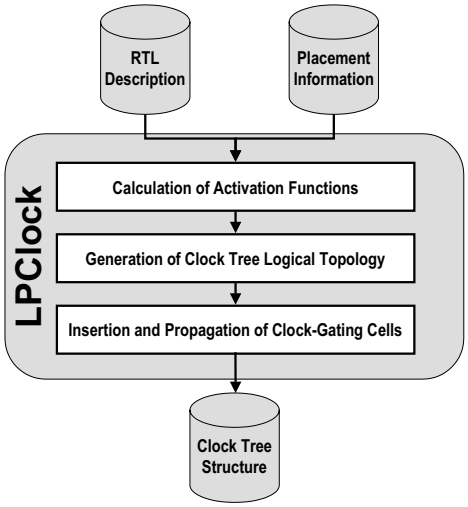

Figure 2: LPClock **Methodology Overview.**

The gated-clock activation functions for all the RTL modules are computed first. This implies the calculation of the observability don't care conditions for the group of flip-flops that belong to each RTL module, which is performed based on the available functional and topological information. Next, according to the activation functions and the physical position of the RTL modules, the logical topology of the clock tree is planned. This entails balancing the reduction in clock switching activity against clock and activation function capacitive loads. Clock-gating cells are then inserted into the clock tree topology and propagated upward in the tree whenever this is convenient, thus balancing the clock power consumption against the power of the gated-clock sub-tree. The information about the gated-clock tree is finally passed to the back-end portion of the flow, which will take care of clock tree routing and buffering.

In the remainder of this section, we illustrate in details the three steps discussed above. Prior to that, we briefly recall the basic principle of clock-gating and the power model that we use to drive the optimization process.

3.1 Clock-Gating: Principle and Power Model

Objective of clock-gating is to reduce power in the logic and in the clock wires by preventing useless transitions.
Let us focus on clock power, and consider the schematic of Figure 3. Assume that module M_i is characterized by an input capacitance $C_i = N_{S_i} \cdot C_{clock}$, where N_{S_i} represents the number flip-flops inside the module 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. Anytime $ACTF_i = 1$, the control input of the AND gate takes on the value 0; this avoids any transition on the gate output, implying that the entire clock network that feeds the flip-flops inside M_i does not experience any transition, thus resulting in a decrease of the power.

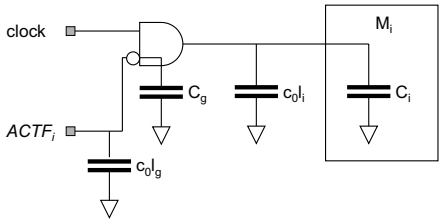

Figure 3: Example of Clock-Gating.

Since our ultimate objective is to reduce clock power consumption, we need a power model to drive the gating logic insertion.
While evaluating clock network power, four contributions are considered: The input capacitances of the module and of the AND gate, 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 again the example of Figure 3; 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 gating 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.

3.2 Calculating the Activation Functions

The clock-gating technique exploits high level information to decide when the clock signal can be shut down. For each module M_i in the design we thus need to calculate its activation function $ACTF_i$.
One option for determining the $ACTF_i$'s for all the modules is to resort to existing tools that are capable of performing clock-gating insertion. In many cases (for example, in Synopsys PowerCompiler), clock-gating 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 the previously stored value: The inverted `enable` signal itself is thus the $ACTF$ for the registers. This approach is purely topological, as it is based on the analysis of the circuit's RTL netlist.

Operand isolation [30] works similarly, as it prevents the switching activity propagation in a module by performing a redundant operation. Again, identifying redundant operations requires the computation of an activation function that is based on a topological analysis of the transitive fanout of the module.
In our flow, the identification of the activation functions for the modules in the RTL description is performed by resorting to the theory of observability don't cares. This allows us to determine more powerful conditions for clock-gating, as calculation of the observability don't care functions considers both topological and functional information about the circuit.

3.2.1 ODC Basics

In logic synthesis, the observability don't care (ODC) of a Boolean variable indicates the conditions under which the logic value of such a variable cannot be observed by the environment. Consider, for instance, a simple two-input AND gate (depicted in Figure 4), whose Boolean function is $z = x \cdot y$. Intuitively, a zero value on input y masks input x at the output z of the gate. On the other hand, input x is not observable by the environment if the output z itself is masked at the primary outputs of the circuit. As a consequence, x becomes unobservable by the environment if just one of the two conditions above does occur.
The observability don't care conditions of x can thus be represented in Boolean form as:

$$ODC(x) = \overline{y} + ODC(z)$$

where the ODC of a variable is assumed to be 1 if the value of the variable itself is not observable at the primary outputs of the circuit, 0 otherwise. Hence, if y is 0 or the output z is not observable at the primary outputs (i.e., $ODC(z) = 1$), then the logic state at the input x is not observable by the environment.

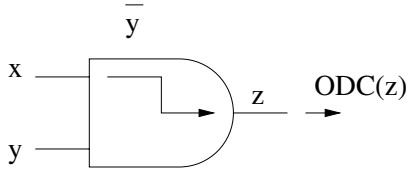

Figure 4: ODC Components of a Boolean Variable.

Generally speaking, given two variables x and z such that $z = f(x)$, the ODC of x is expressed as the logic sum of two terms:

$$ODC(x) = (\overline{f(x)|_{x=1} \oplus f(x)|_{x=0}}) + ODC(z) \qquad (1)$$

where "+" and "$\oplus$" represent the logic OR and the exclusive OR, respectively, "|" represents the restriction condition and the overline denotes the complement of a Boolean function or of a Boolean variable.
We define the ODC function $ODC_{M_i}(x)$ of input x of a module M_i as the first term of equation 1. In particular, $\text{ODC}_M(x)$ represents the conditions under which input x to module M_i is not observable at the module output z. We can then rewrite equation 1 as:

$$ODC(x) = ODC_{M_i}(x) + ODC(z) \qquad (2)$$

Given a module M_i with K inputs, $\{x_1, x_2, ..., x_K\}$, the ODC conditions of the inputs are computed by traversing the fanin cone of M_i backward. At first, the calculation of the ODC of the output (i.e., $ODC(z)$ in equation 2) is performed on the basis of the ODCs of the inputs of the immediate successors of the module. In particular, for a module whose output z fans out to N successors, $\{z_1, z_2, ..., z_N\}$, $ODC(z) = ODC(z_1) \cdot ODC(z_2) \cdot ...ODC(z_N)$, that is, the ODC of output z is the intersection of the ODCs of all its fanout branches. Hence, the ODC of the output takes on the value 0 (i.e., output z is observable) if at least one of its branches drives an observable input of one of the immediate successors of module M_i. Subsequently, $ODC_{M_i}(x_j), \forall x_j$ are computed and $ODC(x_j), \forall x_j$ are determined using equation 2 as follows:

$$ODC(x_j) = ODC_{M_i}(x_j) + \prod_{q=1}^{N} ODC(z_q)$$

3.2.2 Activation Functions and ODCs

According to the definition given in Section 3.1, the activation function $ACTF_i$ of module M_i represents the set of conditions for which the module is idle, that is, it is not supposed to perform any useful computation; thus, its clock input can be masked off when $ACTF_i = 1$.
Given a module M_i with K inputs, $\{x_1, x_2, ..., x_K\}$, and given the observability don't care conditions for all such inputs (i.e., $ODC(x_j), j = 1...K$), the activation function for module M_i is given by the intersection of all the $ODC(x_j)$'s, that is:

$$ACTF_i = \prod_{j=1}^{K} ODC(x_j)$$

In fact, module M_i is idle for all the conditions such that none of its inputs is observable to the environment. In other words, when $ACTF_i = 1$, the clock signal feeding module M_i can be disabled, as no useful computation is going to be performed by the logic in M_i.

3.3 Generating the Clock-Tree Topology

The second stage of the LPClock methodology takes care of generating the logical topology of the clock tree. To this purpose, both activation functions and placement information are used.
For each module M_i in the design, the placed netlist contains information about its position, as well as the physical coordinates of its clock input (which is assumed to be unique and which we call *clock sink* in the sequel), denoted by (x_{s_i}, y_{s_i}). Also available is the capacitance C_i of module M_i, which is proportional to the number of flip-flops that are contained in M_i.
For each pair of RTL modules (M_i,M_j) in the design, we define their *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 ones allowed to the routing tools. Physical closeness means shorter interconnections, hence reduced congestion, shorter interconnection delay and smaller parasitic capacitance.

Besides the physical distance, we also define the *logical distance* between two modules M_i and M_j 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 M_i and M_j to be idle.
If $ACTF_i$ and $ACTF_j$ are completely independent, then $p(i, j) = P(ACTF_i = 1) \cdot P(ACTF_j = 1)$. Since the independence condition is not always satisfied, the probability $p(i, j)$ can be computed in a conservative way by means of RTL simulation: The values of $ACTF_i$ and $ACTF_j$ are collected over N consecutive simulation cycles and the number of times in which the logic AND of the two activation functions takes on the value 1 is calculated. In formula:

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

The logical distance measures the similarity of the activity of the two modules. If two modules with close activities are fed by the same net of the clock tree, the parent node of the net requires the clock signal for a percentage of time comparable to that of the children nodes, leading to a reduction of the overall activity in the tree.
The construction of the clock tree made by LPClock is a search into the space of all topological binary trees associated to the set of clock sinks. The search process is driven by a cost function, shown below, that includes both physical and logical distance information:

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

Parameters α and β allow the tuning of the weight of the wire length between modules (i.e., the physical proximity) versus the common activation of the modules (i.e., the logical proximity), while f and g are normalization functions for D and L.
The clock tree construction algorithm works hierarchically, building a binary topology on a level-by-level basis, proceeding in a bottom up fashion. A `current_set` is associated to each level of the tree that contains all the available sinks for that level. The algorithm aims at building the `current_set` that will contain all the sinks that belong to the next level of the tree.
The algorithm works as follows. Given the `current_set`, the $DIST(i, j)$ cost function is evaluated for every possible pair of sinks (i,j). Then, the pair (i,j) that gives the minimum value for the cost function is moved from the `current_set` to the `next_set` This operation is repeated until the `current_set` becomes empty, that is, all the sinks in that level have been paired and moved one level higher. Then, the newly created `next_set` becomes the `current_set` for the next level, and the process restarts. The construction tree procedure terminates when the `current_set` contains only two sinks and hence the `next_set` will contain the root of the tree.
When completed, the algorithm leads to a fully binary tree structure, whose leaves are all the RTL modules of the design. No clock-gating cells are included in the clock tree at this point. This is the subject of the final stage of the clock tree planning process, which is described next.

3.4 Inserting the Clock-Gating Cells

The last stage of the LPClock methodology targets the insertion and the propagation of the clock-gating cells on the branches of the clock tree in order to guarantee that, at any point in time of circuit operation, the largest possible fraction of the clock nets will be disabled.

Initially, the clock-gating cells are placed right in front of the sinks, i.e., they only condition the clock signals that enter the RTL modules. The gating cells are then repositioned in the tree through a procedure that tries to move them from the leaves of the tree topology towards the upper levels.

The algorithm that we have implemented is heuristic and it is driven by a cost function that, for each possible move, estimates the total clock tree power, using the model described in Section 3.1.

The clock tree is visited in a post-order fashion to search for configurations of the clock-gating cells in the tree corresponding to local minima of the cost function (i.e. the estimated power consumption).

For every node in the tree for which the branches to the two children nodes host a clock-gating cell (see Figure 5-a), three possible transformations can be applied. In case the activation functions of the two children nodes are the same, the best possible solution is certainly the one shown in Figure 5-b, since it guarantees maximum disabling of the clock signals for both children nodes and it requires the insertion of only one clock-gating cell that controls the entire sub-tree. However, there may be cases where the activation functions of the two children nodes differ substantially; in particular, the activation function of the right child may include most of the idle conditions of the left child, and many more. In this case, it may be worth resorting to the configuration shown in Figure 5-c, which allows the disabling of the clock signal to the right branch of the sub-tree even when the left sub-tree actually needs the clock. Clearly, also the symmetric case, shown Figure 5-d, may occur and it is thus handled by the procedure.

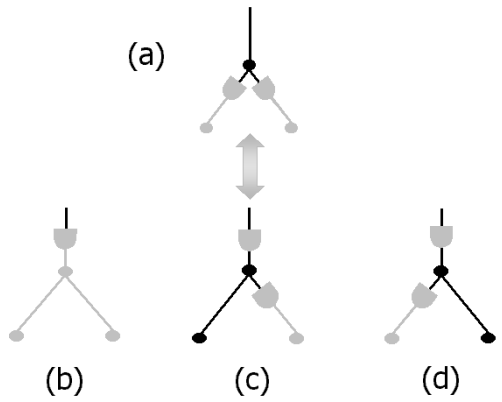

Figure 5: Gating Logic Propagation.

When the final position of the clock-gating cells inside the clock tree is determined, the control logic that combines the activation functions for each clock-gating cell is synthesized and it is passed to the RTL-to-layout synthesis flow, which will then consider the clock tree structure planned by LPClock during both final placement and clock tree synthesis and routing.

3.5 Handling Non-Hierarchical Designs

One fundamental assumption which stands at the basis of the LPClock methodology is that flip-flops belonging to the same RTL module are kept physically contiguous during the RTL-to-layout synthesis step. Unfortunately, there are practical cases in which this does not happen, due to the fact that the hierarchical nature of the design is not enforced during RTL-to-layout synthesis, leading to a layout structure in which physical contiguity of the RTL modules (and of the flip-flops located inside each of them) is lost. The flip-flops belonging to the same RTL module may end-up being spread far apart across the chip, thus making the planned clock tree logical topology highly suboptimal and of no practical use, as the routing of the clock sub-tree to the individual flip-flops contained in the RTL modules can be prohibitively expensive.

This section introduces the enhancements to the LPClock methodology which are needed to prevent the aforementioned undesirable phenomenon, and thus enable the applicability of LPClock also to designs with non-hierarchical (i.e., flat) structure.

The key idea to be pursued is that of forcing physical contiguity for the flip-flops inside an RTL module through the assertion of placement constraints. To this purpose, we introduce the concept of *pseudo-module*, which is defined as a set of flip-flops that are identified (and marked) as belonging to the same RTL module and for which the placement is constrained so that the flip-flops will be placed close to each other. This concept is exploited when the LPClock methodology has to be applied to flat designs, for example those which are produced by RTL synthesis.

Figure 6-a shows a layout where boxes represent flip-flops and different grey levels of color denote flip-flops that belong to different RTL modules. From the picture it is evident that the flip-flops of a given RTL module can be scattered in the final placement, if appropriate countermeasures are not adopted.

Introducing the definition of pseudo-module leads to a more localized layout structure for the flip-flops belonging to each RTL module, as shown in Figure 6(b), thus preserving (or reconstructing), at the physical level, the hierarchical structure that is initially available at the RTL, and that is essential for making the clock tree architecture planned by LPClock effective.

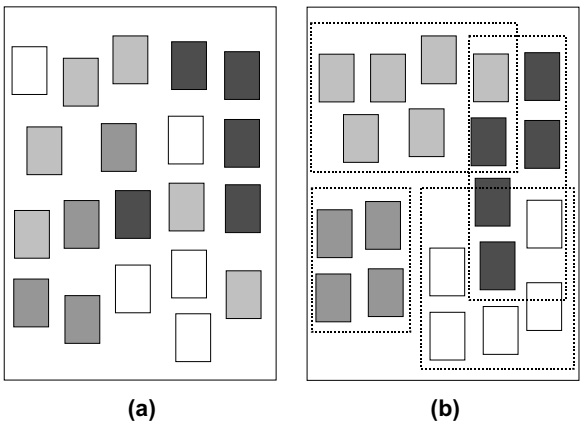

Figure 6: Handling Non-Hierarchical Designs.

4. INTEGRATION OF LPCLOCK INTO AN INDUSTRIAL FLOW

This section describes how the LPClock methodology is integrated into an industrial design flow that relies on commercial tools for RTL synthesis, optimization and physical design.
Figure 7 shows the flow in details.

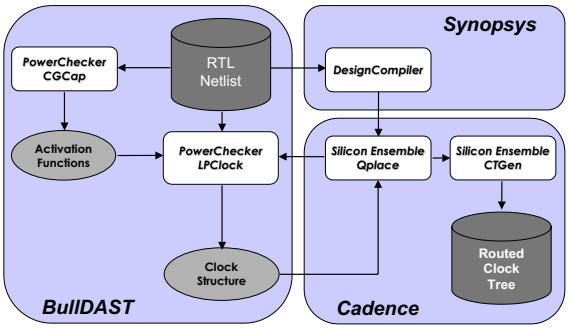

Figure 7: LPClock Integrated Flow.

Starting from a high-level design specification (i.e., VHDL or Verilog), the circuit is first elaborated by Synopsys DesignCompiler to obtain a RTL structural representation from which clocked modules and all nets, including the clock, are extracted. A floorplan and a placement are then initialized by Cadence SE-Qplace.
The LPClock algorithms have been implemented inside BullDAST PowerChecker, an integrated environment for RTL power estimation and optimization. PowerChecker features the CGCap optimization engine, which is capable of generating ODC-based gated-clock activation functions for all the modules in the RTL design starting from the initial specification.
The pre-placed netlist and the module activation functions are fed to LPClock, which generates the clock tree structure according to the methodology described in Section 3. The information about the clock network topology and the position of the clock-gating cells is introduced into the design database. This step requires to first change the `+PLACED` attribute of all the modules in the database to `+FIXED`, in order to avoid that the position of the modules changes during some subsequent optimizations. Next, incremental placement is invoked to include the clock tree structure and the clock-gating logic into the current view of the design.
The updated database is finally fed to Cadence SE-CTGen, which performs buffer insertion and checks for timing closure and final clock skew. 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 planned by LPClock.
By closing this section, we would like to emphasize that the LPClock methodology has general validity, and its usability is not limited to the environment (i.e., tools and flow) we have described above. As LPClock provides, as output, a "plan" of the clock tree consisting of a set of constraints, it can be easily mapped onto any RTL-to-layout flow with very little effort, as no conceptual changes are needed.

5. EXPERIMENTAL RESULTS

We have validated the LPClock flow on some benchmark circuits coming from different sources and domains, as well as on an industrial design case provided by Accent, i.e., an IEEE MAC 802.3 sublayer controller for a VCI bus with 10,100 and 1000 Mbit/s data rates.
Each design was first synthesized and mapped using Synopsys DesignCompiler and PowerCompiler. Then, we generated the placed and routed netlists (including the clock distribution network) using Cadence Silicon Ensemble Qplace for the original descriptions, as well as the netlists for the designs with gating logic inserted at the clock inputs of the RTL modules and with the clock tree structure created by LPClock. Layout extraction was performed next for all the circuits, 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 onto the $0.13\mu m$ HCMOS9 technology library by STMicroelectronics. Clock tree synthesis with Cadence Silicon Ensemble CTGen was performed using a *very tight maximum skew constraint* (less than 0.2% of the clock cycle).
LPClock was run with a value of the α/β ratio equal to one. This choice was made based on previous experience (see the analysis reported in [7]). In practical terms, this means that physical distance (parameter α) and logical distance (parameter β) have equal weight in the cost function that drives LPClock.
In the following sections we present and discuss the results we have achieved for the two classes of circuits.

5.1 Benchmark Circuits

We have considered a total of eight benchmark circuits, characterized by different functionalities and sizes. Some of them are publicly available and are quite simple (no more than 2000 library cells), some others come from industry and are more complex (up to 33000 cells). Details about the circuits are summarized in Table 1.

Benchmark	*# of Gates*	*# of Clock Sinks*
`Simple1`	140	72
`Simple2`	185	68
`Simple3`	1870	624
`Simple4`	1943	680
`Indust1`	13954	1726
`Indust2`	17125	2054
`Indust3`	24587	2963
`Indust4`	33180	5450

Table 1: Characteristics of Benchmark Circuits.

Table 2 collects the results of the experiments. In particular, column *Clock-Gating* shows the savings in the power consumed by the clock tree w.r.t. the original circuit implementation achieved by inserting the clock-gating logic only at the inputs of the RTL modules. On the other hand, column LPClock shows the clock tree power savings against the original circuits obtained by inserting the clock-gating logic as suggested by LPClock. A comparison of the clock power data for the two optimized circuits shows that LPClock offers an additional savings over traditional clock-gating that ranges from 3.58% to 42.03%, depending on the benchmark (column Δ).

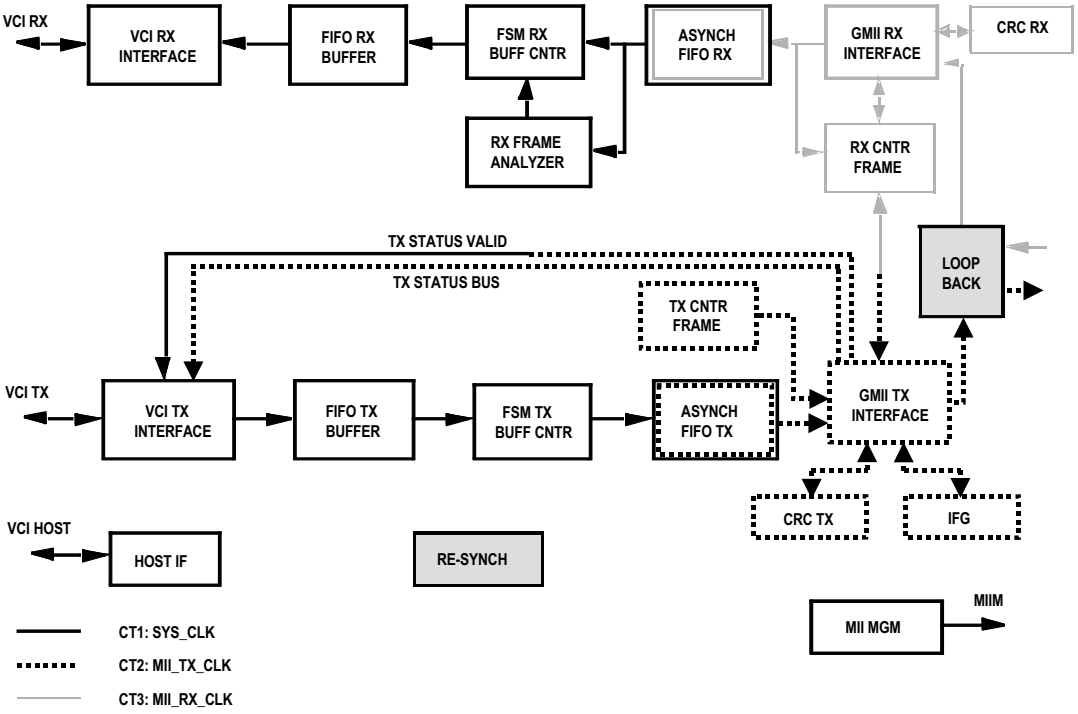

Figure 8: Block Diagram with Clocking Scheme of the MAC 802.3 Controller.

Benchmark	*Clock-Gating*	LPClock	Δ
Simple1	12.24%	18.32%	6.92%
Simple2	43.86%	45.87%	3.58%
Simple3	36.70%	41.78%	8.02%
Simple4	9.94%	22.94%	14.43%
Indust1	25.15%	56.61%	42.03%
Indust2	22.31%	39.88%	22.61%
Indust3	14.38%	37.23%	26.68%
Indust4	17.72%	43.26%	31.04%

Table 2: Results on Benchmark Circuits.

The experimental data show very clearly that the clock trees generated using LPClock as a preprocessor to CTGen are much superior (in terms of power) to those generated by CTGen at the end of the traditional flow for circuits of significant size, while they are limited (i.e., below 15%) on smaller benchmarks. This was somehow expected, as for small circuits the clock distribution networks tend to have very simple structures, and thus the degrees of freedom that are available for the optimization are reduced.
Timing analysis was performed on the synthesized netlists containing capacitance information back-annotated after extraction using Synopsys PrimeTime. The results have shown that *no skew violation* occurred for all the benchmarks. This is a very important result, as it indicates the quality of the constraints for the clock tree structure that LPClock was able to generate.

5.2 Industrial Design: MAC 802.3 Controller

The IEEE 802.3 International Standard for Local Area Network (LAN) employs the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) as the access method. The MAC 802.3 (media access control) controller implements the LAN CSMA/CD sublayer for the following families of systems: 10 Mb/s, 100 Mb/s and 1000Mb/s of data rates for baseband and broadband systems. Half and full-duplex operation modes are supported. The collision detection access method is applied only to the half-duplex operation mode. The frame bursting is supported for half duplex and speed above 100Mb/s. The MAC control frame sublayer (optional) is supported by the current implementation. VCI (Virtual Component Interface) buses (a super set of the standard bus) are used as application and host interfaces. The MII (Media Independent Interface) standard bus is used for the PHY interface.
Figure 8 shows the top-level block diagram of the MAC 802.3 controller, highlighting the implemented clocking scheme. There are three clock domains in the design; the system clock (CT1, indicated by the black, solid lines), the MII TX clock (CT2, indicated by the black dashed lines), and the MII RX clock (CT3, indicated by the grey solid lines). The suggested operating frequency for the system clock is 166MHz; instead, both the MII TX and the MII RX clocks have a suggested operating frequency of 125MHz.
Signals that cross different clock domains are resynchronized in the "RESYNCH" module shown at the bottom of the block diagram (i.e., the configuration bits and the handshaking signals).

The two asynchronous FIFOs are used to detach the data between the system clock and the MII clock domains.
In loopback mode, the MII TX clock is used also on the RX path, therefore the clock trees `CT2` and `CT3` must be balanced starting from the common root `mii_tx_clk` (see the schematic of Figure 9).

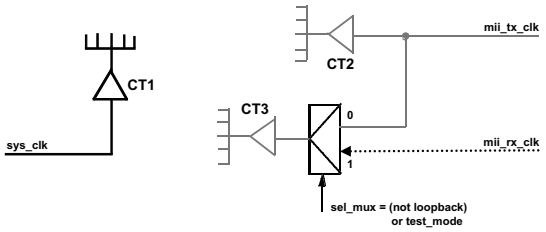

Figure 9: Clock Tree Roots.

The MAC 802.3 controller has been synthesized to the physical level using the same procedure adopted for the benchmark circuits of Section 5.1. The final implementation consists of around 110.000 library cells and around 8.000 clock sinks (for the three clock domains).
Clock tree power consumption results obtained on the original design (with traditional clock-gating cells inserted in front of the RTL module inputs) and on the one for which LPClock was used as a preprocessor are compared in Table 3. Savings are larger for the `CT1` clock tree, mainly due to its larger capacitive load, while they are more limited for clock trees `CT2` and `CT3`. Overall, clock tree power savings are around 16%.

Clock Domain	Δ
`CT1`	18.90%
`CT2`	13.34%
`CT3`	11.21%
`Total`	16.35%

Table 3: Results on the MAC 802.3 Controller.

As for all the benchmarks of Section 5.1, no clock skew penalty is introduced by the adoption of the clock tree structure generated by LPClcok, showing the practical applicability of the LPClock methodology to real-life design cases.

6. CONCLUSIONS

Interconnect capacitance is becoming more and more dominant in very deep-submicron technologies; as a consequence, the clock distribution network currently represents the major performance and power consumption bottleneck in modern processors and SoCs.
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 solve this problem have not found a direct validation into industry-strength design flows.
In this paper, we have introduced a new approach to reduce clock tree power consumption based on clock-gating. More specifically, we have presented the LPClock methodology, which enables us to automatically generate clock tree routing constraints to be fed to the back-end tools starting from a pre-placed RTL specification.
Distinguishing feature of the methodology is its capability of exploiting both physical and logical information of the given RTL design to optimize the clock tree structure. In particular, LPClock takes advantage of innovative techniques for determining clock-gating conditions that are more powerful than existing solutions; in fact, activation functions are calculated by looking at the circuit behavior and functionality, and not just at its topology and structure.
The LPClock methodology has been integrated into an industrial design flow, which adopts Synopsys DesignCompiler as front-end, Cadence Silicon Ensemble as back-end and BullDAST PowerChecker as development framework.
Validation has been carried out on a set of benchmark circuits, as well as on an industrial design case (namely, an IEEE MAC 802.3 controller provided by Accent). For the benchmarks, experimental results showed clock power savings ranging from 3.5% to 40% over the circuits that included traditional clock-gating. Regarding the IEEE MAC 802.3 controller, which contains a total of three clock domains, clock power savings were around 16% w.r.t. traditional clock-gating. In all the cases, no skew increase was observed after the optimization suggested by LPClock.

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.

7. REFERENCES

[1] T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Vol. 34, No. 4, pp. 52-58, April 2001.

[2] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, F. Baez, "Reducing Power in High-Performance Microprocessors," *DAC-35: ACM/IEEE Design Automation Conference*, pp. 732-737, San Francisco, CA, June 1998.

[3] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, R. L. Allmon, "High-Performance Microprocessor Design," *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 5, pp. 676-686, May 1998.

[4] D. R. Gonzales, "Micro-RISC Architecture for the Wireless Market," *IEEE Micro*, Vol. 19, No. 4, pp. 30-37, July-August 1999.

[5] D. Duarte, V. Narayanan, M. J. Irwin, "Impact of Technology Scaling in the Clock System Power," *IEEE Computer Society Annual Symposium on VLSI*, pp. 52-57, Pittsburgh, PA, April 2002.

[6] D. Duarte, V. Narayanan, M. J. Irwin, "A Clock Power Model to Evaluate Impact of Architectural and Technology Optimizations," *IEEE Transactions on VLSI Systems*, Vol. 10, No. 6, pp. 844-855, December 2002.

[7] M. Donno, A. Ivaldi, L. Benini, E. Macii, "Clock-Tree Power Optimization based on RTL Clock-Gating," *DAC-40: ACM/IEEE Design Automation Conference*, pp. 622-627, Anaheim, CA, June 2003.

[8] P. Babighian, L. Benini, E. Macii, "A Scalable ODC-Based Algorithm for RTL Insertion of Gated Clocks," *DATE-04: IEEE 2004 Design Automation and Test in Europe*, pp. 500-505, Paris, France, February 2004.

[9] J. G. Xi, W. W.-M. Dai, "Buffer Insertion and Sizing under Process Variations for Low-Power Clock Distribution," *DAC-32: ACM/IEEE Design Automation Conference*, pp. 491-496, San Francisco, CA, June 1995.

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

[11] 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.

[12] 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.

[13] M. Igarashi, K. Usami, K. Nogami, F. Minami, Y. Kawasaki, T. Aoki, M. Takano, C. Misuno, T. Ishikawa, M. Kanazawa, S. Sonoda, M. Ichida, N. Hatanaka, "A Low-Power Design Method using Multiple Supply Voltages," *ISLPED-97: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 36-41, Monterey, CA, August 1997.

[14] 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.

[15] K. D. Boese, A. B. Kahng, "Zero-Skew Clock Routing Trees with Minimum Wire Length," *IEEE International Conference on ASIC*, pp. 1.1.1-1.1.5, Rochester, NY, September 1992.

[16] T. H. Chao, Y. C. Hsu, J. M. Ho, "Zero Skew Clock Net Routing," *DAC-29: ACM/IEEE Design Automation Conference*, pp. 518-523, Anaheim, CA, June 1992.

[17] M. Edahiro, "A Clustering-Based Optimization Algorithm in Zero-Skew Routing," *DAC-30: ACM/IEEE Design Automation Conference*, pp. 612-616, Dallas, TX, June 1993.

[18] H. Zhang, J. Rabaey, "Low-Swing Interconnect Interface Circuits," *ISLPED-98: ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 161-166, Monterey, CA, August 1998.

[19] A. P. Chandrakasan, S. Sheng, R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473-484, April 1992.

[20] 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, December 1994.

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

[22] F. Theeuwen, E. Seelen, "Power Reduction Through Clock Gating by Symbolic Manipulation," *VLSI: Integrated Systems on Silicon*, pp. 389-400, Gramado, Rio Grande do Sul, Brazil, August 1997.

[23] 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 of Electronic Systems*, Vol. 4, No. 4, pp. 351-375, October 1999.

[24] L. Benini, M. Favalli, G. De Micheli, "Design for Testability of Gated-Clock FSMs," *EDTC-96: IEEE European Design and Test Conference*, pp. 589-596, Paris, France, March 1996.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.