Activity-Driven Clock Design

Amir H. Farrahi, *Senior Member, IEEE*, Chunhong Chen, *Member, IEEE*, Ankur Srivastava, *Member, IEEE*, Gustavo Téllez, and Majid Sarrafzadeh, *Fellow, IEEE*

***Abstract*—In this paper, we investigate reducing the power consumption of a synchronous digital system by minimizing the total power consumed by the clock signals. We construct activity-driven clock trees wherein sections of the clock tree are turned off by gating the clock signals. Since gating the clock signal implies that additional control signals and gates are needed, there exists a tradeoff between the amount of clock tree gating and the total power consumption of the clock tree. We exploit similarities in the switching activity of the clocked modules to reduce the number of clock gates. Assuming a given switching activity of the modules, we propose three novel activity-driven problems: a clock tree construction problem, a clock gate insertion problem, and a zero-skew clock gate insertion problem. The objective of these problems is to minimize system's power consumption by constructing an activity-driven clock tree. We propose an approximation algorithm based on recursive matching to solve the clock tree construction problem. We also propose an exact algorithm employing the dynamic programming paradigm to solve the gate insertion problems. Finally, we present experimental results that verify the effectiveness of our approach. This paper is a step in understanding how high-level decisions (e.g., behavioral design) can affect a low-level design (e.g., clock design).**

***Index Terms*—Activity-driven clock design, algorithms, clock gating, clock tree, power optimization, zero-skew clock.**

I. Introduction

MODERN digital systems are designed with a target *clock period* (or *clock frequency*), which determines the rate of data processing. A clock network distributes the clock signal from the clock generator, or *source*, to the clock inputs or *sinks* of the synchronizing components, or *modules*. The clock distribution network consumes large percentage (20%–50%) of the power consumed by these systems. Therefore, in low-power synchronous systems, we would like to minimize the total power consumed by the clock tree subject to performance constraints on the clock signal, such as the operating frequency and maximum clock skew.

The power consumed by complementary metal–oxide–semiconductor (CMOS) circuits consists of two components: dynamic and static power. The static power is largely determined by the technology. In this paper, we only consider minimizing the dynamic power. The dynamic power consumed by a module clocked at a frequency f is given by $fV_{\rm dd}^2C_L$, where $V_{\rm dd}$ is the supply voltage and C_L is the total load capacitance on the circuit. If a circuit switches α times per clock cycle, then its power consumption is given by $P = \alpha f V_{\rm dd}^2 C_L$, where α is called the *circuit activity*. To minimize the power consumed by a CMOS synchronous system, we would in turn like to minimize its total activity.

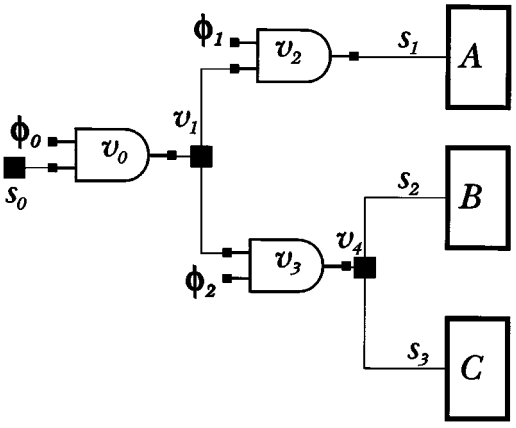

Fig. 1. Gated clock tree, with synchronizing elements A, B and C, source s_0, sinks $\{s_1, s_2, s_3\}$, clock gates $\{v_0, v_2, v_3\}$, control signals $\{\phi_0, \phi_1, \phi_2\}$, and Steiner nodes $\{v_1, v_4\}$.

In a normal clock tree, the clock signal arrives regularly at all of the clock sinks, which means $\alpha = 1$. Suppose that we know the times at which the clocked sinks must be active. We refer to the set of active/idle times for the module as *activity patterns*. They can be obtained by simulation of the design at the behavioral level. The clock signal must be supplied to the modules only during their active times. If the clock signal is gated such that it is only delivered during these times we can reduce the total power consumed by the clock and by the modules themselves. We call a clock tree thus constructed an *activity-driven clock tree*. In this paper, we address the problem of minimizing the power consumption of a synchronous system by minimizing its activity through the use of an activity-driven clock tree. Fig. 1 shows an example of gated clock tree.

Work on clock trees has focused on zero- or near zero-skew routing [4], [6], [13], [21]. In addition to zero skew, further work concentrates on routing clock trees with minimal total wire length [3], [7], [8]. The construction of clock trees that minimize phase delay of the clock signal has been studied in [5] and [9]. Work on buffered clock trees has focused on minimizing the phase delay of the clock tree [20]. More recently, work in [17] considers the minimization of skew and delay in the presence of process variations in buffered clock trees. For a survey on clock network construction issues, see [2], [10], and [15].

Manuscript received June 7, 1999; revised May 8, 2000. This work was supported in part by the National Science Foundation under Grant MIP-9527389. This paper was presented in part at the IEEE International Conference on Computer-Aided Design, San Jose, CA, November 1995. This paper was recommended by Associate Editor M. Pedram.

A. H. Farrahi is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

C. Chen, A. Srivastava, and M. Sarrafzadeh are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA.

G. Téllez is with the IBM Corporation, Essex Junction, VT 05452 USA.

Publisher Item Identifier S 0278-0070(01)03542-4.

0278–0070/01$10.00 © 2001 IEEE

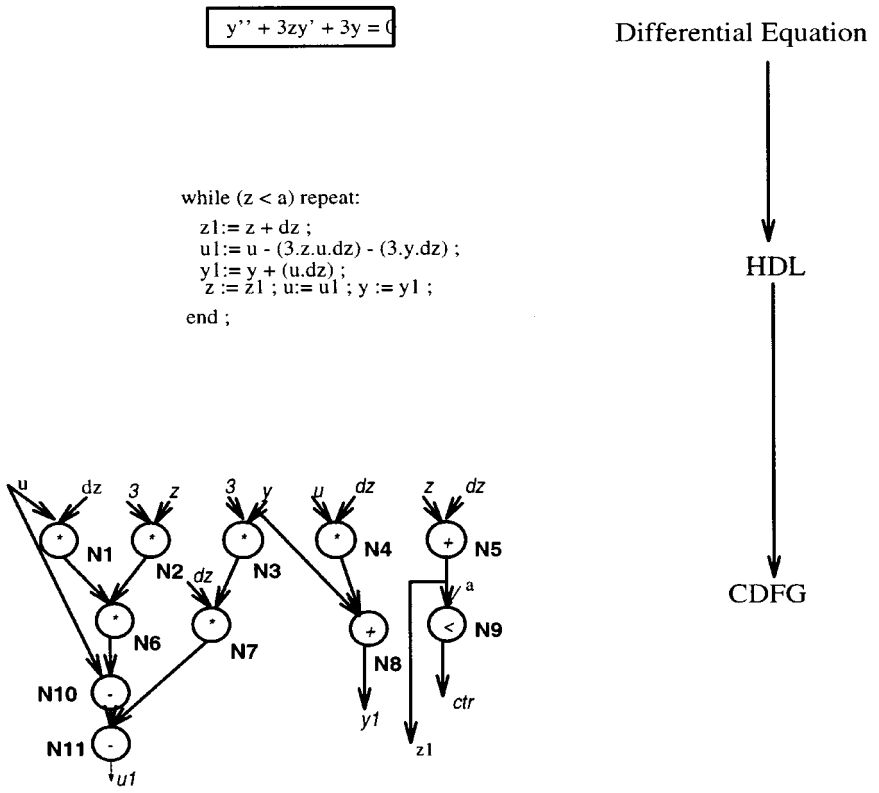

Fig. 2. HDL description of a DE. DE has been transformed to a CDFG. Each CDFG node is labeled with a letter.

The rest of this paper is organized as follows. In Section II, we introduce some definitions and terminology. In Section III, we outline the construction of activity-driven clock tree using a motivational example. In Section IV, we formulate two problems: the activity-driven clock tree construction (ADCTC) problem and the minimum-power activity-driven clock gate insertion problem. We propose an approximate solution to the ADCTC problem in Section V. Section VI presents an exact algorithm that solves the minimum-power activity-driven clock gate insertion problem. The solution is also extended to handle gate insertion with zero skew. Section VII shows experimental results from the proposed algorithms. Finally, we conclude the paper in Section VIII.

II. Terminology

Let a synchronous system be denoted by $\mathcal{C}(\mathcal{M},\mathcal{N})$, where $\mathcal{M} = \{m_i \,|\, i = 1,2,\ldots,M\}$ denotes a set of components (or *modules*) and $\mathcal{N} = \{n_j \,|\, j = 1,2,\ldots,N\}$ denotes a set of nets of the system. A *clock net* N_C consists of a clock source node s_0 and a set of sinks $S = \{s_i \,|\, i = 1,2,\ldots,n\}$, where each sink belongs to a module. For each clock sink, we divide the periods of activity and inactivity. For each sink s_i, we define the *activity pattern* U_i as a bit pattern $U_i = \{a_{ij} \,|\, j = 1,2,\ldots,u, a_{ij} \in \{0,1\}\}$, where a "1" defines an active period, "0" defines an idle period, and u is the total number of periods for the sink. A *clock tree* is a rooted tree $\mathcal{T}(V,E)$ over S with clock source s_0 being the root and S being the leaves of the tree. The *internal* nodes of the clock tree are $\{v \,|\, v \in V - (S \cup \{s_0\})\}$. An edge $e_{ij} \in E$ connects a parent node v_i and a child node v_j. We denote the parent node of v_j as Parent(j). We denote the number of children (or out-degree) of v_i as d_i and the set of children of v_i as Child(i). If node v_i lies in the path from node v_j to the root (leaf), then node v_i is said to lie *above* (*below*) node v_j. A node v_i is said to be at level lev(i) if there are lev(i) edges on the path from v_i to the root. The height h of a tree is the largest level of any node of that tree.

III. Clock Gating—A Motivational Example

In this section, we first show a methodology for obtaining activity patterns which are the inputs of the *activity-driven clock tree* problem. Then, we illustrate the basic idea behind construction of activity-driven clock trees. Also, we look at the effect of clock gates on power consumption.

A. Obtaining Activity Patterns

As mentioned in the introduction, we seek to construct clock trees that can be gated such that the total activity in the tree is reduced. The first step toward constructing such a clock tree is to capture activity patterns for the modules from high-level synthesis. This can be done by the following procedure.

1) Start from a high-level description of the system. Fig. 2 shows an example for a hardware description language (HDL) of a differential equation (DE) and its compilation to the representation of control-data flow graph (CDFG).
2) Schedule and allocate a set of modules into a set of control steps. The result for Fig. 2 is shown in column 2 of Table I.
3) If a module is assigned to a control step, then the module is active during that control step, otherwise the module is idle. Column 3 of Table I shows the activity pattern for each module in Fig. 2

TABLE I
ALLOCATION AND SCHEDULING OF THE DE EXAMPLE CDFG SHOWN IN FIG. 2

Module	Control Step C1	C2	C3	C4	C5	C6	Activity Pattern
M1	N1	N1	N6	N6			111100
M2	N2	N2		N7	N7		110110
M3	N3	N3					110000
M4	N4	N4					110000
A1			N5				001000
A2			N8				001000
S1					N10	N11	000011
C1				N9			000100

This allocation uses six control steps, four multipliers (M1, M2, M3, M4), two adders (A1, A2), one subtractor (S1), and one comparator (C1) for a total of eight modules.

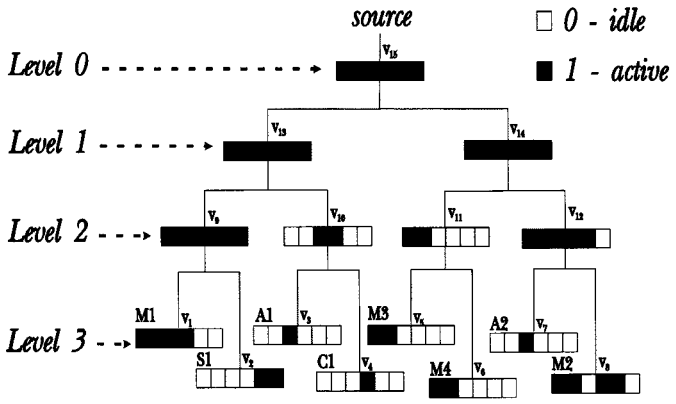

Fig. 3. DE Example 1. Clock tree circuit for the modules of the DE circuit. Each module has a fill-pattern, which depends on its activity pattern. Activity patterns for the subtrees are also shown.

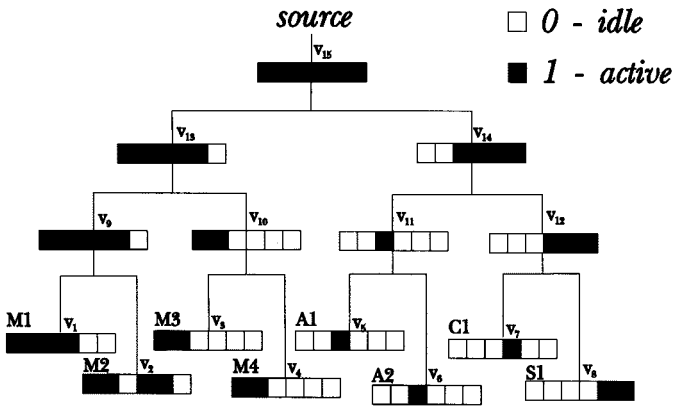

Fig. 4. DE Example 2. Better tree for the DE circuit, which reduces the total activity by clustering modules with similar activity patterns.

The above steps work well with digital signal processor circuits, where the data activity is well known. We assume that the modules have registers on their inputs that are to be clocked. If the clock signal is not fed to the inputs of the module's register, then the outputs of the register will not change and, thus, the module will not consume dynamic power. Given the activity patterns the clock construction stage can be executed at two phases in the design.

1) Before floorplanning/placement of the modules, we seek to minimize the total activity of the clock tree. We then add the results of this stage to the placement problem as follows. Let the graph $G_p(V, E)$ denote a placement graph obtained from the circuit $\mathcal{C}(\mathcal{M}, \mathcal{N})$. Let the edges of this graph be weighted, such that the weights denote priorities in increasing order. Given an activity-driven clock tree $\mathcal{T}(V_C, E_C)$, we use the mapping of the clock tree sinks S to the placement graph vertices V. For each vertex in this mapping, there exists a leaf vertex in the clock tree. For each pair of leaf vertices in the clock tree, we add an edge to the graph G_p with a weight proportional to the length of the simple path between the vertices. Hence, if the vertices are located nearby in the tree, the placement algorithm will tend to place them closely.
2) After floorplanning/placement of the modules is completed, power consumption can be estimated from the placement and routing information. The objective of the clock construction is to come up with a suitable power tradeoff, taking wiring penalties into account. The problem of constructing the clock tree at this stage has been well studied [6], [13], [14].

Fig. 3 illustrates a possible topology of a binary clock tree with eight sinks that correspond to eight modules in Table I. Each module is drawn according to its activity pattern. The activity patterns and modules are obtained from Table I. The tree also contains activity patterns of its internal nodes. The activity pattern of an internal node is calculated by OR-ing (bitwise OR operation) the activity patterns of its two children. We observe that modules connected to the same clock subtree can be driven by one clock gate. The most effective activity pattern of a clock gate is one that feeds the clock signal to the modules only when needed. Consider modules A2 and M2, which have activity patterns 001000 and 110110, respectively. Both modules are connected in the clock tree forming a subtree rooted at the internal node v_{12} with activity pattern 111110. We use the operator $\vee$ to denote the bitwise OR operation that is used to obtain the activity pattern of an internal node (or a subtree rooted at this node). The activity pattern U of a subtree with sinks S is therefore obtained by $U = U_1 \vee U_2 \vee \cdots \vee U_n$.

Fig. 4 shows another clock tree topology for the above example. In the figure, the modules of similar activity are placed close to one another. By doing so recursively, we can increase the total number of idle periods. Since the power consumption of the clock and module can be reduced by gating the clock during idle periods, the total number of idle periods in a clock tree is a measure of the power saving by clock gating. Table II compares the quality of the trees in the DE Examples 1 and 2 (shown in Figs. 3 and 4, respectively) by this criteria. Note that the total number of time periods in the full tree is 90.

It should be pointed out that the total number of idle time periods is just an approximate estimation of power saving by clock gating. The reason is that inserting clock gates introduces additional power consumption of gate control signal and gate control logic. For instance, the different number of active/idle transitions in the patterns can lead to different power consumed

TABLE II
COMPARISON OF THE TWO CLOCK TREES OF FIGS. 4 AND 5 IN TERMS OF TOTAL NUMBER OF IDLE PERIODS

Tree Level	Number of Idle Time Periods	
	DE Example 1	**DE Example 2**
Level 3	31	31
Level 2	9	13
Level 1	0	3
Power Savings	40/90=44%	47/90=52%

Improvement is obtained by matching modules with similar patterns.

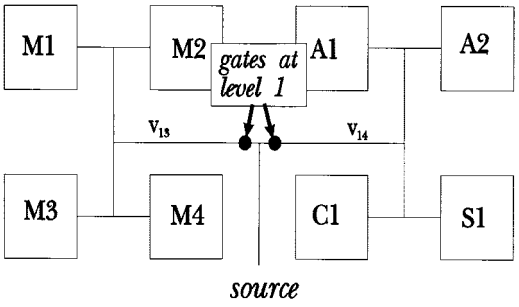

Fig. 5. H-tree placement for DE Example 2 with two clock gates inserted at level 1. Clock gates are drawn as circles.

by the gate control signal. More discussions will be given in Section IV.

B. Locating Clock Gates

Given a clock tree topology, we seek to locate the clock gates in the tree such that the total power is minimized. The objective is to turn off as much capacitance as possible. This motivates us to place the gates close to the parent node. The gate insertion problem requires detailed information about the parasitic capacitances of the clock tree and the control lines of the gates. For this reason, we model the module placement by embedding the clock tree in the plane using an H-tree structure. Example of such an embedding is shown in Fig. 5.

We now show the effect of clock gate placement on power consumption. By placing the gates at level 1 in the H-tree embedding of the DE Example 2 (see Fig. 5), we accomplish two things: the buffer driver (or level-zero gate) only drive the two gates in question, and the activity patterns for the gates v_{13} and v_{14} are 111110 and 001111, respectively. Since there are no other gates under these gates, the activity patterns of all the nodes in their subtrees is inherited from the driving gates at the subtree roots. Assuming that the modules are embedded in a uniform grid with grid spacing one, the wire length of the leaf H-trees is three. An example of a leaf H-tree is the set of vertices $\{v_1, v_2, v_9\}$. At the next level, the total wire-length of the H-tree doubles. Therefore, the total wire-length under the gate for vertex v_{13} is eight. Since the capacitance contribution of a wire of length ℓ is $\beta\ell$, where β is the capacitance per unit length, the power contribution of this wire is $fV_{\rm dd}^2\beta\ell$. We will set $fV_{\rm dd}^2\beta = 1$, and, thus, the power contribution of the wiring is equal to its length. We assume that this is the power contribution of the wiring per activity period. As another example of locating clock gates, Fig. 6 shows four gates inserted at level 2

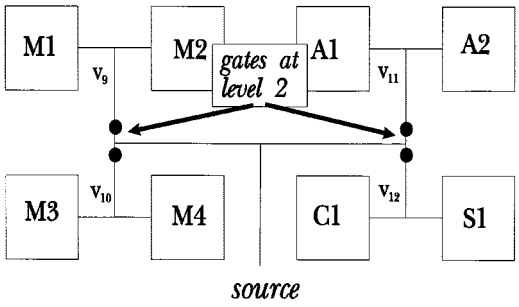

Fig. 6. H-tree placement for DE Example 2 with four clock gates inserted at level 2.

TABLE III
COMPARISON OF POWER CONSUMPTION FOR THE TWO GATED CLOCK TREES OF FIGS. 5 AND 6

Power Contribution	Fig. 5	Fig. 6
Wiring Power	84	69
Gate Power	2	4
Module Power	208	164
Total	294	237

We assume that active power and idle power for a multiplier are $P_A = 8$ and $P_I = 2$, respectively, active and idle power for other modules are $P_A = 2$ and $P_I = 1$, respectively, and the power per gate is one.

of the same tree. Table III compares the power contributions of Figs. 5 and 6.

In the following sections, we will formulate and solve three problems related to activity-driven clock trees: clock tree construction, clock gate insertion, and clock gate insertion with zero skew.

IV. PROBLEM FORMULATION

Assume that for each clock sink we can measure the total power consumed under the following circumstances: the total *active circuit power* P_A, during periods when the circuits are active and the clock, must be supplied for proper function, and the total *inactive or idle circuit power* P_I, when clock supply is unnecessary. For each sink, the periods of activity and inactivity are defined in an activity pattern $U_i = \{a_{ij} \,|\, j = 1, 2, \ldots, u, a_{ij} \in \{0, 1\}\}$. The total power consumed by the clock tree is

$$P = \sum_{i=1}^{n}\sum_{j=1}^{u}[a_{ij}P_{A_i} + (1 - a_{ij})P_{I_i}] + P^T + P^G \quad (1)$$

where

P_{A_i} and P_{I_i} power consumption of the i-th module during each active period and each idle period, respectively, and, hence, the first summation term denotes the power consumed by all modules;

P^T power consumed by tree sections (i.e., wiring power);

P^G power consumed by clock gates (i.e., gate power).

where Let the clock tree be partitioned into a set of r clock tree sections each corresponding to an edge of which r_I tree sections are connected only to internal nodes. Given that the k-th tree

section has activity pattern $U_k^T = \{a_{kj}^T \,|\, j = 1, 2, \ldots, u, a_{kj}^T \in \{0, 1\}\}$ $(k = 1, 2, \ldots, r)$ and consumes $p_{A_k}^T$ power during each active period, its power consumption is given by

$$P_k^T\left(U_k^T\right) = \sum_{j=1}^{u} a_{kj}^T p_{A_k}^T. \tag{2}$$

If a clock gate is added to this section, the power consumed by the gate (denoted by P_k^G) can be computed as follows. Given that the activity of the clock gate is U_k^T, the activity pattern of its input clock signal is $U_k^C = \{a_{kj}^C \,|\, j = 1, 2, \ldots, u, a_{kj}^C \in \{0, 1\}\}$ (where C denotes clock signal) and the input and output capacitances of the gate cause additional input power $P(\text{in})$ and output power $P(\text{out})$ during its active period, respectively, we have

$$P_k^G\left(U_k^C, U_k^T\right) = b_k t\left(U_k^T\right) + \sum_{j=1}^{u}\left[a_{kj}^T P(\text{out}) + a_{kj}^C P(\text{in})\right] \tag{3}$$

where b_k is a constant and $t(U_k^T)$ is a function that measures the changes in activity (and, hence, power consumption) of the control signals of clock gates. If we define a set of transitions $\text{TR}(U_k^T) = \{a_{kj}^T a_{k,(j+1)\bmod u}^T \,|\, a_{kj}^T \neq a_{k,(j+1)\bmod u}^T, j = 1, 2, \ldots, u\}$, then $t(U_k^T) = |\text{TR}(U_k^T)|$. Based on the above discussions, we rewrite (1) as follows:

$$\begin{aligned} P = &\sum_{k=1}^{r_I}\left[P_k^T\left(U_k^T\right) + P_k^G\left(U_k^C, U_k^T\right)\right] \\ &+ \sum_{i=1, k=r_I+1}^{n, r}\left[P_k^T\left(U_k^T\right) + P_k^G\left(U_k^C, U_k^T\right)\right] \\ &+ \sum_{i=1, k=r_I+1}^{n, r}\left[\sum_{j=1}^{u}\left(a_{kj}^T P_{A_k} + \left(1 - a_{kj}^T\right) P_{I_i}\right)\right]. \end{aligned} \tag{4}$$

If we choose not to gate a tree section, then the gate power penalty is zero and the activity pattern of that section is inherited from the parent section. The additional power (P_k^G) consumed by adding a clock gate to the clock tree comes from the additional activity and capacitance added to the circuit by the clock gate control signals and by the input and output clock gate capacitances.

We formulate two general clock tree power minimization problems, namely, the problem of constructing an activity-driven clock tree and the problem of minimizing the total power in an activity-driven clock tree by clock gate insertion.

1) ADCTC Problem : Construct a tree $\mathcal{T}(V, E)$ on a set of sinks S such that the weighted sum of nodes activities

$$\mathcal{A}(\mathcal{T}) = \sum_{v_i \in V}\left[b_i t\left(U_i^T\right) + \sum_{j=1}^{u} q_{ij} a_{ij}\right] \tag{5}$$

in the resulting tree is minimized, where q_{ij} is the weight and a_{ij} is the activity pattern of sinks and/or internal nodes of the tree.

Equation (5) is very flexible, since the weights q_{ij} and b_i can be defined to represent the exact power or an approximate power consumed by the clock tree sections, the clock gates, and/or the circuit modules. This can be done by manipulating (2)–(4) into the form of (5). Hence, the quantity $\mathcal{A}(\mathcal{T})$ can be tuned to measure or estimate as needed the power consumed by the system.

We formulate the next problem using the definition of the activity function given above. Once we have selected a clock tree, we would like to insert clock gates such that the total power consumed by the system is minimized.

2) Activity-Driven Minimum Power Gate Insertion (ADMPGI) Problem : If a gate is added at node $v_i \in V$ in a clock tree $\mathcal{T}(V, E)$, it increases weighted activity to v_i due to its input clock signal and control signal, but may reduce weighted activity to nodes in the subtree rooted at v_i. Find a set of gates inserted in the tree such that $\mathcal{A}(\mathcal{T})$ is minimized.

V. Clock Algorithm with Provable Bounds

In this section, we consider the ADCTC problem. We propose a heuristic algorithm to solve this problem. The clock tree has to be constructed keeping signal skew in mind. The objective is to construct a complete binary tree. The new algorithm, so-called ClockAct, is based on recursive weighted matching. The idea behind the algorithm is to construct a tree in a bottom-up fashion while minimizing the objective on a level by level basis. We model the problem as follows: we would like to match pairs of sinks as well as possible. The matching criteria is the value of the objective given the activity pattern of the subtree containing both sinks.

We can construct a complete weighted graph $G(V, E)$, where V represents the current set of sinks, and the weighted edges $e_{ij} = (v_i, v_j) \in E$ represent all the possible matching pairs. The edge weights are defined as $w(e_{ij}) = \mathcal{A}(\mathcal{T}_i \cup \mathcal{T}_j)$, where $\mathcal{T}_i$ is the subtree rooted at v_i. A *minimum weighted matching* of G selects a set of edges $E_m \in E$, such that $\sum_{e_{ij} \in E_m} w(e_{ij})$ is minimized subject to: 1) $|E_m| = \lfloor |V|/2 \rfloor$ and 2) for any vertex $v_i \in V$, there exists only one edge $e_{ij} \in E_m$. Each selected pair of vertices is then connected in the clock tree to a parent vertex. The algorithm used to connect the vertices to the parent vertex depends on the stage in which the algorithm is being used. If the algorithm is producing a global routing of the clock tree, then this step can be accomplished using a *zero-skew merge algorithm* [21]. The vertex pair is then deleted from the set of matching candidates and the parent vertex is added to the set of candidates to be matched by the next stage. The matching is repeated until only two vertex-matching candidates remain. The pseudocode for this algorithm is shown in Table IV.

In the following, we show that if the weights of the cost function have certain properties, the recursive matching algorithm will yield a δ-approximation [12].[1]

Theorem 1: The recursive matching is a δ-approximation algorithm for the following cases.

Case 1) $\delta = 2$ when all weights $q_{ij} = \kappa$ and $b_i = \gamma < 2\kappa$;

Case 2) $\delta = 3$ when the target tree topology is an H-tree, and the tree topology dominates the values of the q_{ij} weights, or and $b_i = \gamma < 2q_{ij}$.

[1]An algorithm is said to be a δ-approximation algorithm if the cost H of the solution produced by the algorithm is within a factor of δ of the cost H_0 of the optimal solution, i.e., $\max\{(H/H_0), (H_0/H)\} \leq \delta$, where $\delta \geq 1$ is a constant.

TABLE IV
PSEUDOCODE FOR ALGORITHM **CLOCKACT**

Algorithm ClockAct(S)
Input: set of sinks S.
Output: $\mathcal{T}(V,E)$ clock tree on S.

```
V' ← S; V ← ∅;
While (|V'| > 2) do
  Construct complete edge-weighted graph G(V', E') with
    w(e'_ij) = A(T_i ∪ T_j);
  Compute minimum weighted matching E'' ⊂ E' on G;
  For each edge e_ij ∈ E'' do
    V' ← V' − {v_i, v_j};
    V ← V ∪ {v_i, v_j};
    Construct new vertex v_k;
    E ← E ∪ {e_ik, e_jk};
    V' ← V' ∪ {v_k};
  EndFor
EndDo
Construct the tree from the vertices left in V'.
return T(V, E);
```

Proof: We now define the power savings W obtained from an activity-driven tree. From the definition of the cost function

$$\begin{aligned}\mathcal{A}(\mathcal{T}) &= \sum_{v_i \in V} \left[b_i t\left(U_i^T\right) + \sum_{j=1}^{u} q_{ij} a_{ij} \right] \\ &= \sum_{v_i \in V} \sum_{j=1}^{u} q_{ij} - \sum_{v_i \in V} \sum_{j=1} q_{ij}(1 - a_{ij}) \\ &\quad + \sum_{v_i \in V} b_i t\left(U_i^T\right).\end{aligned}$$

Let K denote the total power of the ungated clock tree. For every idle pattern, there can be at most two transitions. Hence

$$\mathcal{A}(\mathcal{T}) = K + \sum_{v_i \in V} \sum_{j=1} (q_{ij} - 2b_i)(1 - a_{ij}).$$

Now let $\kappa_{ij} = (q_{ij} - 2b_i)$. The power saving W of an activity-driven clock tree is as follows:

$$W = \sum_{v_i \in V} \sum_{j=1} \kappa_{ij}(1 - a_{ij}).$$

Furthermore, let $W(l)$ denote the amount of power savings obtained by gating level l of the tree. Now consider the simple case when $\kappa_{ij} = 1$, where we intend to maximize the number of idle periods. Let the function $\upsilon(U_i) = \sum_{j=1}^{u}(1 - a_{ij})$ denote the number of idle periods in an activity pattern U_i. By the definition of the $\vee$ function, the number of idle periods cannot increase when merging two subtrees: $\upsilon(U_i \vee U_j) \leq \max(\upsilon(U_i), \upsilon(U_j))$. Thus, $W(l) = \sum_{v_i \in V, \text{lev}(i)=l} \upsilon(U_i)$ and by the property stated above, $W(l) \leq W(l+1)/2$. Hence, $W(l) \leq W(h)/2^{h-l}$, where h is the height of the tree. Since $W = \sum_{l=1}^{\log n} W(l)$, then $W \leq 2W(h) - 1$. A perfect matching of the leaf level provides the lower bound on the total cost of the tree. Thus the recursive matching algorithm produces a solution that is at worst a two-approximation. Now let $\kappa_{ij} = \kappa - 2\gamma$ and then $W \leq 2(\kappa - 2\gamma)W(h)$, which is also a two-approximation. This approximation is tight, as we will show next. Consider a case where all the activity patterns are equal: any matching in this case will produce a tree with the desired cost. This completes the proof for Case 1.

For Case 2, consider the construction of an H-tree. The H-tree is constructed recursively by connecting the centers of four H-trees with another H-tree. Assuming that the modules are placed on a grid with grid spacing one, the total length for each leaf H-tree is three (since there are three wires on the H). At each subsequent level, the total wire length doubles. In general, the H-tree at level i will have a total of $3n/2^i$ wire length. Furthermore, since each H-tree spans two levels of the clock tree, there are $\log n/2$ H-tree levels. In the leaf level, there will be a total of $n/2$ H-trees for a total of $3n/2$ wire length. Of this wire length $2/3$ end up in the leaf level of the clock tree, which has n wire length. The total wire length in the remaining of the H-tree is $n/2 + 3n\sum_{i=1}^{\log n/2}(1/2^i) \leq 2n$. The total wire length in the leaf level and in H-tree is n and $3n$, respectively. This indicates that the recursive matching algorithm yields a three-approximation in this case. ■

It is known that the weighted matching problem has exact [11] and fast approximate polynomial time solutions (for a survey, see [1]). Given that a matching algorithm takes $O(T_m(|V|,|E|))$ time to complete, we can obtain the time complexity of the ClockAct algorithm that follows.

Theorem 2: Given a matching algorithm that can complete a matching in $O(T_m(|V|,|E|))$ time, the ClockAct algorithm constructs a binary activity-driven clock tree in $O(T_m(|V|,|E|)\log|V|)$ time.

Proof: The recursion cuts the problem size in half at each step. Let $T(|V|,|E|)$ denote the time taken by the ClockAct algorithm. Then we can produce a recursive formula for the time complexity: $T(|V|,|E|) = T_m(|V|,|E|) + T(|V|/2, |E|/4)$. In other words, we have $T(|V|,|E|) = O(T_m(|V|,|E|)\log|V|)$ time. ■

To finish this section, we propose algorithms that compute lower and upper bounds on the clock tree construction objective $\mathcal{A}(\mathcal{T})$. We later use these bounds as experimental evidence on the quality of our algorithms. We compute the lower and upper bounds by considering the construction of an optimal tree with a single time period. In this case, modules will either be constantly active or constantly idle. To minimize the power at each level of the clock tree, we match as many active modules as we can. To maximize the power at each level of the clock tree, we match as few modules as we can at each level. To obtain bounds on the full problem, we apply the algorithm proposed above to each bit of the n modules and sum the objectives values.

VI. ALGORITHMS FOR ACTIVITY-DRIVEN GATE INSERTION

In this section, we propose two exact algorithms. The first algorithm solves the ADMPGI problem, but produces solutions of arbitrary skew. The second algorithm uses the solution to the first to achieve solutions with zero skew.

A. Exact Algorithm for Gate Insertion

Given that a clock tree has been constructed, we must now determine the best locations in the clock tree for clock gates.

TABLE V
PSEUDOCODE FOR ALGORITHM **GATEINSERT**

```
Algorithm GateInsert(T(V,E))
Input: Clock Tree T(V,E).
Output: Clock gates inserted in T in array Soltn.
  For v_i ∈ V ordered by decreasing lev(i) do
    p ← Parent(v_i);
    For k = 1 to lev(i) do
      GP ← P_i^G(U_k^C, U_i^T) + P_i^T(U_i^T);
      noGP ← P_i^T(U_k^C);
      For each v_j ∈ Child(i) do
        GP ← GP + Power[j, lev(i)];
        noGP ← noGP + Power[j, k];
      EndFor
      Power[i,k] ← min(GP, noGP);
      Gate[i,k] ← (GP > noGP) ? "gate" : "nogate";
    EndFor
  EndFor

  Soltn[0] ← "gate";
  Pattern[0] ← 0;
  For v_i ∈ V in depth-first order do
    p ← Parent(v_i);
    Soltn[i] ← Gate[i, Pattern[p]];
    Pattern[i] ← (Soltn[i] = "gate") ? lev(i) : Pattern[p];
  EndFor
```

Our objective is to minimize the total power of the system, as defined in the previous section.

We propose an exact algorithm, called GateInsert, to solve this problem, based on a bottom-up traversal of the clock tree. The algorithm inserts clock gates on the clock tree edges, as close as possible to the parent nodes. The main idea of the algorithm is to keep track, as we go up the tree, of the cost of the two possibilities of adding or not adding a clock gate, which we call the *gate* and *no-gate* choices, respectively.

Consider the problem of measuring the power consumed by a clock gate at the root of a subtree, assuming that we know the solutions for the children nodes. We also know the desired activity pattern for that gate. However, we do not know the activity pattern of the input clock signal since it depends on the choice of gating above the root node of the subtree. We observe that the number of possible activity patterns is equal to the level of the root node of the subtree. Therefore, we can compute and save for later use the best solution for each possible input activity pattern. Similarly, for the no-gate choice, we can make the same computation. Finally, we compute the best solution for each activity pattern by choosing from each child the best from the gate and no-gate solutions. At the end of the traversal, the clock gate insertions can be made by following, in top-down fashion, the choices made at each node in the tree. Table V shows the pseudocode for algorithm GateInsert.

Theorem 3: Algorithm GateInsert finds a set of gates inserted in a tree $\mathcal{T}(V,E)$ such that $\mathcal{A}(\mathcal{T})$ is minimized.

Proof: Let $A_i(U_k)$, $A_{G_i}(U_k)$ and $A_{\mathrm{NG}_i}(U_k)$ denote, for a node v_i and input activity pattern U_k, the minimum weighted activity solution, the minimum weighted activity solution using a gate, and the minimum weighted activity solution with no gate, respectively. At a node, only two solutions exist: either the node uses a gate or it uses no gate. Therefore, $A_i(U_k) = \min\{A_{G_i}(U_k), A_{\mathrm{NG}_i}(U_k)\}$. Given a gate and no-gate choice and the input pattern U_k, we have

$$A_{G_i}(U_k) = P_i^G(U_i, U_k) + P_i^T(U_i) + \sum_{j=1}^{d_i} A_j(U_i) \quad (6)$$

$$A_{\mathrm{NG}_i}(U_k) = P_i^T(U_k) + \sum_{j=1}^{d_i} A_j(U_k) \quad (7)$$

where d_i is the number of children of node v_i. Note that in the case of a gated solution, the gate imposes the activity pattern for the subtree. In the case of the ungated solution, the activity pattern is the input pattern. Also note that these functions are of the general form of (5). We observe that the number of possible input patterns is lev$(i) \leq h$, corresponding to each of the nodes above v_i, where a gate may be placed, thus setting the input activity pattern. The first stage of the GateInsert algorithm computes all possible solutions for each vertex. Since the root node v_0 has only one possible input pattern, the value of $A_0 = \mathcal{A}(\mathcal{T})$ is the minimum weighted activity of the tree. The second stage of the algorithm follows the optimal solution choices in a top-down manner. Since the parent node's activity pattern is set, then the child's solution is also set. ■

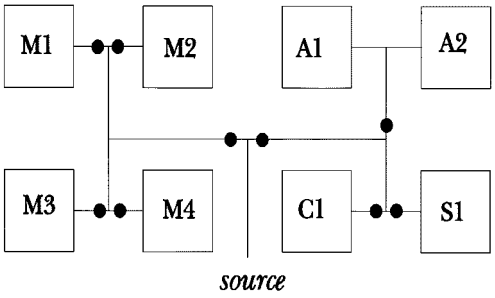

Fig. 7. Optimal solution for sample problem. This solution was obtained with the following parameters. Multiplier active power $P_A = 8$ and idle power $P_I = 2$, other modules active power $P_A = 2$ and idle power $P_I = 1$, the gate control signal power $b_i = 1$, the gate input power $P(\mathrm{in}) = 0.1$, the gate output power $P(\mathrm{out}) = 0.1$, and the tree activity weights are proportional to the wire length on the H-tree.

Fig. 7 shows the optimal solution for DE Example 2.

Theorem 4: Algorithm GateInsert finds a solution to the ADMPGI problem using $O(n \log n)$ time and space.

Proof: By Theorem 3, the algorithm GateInsert computes and saves lev(i) solutions for each vertex $v_i \in V$. All other operations in the algorithm amount to tree traversals which take $O(n)$ time. Therefore, the algorithm takes $O(\sum_{v_i \in V} \mathrm{lev}(i))$ time and space. Since lev$(i) = O(h)$, the time and space complexities becomes $O(nh)$. Also, since the tree is a complete binary tree, the algorithm takes $O(n \log n)$ time and space. ■

B. Exact Algorithm for Gate Insertion With Zero Skew

The GateInsert algorithm as described above does not take skew into account. In this paper, we consider the only effects of *buffer/gate skew* on the real skew. Skew is defined as the largest difference between the source to sink delays. Buffer/gate skew is defined as the largest difference between the number of gates or buffers between the source to sink paths. By making the buffer/gate skew zero and choosing the designs of the gates

and buffers carefully, the remaining skew is due to wiring delay and load capacitance effects, which can be minimized using the approaches proposed in [18].

Since the GateInsert algorithm does not consider the number of gates in the source to sink paths, it may produce solutions with large skew. To solve this problem, we propose the following modified algorithm, so-called zsGateInsert. The idea is to have the same number of buffers/gates in all source to sink paths. We assume in this solution that either a tree level can be gated or it cannot. This assumption is consistent with the objective of constructing zero-skew clock trees. Therefore, the algorithm has three choices at each node.

1) Insert a gate. This instance requires that all nodes at the same level must also have a gate or a buffer.
2) Insert a buffer. This occurs while a gate is not needed at the node, a gate has been inserted at another node of the same level.
3) Insert no gates and no buffers at any nodes in a given level of the tree.

We note that inserting a buffer does not change the activity patterns and only increases the power consumed by the tree. Hence, we propose the following algorithm modifications.

1) Select a combination of levels that are not allowed to have buffers or gates. For all vertices on the remaining levels, add a buffer.
2) Run GateInsert and add gates only in those levels where this is allowed. When a gate is added, it replaces a buffer.
3) Repeat step 1 for all possible combinations of levels; choose the best combination.

The above procedure is repeated for all possible combinations of levels. The following theorem shows the correctness and the complexity of this algorithm.

Theorem 5: Given a clock tree with $O(\log n)$ levels, the algorithm zsGateInsert finds a solution to the ADMPGI problem with zero gate/buffer skew. The algorithm needs $O(n^2 \log n)$ time and $O(n \log n)$ space.

Proof: The algorithm finds a minimum solution by virtue of the exhaustive search of possible solutions. The buffer/gate skew of the solution is zero by the construction of the problem: every vertex in a level will either have buffers and gates or neither. Therefore, all source to sink paths have the same number of active devices. Note that a level allows either gating or no-gating, which implies that there are two possibilities per level. For h levels, there are a total of 2^h possible solutions. Thus, the time complexity is $O(nh2^h)$. Again, since $h = O(\log n)$ for a complete binary tree, the algorithm takes $O(n^2 \log n)$ time. The space complexity of the algorithm is that of GateInsert. ■

VII. Experiments

Since no benchmarks exist that can be used in our test with activity-driven clock design, we used randomly generated bit patterns in our experiments. The bit patterns can be generated by one of the following methods.

1) For each module, randomly set k active periods out of a maximum of u time periods.

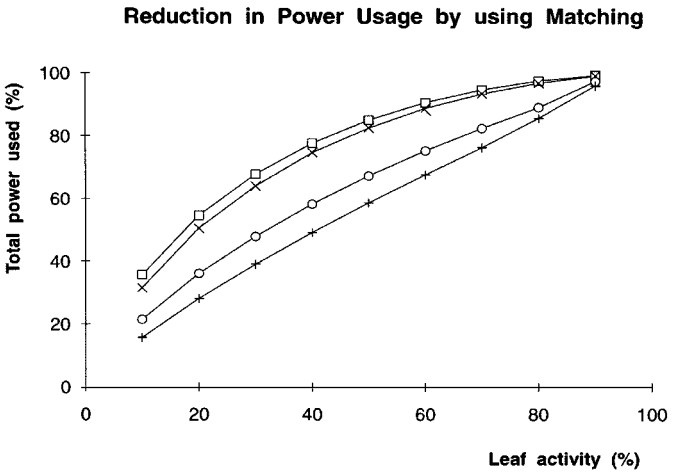

Fig. 8. Graph shows the percentage of ungated tree power consumed by the gated trees. Each point represents the average result over 25 such pattern sets. Experiment was carried out from 10% to 90% activity at the leaves. Curves show, from top to bottom, an average result over 100 randomly generated trees and three results obtained from trees generated using the greedy matching algorithm (where one, zero, and three duplicates of each leaf patterns are generated, respectively).

2) For each module, randomly select k integers in the range $[1, u-1]$. Construct intervals $\{[0, u_1], [u_1 + 1, u_2], \ldots, [u_k, u]\}$ such that $u_i < u_j$ for $i < j$. Alternatively, set the the time periods in each interval as either active or idle.
3) Duplicate d times a given set of activity patterns, thus making $n = n \times 2^d$.

We have observed that the quality of our results changes with the types of patterns and with the percentage of active time periods. To show the properties of our algorithms, we have two sets of results (where the H-tree structure was assumed).

1) We show the power consumption from ungated clock trees, randomly constructed trees with gates inserted using our algorithm, and trees constructed using our matching and gating algorithms. These results are shown in Fig. 8.
2) We compare average matching costs for different problem sizes using randomly constructed clock trees and clock trees constructed using the greedy matching algorithm. These results are compared to the lower and upper bounds computed using the algorithm proposed in Section V. Furthermore, the comparisons are made for different types of patterns. The results are shown in Fig. 9.

The results shown in Fig. 8 indicate that our choice of matching function is a good simplified clock tree construction objective and that the proposed tree construction and gate insertion algorithms are effective in reducing the dynamic power consumption. Furthermore, for design instances where power-driven clock tree construction is not practical, the gate insertion algorithm produces results of independent interest.

The results in Figs. 8 and 9 show that the matching algorithm reduces both the matching objective and the power consumption of gated clock trees. Also, it can be seen from these graphs that the results obtained from this algorithm are very close to optimal results which are indicated as the lower bound graphs

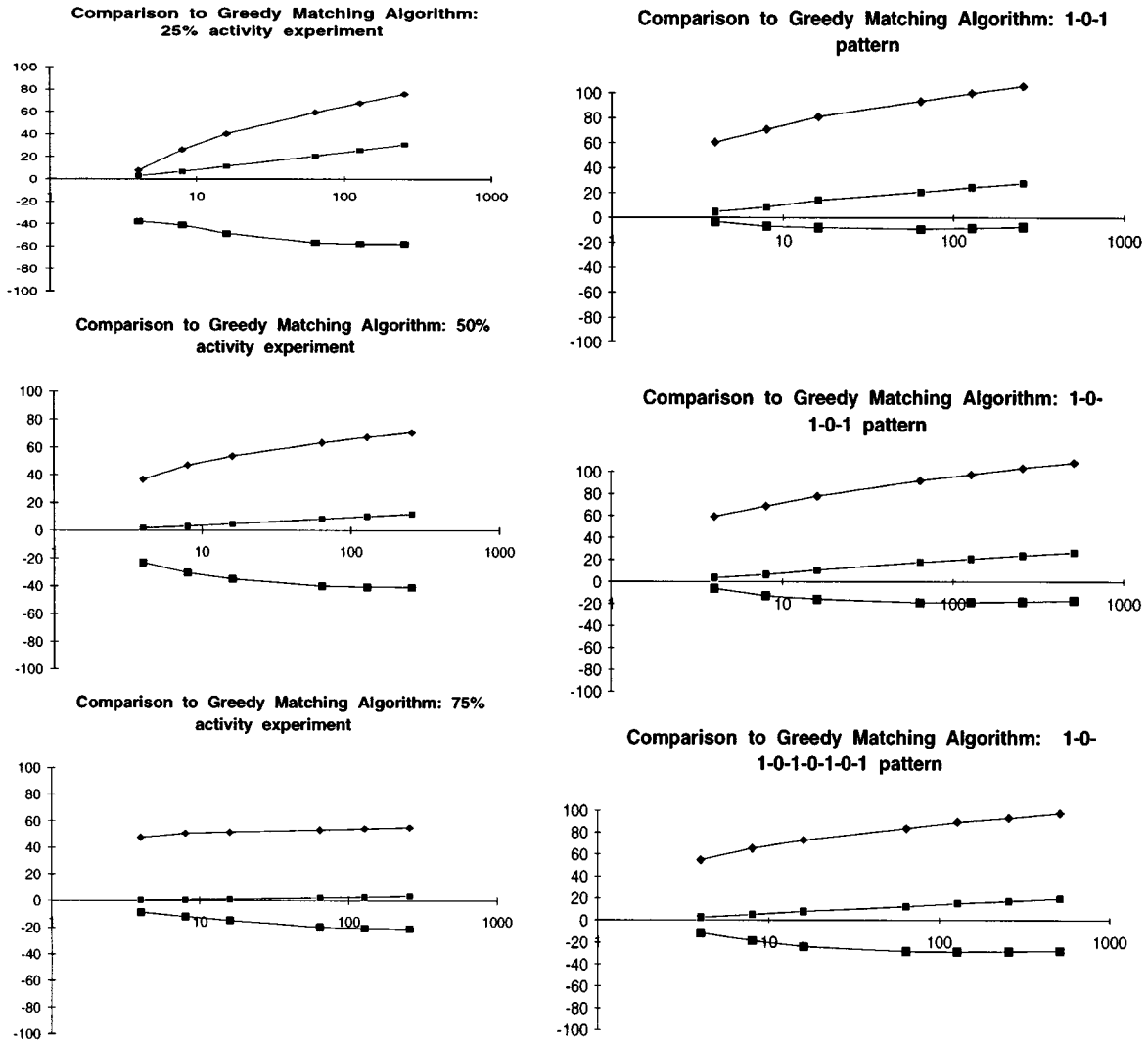

Fig. 9. Graphs show matching cost upper bound and lower bounds and average matching cost for randomly generated trees for various problem sizes. Cost values are normalized as percentages with respect to the cost values of the greedy matching tree solutions.

in Fig. 9. Our experiments show that the greedy matching algorithm produces trees with increased power savings.

VIII. Conclusion and Future Work

In this paper, we have proposed a novel approach for the construction of activity-driven clock trees with the objective of minimizing power consumption. We have developed algorithms that solve the problems of clock tree construction and gate insertion into the clock tree while minimizing power consumption and producing small clock skew.

For future work on activity-driven clock design, the following topics are of interest:

1) scheduling and allocation with low power objectives;
2) other means of obtaining activity patterns, perhaps during different abstraction levels of design such as at the behavioral level or the presynthesis level;
3) extension of our algorithms to handle probabilistic activity patterns;
4) combination of ADCTC with low-power layout design by, for example, extending placement algorithms (such as TimberWolf [19] and Gordian [16]) to deal with these interrelated issues.

References

[1] D. Avis, "A survey of heuristics for the weighted matching problem," *Networks*, vol. 13, pp. 475–493, 1983.
[2] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990, pp. 81–112.
[3] K. D. Boese and A. B. Kahng, "Zero-skew clock routing with minimum wirelength," in *Proc. Int. ASIC Conf. and Exhibit*, Sept. 1992, pp. 17–21.
[4] H. Bakoglu, J. T. Walker, and J. D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ULSI and WSI circuits," in *Proc. Int. Conf. Computer Design*, Oct. 1986, pp. 118–122.
[5] N.-C. Chou and C.-K. Cheng, "Wire length and delay minimization in general clock net routing," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 552–555.
[6] T. H. Chao, Y. C. Hsu, and J. M. Ho, "Zero skew clock net routing," in *Proc. Design Automation Conf.*, June 1992, pp. 518–523.
[7] T. H. Chao, Y. C. Hsu, J. M. Ho, K. D. Boese, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 799–814, Nov. 1992.
[8] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *Proc. Design Automation Conf.*, June 1993, pp. 612–616.
[9] ——, "Delay minimization for zero-skew routing," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 563–566.

[10] E. G. Friedman, "Clock distribution design in VLSI circuits—An overview," in *Proc. Int. Symp. Circuits and Systems*, May 1993, pp. 1475–1478.
[11] H. N. Gabow, "An efficient implementation of Edmonds' algorithm for maximum matching on graphs," *J. ACM*, vol. 23, pp. 221–234, 1973.
[12] D. S. Hochbaum, *Approximation Algorithms for NP-Hard Problems*. Boston, MA: PWS-Kent, 1997.
[13] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high-performance ICs," in *Proc. Design Automation Conf.*, June 1990, pp. 573–579.
[14] A. Kahng, J. Cong, and G. Robins, "High-performance clock routing based on recursive geometric matching," in *Proc. Design Automation Conf.*, June 1991, pp. 322–327.
[15] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*. Norwell, MA: Kluwer, 1995.
[16] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 356–365, Mar. 1991.
[17] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, "Skew and delay optimization for reliable buffered clock trees," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 556–562.
[18] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable nonzero skew clock tree using wire width optimization," in *Proc. Design Automation Conf.*, June 1993, pp. 165–170.
[19] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," *IEEE J. Solid-State Circuits*, vol. 20, pp. 510–522, Apr. 1985.
[20] N. A. Sherwani and B. Wu, "Effective buffer insertion of clock tree for high speed VLSI circuits," *Microelectron. J.*, vol. 23, pp. 291–300, July 1992.
[21] R.-S. Tsay, "Exact zero skew," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1991, pp. 336–339.

Amir H. Farrahi (S'94–M'97–SM'00) received the B.S. degree from the Sharif Institute of Technology, Tehran, Iran, in 1988, the M.S. degree from the Illinois Institute of Technology, Chicago, in 1992, and the Ph.D. degree from in electrical engineering from Northwestern University, Evanston, IL, in 1997, all in electrical engineering.

From 1988 to 1990, he was with the Software and Hardware Support Team at Negareh Computer Corporation, Tehran, Iran. During the summer of 1993 and 1995, he was a Member of the Technical Staff at Vista Technologies Inc., Schaumburg, IL, and Cadence Design Systems, Inc., San Jose, CA, respectively. Upon receiving his Ph.D., he joined the IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1997. His current research interests include the design and analysis of algorithms, design automation in VLSI, graph algorithms, and computational complexity.

Dr. Farrahi is the recipient of the 1994 Design Automation Conference Graduate Scholarship Award, the Best Dissertation Award from the Electrical and Computer Engineering Department, Northwestern University, in 1997, and an IBM Invention Achievement Award. He has served on the Organizing and Technical Program Committees of a number of conferences and symposia in the field of electronic design automation, including the IEEE International Conference on Computer-Aided Design, the Great Lakes Symposium on VLSI, the International Workshop on Interconnect Prediction, and the Southwest Symposium on Mixed Signal Design.

Chunhong Chen (M'99) received the B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China, and the Ph.D. degree in electrical engineering from Fudan University, Shanghai, China.

From 1986 to 1996, he was with the Zhejiang University of Technology, Hangzhou, China, as an Assistant and then Associate Professor. From 1997 to 1998, he was a Research Associate at the Hong Kong University of Science and Technology, Hong Kong. Since 1999, he has been a Postdoctoral Fellow first at Northwestern University, Evanston, IL, and then at the University of California, Los Angeles. His current research interests include physical layout, logic synthesis, timing analysis , and power optimization for integrated circuits.

Ankur Srivastava (S'98–M'98) received the B.Tech. degree from the Indian Institute of Technology, Delhi, India, and the M.S. degree from Northwestern University, Evanston, IL. He is currently working towards the Ph.D. degree at the University of California, Los Angeles.

His current research interests include low-power design and computer-aided design.

Gustavo Téllez received the B.S. and M.S. degrees in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1985 and the Ph.D. degree in computer science from Northwestern University, Evanston, IL, in 1996.

He joined IBM EDA pein 1986 in East Fishkill, NY. He has been with the IBM Corporation, Essex Junction, VT since 1996, where he is currently a Senior Engineer at the Development Laboratory. He holds five patents and has authored or coauthored numerous technical publications. His previous research interests are in the areas of timing-driven physical design, custom layout automation, custom layout compaction and optimization, layout technology migration, and design for yield. His current interests include methodologies and techniques for ASIC design time and manufacturing cost reduction.

Majid Sarrafzadeh (S'82–M'82–SM'92–F'96) received the B.S., M.S. and Ph.D. degrees in electrical and computer engineering from the Univerisity of Illinois at Urbana-Champaign in 1982, 1984, and 1987, respectively.

He joined Northwestern University as an Assistant Professor in 1987. In 2000, he joined the Computer Science Department at University of California at Los Angeles. Dr. Sarrafzadeh He has collaborated with many industries in the past ten years, including IBM and Motorola and many computer-aided design industries and is a technical consultant to Monterey Design Systems. He has authored or coauthored approximately 200 papers, an invited chapter in the *Encyclopedia of Electrical and Electronics Engineering* in the area of VLSI Circuit Layout, and the book *An Introduction to VLSI Physical Design* (New York: McGraw Hill, 1996) and has coedited the book *Algorithm Aspects of VLSI Layout* (Singapore: World Scientific, 1994). His current research interests lie in the area of embedded and reconfigurable computing, VLSI computer-aided design, and design and analysis of algorithms.

Dr. Sarrafzadeh received a National Science Foundation Engineering Initiation Award, two Distinguished Paper Awards at the International Conference on Computer-Aided Design (ICCAD), and the Best Paper Award at the Design Automation Conference (DAC) for his work in the area of Physical Design. He has served on the technical program committee of numerous conferences in the area of VLSI design and computer-aided design, including ICCAD, DAC, EDAC, and the International Symposium on Circuits and Systems. He has served as committee chairs of a number of these conferences, including International Conference on CAD and International Symposium on Physical Design. He was the general chair of the 1998 International Symposium on Physical Design. He is on the editorial board of the *VLSI Design Journal*, Coeditor-in-Chief of the *International Journal of High-Speed Electronics*, an Associate Editor of the *ACM Transaction on Design Automation*, and an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.