Activity-Sensitive Clock Tree Construction for Low Power *

Chunhong Chen
Dept. of Electrical & Computer Eng.
University of Windsor
Windsor, Ontario, Canada
Tel: 1-519-253-3000
cchen@uwindsor.ca

Changjun Kang
Dept. of Electrical and Computer Eng.
University of Windsor
Windsor, Ontario, Canada
Tel: 1-519-253-3000
kang@uwindsor.ca

Majid Sarrafzadeh
Computer Science Department
University of California at Los Angeles
Los Angeles, CA 90095-1596 USA
Tel: 1-310-794-4303
majid@cs.ucla.edu

ABSTRACT
This paper presents an activity-sensitive clock tree construction technique for low power design of VLSI clock networks. We introduce the term of node difference based on module activity information, and show its relationship with the power consumption. A binary clock tree is built using the node difference between different modules to optimize the power consumption due to the interconnections (i.e., clock gating signals and clock edges). We also develop a method to determine gating signals with minimum number of transitions. After the clock tree is constructed, the gating signals are optimized for further power savings.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Styles – *VLSI.*

General Terms
Algorithms.

Keywords
Clock tree, low power, clock gating, activity pattern.

1. INTRODUCTION
In synchronous digital systems, the logic operation is synchronized by clock signal. The clock is generated by a clock generator and distributed across the chip by a clock tree. In system-on-chips, the clock accounts for a large portion of power consumption (20%-50%). The fact that not all of modules are involved in some operations leads to clock gating technique. By controlling the clock fed into modules, the clock gating signals (or control signals) can prevent some unnecessary switching with both clock tree and modules and, hence, save the power consumption.

* This work was supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) Grant #249499-02.

ISLPED'02, August 12-14, 2002, Monterey, California, USA.

In general, the power consumption of a clock network is contributed by three factors: modules, clock edges and control signals. Previous work mainly focused on zero-skew clock trees [1-3]. In [4], the authors proposed the activity-driven clock tree design for low power. A binary clock tree is constructed by combining two modules or internal nodes with similar activity pattern so that the total active periods of the clock are reduced. As a result, the power consumption of the clock tree decreases. However, the effect of control signals was not addressed. Given floor-planning information and activation frequencies of modules, if highly active nodes are merged into the clock tree later than less active nodes, then the overall activity in the tree can be reduced [5].

In this paper, we first introduce the concept of *node difference* based on activity patterns of modules and show how it is related to the power consumption of clock network. Then we propose an algorithm to construct a low-power clock tree. Once the clock tree is available, we perform further power optimization by reducing the penalty of control signals.

2. BACKGROUND
Let $G = \{V, E\}$ denote the clock tree, where $V = \{v_i \mid i = 1, 2, \ldots, m_v\}$ is the set of nodes, and $E = \{e_j \mid j = 1, 2, \ldots, m_v - 1\}$ is the set of clock edges corresponding to each node (except the root node). We use $S = \{v_k \mid k = 1, 2, \ldots, m_s\}$ (where $m_s < m_v$) to denote the set of modules (namely, the leaf nodes). The rest ($m_v - m_s$) nodes are called internal nodes. The root is said to be at level 0. Node v_i is said to be at level n_i if there are n_i edges on the path from v_i to the root. An example of clock tree is shown in Fig. 1, where the tree is a binary tree for which each parent node has no more than two children. The control signal gates a parent node to reach its child node.

For each module, we can obtain the activity information from the behavioral level description [1]. An activity pattern is a set of '1' and '0', where '1' represents an active period while '0' represents an idle period. Each binary bit in the activity pattern stands for a clock period. The number of bits for which the modules have different logic values is referred to as *node difference*. For the binary clock tree, we notice that a parent node must be active whenever its left or right child is active. This means that the activity pattern of the parent node is formed by OR-ing the activity patterns of its left and right children. In contrast, the activity pattern of a child node is obtained by AND-ing the activity patterns of its parent node and control signal. A general

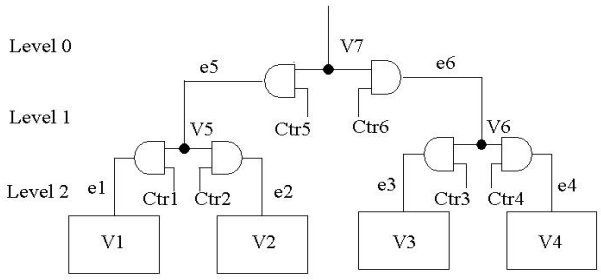

Figure 1. A binary gated clock tree.

Table I. Determining control signals from nodes' activity

Parent node	Child node	Control signal
1	1	1
1	0	0
0	0	unchanged
0	1	(impossible)

rule for determining the control signals from the activity patterns is given in Table I, where the entries "1" (or "0") represent "active" (or "idle") for nodes, and "high" (or "low") for control signals.

3. NODE DIFFERENCE AND POWER CONSUMPTION

3.1 Node-Difference Based Power Analysis

In this section, we show that node difference plays an important role in low-power clock tree construction. For an extreme case where two nodes with the same activity pattern are combined, the activity pattern of parent node is the same as its children's activity patterns. Thus, no transitions occur with the control signals. The number of active periods for the parent node is the same as that for each child. We refer to this extreme case as the *ideal point*. If we change one child's distribution of activity periods while keeping the number of its active periods unchanged, the node difference between two children is increased by exactly the increased number of active periods of the parent node. This is given by the following equation:

$$A_{inc} = L_d = (A_p - A_{ch1}) + (A_p - A_{ch2}) \quad (1)$$

where A_{inc} is the increased number of active periods, L_d is the node difference between child 1 and child 2, and A_p, A_{ch1} and A_{ch2} are the numbers of active periods for the parent, child 1 and child 2 nodes, respectively. Since reducing the A_{inc} can lead to the saved power of the clock edges, the nodes with smaller node difference should be combined with higher priority during the clock tree construction.

An intuitive idea is that minimizing the node difference of neighboring nodes can result in the reduced number of control signals' transitions required. Since the state of a control signal depends on its previous state when both parent and children are idle (refer to Table I), it would be impossible to formulate the relationship between the node difference and transitions of control signals by combinational logic. Therefore, we conducted some experiments to obtain the correlation between the total transitions of control signals and total node difference of all modules (i.e., nodes). Fig. 2 shows the experimental result with 20 nodes and 20 clock periods. In this experiment, we keep the activity densities for all nodes fixed, and the nodes are paired sequentially (i.e., node 1 is paired with node 2, and node 3 is paired with node 4, etc.). The total transition is the sum of transitions of control signals for all nodes. The total node difference is the sum of node differences for all node pairs. We change the total logic distance by shuffling the activity patterns of nodes without changing the activity density, and obtain the corresponding total transitions. It can be seen from Fig. 2 that the number of total transitions of control signals is proportional to the total node difference.

In order to take into account the power consumption for both clock edges and control signals, we define a *merging power*, for each pair of modules to be combined, as a weighted sum of logic distance (i.e., L_d) and the number of transitions of control signals (denoted by T_c):

$$P_{meg} = W_{clk} * L_d + W_{ctr} * T_c \quad (2)$$

where W_{clk} and W_{ctr} are the weights for clock edges and for control signals, representing the power consumption contributed by each active period of clock edges and by each transition of control signals, respectively. These weights are proportional to the wiring capacitances of the associated clock edges or control logic.

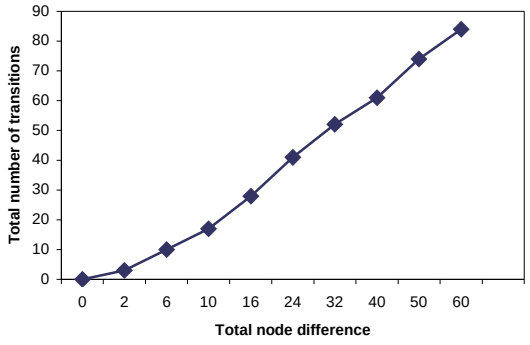

Figure 2. The correlation between total transitions and total node difference.

3.2 Power Consumption of Binary Clock Trees

Consider a typical construction process of the binary clock tree. The top level of the tree is the clock source. Leaves or modules are at the bottom level. Each node has no more than two children nodes. The number of nodes at each level is not necessarily power of 2, depending on the total number of modules. If, at a specific

level, the number of nodes is odd, there must be one node that doesn't combine with any other node.

To estimate the physical wiring information, we take an H tree for example. Fig. 3 shows the H tree with eight modules that are placed in uniform grids. Assuming the clock wiring length at the bottom level is l_{clk}, the clock wiring length at level *n* is given by

$$L_{clk(0)} = 0 \tag{3}$$

$$L_{clk(n)} = 2^{fl[(N-1-n)/2]} l_{clk} \tag{4}$$

where *N* is the number of levels ($0 \le n < N$), and $fl[x]$ is the floor value of positive number *x*. Wire length of the source is 0. Eq. (4) means that the wire length of clock edges doubles at every other level upward. Assuming all control signals are generated from the central [5]. The routing of the control signals is parallel to the clock routing.

The wire length of each control signal at level *n* is given by

$$L_{ctr(n)} = 0 \qquad for\ n = 0,1 \tag{5}$$

$$L_{ctr(n)} = l_{clk} \sum_{k=1}^{n-1} 2^{fl[(N-1-k)/2]} \quad for\ n > 1 \tag{6}$$

Since there is no control signal for the source, $L_{ctr(0)}$ is 0. If we assume the clock gate is placed as close as possible to the upper level, then $L_{ctr(1)}$ can also be considered as 0. As to other levels, the wire length of control signals is the sum of the wire lengths of upper level clock edge all ways up to the root. Therefore, the power consumed by clock edge *i* at level *n* can be expressed as:

$$P_{clk(i)} = k_{clk} A_i L_{clk(n)} = W_{clk(n)} A_i \tag{7}$$

where k_{clk} is a constant, $W_{clk(n)} = k_{clk} L_{clk(n)}$, and A_i is the number of active periods of the node connected to clock edge *i*. Similarly, the power consumed by control signal *j* at level *n* is given by

$$P_{ctr(j)} = k_{ctr} T_j L_{ctr(n)} = W_{ctr(n)} T_j \tag{8}$$

where k_{ctr} is a constant, $W_{ctr(n)} = k_{ctr} L_{ctr(n)}$, and T_j is the number of transitions of control signal *j*. The total power consumption, P, of the clock network consists of the contributions by modules (P_m) and by interconnections, i.e.,

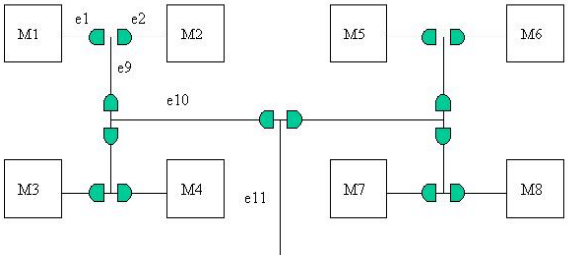

Figure 3. The topology of H tree.

$$P = \sum P_m + \sum P_{clk(i)} + \sum P_{ctr(j)} \tag{9}$$

4. ALGORITHMS

Based on the above discussions, we develop a so-called Merging Algorithm (MA) for the low-power clock tree construction. The algorithm consists of two parts. Given the total number of clock periods, the number of modules and their activity patterns, we first construct the binary clock tree based on Eq. (6) in a bottom-up manner. This is done level by level toward the root node of the tree. Then we perform an optimization step by looking at the tradeoff between the power penalty from control signals and the power savings from gated clock edges.

4.1 Clock Tree Construction

The basic idea for the clock tree construction is to combine every two nodes with smallest value of P_{meg} from leaf nodes towards the root node. All node pairs at one level are combined to obtain their parent nodes at the next level. If the number of nodes at a specific level is odd, then one node at this level will combine with a dummy node that has no active periods and no transition on its control signal. Each parent's activity pattern is obtained by OR-ing the activity patterns of the children. The control signals of children are determined by the method described in Section 2.

From Eq. (2), the *merging power* for node i and j at level n is written as:

$$P_{meg(i,j)} = W_{clk(n)} L_d + W_{ctr(n)} (T_i + T_j) \tag{10}$$

where T_i (T_j) is the number of transitions of the control signal for node *i* (node *j*). If there are *M* nodes at a given level, the number of possible *merging powers* values is $M(M-1)/2$. We sort all these values in ascending order. The two nodes with smallest value are paired to form their parent node, and then are deleted from the list. This process repeats until all nodes are paired. If the number of nodes is odd, the last node pairs with a dummy node. All parent nodes so obtained at the upper level can be used to determine the control signals of their children. The depth of the tree is $\lceil \log m_s \rceil$, where m_s is the number of modules.

4.2 Local Ungating

After the clock tree is constructed, every node (except the root) has an associated control signal. The control signals can reduce the power consumption due to clock edges and/or modules. However, the transition of control signals consumes additional power, which may offset the power savings. This requires more attention especially at some levels close to the leaf nodes, where the weights of control signals are relatively high (see Eq. (6)). The control signals' power penalty caused by their transitions can be reduced by incrementally changing some of their periods from '0' to '1'. This operation is called *local ungating*, which is acceptable if it can result in the overall power reduction.

5. EXPERIMENTAL RESULTS

We created the *CDFG* (Control Data Flow Graph) files from the benchmarks described in VHDL by using *CDFG Tool, version*

1.0. [6]. The ASAP (i.e., *As Soon As Possible*) scheduling was done based on the CDFG files. After the scheduling, we obtained the number of periods and minimum resource required, i.e., modules that include adder, subtractor, multiplier and multiplexer etc. The merging algorithm is applied on the activity patterns followed by local ungating. We compared the merging algorithm with conventional clock construction that does not account for node difference. The results are shown in Table II, where the power savings and wire-length are relative to that from a standard clock with gating. The merging algorithm results in fewer active periods on clock tree and fewer transitions of control signals, reducing power consumption of the clock tree. By combining the nodes with smallest node difference, it is much likely to produce the control signals that are always high at most periods. Thus, the ungating reduces the number of control signals and total wire length as well. As can be seen from Table II, there is also a significant decrease in control signals' wire length.

Table II. Performance of the proposed algorithm on benchmarks

Bench-mark	# Periods	# Modules	Power savings	Wire-length red.
iir7	13	19	13.5%	50.9%
ellipf	14	8	21.2%	30.9%
diffeq	5	8	23.1%	57.1%
parallel	9	17	24.8%	73.9%
nc	12	26	28.9%	55.0%

6. CONCLUSIONS

In this paper we have dealt with the activity-sensitive clock tree problem. We have presented the method of determining the control signals such that their transitions are reduced. By using modules' activity information, the clock tree construction algorithm and local ungating technique have been introduced for power savings. It has been shown that node difference plays an important role in low-power clock tree design.

7. REFERENCES

[1] Tsay R. S. An exact zero-skew clock routing algorithm. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 2 (February 1993), 242-249.

[2] Chao T. H., Hsu Y. C., Ho J. M., Boese K. D., and Kahng A. B. Zero skew routing with minimum wirelength. IEEE Trans. on Circuits and Systems, vol. 39, no. 11 (1992), 799-814.

[3] Edahiro M. Delay minimization for zero-skew routing. in Proceedings of ICCAD (Nov. 1993), 563-566.

[4] Farrahi A. H., Chen C. H., Sarrafzadeh M., and Tellez G. Activity-driven clock design. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 6 (June 2001), 706-714.

[5] Oh J. and Pedram M. Gated clock routing for low-power microprocessor design. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 6 (June 2001), 715-722.

[6] http://poppy.snu.ac.kr/~shlee/class/icda2001/CDFGTool.pdf