Clock Skew Minimization During FPGA Placement *

Kai Zhu and D.F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

Abstract

Unlike traditional ASIC technologies, the geometrical structures of clock trees in an FPGA are usually fixed and cannot be changed for different circuit designs. Moreover, the clock pins are connected to the clock trees via programmable switches. As a result, the load capacitances of a clock tree may be changed, depending on the utilization and distribution of logic modules in an FPGA. It is possible to minimize clock skew by distributing the load capacitances, or equivalently the logic modules used by the circuit design, carefully according to the circuit design. In this paper we present an algorithm for selecting logic modules used for circuit placement such that the clock skew is minimized. The algorithm can be applied to a variety of clock tree architectures, including those used in major commercial FPGAs. Furthermore, the algorithm can be extended to handle buffered clock trees and multi-phase clock trees. Experimental results show that the algorithm can reduce clock skews significantly as compared with the traditional placement algorithms which do not consider clock skew minimization.

1 Introduction

Clock skew is defined as the maximum difference of the delays from clock source to clock pins of latches in a clock tree. Since clock skew may affect system performance and even cause system failure, it is important to reduce clock skew for achieving high performance. In Field-Programmable Gate Arrays (FPGAs), the routing resources are fixed after chip fabrication. The population and distribution of programmable switches are usually restricted in order to reduce interconnection delays. The routing resources in FPGAs, therefore, lack the flexibility required by the clock tree construction algorithms (e.g., [11]) for traditional ASIC technologies to achieve zero skew. As a compromise, the geometrical structures of clock trees in many commercial FPGAs [1, 2, 9] are fixed and hence cannot be changed for different circuit designs.

Usually, there are several clock trees in an FPGA chip. Clock pins of latches in logic modules of an FPGA are connected to the designated clock trees via programmable switches. Figure 1 illustrates an architecture of connections of clock pin (CLK) in a logic module to clock trees in an FPGA. There are two clock signals: CLK1 and CLK2. Programmable switches are shown in circles. The clock pin of logic module can be connected to any of these two clock signals, as well as to the normal routing tracks. There are several reasons for such architecture of clock pin connection. One is multi-phase clocks. The total number and distribution of logic modules connected to a particular clock signal network in multi-phase clocks depend on circuit design and cannot be predetermined before FPGA fabrication. We thus need to allow each individual logic module to have the freedom of selecting clock signal to connect to [1, 2, 3, 4, 9]. Another reason is that circuit designer may occasionally want to drive clock pin by the outputs of logic modules for local asynchronous clocking [1, 2]. This requires that clock pins can be connected to normal routing tracks as well as clock trees. Furthermore, clock trees may also be used for distributing other global signals, such as reset and enable, rather than clocks [1, 2, 9]. Due to such clock pin connection architecture, the load capacitance seen at a clock pin of a logic module by a particular clock tree can be changed in different circuit designs, depending on whether the clock pin of the logic module is connected to the clock tree. The load capacitances of a clock tree therefore depend on the utilization and distribution of logic modules in the circuit design.

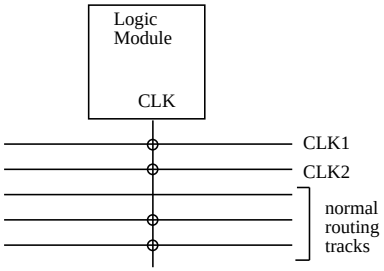

Figure 1: Connection architecture of clock pin to clock networks.

For a given circuit design, the total number of clock pins to be connected to a particular clock tree in an FPGA is available prior to circuit placement. The distribution of clock pins connected to a clock tree, which is determined during placement, will determine the load capacitance of the clock trees and is an important factor affecting clock skew. In order to reduce clock skew of a clock tree, it is necessary to distribute

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459.

clock pins connected to the clock tree carefully during placement to balance the load capacitances in the clock tree.

Traditionally, the objectives of placement algorithms are minimizing areas, total wire lengths, or satisfying timing requirements, but not minimizing clock skew. Therefore, traditional placement algorithms are not readily applicable to minimizing clock skew in FPGAs. Thus, it is necessary to develop new placement algorithms for FPGAs with primary objective of minimizing clock skew. In this paper, we present an algorithm for selecting logic modules in FPGA which will be the only logic modules used for circuit placement such that the clock skew is minimized. The selection of logic modules is dependent on the circuit design as well as the clock trees. The Elmore distributed RC delay model is used to compute the delays in clock tree. The algorithm can be either used as a preprocessing step before placement, or combined with min-cut placement algorithm for a special class of clock trees to minimize clock skew and wire length simultaneously.

The remainder of this paper is organized as follows. First, we describe the clock trees in several commercial FPGAs. In addition, We also propose a class of clock tree architectures suitable for minimizing clock skew and wire length simultaneously during placement. Next, we describe the algorithm for selecting logic modules for minimum clock skew placement. The algorithm can be extended to buffered clock trees and multi-phase clock trees. Finally, we present experimental results and the comparisons with the traditional placement algorithms on several industrial circuits.

2 Clock Tree Architecture

In this section, we describe several clock tree architectures. We assume that the logic modules in an FPGA are identical and are laid out in regular structure defined by a rectangular grid. The symmetrical array [4, 8] and row-based architectures [3, 7, 9] used in popular commercial FPGAs belong to such regular structures.

First, we describe two clock tree architectures used in commercial FPGAs. See Figure 2. The *comb* clock tree architecture (Figure 2(a)) is used in row-based FPGAs [1, 2]. Clock signal is passed to the *spine* on one side of the chip. Inside each channel, there is a special track, called *branch*, connected to the spine. The clock pins in logic modules can be connected to the branches in the adjacent channel via switches. The *dual comb* clock tree architecture (Figure 2(b)) is used in symmetrical-array based FPGAs [4] [1]. There is a vertical spine in a vertical channel of the chip to distribute clock signal to the branches in horizontal channels. Unlike comb architecture, the spine in general intersects with branches, instead of being connected to the endpoints of the branches. Note that a comb

[1] The clock trees in [4] are constructed "on-line" using normal routing resources. The constructed clock trees, however, always remain the dual comb structure as shown in Figure 2(b).

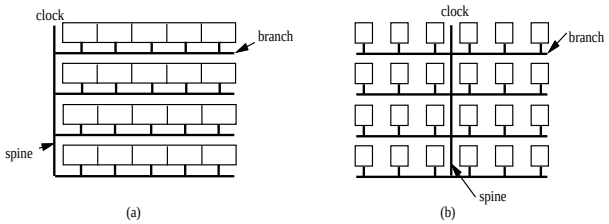

Figure 2: (a) Comb clock tree. (b) Dual comb clock tree.

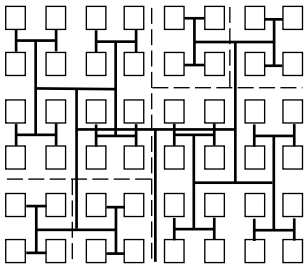

Figure 3: An 8×6 array of modules and a slicing binary clock tree.

clock tree is a binary tree and a dual comb clock tree is a 3-ary tree.

We now propose a class of clock trees which is designed for minimizing clock skew as well as wire length simultaneously. We denote the subtree rooted at an internal node u of a clock tree T by T_u. A logic module is *in* T_u if the clock pin of the logic module can be connected to a leaf node of T_u. The proposed clock trees, called *slicing binary clock trees*, have the following properties. The clock trees are binary trees. The root node of the clock tree thus has two children subtrees. All the logic modules in one subtree of the root node are separated by a straight line from all the logic modules in the other subtree of the root node. The root node of the clock tree thus represents a cut line of the entire circuit. Similarly, every internal node u represents a cut line on all the logic modules in the subtree T_u. The clock tree therefore defines a hierarchy of cut lines on the entire FPGA chip. Let W and H be the width and height of a FPGA chip, respectively, in terms of the number of logic modules in horizontal and vertical dimensions. As an example, Figure 3 shows an 8×6 ($W \times H$) FPGA. The clock tree is constructed from H-trees. The first few levels of cut lines are shown in dotted lines.

3 Selection of Logic Modules

3.1 Preliminary

A circuit design consists of a set of *circuit modules* interconnected by nets. We assume that a circuit module contains only one clock pin and can be realized by one logic module. For a circuit design with N circuit modules, we want to select N logic modules in the FPGA chip such that placing the N circuit modules on the selected logic modules will minimize clock skew. The

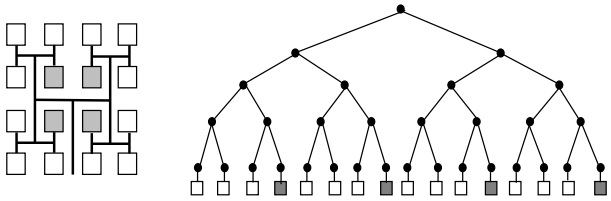

Figure 4: Selection of logic modules.

selected N logic modules can be viewed as an independent chip itself, and we can then use traditional placement algorithms to place the circuit modules on the selected N logic modules. As an illustration, Figure 4 shows a 4×4 FPGA with a slicing clock tree. It also shows the binary tree representation of the clock tree. The shaded leaves correspond to the selected modules. Note that at every node, the numbers of circuit modules allocated to each of subtrees are known. Min-cut placement algorithm (e.g. [5]) can then be used to assign circuit modules for wiring optimization. Given a clock tree T and N circuit modules, we define a *min-skew module selection* as the selection of N leaf nodes of T such that placing N circuit modules on the logic modules corresponding to the selected N leaf nodes results in minimum clock skew of T.

Before describing the algorithm for computing min-skew module selection, we briefly review the Elmore delay model [6, 10, 12] used in computing the delays in clock tree. To use Elmore delay model, the clock tree is modeled by a RC tree in which each edge is associated with a resistance and each node is associated with a capacitance. We denote the edge between node v in a tree and its parent node by e_v. For a node v in the RC tree, let r_v be the resistance of edge e_v and C_v be the total capacitance of the subtree T_v, including the node capacitance at v. The Elmore delay from a node u of the clock tree to a descendant node v of u, denoted by $d(u, v)$, is

$$d(u, v) = \sum_{u' \in path(u,v)} r_{u'} C_{u'}, \qquad (1)$$

where $path(u, v)$ is the set of nodes along the unique path from u to v, excluding u. Note that by replacing u with the root node of clock tree and v with a leaf node, we can compute the delay from the root node to the clock pin connected to the leaf node, if any. By computing the delays from root node to all the clock pins, we can compute the clock skew. The clock skew of a given clock tree T can be computed efficiently in linear time $O(|V|)$, where V is the set of nodes in T [10, 11].

3.2 Min-Skew Module Selection

To compute min-skew module selection, we associate nodes of the RC tree modeling a clock tree T with attributes. The *node attribute* of a node u in the RC tree is a set of ordered quadruples of the form (n_u, C_u, d_u, S_u), where C_u is the total capacitance of the subtree T_u with n_u circuit modules allocated to T_u,

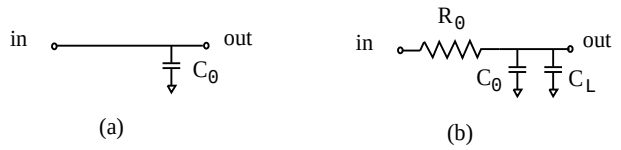

Figure 5: Equivalent circuits of leaf nodes when the clock pin (a) disconnected or (b) connected to clock tree.

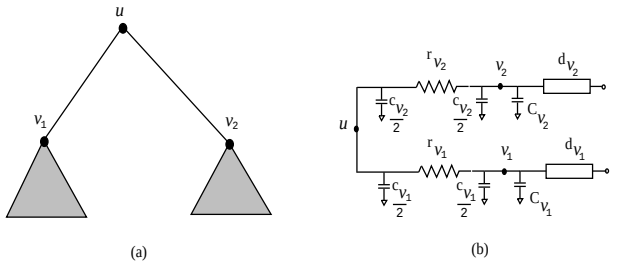

Figure 6: (a) A subtree T_u. (b) Equivalent circuit of T_u.

d_u is the maximum delay from u to clock pins in T_u, and S_u is the clock skew from u to clock pins in T_u. The basic idea of the algorithm is to compute node attributes bottom-up on the RC tree using dynamic programming, and then determine the n_u's using backtracking to obtain a min-skew module selection. Now we describe the dynamic programming for computing the node attributes.

First, we consider node attributes of leaf nodes. Let R_0 and C_0 be the on-resistance and capacitance of a programmable switch, respectively. There are two possible states for a leaf node u which can be connected to a clock pin of a logic module via a programmable switch. If a leaf node of the clock tree is disconnected from clock pin, i.e. $n_u = 0$, the leaf node is modeled by an equivalent circuit with only a load capacitance C_0. (See Figure 5(a).) The quadruple corresponding to this state is $(0, C_0, 0, 0)$. If the leaf node is connected to the clock pin, i.e. $n_u = 1$, the leaf node is modeled by an equivalent circuit shown in Figure 5(b), where C_L is the increase of load capacitance to a leaf node when a clock pin is connected to the leaf node. Note that the delay in the RC circuit of Figure 5(b) is $d_u = R_0(C_0 + C_L)$. The corresponding quadruple is $(1, C_0 + C_L, R_0(C_0 + C_L), 0)$. The node attribute for a leaf node in the clock tree thus contains two quadruples: $\{(0, C_0, 0, 0), (1, C_0 + C_L, R_0(C_0 + C_L), 0)\}$.

Next, we consider node attributes of internal nodes. Consider an internal node u with two subtrees T_{v_1} and T_{v_2}. (See Figure 6(a).) In the bottom-up process, the node attributes of nodes v_1 and v_2 are available before we compute the the node attribute of u. The node attribute of u can be computed from the node attributes of v_1 and v_2 and other known data. Let $(n_{v_1}, C_{v_1}, d_{v_1}, S_{v_1})$ and $(n_{v_2}, C_{v_2}, d_{v_2}, S_{v_2})$ be two quadruples of node attributes of v_1 and v_2, respectively. The subtree T_{v_1} is modeled by an equivalent

circuit of capacitance C_{v_1} and delay element d_{v_1}. Note that for a node v in the clock tree, the capacitance C_v of T_v with n_v circuit modules connected to leaf nodes of T_v can be computed by $C_v = \hat{C}_v + n_v C_L$, where $\hat{C}_v$ is the capacitance of T_v when no circuit module is allocated to T_v. Thus $C_{v_1} = \hat{C}_{v_1} + n_{v_1} C_L$. Similarly we model subtree T_{v_2}. The routing segment e_{v_1} is modeled by a π-model equivalent circuit with resistance r_{v_1} and capacitance c_{v_1}, where c_{v_1} is the capacitance of routing segment e_{v_1}. Similarly we model routing segment e_{v_2} by another π-model equivalent circuit with resistance r_{v_2} and capacitance c_{v_2}. Combined these equivalent circuits, we have an equivalent circuit for T_u as shown in Figure 6(b). The quadruple (n_u, C_u, d_u, S_u) for the node attribute of u derived from $(n_{v_1}, C_{v_1}, d_{v_1}, S_{v_1})$ and $(n_{v_2}, C_{v_2}, d_{v_2}, S_{v_2})$ is computed by the following equations:

$$\begin{aligned} n_u &= n_{v_1} + n_{v_2}, && (2)\\ C_u &= C_{v_1} + C_{v_2} + c_{v_1} + c_{v_2}, && (3)\\ d_u &= \max\{D_1, D_2\}, && (4)\\ S_u &= \max\{S_{v_1}, S_{v_2}, D_1 - D_2 + S_{v_2}, \\ &\qquad D_2 - D_1 + S_{v_1}\}, && (5) \end{aligned}$$

where

$$\begin{aligned} D_1 &= r_{v_1}(\frac{c_{v_1}}{2} + C_{v_1}) + d_{v_1}, && (6)\\ D_2 &= r_{v_2}(\frac{c_{v_2}}{2} + C_{v_2}) + d_{v_2}. && (7) \end{aligned}$$

Equations (2)-(4) and (6)-(7) are simply the results of the related definitions and Elmore delay formula. Equation (5) can be obtained by analyzing total 6 possible cases of the delay relationship between T_{v_1} and T_{v_2} and using the definition of clock skew. The node attribute of u is obtained by considering all pairs of quadruples between the node attribute of v_1 and the node attribute of v_2. An important observation is that for quadruples with same n_u and d_u [2], we need to retain only the quadruple with the smallest skew S_u in the attribute computation of u. This is because that any other quadruple with larger skew will lead to a suboptimal solution to min-skew module selection.

After computing node attributes for all nodes in the clock tree, we can determine the n_u's by backtracking from the root node. This can be done easily using the information stored during the bottom-up computing phase. The following theorem states the optimality of the algorithm. The proof of the theorem is omitted.

Theorem 1 *The min-skew module selection algorithm is optimal for selecting logic modules to produce minimum clock skew placement.*

In practice, the computation of delay d_u need to be discretized. Let $\Delta_d = [d_{min}, d_{max}]$ be the range of d_u, where d_{min} and d_{max} are the minimum and maximum possible phase delays on the clock tree, respectively. We partition Δ_d into L intervals, $[i\delta + d_{min}, (i+1)\delta + d_{min}), 0 \le i \le L-1$, where $\delta = \Delta_d / L$. A delay d_u which falls within an interval is rounded down to the low end of the interval. Note that $0 \le n_u \le N$ for every node u in T, where N is the number of circuit modules. With discretization, the node attribute of an internal node need memory $O(NL)$ and can be computed in time $O(N^2L^2)$. We thus have the following theorem on the efficiency of the min-skew module selection algorithm.

Theorem 2 *The min-skew module selection algorithm runs in time $O(MN^2L^2)$ using $O(MNL)$ memory, where M is the number of internal nodes in the clock tree and N is the number of circuit modules.*

The memory requirement of the algorithm can be further reduced by taking advantage of possible symmetry of the clock trees. Due to space limit, we omit further discussion on this subject.

4 Extensions

4.1 Buffered Clock Trees

We assume that buffers can be only placed at the locations of internal nodes of the clock tree. Figure 7(a) shows a subtree T_u with a buffer at node u. The buffer can be modeled by a two-terminal equivalent circuit with input capacitance C_b, delay d_b, and output resistance r_b [11]. See the circuit between points p and u in Figure 7(b). Combined with the equivalent circuit in Figure 6(b), we have an equivalent circuit as shown in Figure 7(b) for the buffered subtree T_u of Figure 7(a).

Let A_u and A_p be the node attributes of nodes u and p, respectively. The computation of A_p is a transformation from A_u. The node attribute A_u can be computed by the algorithm described in Section 3. For each quadruple $(n_u, C_u, d_u, S_u) \in A_u$, construct a quadruple (n_p, C_p, d_p, S_p) for A_p such that

$$\begin{aligned} n_p &= n_u, && (8)\\ C_p &= C_b, && (9)\\ d_p &= d_b + r_b C_u + d_u, && (10)\\ S_p &= S_u. && (11) \end{aligned}$$

Equation (9) is from the fact that the capacitance seen at p is the input capacitance C_b of the buffer. Equation (10) is a direct application of Elmore delay formula. After computing node attributes of the multi-stage clock tree, we then use backtracking to compute the min-skew logic module selection.

4.2 Multi-Phase Clock

Assume there is a set $\{T_1, \ldots, T_K\}$ of K clock trees in the multi-phase clock, where $T_i = (V_i, E_i), 1 \le i \le K$. Each clock tree is used to transmit a clock signal. Let $S_i, 1 \le i \le K$, be the clock skew of T_i. The minimum skew logic module selection problem for multi-phase clock is to select logic modules for every clock tree such

[2] Note that capacitance C_u is uniquely determined by n_u for a node u.

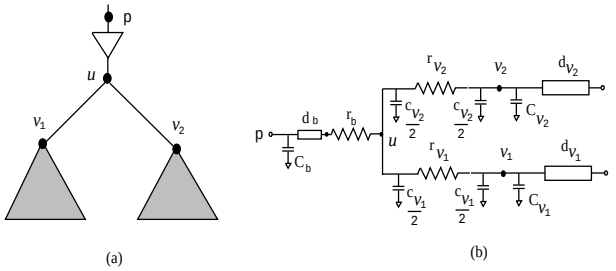

Figure 7: (a) A buffered subtree T_u. (b) Equivalent circuit of buffered subtree T_u.

that $\max\{S_1, \ldots, S_K\}$ is minimized. For simplicity of presentation, we describe the case $K = 2$, i.e., two-phase clock. Extension from the case $K = 2$ to the case $K > 2$ is straightforward.

To extend the min-skew module selection algorithm of Section 3, we need an assumption on the structure of two-phase clock trees. We say two clock trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ to be *isomorphic* if there is an one-to-one mapping $\sigma : V_1 \mapsto V_2$, such that (1) if $v_1 \in V_1$ is a leaf node of T_1, then $\sigma(v_1)$ is the same leaf node, i.e., $v_1 = \sigma(v_1)$; (2) for any $v_1, v_1' \in V_1$, $(v_1, v_1') \in E_1$ if and only if $(\sigma(v_1), \sigma(v_1')) \in E_2$. The corresponding nodes with respect to the mapping σ in T_1 and T_2 are said to be the *equivalent nodes*. Figure 8 shows an example of two isomorphic clock trees, transmitting clock signals *clk*1 and *clk*2. A pair of equivalent nodes is indicated by a circle.

Now we describe the the algorithm of min-skew module selection for two-phase clock. We associate node attribute with each pair of equivalent nodes. For a pair of equivalent nodes $v_1 \in V_1$ and $v_2 \in V_2$, the node attribute is a set of tuples of the form $(n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}, S_{v_1}, S_{v_2})$, where n_{v_1} and n_{v_2} are the numbers of circuit modules allocated in T_{v_1} and T_{v_2}, d_{v_1} and d_{v_2} are the maximum delays from v_1 and v_2 to the clock pins in T_{v_1} and T_{v_2}, S_{v_1} and S_{v_2} are the clock skews of T_{v_1} and T_{v_2}, respectively. By defining node attributes for the equivalent nodes in this way, we ensure that no module will be selected by both clock trees simultaneously, and thus avoid module selection conflicts.

The node attribute for a leaf node is the set $\{(0, 0, 0, 0, 0, 0), (1, 0, R_0(C_0 + C_L), 0, 0, 0), (0, 1, 0, R_0 (C_0 + C_L, 0, 0)\}$. The node attribute for a pair of internal equivalent nodes is computed similar to that in Section 3. Each of the subtrees T_{v_1} and T_{v_2} is modeled by an equivalent circuit, constructed from the parameters provided by the corresponding node attributes of the children nodes. The tuple $(n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}, S_{v_1}, S_{v_2})$ is computed using the formula similar to Equations (2)-(7), on the corresponding equivalent circuits. For tuples with same $n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}$, we need to keep only the tuple with minimum clock skew in node attribute. More specifically, for two tuples $(n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}, S_{v_1}, S_{v_2})$ and $(n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}, S'_{v_1}, S'_{v_2})$, if $\max\{S_{v_1}, S_{v_2}\} < \max\{S'_{v_1}, S'_{v_2}\}$, then the tuple $(n_{v_1}, n_{v_2}, d_{v_1}, d_{v_2}, S'_{v_1}, S'_{v_2})$ can be removed from the node attribute.

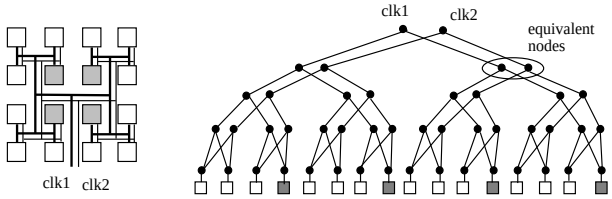

Figure 8: Two isomorphic clock trees.

5 Experimental Results

The min-skew module selection algorithm was implemented in C on a SUN SPARC station 1 and was tested on comb clock tree architecture and slicing clock tree architecture. We assume the chips are fabricated in 1.0 μm CMOS technology. The load capacitance is 100 fF. The on-resistance and capacitance of a programmable switch are 500 Ω and 10fF, respectively. The per-unit resistance and capacitance of routing metal wire are assumed as 3 mΩ and 0.05 fF, respectively. All logic modules are assumed identical. An FPGA chip can be viewed as being constructed by repeating a basic building block called *tile* which is composed of a logic module and the surrounding routing area. A tile is assumed to be a square with 200 μm in each dimension.

The row-based FPGA chip with comb clock tree contains 10 rows, with 30 logic modules in each row. The comb clock tree architecture is as that shown in Figure 2(a). The size of the FPGA chip with slicing clock tree contains 20×20 logic modules. The slicing clock tree structure is shown Figure 9, where the dot lines indicate the first few levels of cut lines represented by the clock tree. The sub-clock trees in all the squares are constructed from the standard H-trees. The entire clock tree of the chip in Figure 9 is zero-skew when no clock pin is connected to the leaf nodes and is constructed by using the algorithm of [11].

The experimental results are shown in Table 1 and Table 2 for slicing clock tree and comb clock tree, respectively. The second to the fourth columns in Table 1 list the number of modules in each experimental circuit and the corresponding utilization rate of 400 and 300 logic modules. The results are measured in phase delays, clock skews and wire lengths, and are shown in the columns "Min-skew". For slicing clock tree, we also implement a classical min-cut placement algorithm which chooses cut lines and partition ratios evenly. The corresponding results are shown in the columns "Min-cut" in Table 1. As a comparison for comb clock tree architecture, we place the circuit modules uniformly for each circuit and the corresponding results are listed in the columns "Uniform" in Table 2. The last rows of Table 1 and Table 2 show the comparisons on the results of all the circuits. The clock skew reductions for slicing clock tree and comb clock tree architectures are on average 14 times and 47%, respectively, as compared with the algorithms without clock skew consideration. Meanwhile, the phase delays are similar. The wire lengths for the chip with slicing

Circuits	No. of Modules	Utilization (out of 400)	Utilization (out of 300)	Phase Delay (ps)		Clock Skew (ps)		Wire Length ($10^5 \mu m$)	
				Min-cut	Min-skew	Min-cut	Min-skew	Min-cut	Min-skew
BUSC	152	39.0%	50.7%	651.5	640.0	78.7	1.8	3.6	3.1
ALU2	159	39.8%	53.0%	663.0	681.4	75.1	1.7	8.6	8.2
Example2	164	41.0%	54.6%	674.5	655.4	70.0	2.9	15.1	14.8
X1	168	42.0%	56.0%	680.6	681.4	69.0	4.6	11.5	11.6
TooLarge	189	47.3%	63.0%	727.6	698.2	65.8	2.5	7.8	7.6
DMA	261	65.3%	87.0%	886.3	883.2	70.2	6.2	13.1	12.8
VDA	262	65.5%	87.3%	886.3	865.6	70.2	4.3	12.5	12.1
ALU4	264	66.0%	88.0%	888.1	871.8	65.9	5.5	17.8	17.7
Ebnr	390	97.5%	—	1133.0	1128.0	21.4	7.7	6.8	6.7
total	—	—	—	7190.9	7105.0	586.3	37.2	96.8	94.6
comparison	—	—	—	+1.2%	1	+1476%	1	+2.3%	1

Table 1: Experimental circuits and results for slicing clock tree.

Circuits	Phase Delay (ps)		Clock Skew (ps)	
	Uniform	Min-skew	Uniform	Min-skew
BUSC	747.1	809.1	654.3	355.0
ALU2	772.3	859.3	677.7	415.2
Example2	800.5	880.2	705.9	433.7
X1	807.7	905.1	713.1	453.8
TooLarge	889.9	884.5	805.7	444.0
DMA	1040.0	944.7	964.4	754.4
VDA	1042.0	966.5	965.6	750.2
ALU4	1044.0	966.5	968.0	776.2
total	7143.5	7215.9	6454.7	4382.5
comparison	-1.0%	1	+47.3%	1

Table 2: Experimental results for comb clock tree.

clock tree are also computed based on the perimeters of net bounding boxes and the results are comparable. It is noticed that the clock skew reduction for comb clock tree is much smaller than that for slicing clock tree. This might be due to the fact that the slicing clock tree used in the experiment is zero-skew when no logic module connected to the clock tree, while the comb clock tree is not zero-skew and thus leaves much less room for improvement.

Figure 9 shows the module selection for circuit ALU4 in the slicing clock tree chip. The shaded logic modules are selected.

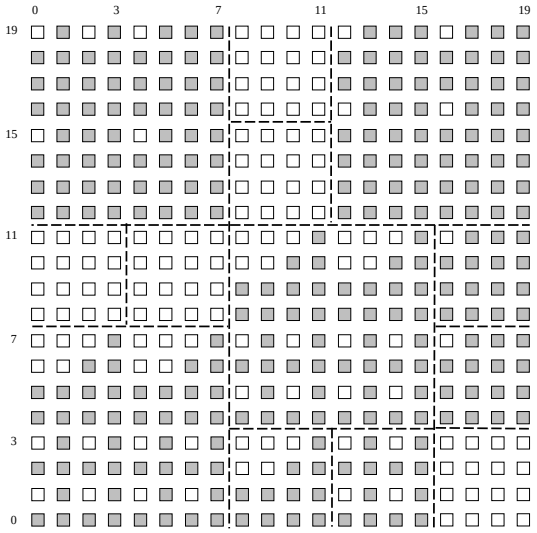

Figure 9: A slicing clock tree and module seletion result for ALU4.

Acknowledgement

We would like to thank Steve Brown and Baharam Fallah of University of Toronto for providing us with the benchmark circuits, Nick Haruyama of AT&T Bell Lab and Sina Kaptanoglu of Actel for the helpful discussions.

References

[1] Actel Corporation, *ACT 3 Field Programmable Gate Array*, Preliminary, January 1993.

[2] Actel Corporation, *ACT Family Field Programmable Gate Array Data Book*, April 1992.

[3] M. Ahrens, et al., "An FPGA Family Optimized for High Densities and Reduced Routing Delay", *CICC*, pp.31.5.1-31.5.4, 1990.

[4] AT&T Microelectronics, "Optimized Reconfigurable Cell Array (ORCA) Series Field-Programmable Gate Arrays", Advance Data Sheet, February 1993.

[5] A.E. Dunlop and B.W. Kernighan, "A Procedure for Layout of Standard-Cell VLSI Circuits", *IEEE Trans. CAD*, Vol. 4, No. 1, pp. 92-98, 1985.

[6] W.C. Elmore, "The transient response of damped linear networks with particular regard to wide band amplifiers", *J. Appl. Phys.*, Vol. 19, pp.55-63, 1948.

[7] A.E. Gamal, et al., "An Architecture for Electrically Configurable Gate Arrays", *IEEE Journal of Solid-State Circuits*, Vol.24, No.2, pp. 394-398, 1989.

[8] H.C. Hsieh, et al., "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays", *CICC*, pp.31.2.1-31.2.7, 1990.

[9] D. Marple and L. Cooke, "An MPGA Compatible FPGA Architecture", *First International ACM/SIGDA Workshop on FPGA*, pp. 39-44, 1992.

[10] J. Rubinstein, P. Penfield and M.A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. CAD*, Vol. 2, No. 3, pp. 202-211, 1983.

[11] R.S. Tsay, "Exact Zero Skew", *ICCAD*, pp. 336-339, 1991.

[12] J.L. Wyatt, Jr., "Signal propagation delay in RC models for interconnect", in A.E. Ruehli, *Advances in CAD for VLSI*, vol.3, part 2, *Circuit Ananlysis, Simulation and Design*, 1987.