A Practical Clock Tree Synthesis Flow

Mely Chen Chi
Department of Information and Computer Engineering
Chung Yuan Christian University,
Chung Li,
Taiwan
mlchen@ice.cycu.edu.tw

Shih-Hsu Huang
IC Design Department
Computer & Communications Research Laboratories
Industrial Technology Research Institutes
Hsinchu, Taiwan
shhuang@itri.org.tw

Abstract

In deep sub-micron era, an ASIC chip may contain millions of gates and have the requirements of low power and high performance. The ability to construct multiple clock trees effectively is very important.

After conducting many clock tree synthesis experiments, which explore various configuration of clock tree structures and layouts, a guidance for clock tree synthesis is generated. By applying this guidance, the clock tree design procedure is simplified and the design time is shortened. This methodology has been used to implement clock trees on the chips designed in the Computer and Communications Research Laboratories. Our experience shows that for single clock trees the intra-clock skew is confined within 0.1ns in one design pass for 0.35μ CMOS technology chips. For multiple clock trees, which are originated from the same clock source, the inter-clock skew may also be controlled easily. This design methodology is proven to be an effective method to implement clock trees on ASIC chips.

Keywords: Clock Tree Design, Clock Tree Structure, Clock Skew Minimization, Clock Tree Synthesis, ASIC Design

I. Introduction

Due to the progress of IC processing technology, a chip may have millions of gates with a very complex structure. The synchronization of clocks on a chip is critical to the performance and reliability of the chip [1,2,3]. There are different ways to implement a distributed clock network . A distributed clock tree structure, as shown in Fig.1 is commonly used in VLSI designs. Clock tree synthesis is a technique that is used to dynamically insert clock drivers between the clock source pin and multiple receiver pins, physically place the drivers on the chip, and route the clock net. After routing, the largest difference in delay among all the branches from the source pin to driver pins is called clock skew. The clock skew should be minimized in order to speed up the circuit.

In many ASIC applications low power is a very important requirement. The gated-clock technique is used in many designs in order to save power. When a chip has multiple clocks how to control the inter-clock skew is a challenge work. Designers not only need to construct individual balanced clock trees to minimize intra-clock skew but also to balance multiple clock trees simultaneously to minimize the inter-clock skew.

In a traditional ASIC design flow, a clock tree generation is done after the logic synthesis before the layout stage. There are two drawbacks in this approach. It lacks of physical locations of the receiver elements such that the wire delay can not be obtained during the clock tree driver assignment. Also it will require engineers to spend efforts to balance the clock trees on the chip during layout. Usually it takes many iterations between synthesis and layout before a satisfactory result is achieved.

In order to solve this problem, a series of experiments were conducted. As the result of these experiments, a clock tree design methodology is developed. It helps designers to construct balanced clock trees effectively. This

methodology is described in this paper. This procedure has been applied to many chips designed in the Computer and Communications Research Laboratories of Industrial Technology Research Institute (CCL/ITRI).

! .Clock Tree Synthesis Flow

As shown in Fig. 2, in an ASIC design procedure, after the logic synthesis the logic designers will define clock tree structures and pass it to a layout engineer. After all the cells are placed, the layout engineer will input the information of the tree structure to a layout tool. The tool will insert drivers in the placement and update the net list. Then all the nets are routed. We use Avant!/Apollo for the clock tree synthesis. After layout is done, the distributed RC of the clock network is extracted by a distributed RC extractor (Avant!/StarRC). Then we use a delay calculator (Ultima/MDC) to calculate timing delay for all branches and generate a Standard Delay Format (SDF) file which stores the timing delay data of the drivers and the interconnects. The clock skew is the maximum difference in delay among all branches. Then the updated netlist, which includes the clock drivers, and the SDF file are fed to a timing simulator (Cadence/Verilog) for post-layout simulation.

If the delay or skew requirements are not satisfied then the clock tree structure will be modified. The loop starting from the clock tree synthesis to the timing analysis will be repeated until the timing constraints are met. Usually it took several loops to complete the design. It is very time consuming. To reduce the number of iterations, we developed a methodology that is described in the next section.

" . Clock Tree Design Methodology

In order to reduce the iterations of the clock tree synthesis, a clock tree design methodology is developed. It provides a guidance to layout engineers to synthesize clock trees for each chip without extra effort from logic designers.

To construct clock trees in an ASIC chip, two major techniques are required. First technique is to design the tree structure and the second technique is to layout the tree. These two techniques are described below.

To design the tree structure, as shown in Fig. 1, includes the decision of how many levels of drivers are in the tree, how many branches of fanouts are at each level, and which driver to be used at each branch. The design of the tree should conform to the skew and delay constraints. In the tree structure, each level is balanced one by one. There will have some amount of capacitive variation at each level. So less levels means less chances to have variability, that means less skew. The number of branches at each level is also an important factor. High fanouts means more drivers and more interconnects. It will result in a larger area and more potential for skew variability due to interconnects. Low fanouts may results in many levels, which may also cause the skew variation. Therefore, designing a clock tree structure which satisfies the timing requirements usually take iterations to complete. In order to reduce the design time, a series of experiments which simulate clock tree design in real circuits are performed and the results are summarized as a guidance for clock tree design. This guidance provides data for tree construction according to the number of clock receivers. After logic synthesis, logic designers do not need to design the tree structure. Layout engineers just follow the guidance and pass the tree structure data to the P&R tool. As to our experience, all chips design in the Computer and Communications Research Laboratories satisfied the clock skew requirement in one pass.

The clock tree structure design includes two parts. Part one is for the synthesis of a single clock tree. The second part is for synchronizing multiple clock trees of the same clock source. They are described in Section # and Section $ respectively.

Design Methodology for a Single Clock Tree

We first conduct many experiments that resemble the clock trees in real chips to generate a guidance table for the clock tree structure design. The methodology is described in this section.

.1 Clock Tree Structure Guidance Generation for a Single Clock Tree

Our goal is to generate a clock tree design guidance, which may be applied to the majority of chips designed in the Computer & Communication Research Laboratories. To generate the guidance, first a set of netlists which resemble real circuits of different sizes are generated. Then various clock tree structure are explored on each netlist. Clock tree synthesis and timing analysis for each clock tree structure are performed. Finally, among all results the best tree structures for different number of receivers are extracted which to be used later as guidance. The details of these steps are described below.

.1. 1 Generate Netlists with Specific Number of Receivers in a Clock Tree

In order to resemble real designs, a netlist generator is developed. By specifing the number of clock receivers as input, it will generate different netlists with logic

connectivity distribution similar to the connectivity distribution of real circuits. The distribution of nets with specific number of fanouts is summarized from the statistics of the chips designed in the Computer and Communications Research Laboratories. The percentage range of the net distribution in a netlist is listed in Table 1. Different netlists are randomly generated for the same number of clock receivers. This netlist will not only have similar logic connectivity to real circuits but also have reasonable chip areas for constructing clock trees.

.1.2. Design the tree structure using bottom-up process

We first define different levels of tree structures to be used in one experiment, for example, using 2, 4, 6, 8 levels in a tree. For each type of tree structure, we build the tree level by level from the leaves (flip-flops) to the root (clock source). In ASIC designs, the drivers are selected from the standard cell library. In our design, inverter cells are used as drivers. We study the driving capability and timing characteristics of the available inverters. We first select drivers for the lowest level which directly drive the flip-flops. The number of flip-flops may be driven by each driver is calculated by equation (1) where the wire load capacitance is an estimated value.

Maximum Number of flip-flops
= (Maximum driving capacitance - Wire load capacitance) / Input pin capacitance of flip-flops (1)

We select drivers with lower driving capability at the bottom level. After the bottom level drivers are defined, then the input capacitance of these drivers are used as the capacitance load for the selection of drivers of one level above it. Repeating this process until the clock source is reached. The choice of driver used in different levels is made according to the following considerations. First, the driver should have sufficient driving capability to handle the variation of the loading capacitance. High drive inverters will add high capacitance load to the driver of the previous level. It will increase insertion delay. Also the use of strong drivers may result in large power dissipation. So at different levels, inverters are chosen to balance the driving capability and the capacitance load to minimize insertion delays.

.1.3. Use layout tool to synthesize a clock tree

After choosing one clock tree structure, we supply the number of level, number of fanout at each level, and the driver type at each level to a layout tool. We also supply a netlist, which has the correspondent number of flip-flops, to the tool as an input. We use Avant!/Apollo as P&R tool to insert the driver cells after logic cell placement. It will also perform the clock routing, and update the netlist.

.1.4. Analyze the delay and the skew of the clock net

After layout is completed, the distributed RC network of the clock tree is extracted and fed to a delay calculator. The calculator will also use the timing information of the driver cell provided by the standard cell library to calculate the timing delay from the clock source to the driver pins of all branches. We can then calculate the skew of the clock tree. This calculation is performed on all timing corners including best, typical, and worst cases of the driver cells.

.1.5. Construct the clock Tree Guidance after many layouts

As described in the previous sections, different clock tree structures are explored. Different combination of driver types are applied to many netlists. That means given a specific number of flip-flops, the total number of layout may be calculated using the equation (2).

No. of layout = No. of the sample netlist × No. of clock tree structure ×
No. of different combination of driver types (2)

The delay and the skew of the clock tree are calculated for each layout. We look for a specific tree structure and types of driver that consistently results in small values of timing delay and skew for all layouts. The design of this tree is recommended as the guidance for constructing the clock tree with the specific number of receivers (flip-flops). From our study, it is found that using 2-level tree structure is the best choice. The clock skew is found to be less than 0.1ns for 0.35% CMOS technology designs. The experiments are done for different numbers of receivers (flip-flops). The recommended tree designs are summarized in a table. The table includes the number and the type of drivers at each level for different number of receivers (flip-flops). Table 2, lists the tree structure for clocks with receivers (flip-flops) ranging from 200 to 900. If a clock has 200 receivers (flip-flops), the tree structure has two IND type drivers at level 1 and twenty INC type drivers at level 2. Each INC driver at level 2 will drive about 10 flip-flops. The driver IND has higher driving capability than a driver INC does. The tree structure is similar to that in Fig. 1. If the clock has 250 receivers, then the clock tree structure of 200 receivers is used. The tree structure is customized for the specific library.

.2 Clock Tree Synthesis

The second technique in clock tree implementation is to layout a balanced clock tree. The layout technique is to

place the drivers such that after the clock routing the clock tree branches are balanced and the signal propagation delays from the clock source to all receiver pins are as equal as possible. Both interconnect wire capacitance and input loading capacitance affect the signal delay. The maximum difference among all delays is the clock skew of the clock tree. To balance the tree network, the root driver is confined at the center of the chip by layout engineer. The rest of work is usually done by an automatic placement & route tool. Some commercial tools can do the job very well. In general, if the receiver pins are unevenly distributed in a large area or there are routing blockages on the chip then it is difficult to have balanced distribution of the tree.

V. Multiple Clock Tree Design Methodology

In the previous section, we have discussed how to implement each individual clock tree. In this section, we describe procedures of how to balance multiple clock trees which are originated from the same clock source. The goal is within the allowable bound of delay of a chip, we try to develop a procedure which may be used to minimize the inter-clock skew effectively.

Because the skew of each individual clock tree is confined within a small value, so the structure of each individual clock tree is preserved. To equalize the timing delay of all trees, a series of 4 larger drivers are used to replace the root driver of each tree. We use inverters as drivers. Because at the root level the interconnect wire is longer, the capacitive load and capacitive variance is larger. The larger drivers have larger driving capability and are less sensitive to the capacitive variance. These four inverters are called delay-control drivers of the individual clock tree.

After the delay-control drivers are inserted in all clock trees, the synthesis of each clock trees is performed as described in the previous section. The timing delays of all branches of all clock trees are calculated and sorted by the descending order. We use the longest path as the reference to calculate inter-clock skews. If all inter-clock skews are within the range of timing constraint then the clock tree construction is done. If some of the inter-clock skews are larger than the requirement, the delay-control drivers of a shorter delay tree are replaced by smaller inverters one at a time. This new clock structure may be analyzed using Synopsys /DesignTime without re-do the clock synthesis.

This process is repeated until all the inter-clock skews satisfy the timing constraint. Replacing the delay-control inverters with smaller inverters has the advantages of less power consumption and smaller area. Also in physical design, it is easier to replace big inverters with smaller ones than vice versa. The algorithm is described in the next paragraph.

Suppose *CLKi* is a clock tree. We define its delay-control buffers, i.e., four inverters, as *CLKi[1], CLKi[2], CLKi[3],* and *CLKi[4].* The output of inverter *CLKi[4]* is the starting point of a single clock tree. Following clock tree guidance table, the skew of a clock tree is confined within a very small value. Suppose delay*(CLKi)* is the delay of clock tree *CLKi.* Assume there are m clock trees on a chip. We sort all the clock trees such that *delay(CLK1)& delay(CLK2)& ... & delay(CLKm).* If using *delay(CLKm)* as the reference, assume *n* clock trees exceed inter-clock skew requirement, i.e., *CLK1*, *CLK2*, ..., and *CLKn*. Fig. 3 shows the pseudo code of Procedure minimize_inter_clock_skew. As shown in Fig. 3, the procedure minimize_inter_clock_skew minimizes inter-clock skew by adding delay to *CLK1*, *CLK2*, ... ,and *CLKn*. Let's define the term *size(bufferi)* is the size of *bufferi*. By reducing the size of delay-control buffer, we can add delay to the clock tree. The methodology of adding delay to a clock tree is shown in Fig.4.

. Example & Results

After applying the methodology to all 0.35μ technology chips designed in the CCL, it is found that the skews of all single clock trees are less than 0.1ns in one pass of clock tree synthesis. For the application on multiple clock trees design, we use a wireless communication chip as an example for illustration. The chip has 12,611 nets, 11,090 standard cells, and 4 macro cells. Because of the low power requirement, the system clock is divided into 6 gated clocks, Each gated clock is enabled at different condition for power saving. The architecture of the clock enabling circuit is the same as the concept of gated clocks. Because these 6 clocks are originated from the same clock source, they are required to remain synchronized.

The clock tree design methodology is applied. The intra-clock skews of all the individual clocks are confined within 0.1ns in one pass. The requirement of inter-clock skew is also met in the second pass. Some results of delays of all clocks are listed in Table 3 and Table 4. Table 3 shows the delay of all clocks by applying 3.3 volts of voltage and using the worst case timing corner of drivers. The second column in Table 3 shows the minimum delay among all branches of the correspondent clocks. For example, the minimum delay among all branches of clock1 is 1.013983 ns. The same explanation holds for the maximum delay. It shows the longest delay occurs on

clock5 and the shortest delay occurs on clock1. The inter-clock skew is within 0.3 ns.

Table 4 shows the clock delays by applying 3.3 volts of voltage and using the best timing corner of the driver cells. It shows the longest delay still occurred on a branch of clock5. The shortest delay is also on a branch of clock1. The inter-clock skew is less than 0.2 ns.

(. Conclusion

An effective clock tree design methodology is developed and it is in production use. A clock tree structure design guidance is summarized from the results of many clock tree synthesis experiments. Logic designers do not need to spend time to design the clock tree structures for each chip. With the help of the guidance, layout engineers may obtain the tree structure from the guidance. From the past experiences, the inter-clock skew of an individual clock tree may be confined within 0.1ns for 0.35u CMOS technology chip in one pass of clock tree synthesis. The inter-clock skew among multiple clock trees, which are originated from the same clock source, may be controlled easily. It simplifies the design process and reduces the number of iterations of design cycle. The clock tree design methodology is proven to be very effective in ASIC chip design.

VIII. References

[1] J. Burkis, "Clock Tree Synthesis for High Performance ASIC", in Proc. The 4th Annual IEEE ASIC Seminar & Exhibits, pp. p9-8.1~p9-8.3, 1991.

[2] E.G. Friedman, editor, "Clock Distribution Networks in VLSI Circuits and Systems: A Selected Reprint Volume", IEEE Press, 1995.

[3] A. Takahashi and Y. Kajitani, "Performance and Reliability Driven Clock Scheduling of Sequential Logic Circuits", in Proc. of ASP-DAC, pp. 37-42, 1997.

[4] C. Y. Lee, Y. R. Lin, M. C. Chi, S. H. Huang, K. C. Jung, "Clock Tree Synthesis Flow" , in Technical Journal of Computer & Communications Research Laboratories, pp. 28~31, 1997.

[5] S. H. Huang, " A Design Methodology for a 0.35um IC Project", in Technical Journal of Computer & Communications Research Laboratories, pp.42-47, 1999

[6] C.H. Chien, "CCL 0.35um Standard Cell Library", in Technical Journal of Computer & Communications Research Laboratories, pp. 32-40, 1998.

[7] Apollo User Guide 1998.4, Avant! Inc.

NSC Project Number: 89-2215-E-033-008

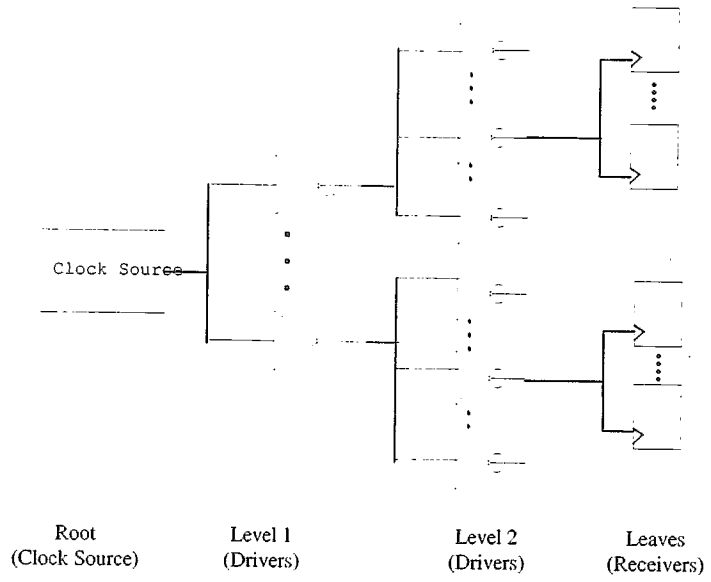

Fig.1. A two level clock Tree structure

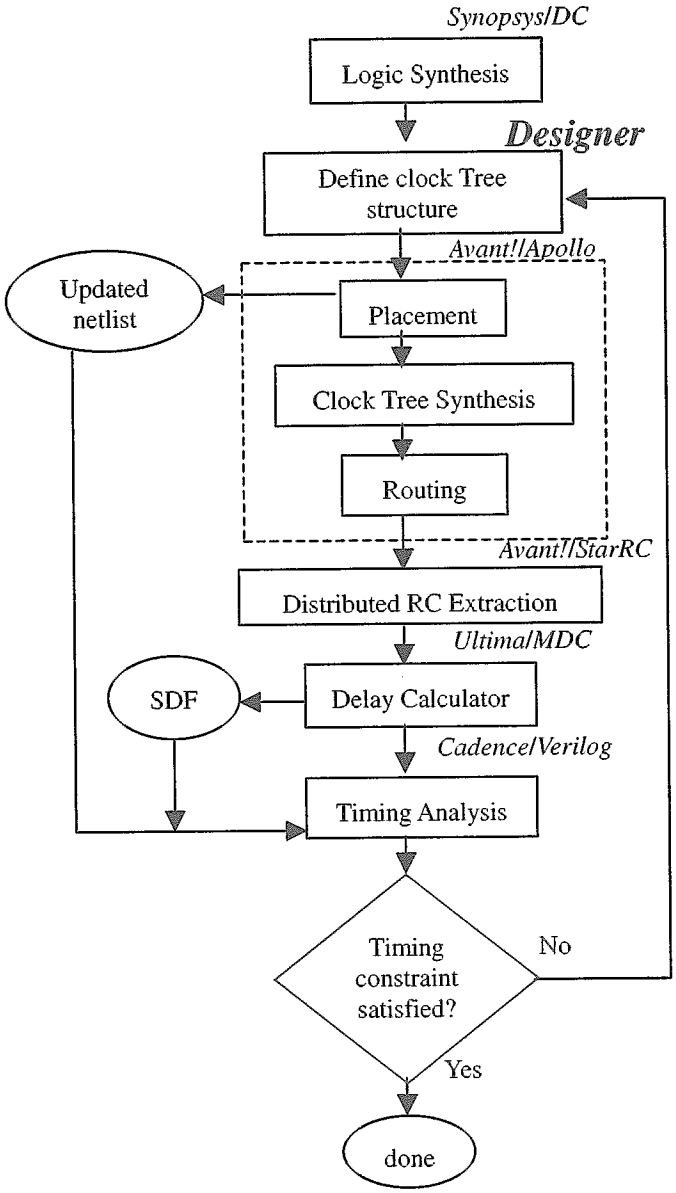

Fig.2 Clock Tree Systhesis Design Flow

```
Procedure minimize_inter_clock_skew()
{
integer index;
        for index = 1 to n do   /* assume n clock trees exceed skew requirement*/
          while (delay(CLK_m)-delay(CLK_index)>skew_requirement)
              call procedure downsize_maxbuffer(CLK_index);   /*as shown in Fig.4*/
}
```

Fig.3 The pseudo code of Procedure minimize_inter_clock_skew.

```
Procedure downsize_maxbuffer(CLK_i)
{
integer maxbuffer_index , index ;
        maxbuffer_index = 1;
        for index = 2 to 4 do
          if (size(CLK_i[maxbuffer_index])< size(CLK_i[index]))
             maxbuffer_index = index;
        reduce the buffer size of CLK_i[maxbuffer_index];
}
```

Fig.4 The pseudo code of Procedure downsize_maxbuffer.

Table 1. Distribution of nets with corresponding fanout numbers in generated netlists.

Number of fanout per net	Percentage range of the type of net in a netlist
1	70%) 75%
2	10%) 15%
3	6%) 10%
5	1%
6	1%
7	1%
8	1%

Table 2. Clock tree guidance table for number of receivers ranging from 200 to 900.

Number of Receivers	Clock Tree Structure			
	Level 1		Level 2	
	Driver Type	Number	Driver Type	Number
200	IND	2	INC	20
300	IND	3	INC	24
400	IND	4	INC	32
500	INE	4	IND	20
600	INE	4	IND	24
700	INE	4	IND	28
800	INE	4	IND	34
900	INF	4	IND	36

Table 3. The minimum and maximum delays of each clock by applying 3.3v and using the worst timing corner of clock drivers.

Clock tree	Minimum delay (ns)	Maximum delay (ns)	Skew (ns)
Clock0	1.099331	1.128136	0.0288050
Clock1	1.013983	1.045422	0.0414390
Clock2	1.165690	1.196863	0.0311730
Clock3	1.118894	1.153306	0.0344120
Clock4	1.104621	1.159723	0.0551020
Clock5	1.231235	1.313325	0.0820900

Table 4. The minimum and maximum delays of each clock by applying 3.3v and using the best timing corner of clock drivers.

Clock tree	Minimum delay (ns)	Maximum delay (ns)	Skew (ns)
Clock0	0.5948546	0.6149276	0.0200730
Clock1	0.5240446	0.5415518	0.0175072
Clock2	0.6037482	0.6282231	0.0244749
Clock3	0.6153231	0.6327919	0.0174688
Clock4	0.5769332	0.6118916	0.0349584
Clock5	0.6381470	0.7058691	0.0677221