

An Automatic and Practical Flow for Clock Tree Construction in Physical Design

Meng Liu, Wenqin Sun, Wuqi Wang, Zhiwei Zhang, Donglin Wang

Institute of Automation, Chinese Academy of Sciences

University of Chinese Academy of Sciences

Beijing, 100190, China

liumeng2013@ia.ac.cn

Abstract—High performance chip design is always a hot topic in integrated circuit (IC) field. Clock design plays a critical role in improving chip performance. In this paper, we propose an automatic and practical flow for clock tree design, which is a popular clock structure. Based on deferred-merge embedding (DME) algorithm, we integrate buffer insertion and obstacle processing into the layout design. The clock skew is guaranteed to be small in the presence of correlated process variations. Our work also combines the algorithm results and SPICE-driven simulation results into a practical flow.

Keywords—clock design; buffer insertion; obstacle processing

I. INTRODUCTION

As the technology of semiconductor process is scaling down to 10nm and below, it is possible to assemble systems with high performance processors that can theoretically provide computational power of up to tens of PLOPS [1]. In processor design flow, physical design plays an important role in controlling power and improving its frequency, although the processor architecture may determine the scale of performance. And clock distribution network (CDN) contributes more than 40% of processor power, mainly due to the constant switching and large capacitance loads from clock drivers or buffers [2]. CDN design always aims at finding the trade-off of low power and high performance. Apparently, this is a green computing topic in EDA (Electronic Design Automation) area. As a physical design part in EDA flow, CDN design has several structure styles to choose, including clock tree, clock mesh, clock spine and etc.

Clock tree design is the most common CDN structure for the following advantages. Firstly, it is easy to design and analyze by using tree-based distributions (single path between source and sinks). Secondly, the power of tree style is much less than non-tree style because of the less wire spending. Clock tree design in VLSI (Very Large Scale Integration) is also called clock tree synthesis, which is used to dynamically insert clock drivers between the clock source pin and multiple receiver pins, physically placing the drivers in the optimized locations, and routing the clock nets [3], as shown in Fig. 1.

Several previous works have contributed to clock tree design. The most influential algorithm is called DME (Deferred Merge Embedding) which aims at the minimum wire length and zero clock skew [4]. Several related papers extend

DME algorithm to some situations which engineers will encounter in real design. In [5], obstacles are considered in the Ex-DME algorithm when there are obstacles in the routing plane. In [6], buffer insertion is added into the original DME algorithm for minimizing clock phase delay and wire length. However, they did not use the simulation method to get accurate results.

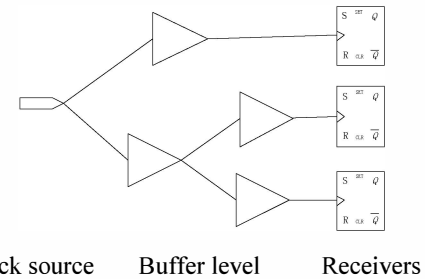


Figure 1 Example of a clock tree.

In commercial EDA tools, clock tree synthesis is integrated into the design flow, users only need and have to use the major commands to design. However, by ignoring too many details, the result cannot guarantee the critical timing requirement and will bring a lot of superfluous buffers which improve power, especially for 10nm and below. So developing a custom design flow of clock tree is a trend and can also make hybrid CDN structure like clock mesh and clock spine implement easily.

II. DESIGN FLOW AND SPECIFICATION

A. Problem Formulation

The sinks represent the registers in the layout design. Skew is because the difference of the clock arriving time from sinks. Let r be the root, N be the number of clock pins, and l_i be a leaf. Clock skew is defined as

$$Skew = \max_{1 \leq i, j \leq N} |d(r, l_i) - d(r, l_j)| \quad (1)$$

A set of obstacles are given. The constraints are the timing constraint and no overlap with obstacles. We choose appropriate clock buffers in 65nm TSMC library. Our objective is to minimize the worst skew and the total clock wire length. The information in netlist, timing constraint file, DEF (Design Exchange Format), logic library and physical library have been parsed into necessary files. We map this clock tree construction

into a commercial place and route tool to help us tape-out. The design flow is as in Fig. 2. The detail approaches will be discussed in following parts.

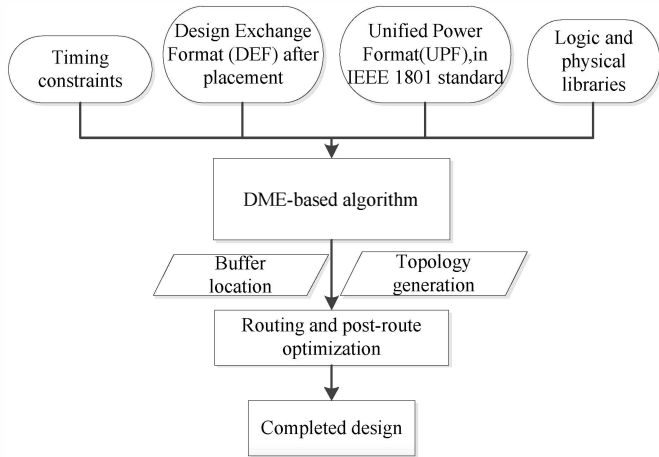


Figure 2 Algorithmic flow

B. Deferred-Merge Embedding(DME) Algorithm

The deferred-merge embedding (DME) algorithm defers the choice of merging points for sub-trees of the clock tree. The algorithm framework includes two phases: the first phase of DME is bottom-up, and determines all possible locations of internal nodes of topology that are consistent with a minimum-cost zero-skew tree. A TRR (Tilted Rectangular Region) is a collection of points within a fixed distance of a Manhattan arc. The merging segment of a node v in the topology is denoted by $ms(v)$. The output of this phase is sets of line segments, with each line segment being the locus of possible placements of an internal node (Fig. 3).

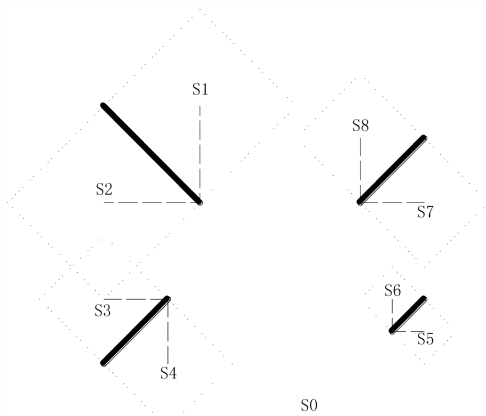


Figure 3 Construct merging segments for sinks.

In the next phase is called top-down phase or embedding phase, exact locations of internal nodes in topology are determined. Any point on the root merging segment from the bottom-up phase is consistent with a minimum-cost tree. This phase is illustrated in Fig.4.

DME requires an input topology. Several works have studied topology constructions that lead to low-cost routing

solutions and the most successful is the Greedy-DME method [4]. Greedy-DME iteratively finds the pair of nearest neighbors among the merging segments. We have integrated this algorithm into our flow.

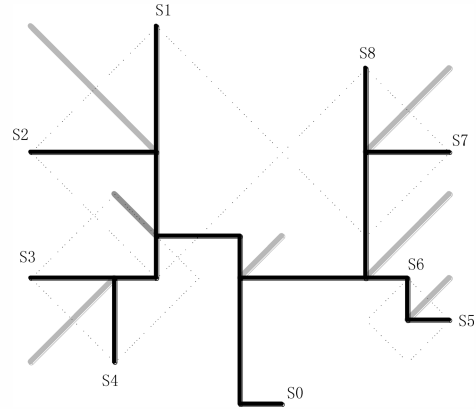


Figure 4 Embedding phase.

C. Buffered Clock Tree Modeling

In clock tree design, we usually insert buffers to control delay and slew. Our buffer insertion is implemented in DME phase. We design two modes of buffer insertion. Based on the constructed tree topology, buffers can be straightforwardly placed according to tree roots. This is called root mode. Otherwise, designers can set buffers according to clustering trees which can save a lot of buffers. This is called level mode. Because our clock tree is a binary tree structure, buffer insertion will not influence skew by using the same buffer type. The timing delay of buffer can be computed by using Elmore delay model and the wiring parasitics are modeled using distributed RC model (Fig.5). Buffers of different sizes have different parameters, so we build a buffer library to choose the suitable one. Before buffer insertion, the capacitance under tree node v is C , which equals the total capacitance of its sub-tree rooted by v . After buffer insertion, C reduces to c . Capacitance c is usually much less than C , therefore clock tree latency can be reduced. The delay through the buffer is $rc+d$. Given the driver resistance R and the load C of the wire segment, the path delay can also be computed according to π -type RC model. This analytical approach during DME flow can save much time compared to simulation method [6]. However, clock slew parameter cannot be avoided in real design.

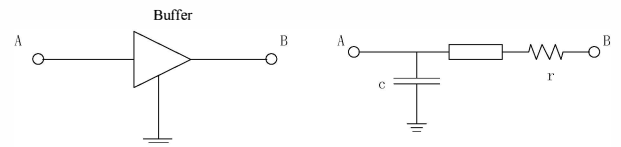


Figure 5 A clock buffer model.

In our buffer insertion flow, we first choose level mode. If the result can satisfy timing requirement, the program will go on for IC Compiler (a commercial place and route tool) stage. Level mode will use much less buffers than root mode.

However, root mode can guarantee better timing result. After analytical computing, Monte-Carlo SPICE simulation can help verify the final result. Monte-Carlo SPICE simulation is necessary to achieve relatively accurate buffer's output slew. The design flow is as followings in Fig.6. Our clock tree construction with buffers is shown in Fig.7.

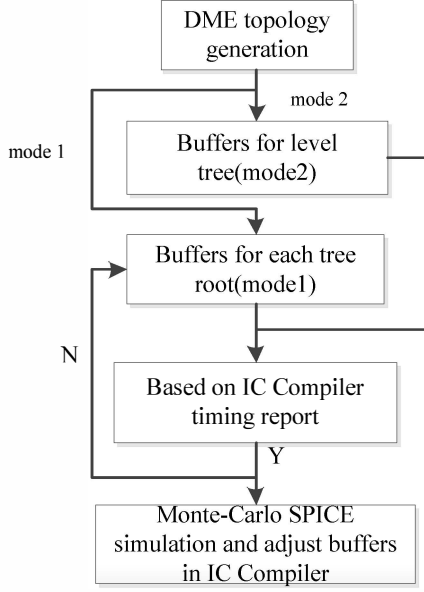


Figure 6 Buffer insertion flow.

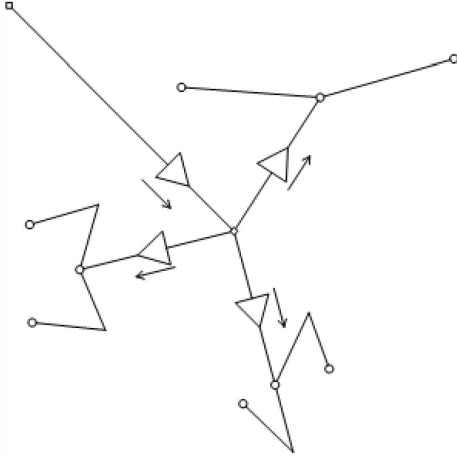


Figure 7 Clock tree construction with buffers.

D. Obstacle-avoiding Clock Routing

In our design flow, we regard blockages and UPF (Unified Powered Format) regions as obstacles. Apparently, obstacles are common met in real design. We integrated this process into the DME algorithm. After generating the clock tree topology and merging segments, the intersections can be computed based on obstacles and merging regions. SDR (Shortest Distance Region) is the set of points that have minimum sum of Manhattan distances. If SDR does overlap with obstacles, select the embedding point in merging lines according to DME rule. Otherwise, we need to compute the

shortest paths from p to merging segments, and choose the suitable point with the least cost to be the embedding point. The specific algorithm part is as followings.

Algorithm Top-down Embedding with obstacles consideration

Input: clock topology G , merging segment lines

Output: the embedding location of v

Let C be the possible embedding points of v

for each merging segment $ms_v \in ms_set(v)$ **do**

if $SDR(p, ms_v) \cap obsts = \emptyset$ **then**

 Select the embedding point lv by DME rule

$C \leftarrow C \cup l_v$

else

 Find the shortest paths from p to ms_v

 Select the target point lt with minimum cost

$C \leftarrow C \cup l_t$

Choose the point nearest to p in C

III. EXPERIMENTAL RESULTS

We implement our algorithm in C++ on a PC workstation of Intel Core i7-3537U CPU. We have ran experiments on benchmark circuits r1-r5 [7]. The parameter of metal layer is 0.003Ohm/nm and 0.02fF/nm. The output resistance of clock buffer we use is 100 Ω , the capacitance is 50 fF, and the buffer delay is 100 ps. The results are shown in TABLE I. All experiments are under 50ps of skew bound. Simulation graph is shown in Fig.8. We also use TSMC 65nm technology to finish the physical design of benchmark circuit s1196 in ISCAS89. The layout result is shown in Fig.9.

TABLE I RESULTS FOR BENCHMARK CIRCUITS

Circuits	Sink number	Wirelength (nm)	Delay without buffers (ps)	Delay with buffers (ps)
r1	267	969439.23	4702.14	4809.33
r2	598	1918564.36	10444.91	10545.11
r3	862	2405578.43	13384.10	13490.84
r4	1903	4946409.70	34161.80	34264.05
r5	3101	7267365.01	54993.40	55097.33

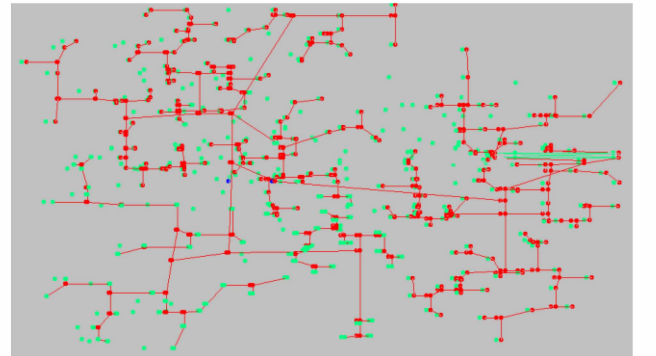


Figure 8 Simulation with obstacles consideration.



Figure 9 A clock tree of s1196 benchmark circuit.

IV. CONCLUSION

We have presented an automatic and practical flow for Clock Tree Construction in physical design with buffer insertion and obstacles avoiding features. By using our flow, it is easy to synthesize the circuit with low skew requirement. In future, we plan to apply the flow into clock mesh and clock spine design. This method provides the possibility of designing the local clock trees in hybrid clock structure.

REFERENCES

- [1] D. Wang, X. Du, L. Yin, C. Lin, H. Ma, W. Ren, et al., "MaPU: A novel mathematical computing architecture," in 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2016, pp. 457-468.
- [2] U. R. Tida, V. Mittapalli, C. Zhuo, and Y. Shi, "Opportunistic through-silicon-via inductor utilization in LC resonant clocks: concept and algorithms," in Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design, 2014, pp. 750-757.
- [3] M. C. Chi and S.-H. Huang, "A reliable clock tree design methodology for ASIC designs," in Quality Electronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on, 2000, pp. 269-274.
- [4] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 3, pp. 341-388, 1998.
- [5] A. B. Kahng and C.-W. A. Tsao, "More practical bounded-skew clock routing," in Proceedings of the 34th annual Design Automation Conference, 1997, pp. 594-599.
- [6] Y. Chen and D. Wong, "An algorithm for zero-skew clock tree routing with buffer insertion," in European Design and Test Conference, 1996. ED&TC 96. Proceedings, 1996, pp. 230-236.