Skew Bounded Buffer Tree Resynthesis for Clock Power Optimization

Subhendu Roy, Pavlos M. Mattheakis[†], Peter S. Colyer[‡], Laurent Masse-Navette[†], Pierre-Olivier Ribet[†], David Z. Pan
Department of Electrical and Computer Engineering, University of Texas at Austin, USA
[†]Mentor Graphics, Grenoble, France [‡]Mentor Graphics, Fremont, USA
subhendu@utexas.edu, {Pavlos_Matthaiakis, Peter_Colyer}@mentor.com, {Laurent_Masse-Navette, Pierre-Olivier_Ribet}@mentor.com and dpan@ece.utexas.edu

ABSTRACT

With aggressive technology scaling in nanometer regime, a significant fraction of dynamic power is consumed in the clock network due to its high switching activity. Clock networks are typically synthesized and routed to optimize for zero clock skew. However, clock skew optimization is often accompanied with routing overhead which increases the clock net capacitance thereby consuming more power. In this paper, we propose a skew bounded buffer tree resynthesis algorithm to optimize clock net capacitance after the clock network has been synthesized and routed. Our algorithm restricts the skew of the designs within a specified margin from its original skew, and does not introduce any additional Design Rule Check (DRC) violation. Experimental results on industrial designs, with clock networks synthesized and routed by an industrial tool, have demonstrated that our approach can achieve an average reduction of 5.6% and 3.5% in clock net capacitance and clock dynamic power respectively with a marginal overhead in the clock skew.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits**]: Design Aids;

Keywords

Clock tree, dynamic power, post-CTS optimization

1. INTRODUCTION

Clock network synthesis is a fundamental design step in modern ICs. In any synchronous VLSI circuit, the clock network provides the synchronizing signals to the sequential elements (flip-flops). Clock skew is the difference in the clock arrival times between two flip-flops. Although skew has been exploited to cope with the unbalanced data-path delays between the launch and the capture flops [1][2][3][4], skew minimization has long been a prime focus for the CAD engineers [5][6][7][8][9].

GLSVLSI'15, May 22 - 22, 2015, Pittsburgh, PA, USA.

http://dx.doi.org/10.1145/2742060.2742119

Commercial tools synthesize buffered clock trees with specific skew targets in presence of process variations and on-chip-variations [10][11] which along with clock latency (delay from clock root to sink *i.e.*, flip-flops) have been key objectives in ISPD'09 and ISPD'10 contests [12][8]. However, this stringent skew target may not be needed in the late design stage, *i.e.*, after clock tree synthesis (CTS), to achieve the timing closure. Rather, a marginal relaxation in the clock skew might not hurt the timing profile of the design, but could be exploited to optimize the clock net capacitance. Wire sizing and the use of non-default routing with more spacing can potentially improve skew and reduce overall clock tree capacitance [7][13][14], but at the cost of congestion and higher chip area. Several other techniques have been explored in the past to reduce clock dynamic power, such as synthesis-based clock gating [15], data-driven clock gating [16], clock gate cloning [17] and usage of multi-bit-flip-flop (MBFF) [18], but none of them focus primarily on the optimization of the clock net or wire capacitance at the post-CTS stage.

Due to the high switching activity and larger capacitive loads, 30-70% of the dynamic power is dissipated in the clock network [19][20]. The pin capacitance of the clock elements, such as the clock buffers, inverters, clock gates etc., and flip-flops, and the wire capacitance of the clock nets contribute to the clock dynamic power consumption. Recently, the net capacitance has become comparable, and sometimes even higher than the pin capacitance due to several reasons. Firstly, device size is shrinking with technology scaling, and as a result pin capacitance has reduced at much faster rate than the net capacitance. Secondly, with growing design complexity and aggravating variation effect, massive clock gating and skew balancing introduce more and more routing overhead. Consequently, clock net capacitance contributes a significant fraction of the clock dynamic power.

In this paper, we formulate a problem with the target objective of reducing the clock net capacitance of an already synthesized and routed clock tree network, given a specified relaxation margin in the clock skew. A buffer-tree resynthesis algorithm has been proposed which traverses the clock network in a bottom-up fashion, and relocates the clock buffers/inverters guided by mean-centric grid based placement (MCGBP). Since this resynthesis approach has been exercised in the post-CTS stage, semi-global optimizations by moving multiple clock buffers/inverters at a time may not be suitable as it would be disruptive to the timing profile of the design. Instead, we have explored the movement of the clock buffers/inverters one-by-one and ensured that these

local transformations do not (i) add any design-rule-check (DRC) violation, such as the maximum load or maximum slew violation etc., and (ii) increase the clock latency apart from meeting a specific skew bound. The key contributions of our paper are summarized as follows:

- To the best of our knowledge, this is the first problem formulation to optimize the clock net capacitance at the post-CTS stage given a specific skew bound.
- A buffer tree resynthesis algorithm has been proposed by mean-centric grid based placement of the clock buffers and inverters in bottom-up fashion without introducing any new DRC violation and increasing the clock latency.
- Our approach has been integrated into an industrial tool, and the execution of this algorithm on 22-65nm industrial designs resulted an average 5.6% reduction in the clock net capacitance with small improvement in the pin capacitance of the clock network.

The rest of the paper is organized as follows. Section 2 describes the problem formulation. Section 3 presents our skew bounded buffer tree resynthesis algorithm. Section 4 presents the experimental results for industrial designs with conclusion in Section 5.

2. PROBLEM FORMULATION

Suppose there are n clock domains in an already synthesized and routed clock tree network. Let WC_{clk} be the total wire capacitance of the clock network. Then the formulation of our problem is as follows:

$$\begin{aligned} &\text{minimize:} && WC_{clk} \\ &\text{subject to:} && \forall i \in [1,n] \quad Skew_i^{'} \leq Skew_i + \Delta_{margin} \end{aligned} \tag{1}$$

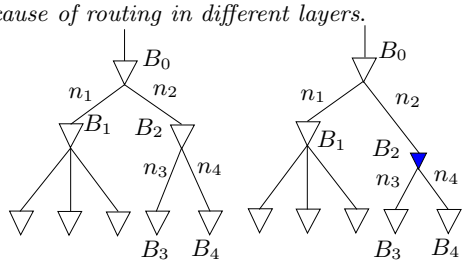
where, $Skew_i$ and $Skew_i^{'}$ are the skew of the i^{th} clock domain before and after the DRC-aware clock network modification respectively and Δ_{margin} is the margin allowed in the skew. By DRC-aware, we mean that the clock network modification is not allowed to introduce any new DRC violation, and in addition not to increase the clock latency. Note that we have used the terms net capacitance and wire capacitance interchangeably throughout the paper.

3. SKEW BOUNDED BUFFER TREE RESYNTHESIS

In this section, we describe our approach. At first, we present the key idea behind the incremental clock tree modification. Then we illustrate why and how the clock network is traversed in the bottom-up fashion. Next, we describe the algorithm for single buffer migration followed by mean-centric grid based placement mechanism to scale this approach.

3.1 Incremental Clock Tree Modification

The key idea of our approach is to size and move the clock buffers or inverters towards its loads. Consider Fig. 1 to illustrate the incremental clock tree modification. For instance, B_2 drives two other clock buffers B_3 and B_4, and B_2 is placed closer to B_3 and B_4 in Fig. 1(b), followed by blockage aware re-routing of the nets n_2, n_3 and n_4. The new placement of B_2 and the re-routing of the nets (n_2, n_3 and n_4) will impact the clock arrivals in the flip-flops which are at the transitive-fanout (TFO) cone of B_0. Since B_2 is placed closer to its loads, intuitively the wire-load for B_2 would decrease whereas the same for B_0 may increase. However, it ultimately depends on how the re-routing of those nets occurs in the placed design with existing routing. It should be stressed that *global wire-load minimization might not always happen due to the change in routing topology, but also because of routing in different layers.*



(a) Original clock tree (b) Moving and sizing B_2

Figure 1: Incremental clock tree modification

The benefit of this incremental clock tree modification can be two-fold: (i) it can reduce the total wire-capacitance of the clock nets, and (ii) if the size of B_2 is reduced, it will reduce the pin-capacitance as well. But this can increase the skew of the clock network, and can introduce new DRC violations. So we need to respect these two constraints while physically realizing the transformation.

3.2 Bottom-up Traversal

We adopt a bottom-up traversal of the clock network for performing the incremental modification in the clock tree. This is due to most of the total capacitance of the clock network is associated with the bottom-levels, *i.e.*, closer to the flip-flops. For instance, [9] has shown with one benchmark from ISPD'10 contest [8] that 71% and 12% of the total capacitance are respectively linked with the bottom-most level and second bottom-most level. For each clock domain the incremental modification is performed within the scope of a hyper-net. A hyper-net is a set of logically equivalent or opposite polarity nets which are separated by buffers or inverters in the same physical partition as the root driver of the top net [21]. Note that we do not perform the transformation on any clock-gate or clock-multiplexers, as such an action could worsen the timing pertaining to the enable signals. Fig. 2 shows three hyper-nets, and the traversal order for these hyper-nets can be $HyperNet_1 \rightarrow HyperNet_2 \rightarrow HyperNet_0$. The relative order between $HyperNet_1$ and $HyperNet_2$ is random. But a prioritized ordering can be imposed giving more precedence to the hyper-net with higher switching activity to facilitate more savings in dynamic power.

3.3 Single Buffer Migration (SBM)

Algorithm 1 presents the key steps of Single Buffer Migration (SBM). It works on a buffer or inverter (*cell*) in the clock network. First, the output net of *cell* is obtained, and the bounding box (*bbox*) of the net is calculated (Lines 2-3). At the next stage, we try to find the best position within *bbox* for *cell* for migration, such that the resultant wire-capacitance of the clock network is optimized. However, since this migration occurs after placement in post-

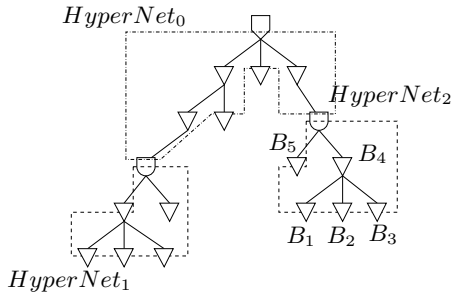


Figure 2: Bottom-up traversal within and across hyper-nets

CTS stage, *cell* can not be moved to any position in order to avoid the placement overlap with the already placed cells. So we find the set of congestion free positions (*positionSet*), and attempt to explore those positions (Lines 4-21). The detailed mechanism for finding these positions will be discussed in Section 3.4.

Algorithm 1 Single Buffer Migration

```
1: Procedure SBM(cell);
2: opnet ← output net of cell;
3: bbox ← bounding box of opnet;
4: positionSet ← congestion free positions within bbox;
5: for all pos ∈ positionSet do
6:    cellTypeSet ← available buffers or inverters in the library;
7:    for all cellType ∈ cellTypeSet do
8:       substitute the cell-type of cell with cellType;
9:       place cell in pos;
10:      re-route the nets connected to cell;
11:      if isDRCViolation then
12:         continue;
13:      end if
14:      if Skew' > Skew + Δ_margin then
15:         continue;
16:      end if
17:      if cost(currentSol) < cost(bestSol) then
18:         bestSol ← currentSol;
19:      end if
20:   end for
21: end for
22: return
23: end Procedure
```

For each available position, we also explore different sizing options for *cell*, followed by re-routing of the clock nets connected to *cell* (Line 10). For instance, if we consider the cell B_2 in Fig. 1, then n_2, n_3 and n_4 will be re-routed. If this migration with sizing and re-routing introduces any new DRC violation, then this move is not accepted (Lines 11-13). We also do not up-size the cells as it could increase the total pin-capacitance of the clock network. In addition, we check if the skew exceeds the pre-specified margin by the move (Lines 14-16). Among all these moves (with sizing), the solution giving the best improvement in wire-capacitance of the clock network is committed by backtracking mechanism (Lines 17-19). It should be stressed that *the algorithm has been integrated to an industrial tool and all these moves are followed by the re-routing of the clock nets along with routing layer assignment by the tool* to get the accurate skew measurement and the improvement in wire-capacitance of the clock network in a fast and incremental way.

However, it is still computationally intensive to explore all congestion free positions and all sizing options. So we propose mean-centric grid-based placement along with pruned sizing options to scale the approach.

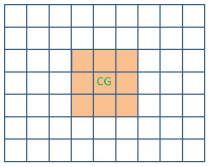


Figure 3: Mean-centric grid exploration with $gridParam = 1$

3.4 Mean-Centric Grid-Based Placement (MCGBM)

In modern designs, a clock net can drive tens of clock buffers or inverters. Although the CAD tools try to place those buffers in geometrical proximity, it may not be always possible, and thus the buffers driven by a net can be placed wide apart. As a result, the area of the bounding box of the clock net can be large, and attempting all congestion free positions in the bounding box can be run-time intensive. On the other hand, the sum of the euclidean distances from a point (x, y) to a set of n points, such as (x_1, y_1), (x_2, y_2)...(x_n, y_n) is given by $\sum_{i=1}^{n} \sqrt{(x-x_i)^2 + (y-y_i)^2}$ and minimized for the mean position, *i.e.*, $x = \frac{\sum_{i=1}^{n} x_i}{n}$, $y = \frac{\sum_{i=1}^{n} y_i}{n}$. So the neighborhood locations of mean position are preferred for capacitance cost optimization.

To move the targeted clock cell accordingly, we divide the bounding box (found in Algorithm 1) into grids of ($2h \times 2w$), where h and w are the height and width of the cell respectively. Note this grid-granularity is experimentally found to be suitable for solution quality and run-time. To spot the neighborhood of the mean position, we define a parameter called $gridParam$. Suppose the mean position of the net is in a certain grid gr. In that case, we will explore all the grids for which the grid distance in x and y co-ordinates from gr are both less or equal to $gridParam$. This is explained in Fig 3 representing the grid-based partitioning and the centre of gravity (CG) or mean position of the bounding box. For instance, if $gridParam = 1$, then all shaded grid positions in Fig 3 will be explored for placement.

Additionally, instead of trying with all down-sizing options for the targeted clock buffer or inverter (Line 7 in Algorithm 1), we explore the pre-sorted (according to size) library cells in decreasing order. Once we get a higher cost compared to earlier sizing options, we do not further try to down-size. Experimentally we have observed that this pruning technique does not practically incur any compromise in the solution quality, but saves run-time.

4. EXPERIMENTAL RESULTS

We have implemented the algorithms presented in this work in C++ and run it on a Linux machine with 16-Core 3GHz CPU and 256GB RAM. Table 1 presents the characteristics of 6 industrial designs using cutting-edge technology nodes (22-65nm) along with experimental results run on those designs. The designs are sorted according to the number of cells in the design (Column 3). Columns 2 and 4 represent the technology node and the total number of flip-flops in the designs respectively. Column 5 shows the maximum clock latency (maximum of rise latency and fall latency) in the clock networks of the respective designs. Design 'F' is the biggest design in terms of the total number of cells, whereas design 'D' has biggest clock network with around 136k flip-flops. Column 6 presents the ratio of the clock net capacitance (cap_{net}) to the clock pin ca-

Table 1: Design characteristics and clock capacitance/dynamic power reduction

Design	Technology (nm)	# of cells	# of flops	Clock latency (ps)	$\frac{cap_{net}}{cap_{pin}}$	cap_{net} imprv. (%)	cap_{pin} imprv. (%)	P_{dyn} reduction (%)	Run-time (min)
A	65	68,945	11,937	791	0.60:1	4.0	1.0	2.13	25.0
B	65	79,055	11,741	1573	0.66:1	4.7	1.6	2.83	22.4
C	28	128,444	18,967	1034	0.79:1	6.0	0.03	2.66	37.9
D	22	590,392	135,937	1227	2.26:1	3.3	0.4	2.41	245.0
E	28	687,221	34,399	738	2.60:1	9.6	0.84	7.17	56.7
F	28	859,833	37,239	1554	2.25:1	5.7	0.2	4.01	46.3
Average						5.55	0.52	3.54	

pacitance (cap_{pin}). Columns 7 and 8 respectively represent the percentage reduction in cap_{net} and cap_{pin} in the clock network of the designs. Considering the switching activities to be same across all hyper-nets in the clock network, the clock dynamic power (P_{dyn}) improvement is calculated as the weighted sum of the improvements in the net capacitance and the pin capacitance in Column 9. For instance, the improvement in cap_{net} and cap_{pin} for the design 'C' are 6.0% and 0.03% respectively, and the ratio of net to pin capacitance is 0.79:1. So the clock dynamic power improvement would be $6.0 \times \frac{0.79}{1.79} + 0.03 \times \frac{1}{1.79} = 2.66\%$. Column 10 shows the run-time of our approach in each design.

On average, our algorithm achieves respectively 5.55%, 0.52% and 3.54% improvement in cap_{net}, cap_{pin} and P_{dyn}. The pin-capacitance improvement is much smaller as downsizing the buffers to improve cap_{pin} typically (i) introduces maximum slew and maximum load violations and (ii) changes the clock arrivals to a greater extent than just by local move of the buffers, violating the constraints related to the skew and the clock latency. Due to the higher net-to-pin capacitance ratio, the power improvement is higher in the lower technology node, most being for the designs 'E' and 'F' with around 7.2% and 4.0% respectively. It should be stressed that these improvements are *over the baseline clock network synthesized and routed by an industrial tool*. The skew margin is kept at $20ps$. $gridParam$ is set to 10 for all these runs. To justify the selection of $gridParam$, we take the smaller design A and run our algorithm with $gridParam = 1, 2, 5, 10, 20$ and covering the entire bounding box for each of the nets. Fig. 4 shows the curve for run-time vs. percentage wire-cap reduction. The curve is initially monotonic with $gridParam$, but starts to flatten beyond $gridParam = 10$. The maximum run-time is around 4 hours for the design 'D'. This is because although it is not the biggest design, it has the biggest clock network in our benchmark suite.

5. CONCLUSION

To our best knowledge, this is the first work to optimize the clock net capacitance of already synthesized and routed clock networks by a buffer tree resynthesis algorithm. Our approach has been integrated into an industrial tool, and provided an average 5.6% and up to 9.6% improvement in clock net capacitance over the baseline clock networks of industrial designs synthesized and routed by the industrial tool. In terms of clock dynamic power reduction, our algorithm can reduce the clock dynamic power up to 7.2% and 3.5% on average. Since with technology scaling, the percentage contribution of net capacitance is becoming higher and higher in comparison to the pin capacitance, we believe that our approach can potentially reduce more dynamic power in modern designs.

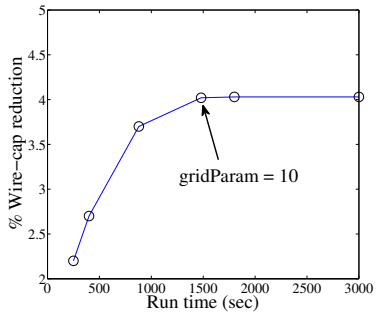


Figure 4: Percentage wire-cap improvement vs. run time trade-off for design A

6. REFERENCES

[1] L. F. Chao and H. M. Sha, "Retiming and clock skew for synchronous systems," *ISCAS*, pp. 283–86, 1994.

[2] V. Nawale and T. W. Chen, "Optimal useful clock skew scheduling in the presence of variations using robust ILP formulations," *ICCAD*, pp. 27–32, 2006.

[3] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–51, 1990.

[4] X. Liu *et al.*, "Maximizing performance by retiming and clock skew scheduling," *DAC*, pp. 231–36, 1999.

[5] R. Tsay, "Exact zero skew clock routing algorithm," *TCAD*, vol. 12, no. 2, pp. 242–249, 1993.

[6] K. D. Boese and A. B. Kahng, "Zero skew clock-routing trees with minimum wirelength," *ASIC Conference and Exhibit*, pp. 17–21, 1992.

[7] J. L. Tsai *et al.*, "Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing," *TCAD*, vol. 23, no. 4, pp. 565–572, 2004.

[8] C. Sze, "ISPD-2010 high performance clock network synthesis contest," *ISPD*, pp. 143–143, 2010.

[9] R. Ewetz and C. K. Koh, "Local merges for effective redundancy in clock networks," *ISPD*, pp. 162–67, 2013.

[10] X. Shih and Y. Chang, "Fast timing-model independent buffered clock-tree synthesis," *DAC*, pp. 80–85, 2010.

[11] X. Shih *et al.*, "Blockage-avoiding buffered clock tree synthesis for clock latency range and skew minimization," *ASPDAC*, pp. 395–400, 2010.

[12] C. Sze *et al.*, "ISPD-2009 clock network synthesis contest," *ISPD*, pp. 149–50, 2009.

[13] Y. M. Lee *et al.*, "Simultaneous buffer-sizing and wire-sizing for clock trees based on lagrangian relaxation," *VLSI Design*, pp. 587–594, 2002.

[14] A. Kahng, S. Kang, and H. Lee, "Smart non-default routing for clock power reduction," *DAC*, pp. 1–7, 2013.

[15] A. H. Farrahi *et al.*, "Activity-driven clock design," *TCAD*, pp. 705–714, 2001.

[16] S. Wimer and I. Koren, "Design flow for flip-flop grouping in data-driven clock gating," *TVLSI*, pp. 771–78, 2014.

[17] R. Visweshwara *et al.*, "Placement aware clock gate cloning and redistribution methodology," *ISQED*, pp. 432–436, 2012.

[18] S. Lo *et al.*, "Power optimization for clock network with clock gate cloning and flip-flop merging," *ISPD*, pp. 77–84, 2014.

[19] V. G. Oklobdzija *et al.*, *Digital System Clocking: High-Performance and Low-Power Aspects*. Wiley, 2003.

[20] D. Liu and C. Svensson, "Power consumtion estimation in CMOS VLSI chips," *JSSC*, pp. 663–70, 1994.

[21] S. Roy *et al.*, "Clock tree resynthesis for multi-corner multi-mode timing closure," *ISPD*, pp. 69–76, 2014.