System Level Clock Tree Synthesis for Power Optimization

Saif Ali Butt, Stefan Schmermbeck, Jurij Rosenthal, Alexander Pratsch, Eike Schmidt
ChipVision Design Systems AG
Oldenburg, Germany

Abstract

The clock tree is the interconnect net on Systems-on-Chip (SoCs) with the heaviest load and consumes up to 40% of the overall power budget. Substantial savings of the overall power dissipations are possible by optimizing the clock tree. Although these savings are already relevant at system-level, only little effort has been made to consider the clock tree at higher levels of abstraction. This paper shows how the clock-tree can be integrated into system-level power estimation and optimization. A clock tree routing algorithm is chosen, adapted to the system-level and then integrated into an algorithmic-level power optimization tool. Experimental results demonstrate the importance of the clock tree for system-level power optimization.

1 Introduction

The rising complexity of modern Systems-on-Chip (SoCs) is a challenge for electronic design automation (EDA). A possible solution to handle the complexity is to raise the level of abstraction for the design and optimization. At higher abstraction levels, better optimizations of performance and power can be achieved by applying algorithmic level transformations.

With shrinking technology sizes of 65nm and below, the impact of interconnects is growing as their delay no longer scales with the gate delay. That is, the impact of interconnects must be included in optimizations at higher levels of abstraction. Interconnect-aware design techniques [13] have been introduced to meet this challenge.

The clock tree is the set of interconnect nets on an SoC with the heaviest load [9]. The routing of the clock tree is performed in a way such that the clock signal is distributed to all registers on dependent data paths in a synchronous fashion. That means no clock skew or a skew within tolerable bounds that still guarantees correct operation of the logical units is allowed.

The problem of clock tree routing has gained much interest in research [2–5, 11] as the clock tree not only greatly influences the system performance, but it also consumes up to 40% of the overall energy of SoCs and microprocessors [7]. The power dissipation of the clock tree is dominated by the switched capacitances, which are influenced by the overall length of the clock distribution network.

This has led to clock tree routing algorithms which reduce the overall length of the clock tree and this way the delay on the net by allowing a properly chosen clock skew [3, 11]. Considerable power savings can be obtained by using these so-called bounded skew clock tree routing algorithms as its length can be reduced by up to 50% compared to zero skew clock trees [10].

The high power dissipation of the clock tree and the high potential for optimization imply that the design of the clock tree should already be included into design and optimization flows at system-level. If combined with further techniques, like clock gating [17], substantial power savings become possible [8]. A major problem for the application of state-of-the-art clock tree routing algorithms for system-level design is that the routing algorithms require gate-level information which is not yet available at system-level.

The contribution of this paper is to include the clock tree into the cost function of system-level design. The focus of our analysis is on power optimization. First, different clock tree routing algorithms and their applicability to system-level design are investigated. Second, it is shown how a gate-level clock tree routing algorithm can be adapted to the system-level. Third, the integration of such a modified algorithm into an algorithmic-level power estimation and optimization tool is presented. Finally, algorithmic level transformations applied to a Fast Fourier Transform (FFT) demonstrate the impact of the clock tree for the system-level power optimization process.

The remainder of the paper is organized as follows: In Section 2, different state-of-the-art clock tree synthesis algorithms, their applicability to system-level design and the integration into a system-level design tool are described.

978-3-9810801-2-4/DATE07 © 2007 EDAA

Experimental results which show the accuracy of the presented approach and potentials for power savings are presented in Section 3. The paper closes with the conclusions and an outlook to future work.

2 System Level Clock Tree Synthesis

2.1 Problem Description

If a circuit is designed or optimized at system-level, an algorithmic, sequential description is used as design entry point. This algorithm will be mapped to a hardware which is synchronized by a set of clocked registers during the later synthesis. Various hardware mappings of the same algorithm are possible which lead to different numbers and topologies of registers. The optimum clock tree depends on the number as well as on the distribution of registers. That is, the problem of finding the optimum hardware mapping therefore needs to take the clock tree topology into account. This implies that state-of-the-art routing algorithms should be considered for system-level design.

Despite that, only little research has been performed to include these approaches into analysis and optimization at higher levels of abstraction. Instead, buffered tree structures [8] or H-trees [14] are used and the geometry information is only considered based on statistical information, thus neglecting the topology.

The reason for this is that detailed layout information is required as input to the clock tree routing algorithms. The following requirements must be met to apply a clock-tree estimation at system-level:

1. **Position of the clock sinks**

   The clocked registers are the clock sinks of a design. They are the leaves of the clock tree and therefore the input of the clock tree routing algorithm. The positions of the registers is usually not determined at system-level, so that an extension is required.

2. **Technology dependent parameters**

   For a power estimation of the clock tree, technology dependent parameters are required. For example, per-unit resistance and capacitance and the fan-in capacitances of the registers are required to determine the overall switched capacitance of the clock tree.

3. **Speed issues**

   If the clock tree routing problem is applied to the system-level, speed issues must be taken into account. At system-level the clock-tree routing will probably be integrated into an iterative process, so that the time budget is much slower than at gate level synthesis.

4. **Choice of a clock tree routing algorithm**

   A suitable low-power, state-of-the-art clock tree routing algorithm must be chosen. It must be possible to modify the algorithm such that it can be utilized in system-level design. It must either be abstracted from the post-layout level or the unknown parts must be estimated.

5. **Choice of a delay model**

   The question of measuring the skew on a given node on the tree is closely connected to the delay model. In our implementation, we have applied the Elmore delay model because of its better accuracy compared to the path-length delay model. For a detailed comparison of the two models please refer to Cong et al. [3].

6. **Integration**

   Finally, the algorithm must be integrated into the flow of a system-level power estimation tool.

Therefore, different Clock Tree Synthesis (CTS) algorithms and their applicability to the system-level are investigated in the next section and a suitable candidate is chosen. After that it is shown in Section 2.3 how this algorithm will be applied to system-level in order to meet the requirements above and to fit into the flow of a system-level power estimation and optimization tool.

2.2 Comparison of Different CTS Algorithms and their Application to the System-Level

In this section, it is evaluated whether different well known clock tree routing algorithms can be applied for system-level power optimization.

2.2.1 Zero Skew CTS

In order to ensure a synchronous clock distribution among the chip area, symmetric H-trees are used which guarantee equal distances from the clock source to all registers on the chip. This way, no clock skew is caused by unequal interconnect lengths. On the downside, the zero-skew constraint of H-trees leads to a larger overall clock tree length and thus to a larger delay and larger switched capacitances.

Liu and Svensson [14] already applied an H-tree routing algorithm for system-level power estimation. In their approach, the capacitance of the clock tree was estimated based on the chip dimensions and on statistical data. The results can easily be incorporated in a system-level power analysis, but the floorplanning information is ignored which reduces accuracy.

2.2.2 Bounded Skew CTS

If the constraint on the clock skew is relaxed within well controlled borders, correctly working circuits with shorter clock tree lengths compared to H-trees become possible. This will in turn lead to smaller overall capacitances and thus to lower power dissipation. For most applications, a bounded clock skew is tolerable [11].

First attempts of bounded skew routing algorithms [10] were based on the Deferred Merge Embedding (DME) algorithms which were originally derived for zero skew constraints. DME identifies the possible loci for placing internal nodes of the topology tree (i.e. the branching points) in a bottom up phase, before fixing the placement in a top down phase. Edahiro [5] introduced on-the-fly topology tree construction into DME. While the loci form segments for zero skew, they form polygons ("merging regions") for bounded skew.

Cong et al. [3] presented a comprehensive comparison of different bounded skew routing tree (BST) algorithms. They showed that while in principle interior points of merging regions should be considered (IME), restricting the solutions space to the boundaries (BME) saves significant computation time with negligible degradation of results.

2.2.3 Clock Scheduling

A clock skew can furthermore be exploited by deliberately introducing a positive or negative skew in distinct paths on an SoC. Clock skew is positive or negative, depending on whether the clock signal of the final register of a sequential data path leads or lags the clock signal of the initial register [9]. That is, the total delay on that data path is either enlarged or decreased by the clock skew. This way, the delays in a circuit can be balanced individually which enables higher clock frequencies [16].

A drawback of the first clock scheduling algorithms was their higher power consumption due to an increased length of the clock net. If the positioning of registers with equal clock timing is optimized, the length of the clock net can be reduced again, so that high speed and low power clock trees can be achieved [6].

2.2.4 Discussion

As stated above, the clock tree routing algorithm shall be integrated into a system-level power estimation and optimization flow. Zero skew routing algorithms do not fit into this profile as they are no longer state-of-the-art, neither in literature, nor in current design tools. Furthermore, a bounded clock skew is tolerable for most applications, so that the larger clock tree lengths that are required to fulfill the zero skew constraint are not acceptable in terms of power dissipation.

Other routing algorithms, like the bounded skew clock tree algorithm or clock scheduling approaches permit better results in terms of both performance and power dissipation. For this reason, pure zero skew algorithms will not be used for our approach. Note that a zero skew condition can be modeled as a special case of the bounded skew algorithm if the skew bound is set to zero.

Clock scheduling algorithms offer the greatest potential for optimizations of the clock tree, because single data paths are treated individually. Despite this advantage, they are not a suitable candidate for system-level clock tree synthesis, because the delays of single data paths are still unknown at system level.

This means that bounded skew algorithms are best suited for a system-level clock tree estimation. First, the clock tree length can be reduced compared to H-trees which leads to better results in terms of power. Second, bounded skew clock tree routing algorithms are utilized in state-of-the-art design tools, like Cadence SoC Encounter. That is, they are suitable candidates with practical relevance for system-level clock tree synthesis.

2.3 Applying Clock Tree Synthesis at System-Level

Our approach was implemented in the high level power estimation tool ORINOCO [1], which maps each sequential algorithm to a possible RT level power optimized hardware solution to help designers find a power-aware architecture for their system. In order to get a realistic estimation of the power consumption, the design is floorplanned by placing and reshaping the RT components (registers, adders, multipliers etc.). This floorplan has been verified to be a good estimation of the final floorplan [15]. Our CTS algorithm takes this floorplan as an input and generates flip flop locations (named as set of sinks S in the following description) from the register locations by assuming an equal distribution of flip flops within the register boundaries.

2.3.1 CTS Algorithm

For a given set of sinks S, the goal of any bounded skew clock tree synthesis algorithm is to calculate a clock tree $T(S)$, which embeds all sinks and meets the skew bound condition for all possible paths from root to each sink. Our algorithm is a slightly modified form of the greedy BST/DME algorithm suggested by Cong et al. [3]. The basic idea is to iteratively merge subtrees, physically represented by merging regions, to form a single tree of merging regions. The algorithm is described in an abstract form in Algorithm 1 and Function 1 in pseudo-code. As an initialization of the algorithm, it takes all the sink locations $l(s)$ and for each sink, it defines a point-merging-region

$mr(s)$ representing a subtree, which consists of the sink only $[Algorithm1, lines3-5]$. After that, nearest neighbor pairs of all available merging regions are formed iteratively $[Algorithm1, lines6-19]$ and in every iteration, the first one-third of the closest pairs are merged together $[Function1]$ to form new merging regions. The new merging region represents a set of all points, where the root of the two subtrees can be placed without violating the skew bound condition. Each time a pair is merged, it reduces the total count of merging regions by 1. Therefore, after some iterations, only 1 merging region is left, where the root of the whole tree $T(S)$ can be placed.

Each time a new merging region is formed, the capacitance of the subtree represented by that merging region is calculated and saved as a property of the merging region. Further, this capacitance value is compared with a technology dependent threshold and a buffer is inserted into the tree when that threshold is reached, which ensures acceptable transition times for the subtree it drives. On every buffer insertion, dynamic and cell power values are incremented for the values consumed by the new buffer.

When only one merging region is left after iteratively merging the nearest pairs of the merging regions, the iteration loop is terminated and the power is reported after inserting the root buffer.

2.3.2 Clock-Tree Power Model

On every buffer insertion, power values are incremented according to the following model. The average clock tree power P_{clk} is determined as the sum of the power of all buffers. The buffer power is the sum of the static buffer power, the cell power and the power that is required for driving the capacitances at the output of the buffer:

$$P_{clk} = \sum_{Buffers} (P_0 \cdot k_i + P_{cell} + \frac{1}{2} \cdot C_{on} \cdot f_{on} \cdot V_{DD}^2) \quad (1)$$

Where P_0 is the static power dissipated by a minimum size buffer, k_i is the the size of the i-th buffer, P_{cell} is the cell power, C_{on} is the capacitance that is driven by the buffer, f_{on} is the frequency of signal transitions and V_{DD}^2 is the operating voltage.

3 Experimental Results

In this section the presented clock power estimation approach will be validated. First, the accuracy evaluated by comparison to layout level. Second, the relevance of the proposed concept is demonstrated through a design example.

Data: Register sizes and position in the floorplan, skew bound B, technology parameters
Result: Dynamic and cell power consumed in the clock tree embedding all sinks $T(S)$

```
1  Generate set of sinks S from register sizes and locations;
2  n = |S|;
3  forall sinks s ∈ S do
4  |  mr(s) = {l(s)} /* location of sink s */
5  end
6  while n > 1 do
7  |  Construct nearest neighbor graph H;
8  |  A = sorted edges of H in non-decreasing order of edges weight;
9  |  for i = 1 to min{max(1, n/3), n − 1} do
10 |  |  Take edge E_uv with smallest weight from A;
11 |  |  Delete all edges incident to u or v from A;
12 |  |  mr(w) = Merge(mr(u), mr(v));
13 |  |  Cap(w) = Cap(u) + Cap(v) + c * d(mr(u), mr(v));
14 |  |  /* d(a, b) being the distance between a and b */
15 |  |  Insert buffer if required and increment power;
16 |  |  n = (n − 1);
17 |  |  /* one less subtree */
18 |  end
19 end
20 Insert root buffer and report power;
```

Algorithm 1: Bounded skew clock power estimator

Input: Merging regions $mr(a), mr(b)$
Output: Resulting merging region $mr(w)$

```
1  Make shortest distance region
   P = SDR(mr(a), mr(b)) =
   {p|d(p, mr(a)) + d(p, mr(b)) == d(mr(a), mr(b))}
2  Calculate Q, the set of all points for which the skew condition is met:
3  Q = {q|skew(q) ≤ B}
4  if P ∩ Q = ∅ then
5  |  Insert detour routing, such that P ∩ Q ≠ ∅;
6  end
7  return ( mr(w) = P ∩ Q );
```

Function1 : Merge($mr(a), mr(b)$)

3.1 Evaluation of the CTS Algorithm

The evaluation was performed as follows: The benchmark designs were simulated and estimated with ORINOCO. The ORINOCO placement information was subsequently read into Cadence SoC Encounter where a CTS run was executed. The same constraint on the register

placement was put on the system-level clock-tree algorithm which was then executed as explained in Section 2. Finally, the clock tree power was compared for both variants.

The ORINOCO floorplan was forced onto Encounter and onto the system-level algorithm to compare the clock trees without the influence of floorplanning. An example for an ORINOCO RT-level floorplan can be seen in Fig. 1. The light boxes indicate the size and the positions of the registers of the design. The equivalent in Encounter is shown in Fig. 2. The assumption of uniform distribution of the flip-flops seems justified. Note that while Encounter also performs bounded skew CTS, it does not always fully exploit the skew bound. That is, the maximum skew that actually occurs in the clock tree from SoC Encounter must be used as skew bound for the system-level clock tree algorithm to obtain comparable results.

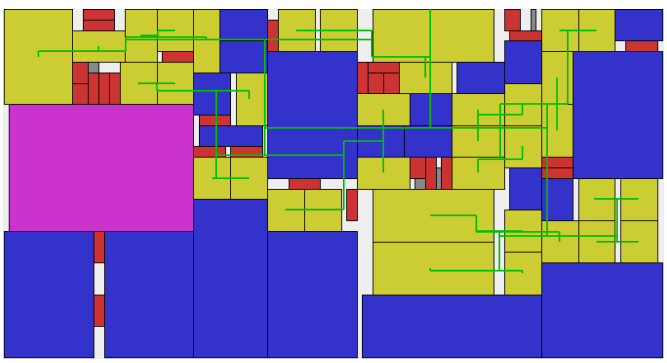

Figure 1. Floorplan extracted from ORINOCO. The registers are marked as light boxes.

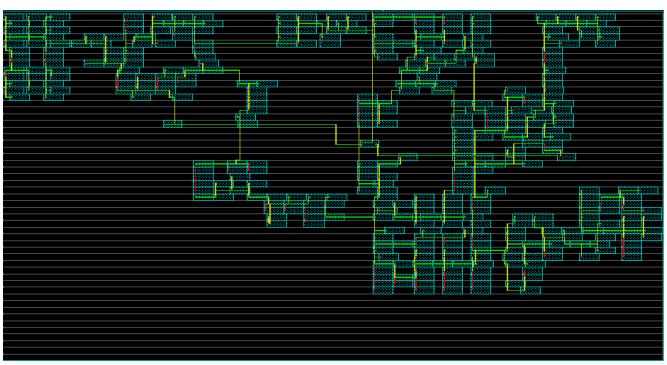

Figure 2. Clock tree extracted from Cadence SoC Encounter

The clock tree generated in our approach for this example is shown in Figure 3 (flip-flops visible as dots). It can be seen that a very similar clock tree structure is obtained. The overall clock tree length deviates from the results of SoC Encounter by only 10%.

The accuracy of the proposed algorithm was analyzed for the following benchmark programs: A JPEG compression algorithm, a wavelet transform and a radix-2 Fast Fourier Transform (FFT). The FFT is evaluated in two variants: The first one uses a shift register as delay element, while the second one uses a ring buffer instead.

The results were obtained for a 90 nm technology and a supply voltage of $V_{DD} = 1V$. For the clock tree model, a constant transition time of 25 ps was assumed. The skew bound B for SoC Encounter was set to 100 ps, but the actual maximum skew was always lower and is given in Table 1 below. That is, the skew bound for the system-level clock tree algorithm was also set to the value from the table.

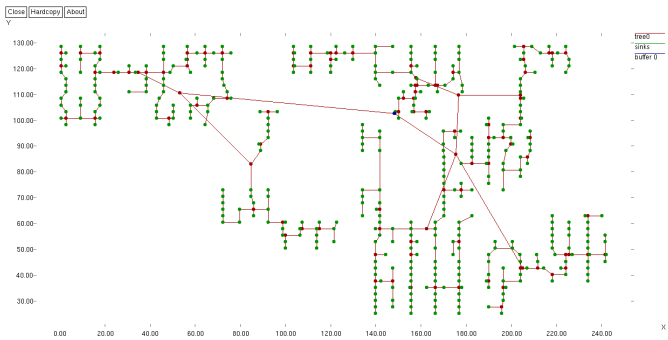

Figure 3. Clock tree determined by the proposed, Cong-based system-level algorithm

Design	Clock MHz	B ps	Power Enc.	Power prop. alg.	Diff. %
JPEG	200	28	165 μW	187.5 μW	+ 6.7
Wavelet	200	12	75.5 μW	73.66 μW	-2.4
FFT Shiftregister	102.4	48	1.5 mW	1.18 mW	- 21.3
FFT Ringbuffer	102.4	24	311.4 μW	322 μW	+ 3.4

Table 1. The table shows the clock tree power from Cadence SoC Encounter and from the proposed system-level algorithm for different benchmark designs.

The results show a good accuracy of the proposed system-level algorithm with a deviation from the SoC Encounter results between 2.4 % and 21.3 %.

3.2 Impact of the Clock-Tree on System-Level Optimizations

In this section, the impact of the clock tree for system-level low-power optimizations is shown for a radix-2 module of a 128-point mixed radix Fast Fourier Transform (FFT) processor [12]. One of the major consumers of energy will be the registers required to store values across

clock boundaries. ORINOCO provides the information about which registers have to be updated with new values given the schedule of the design. In particular, the registers for the delay feedback are consuming most of the design energy. Two different implementations of the delay feedback registers are compared in this analysis. The first version uses a shift register, while the second version of the FFT uses a ring buffer for the delay element.

Each variant leads to different numbers of registers in the design which is reflected in the power results (Table 1). The results obtained with SoC Encounter show that the clock tree power of the FFT with the shift register as delay element is almost five times higher than that of the variant with a ring buffer. This makes FFT with ring buffer an obvious choice for the low-power design. This demonstrates the high impact of the clock tree for the power aware system architecture.

4 Conclusion and Outlook

This paper proposes including the clock tree into the cost function of system-level low-power design. A power-optimal, state-of-the-art clock tree routing algorithm was chosen and then adapted to fit into the flow of an algorithmic-level power estimation and optimization tool. Comparisons with results obtained from Cadence SoC Encounter show only small average deviations of less than 10 %. To show the importance of the clock tree at system-level, two different implementations of a delay element in an FFT were compared. It was shown that a change of the delay element of the FFT architecture might increase the clock tree power by a factor of five. Future work will furthermore include clock gating in the analysis. This way, an even higher potential for system-level power optimization will be accessible.

References

[1] ChipVision Design Systems AG. http://www.chipvision.com, 2006. Oldenburg, Germany.

[2] R. Chaturvedi and J. Hu. Buffered clock tree for high quality ic design. In *5th International Symposium on Quality Electronic Design*, 2004.

[3] 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*, 3(3), July 1998.

[4] D.E. Duarte, N. Vijaykrishnan, and M.J. Irwin. Formulation and validation of an energy dissipation model for the clock generation circuitry and distribution networks. In *Proceedings of the International Conference on VLSI Design*, 2001.

[5] M. Edahiro. An efficient zero-skew routing algorithm. In *Proceeding of the 31st ACM/IEEE Design Automation Conference*, 1994.

[6] K. Kurokawa et al. A high-speed and low-power clock tree synthesis by dynamic clock scheduling. *IEICE Transactions on Fundamentals*, E85-A(12), December 2002.

[7] P.E. Gronowski et al. High performance microprocessor design. *IEEE Journal on Solid State Circuits*, 33, May 1998.

[8] T. Yamada et al. Low-power design of 90nm superh processor core. In *ICCD Dig. Tech. Papers*, 2005.

[9] E.G. Friedman. Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE*, 89(5), May 2001.

[10] D.J.H. Huang, A.B. Kahng, and C.W.A. Tsao. On the bounded-skew clock and steiner routing problems. In *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, 1995.

[11] A.B. Kahng and G. Robbins. *On Optimal Interconnections for VLSI*. Kluwer, 1994.

[12] Y.W. Lin, H.Y. Liu, and C.Y. Lee. A 1-gs/s fft/ifft processor for uwb applications. *IEEE Journal of Solid State Circuits*, 40(8), August 2005.

[13] Z. Lin and N.K. Jha. Interconnect-aware low power high-level synthesis. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 24(3), March 2005.

[14] D Liu and C. Svensson. Power consumption estimation in CMOS VLSI chips. In *IEEE Journal of Solid-State Circuits*, volume 29, no. 6, pages 663–670, june 1994.

[15] A. Stammermann, D. Helms, M. Schulte, A. Schulz, and W. Nebel. Binding, allocation and floorplanning in low-power high-level snythesis. In *Pro. Int. Conf. Computer Aided Design*, November 2003.

[16] A. Takahashi, K. Inoue, and Y. Kaijtani. Clock-tree routing realizing a clock-schedule for semi-synchronous circuits. In *IEEE/ACM International Conference on Computer Aided Design*, 1997.

[17] Q. Wu, M. Pedram, and M.X. Wu. Clock-gating and its application to low power design of sequential circuits. *IEEE Transactions on Circuits and Systems I*, 47(3), March 2000.