An Automated Approach for Minimum Jitter Buffered H-Tree Construction

Ayan Mandal†, Nikhil Jayakumar*, Kalyana Bollapalli+, Sunil P Khatri† and Rabi N Mahapatra†
† Department of Electrical & Computer Engineering, Texas A&M University, College Station TX 77843.
* Juniper Networks Sunnyvale, CA 94089. + NVIDIA Corp, Santa Clara, CA 95050.

***Abstract*— In recent fabrication technologies, buffered clock distribution networks have become increasingly popular due to increasing on-chip wiring delays. Traditionally, clock distribution networks has been optimized to minimize end-to-end skew of the distribution network. However, since most ICs have an on-chip PLL, we argue that the design goal of minimizing end-to-end jitter is more relevant. In this paper, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. Our cost functions are a weighted sum of power and jitter, and a weighted sum of power and end-to-end delay of the distribution network. Our approach is based on pre-characterizing the delay, jitter and power of buffered segments of different lengths, topologies, buffer sizes and wire-codes. Using this information, a dynamic programming (DP) engine automatically generates the optimal H-tree that minimizes the appropriate cost function. Compared to a manually constructed buffered H-tree network, our approaches are able to reduce both jitter (by as much as 28%, and power by as much as 46%. When optimizing for minimum jitter, the DP engine generates a H-tree with lower jitter than when optimizing for minimum delay, thereby validating our approach, and proving its usefulness.**

I. Introduction

In recent VLSI fabrication processes, with decreasing feature sizes, wire widths have been shrinking. The direct consequence of this is an increase in wire resistance (R_w), since $R_w = \frac{\rho \cdot L}{W \cdot T}$, where ρ is the resistivity of the wire material, and L, W and T are the length, width and thickness of the wire respectively. Since wiring delays on a die are proportional to the RC time constant of the wires, this increase in wire resistance results in increased wiring delays, especially for long wires on a chip (which do not scale with technology). The classical approach is to partially compensate for the reduction of W by increasing T, but this results in wires which are thin and tall, leading to an increased cross-coupling capacitance among wires, thereby again resulting in an increased RC time constant. The design of the wiring stack (i.e. the height and minimum width of wires on each metal layer of the IC) is therefore a complex problem. Despite significant research in academia as well as industry, it remains true that wiring delays in VLSI are on the rise [1], and this problem is particularly acute for global signals which need to be driven at high frequencies with tight slew-rate constraints.

The global clock signals on an IC are significantly impacted by this problem. The clock signal is operated at high frequencies, and in addition to having stringent slew-rate constraints, the clock signal has very tight jitter constraints. Consider the rising (falling) edge of the clock. Over a number of cycles, this edge may appear early or late. Jitter is difference between the latest arrival time and the earliest arrival time of a clock signal edge. The clock period of the IC is determined by adding the worst-case clock jitter value to the longest delay between any two sequentially adjacent flip-flops in the circuit. The longest delay between two sequentially adjacent flip-flops is computed as the sum of the longest combinational delay of the circuit plus the setup time and the clock-to-output delay of the flip-flops of the design. As a result, a high clock jitter results in a reduced frequency of operation of the IC, and hence it is important to keep jitter to a minimum.

Given the timing critical nature of the clock signal, it is very important to distribute this signal carefully on an IC, to ensure high slew-rates, high frequency and low jitter. Over the years, several clock distribution methodologies have been developed, such as the H-Tree, mesh and star, among others [2], [3]. The H-tree is a popular clock distribution approach, due to its simplicity and low power and area requirements. In this paper, we focus our attention on an H-tree based clock distribution network.

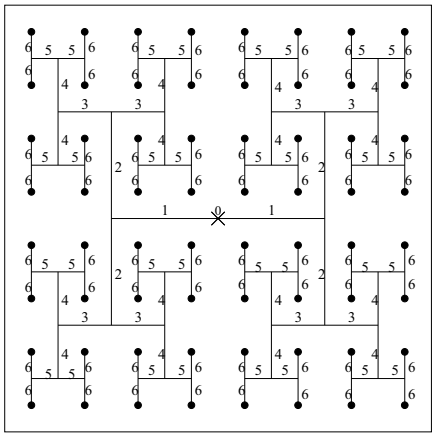


Fig. 1. Buffered H-tree clock distribution network with 6 levels

The structure of a 6-level H-tree clock distribution network is shown in Figure 1. The numbers on each of the branches of the H-tree indicate the level of that branch. The × notation in the center indicates the source of the H-tree. There are $2^6 = 64$ sinks (end-points) in the H-tree, indicated by dots. Note that the end-points of the H-tree distribution network are uniformly spaced, and cover the entire IC area. The lengths of the i^{th} branch of the H-tree is given by $L_i = \frac{S}{2^{\lceil \frac{i}{2} \rceil + 1}}$, where S is the dimension of any side of the (square) die. For example, for a chip which is 20mm wide, the lengths of the branches of a 6-level H-tree are 5mm, 5mm, 2.5mm, 2.5mm, 1.25mm and 1.25mm, for levels 1 through 6 respectively.

The unbuffered H-tree network by itself is a zero-skew balanced network (if process and temperature variations are not considered). A traditional H-tree is designed assuming that the clock driver at the center of the H-tree is large enough to drive the entire clock tree. Wire widths and driver sizes are fixed to make sure that the clock signal can drive the local clock regenerators at the leaves of the H-tree for the required frequency of operation, with a sufficiently high slew-rate. The optimal (with respect to having a high slew-rate clock signal and also in terms of reducing the clock distribution delay) wire sizing methodology dictates that we utilize wider wires near the center

1063-9667/11 $26.00 © 2011 IEEE
DOI 10.1109/VLSID.2011.69

of the H-tree, and narrower wires as we get closer to the leaves. This is referred to as a *tapered* wire sizing approach.

Note that with increasing wiring delays in recent technologies, it is no longer feasible to utilize unbuffered H-trees for ICs of reasonable size (S $> \sim$ 5mm). For this reason, buffered H-trees are commonly utilized in today's IC designs. The buffers of the H-tree are not shown in Figure 1, and can be at arbitrary locations along any path from the source to a sink, provided the locations and sizes of each buffers is the same for all 64 branches of the H-tree. Typically, when an H-tree is manually designed (which is the common practice), designers typically place identical buffers are regular intervals along the H-tree, so as to simplify the design process. Also, in a manually designed H-tree the wire topologies are typically fixed from source to all sinks. For example, a common practice is to place buffers at each branching site in the H-tree. Note that with a buffered H-tree, the tapered wire sizing (commonly used in unbuffered H-trees) is no longer applicable, and the wire widths of each level of the H-tree are typically dependent on the size of the driving buffer.

The buffered H-tree alleviates the wiring delay problem in recent technologies, and enables a designer to deliver a high frequency, high slew-rate clock at all the sinks, with low end-to-end skew. However, the disadvantage of such a buffered H-tree is an increase in the jitter at the sinks. In practice, there are cycle-to-cycle variations in the VDD value across the die. On one hand, the wire segments in a buffered H-tree do not exhibit delay variations as a consequence of the cycle-to-cycle VDD variations along the die, since the delay of a wire depends solely on the RC parasitics of the wire. However, the cycle-to-cycle variations in the VDD value across the die affect the delay of the buffers in a buffered H-tree, since the delay of an inverter depends on its supply voltage. Hence, in practice, the different buffers in a buffered H-tree experience different delays on a cycle-to-cycle basis. This results in an increased jitter at the sinks of the buffered H-tree. This can reduce the maximum operating frequency of the IC, since the operating frequency has to be guard-banded to account for worst-case jitter, as discussed earlier.

Typically a buffered H-tree network is designed to minimize the end-to-end skew of the tree. However, most modern ICs have an on-chip Phase Locked Loop (PLL) to synchronize the off-chip clock (typically generated by a crystal oscillator) to the on-chip clock. As a consequence, the end-to-end skew of the H-tree (or any other clock distribution network) is not of consequence. Rather, our position is that the H-tree should be designed for minimum jitter. This is the key thesis of this paper. We show that designing a H-tree for minimum end-to-end skew (delay) can yield different results compared to designing the H-tree for minimum jitter.

Our approach automates the design of the H-tree, thereby availing significant optimization flexibility that is not practical to leverage in a manual H-tree design process. In particular, we select the best H-tree by exploring various a) wiring configurations (referred to as *wire-codes* in the sequel), b) buffer sizes c) segment lengths and d) segment topologies. Any given wiring segment in our approach has a total of 2745 choices for its implementation. The automatic H-tree synthesis procedure utilized in this paper is based on dynamic programming (DP). First, we statically characterize a large number of ways of implementing a segment, by computing their power, segment delay and segment jitter and output slew, as a function of input slew. This is done for all 2745 segment choices up-front. Then a dynamic programming engine selects the lowest cost H-tree implementation that satisfies constraints such as output slew limits and cycle-to-cycle "wake-up" jitter. Our approach is implemented to minimize one of two cost functions – a weighted sum of power and jitter, and a weighted sum of power and delay. The resulting H-trees are simulated in HSPICE, and the end-to-end delay, power and end-to-end jitter estimated by our DP engine matches the HSPICE results with a maximum error of 4.6%. We also compare our DP based buffered H-tree synthesis with a manually designed H-tree. Results demonstrate that our approach, when run with jitter minimization as its goal, produces H-trees with strictly better jitter (by as much as 28%) and power (by as much as 46%) compared to a manually designed H-tree. Also, it achieves better jitter performance than the DP based approach run with delay minimization as its goal.

The key contributions of this paper are:

- We argue that buffered H-trees should be designed to minimize jitter (as opposed to minimizing end-to-end skew which is the typical approach). We demonstrate that synthesizing an H-tree for minimal end-to-end skew does not necessarily minimize jitter, underscoring the importance of the above observation.
- We present a DP based automated approach to synthesize a buffered H-tree. This approach simultaneously explores a large number of wiring wire-codes, buffer sizes, segment lengths and segment topologies.
- With the cost function of minimum jitter, our approach produces an H-tree which has up to 28% lower jitter than a manually designed H-tree. The power as well as the end-to-end skew of our approach is better by up to 46% and 16.6% respectively.
- Our DP engine can produce H-trees with a cost functions of a weighted sum of power and jitter, and a weighted sum of power and delay.

The rest of the paper is organized as follows. Section II describes previous approaches in this area. Section III describes our approach, while Section IV describes the results of experiments which we performed to validate our approach. In Section V, we draw conclusions and discuss avenues for future work.

II. Previous Work

Several clock distribution methods (such as the H-tree, Meshes, Stars etc) have been studied in the past. Dome excellent survey style papers in this area is [2], [3]. Most of previous works on clock network design [4], [5], [6] attempt to obtain zero skew, while others try to bound the clock skew within a given hard constraint [7]. Some clock synthesis works generate unbuffered clock networks that require a separate buffer insertion stage using conventional or specialized buffer insertion algorithms. In other works [8], [9], buffer insertion and clock tree routing are integrated. The Deferred-Merge Embedding (DME) algorithm [10] is the fundamental technique used in many later clock tree works. It consists of a bottom-up stage and a top-down stage. A tree of merge segments, which represent the possible locations of merge nodes, is constructed in the bottom-up stage. Once all the merge segments are constructed, the exact locations of merge nodes are determined in the top-down stage. Merge

segments are calculated to balance the delays of the two sub-trees. A key difference between DME and our approach is that DME is used for clock tree synthesis for ASICs, unlike the H-tree approach of our work which is more pertinent to custom designs. In this work we intend to develop an optimal H-Tree that will minimize the appropriate cost function by exploring a large number of wiring wire-codes, buffer sizes, and segment length and segment topologies. This automatic tool gives the flexibility to the designer to optimize delay, power and or jitter across the clock tree.

III. Approach

Traditionally, buffered H-trees have been designed manually, with the goal of minimizing end-to-end skew. Since the end-to-end skew is unimportant in an IC which has a PLL, and since the buffers in the H-tree introduce jitter at the clock sinks, the goal of this paper is to automatically synthesize a buffered H-tree. The primary goal of minimizing the end-to-end jitter of the synthesized H-tree, while the secondary goal is to minimize power as well.

In the remainder of this section, we will discuss the design scenario for this effort, along with the electrical constraints which we impose on the resulting synthesized buffered H-tree. We end with a discussion of our dynamic programming (DP) based automated buffered H-tree synthesis approach.

We conduct our experimentation on a 6 level buffered H-tree, implemented in a 45 nm PTM [11] fabrication process. The H-tree is implemented on METAL9, and we assume that 10 metal layers are available. A variety of buffers are used. Each buffer consists of two inverters, with the driving inverters of size 32× to 51264× of a minimum sized inverter, in increments of 32×. The other inverter is sized 3× smaller than the driving inverter.

A total of 6 wiring configurations (referred to as wire-codes) are available for any buffered segment of the H-tree. There is no restriction on the number of wire-codes utilized for the synthesized H-tree. These wire-codes are referred to as WC1 through WC6, and their details are described next. Each wire-code consists of a METAL8 wire, shielded on either side as well as above and below by grounded wires, as shown in Figure 2.

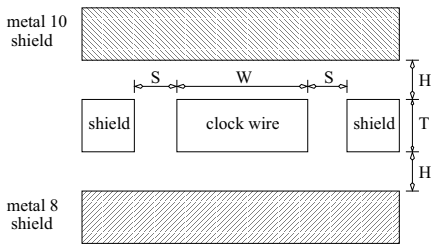


Fig. 2. Cross section of Wires in any H-tree Segment

Table I describes the details of the 6 wire-codes, along with their parasitic resistance and capacitance values, which are extracted using Raphael [12]. All resistive values are reported for an operating temperature of 100∘C, to model the worst case wiring delay condition. For all wire-codes, we assumed $T = 0.9$ μm, H = 1.4 μm and a dielectric constant $\kappa = 2.5$.

A variety of segments are selectable during the automatic synthesis of the buffered H-tree using our approach. These segments fall into three categories, as shown in Figure 3. These

Wire-code	$W(\mu)$	$S(\mu)$	Res. (Ω/mm)	Cap. (F/mm)
WC1	0.45	0.45	69.78	1.68E-13
WC2	0.45	0.9	69.78	1.08E-13
WC3	0.9	0.45	34.89	1.84E-13
WC4	0.9	0.9	34.89	1.23E-13
WC5	1.35	0.45	23.27	1.98E-13
WC6	1.35	0.9	23.27	1.38E-13

TABLE I
Wire-codes and their Characteristics

segments are a) a straight segment b) an I segment and c) a T segment. The total length of these segments is referred to as x. The straight portion of a T segment has a length y. For all segments, $x \in [1mm, 2.5mm]$, with increments of $0.25mm$. For a T segment, $y \in [0.25, x - 0.5mm]$, with increments of $0.25mm$.

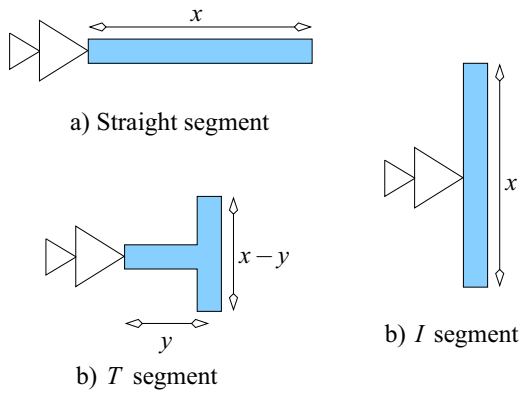


Fig. 3. Types of Wire Segments used For Buffered H-Tree Construction

As discussed earlier, jitter in a buffered H-tree occurs due to cycle-to-cycle local variations of VDD across the die, which result in relative changes in the clock buffer delays. We next discuss our methodology for computing jitter. For our 45 nm process, the supply voltage is 1.1V. The maximum variation in the VDD value is assumed to be 10% of VDD, or 110 mV. This variation is due to various sources, including variations in the voltage regulator output, IR drops in the power distribution network, etc. Of this 110 mV, about 70% is attributed to cycle-to-cycle variations, and 30% constitutes long term variations [13]. Therefore the cycle-to-cycle variation of the supply voltage is 77 mV. For purposes of jitter measurement, we therefore consider jitter to be the difference in delay of the clock buffer for VDD values of 990 mV and 1067 mV (i.e. 990 mV + 77 mV).

The jitter measurement methodology is shown in Figure 4. Consider the clock buffer operating at 1067 mV. Assume that the solid thick line is the input to the buffer, and the dashed thick line is its output. Therefore its propagation delay at 1067 mV is T_{pd}^{1067} as marked. Now consider the clock buffer operating at 990 mV. Assume that the solid thin line is the input to the buffer, and the dashed thin line is its output. Therefore its propagation delay at 990 mV is T_{pd}^{990} as marked. The jitter of the clock buffer is therefore $J = T_{pd}^{990} - T_{pd}^{1067}$.

Note that the input to the clock buffer operating at 1067 mV and 990 mV in our experiments could be driven in one of two ways - with the same slew-rate (Method A) or with the same transition time from rail to rail (Method B). Figure 4 shows the

input of the clock buffer being driven using Method B. We use this method based on experiments in which we drove a long chain of identical H-tree segments with inputs of both kinds. We noticed that the output after the signals traverse 5 H-tree segments closely match Method B, which is why we utilize this method to drive the clock buffer inputs when measuring jitter.

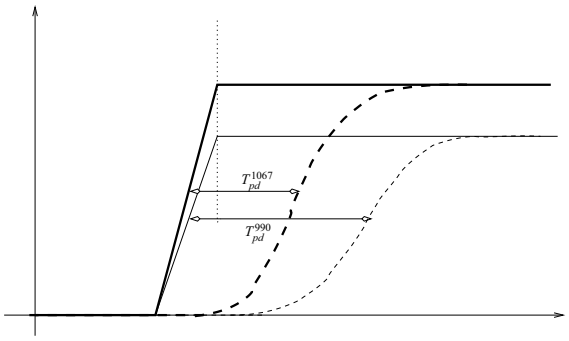


Fig. 4. Measuring Jitter

We additionally invoke several constraints on the wire segments utilized by our buffered H-tree synthesis algorithm

- First, if any segment being considered by the synthesis algorithm has a output slew greater than 150 ps, we reject that segment. The output slew is computed as the 10% to 90% (or 90% to 10%) transition time for the output of the segment, divided by 0.8. This constraint ensures that the clock signal at the sinks has a high slew-rate which is essential for a clock signal, and also results in reduced crow-bar currents and hence lower power.
- We also ensure the cycle-to-cycle "wake-up" jitter is less than 1ps. Figure 5 illustrates this idea. Modern ICs frequently need to design in a requirement to be able to put the part to sleep, in order to save power. In such a case, the clock signal is held static during the sleep mode. After the IC wakes up, the pulse widths of the first two clock pulses must be tightly controlled for correct functionality. The wake-up jitter is defined as $|T_2 - T_1|$. We constrain each segment to have a wake-up jitter of less than 1ps. All segments which fail this requirement are rejected. This requirement essentially requires that each segment be able to drive its clock signal to rail values during normal operation of the IC. If this is not true for any segment, such a segment will fail the wake-up jitter test.

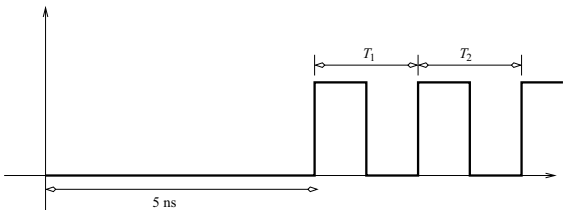


Fig. 5. Wake-up Jitter Experiment

Our DP based algorithm for automated synthesis of the buffered H-tree is described next. We begin by first characterizing (using HSPICE [14]) each possible segment configuration, which consists of the different wire-codes, buffer sizes, segment lengths and segment types described earlier in this section. There are a total of 2745 unique segment configurations available to the DP algorithm. For each such segment configuration, we drive it with varying input slews (ranging from 20 ps to 300 ps, with 1 ps increments). We record the jitter, output slew, delay (skew) and the power of the segment configuration as a function of its input slew. This characterization data is now used by the DP algorithm. We implement two cost functions for the DP algorithm. Suppose that the cost is being computed for a segment configuration i which builds upon a partial solution with total jitter J, total delay D and total power P. The segment configuration i has jitter J_i, delay D_i and power P_i.

- A weighted sum of power and jitter. This cost function is expressed as
  $C_{PJ} = \alpha\frac{(P+P_i)}{P^*} + (1-\alpha)\frac{(J+J_i)}{J^*}$
  Note that $P+P_i$ and $J+J_i$ are the total power and total jitter after segment i is appended to the partial solution. Also note that P^* and J^* are the average power and jitter values respectively, for all candidate segment configurations. Since the scale of power and jitter are different, these values are introduced in order to equalize the contribution of power and jitter to the cost function.
- A weighted sum of power and delay (or slew). This cost function is expressed as
  $C_{PD} = \alpha\frac{(P+P_i)}{P^*} + (1-\alpha)\frac{(D+D_i)}{D^*}$
  Note that $P+P_i$ and $D+D_i$ are the total power and total delay after segment i is appended to the partial solution. Also note that P^* and D^* are the average power and delay values respectively, for all candidate segment configurations. Since the scale of power and delay are different, these values are introduced in order to equalize the contribution of power and delay to the cost function.

Suppose we have a optimal solution to the buffered H-tree construction problem S. Clearly, this solution consists of making a choice of the last segment s (closest to the sink) to be selected. Hence the solution to the buffered H-tree construction problem from the source of the H-tree to the input of segment s must also be optimal (otherwise we contradict the optimality of S). Since the cost functions of jitter, delay and power are additive, an optimal solution to the buffered H-tree construction at any location n can be constructed from the optimal solution of the buffered H-tree construction problem for every location m which is downstream from n (i.e. m is closer to the source of the H-tree than n). Hence the criterion for optimality of DP is satisfied and as a result, DP can yield an optimal solution (for the cost functions of delay, jitter or power) to the buffered H-tree construction problem.

The DP based algorithm selects the lowest cost buffered H-tree implementation that satisfies constraints such as output slew limits and cycle-to-cycle wake-up jitter. The resulting H-trees are simulated in HSPICE, and the end-to-end delay, power and end-to-end jitter estimated by our DP engine are compared with the corresponding values obtained via HSPICE.

IV. Experimental Results

We implemented our DP algorithm in the C programming language. We assumed that the H-tree is implemented in a 45 nm PTM [11] technology, with $VDD = 1.1V$. All the segment configurations were pre-characterized using HSPICE [14]. We used the *accurate* option in HSPICE. We set the parameter

delmax set to 1 ps, for maximum accuracy, which ensures that the maximum timestep that HSPICE takes is limited to 1 ps. We observed that the slew rate at the end of each segment was substantially constant for a given segment configuration, independent of the slew rate at the input to the segment.

In order to compare our results with a manually generated buffered H-tree, we consulted an industrial clock tree designer and constructed a minimum delay H-tree based on their input. This H-tree utilized the wire-code with the lowest RC delays (WC-2), with the first 6 segments of length 2.5mm, and the last 2 segments of length 1.25mm. 512× buffers used were used exclusively to minimize the delay. The DP algorithms' results were compared with the manually generated H-tree, and the comparison is presented next.

We split the experiments into three sets. In the first set, the maximum buffer size allowed was limited to 512×, while in the second and third sets, up to 256× and 128× buffers were allowed respectively. Figure 6 reports the results for these three sets of experiments. In this figure, the first row of figures corresponds to the first set of experiments (with up to 512× buffers), while the second and third rows present plots for the second and third sets of experiments. In each row, the left plot presents power results, the middle plot presents delay results, and the right plot presents jitter results. For each plot, the x-axis corresponds to α, while the y-axis corresponds to the quantity expressed by the plot. Also, for each plot, the DP results are presented for both cost functions described in Section III. The results of the manual buffered H-tree are reported in each plot as well.

From these plots, we note that for all 3 experiments, the lowest jitter of the buffered H-tree returned by the DP engine is strictly lower than that of the manually synthesized design, by as much as 28%. This indicates that our DP based buffered H-tree synthesis technique is of value. Further, since our approach is able to select segment configurations freely, it is able to produce strictly better values of power compared to the manual approach, for *all* values of α. The C_{PJ} cost function yields lower jitter values compared to the C_{PD} cost function, especially for $\alpha > 0.5$. For values of $\alpha < 0.5$, C_{PJ} achieves better results than C_{PD}, but the improvement is lower. However, for power-constrained designs, it may be desirable to use $\alpha > 0.5$. The delays of the DP based approaches are higher than the manual design for $\alpha > 0.5$, but this is not of importance for PLL based designs, as we have mentioned earlier.

To validate the accuracy of our DP engine, we simulated the buffered H-tree returned by our DP engine (for the cost functions of minimum jitter, minimum power and minimum delay) in HSPICE. The results were compared for the DP engine running with a maximum allowed buffer size of 128×, 256×, and 512× respectively. Table II reports the results of this comparison. The delay, jitter and power estimated by our DP engine are reported in Columns 3, 4 and 5 respectively. Columns 6, 7 and 8 report the error in the DP estimate of delay, jitter and power respectively, compared to the HSPICE value. Note that the maximum error in any of these estimates is 4.6%. The DP's estimate of these three quantities was always lower than the value returned by HSPICE.

The segments selected by the DP engine (for a maximum allowable buffer size of 512×) are reported in Table III and IV (for the cost function of minimum jitter and minimum delay respectively). Segments with lower indices (Column 1) are closer to the H-tree source. For each segment, we report the segment index, wire-code (Column 2), segment length in mm (Column 3), segment style (one of Straight, *T*, and *I*) (Column 4) and buffer size (Column 5). Note that the segments for Table III and IV are quite different, indicating that the minimum delay buffered H-tree is not the same as minimum jitter buffered H-tree.

Seg. #	Wire-code	Length (mm)	Seg. type	Buf. size
1	1	1	I	512×
2	2	2	S	512×
3	2	2	S	512×
4	1	1	I	512×
5	2	2	S	512×
6	2	2	S	512×
7	1	1	I	512×
8	2	1.5	S	512×
9	1	1	I	512×
10	2	1.5	S	512×
11	2	1.25	I	512×
12	2	1.25	I	512×

TABLE III
SEGMENTS SELECTED BY OUR DP ALGORITHM (MIN. JITTER, 512× MAX

Seg. #	Wire-code	Length (mm)	Seg. type	Buf. size
1	1	1	I	512×
2	2	1.75	S	512×
3	2	2.25	S	512×
4	1	1	I	512×
5	2	1.75	S	512×
6	2	2.25	S	512×
7	1	1	I	512×
8	2	1.5	S	512×
9	1	1	I	512×
10	2	1.5	S	512×
11	2	1.25	I	512×
12	2	1.25	I	512×

TABLE IV
SEGMENTS SELECTED BY OUR DP ALGORITHM (MIN. DELAY, 512× MAX

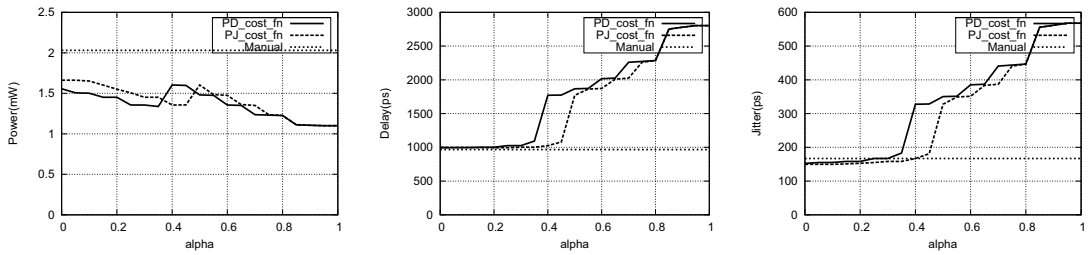
V. CONCLUSIONS

Buffered clock distribution networks have become increasingly popular due to increasing on-chip wiring delays. Since clock buffers are liable to add jitter in the clock signal on account of cycle-to-cycle *VDD* variations in the die, we argue that the design goal of minimizing end-to-end jitter is more relevant for buffered H-tree synthesis than minimizing end-to-end slew (which is irrelevant for PLL based designs). In this paper, we present a dynamic programming based approach to synthesize a minimum cost buffered H-tree clock distribution network. Our cost functions are a weighted sum of power and jitter, and a weighted sum of power and end-to-end delay of the distribution network. After pre-characterizing the delay, jitter and power of buffered segments (2745 in all) of different lengths, topologies, buffer sizes and wire-codes. Using this information, a dynamic programming (DP) engine automatically generates the optimal H-tree that minimizes the appropriate cost function. Compared to a manually constructed buffered H-tree network, our approaches are able to reduce both jitter (by as much as 28%, and power by as much as 46%. When optimizing for minimum jitter, the DP engine generates a H-tree with lower jitter than when optimizing for minimum delay, thereby validating our approach, and proving its utility.

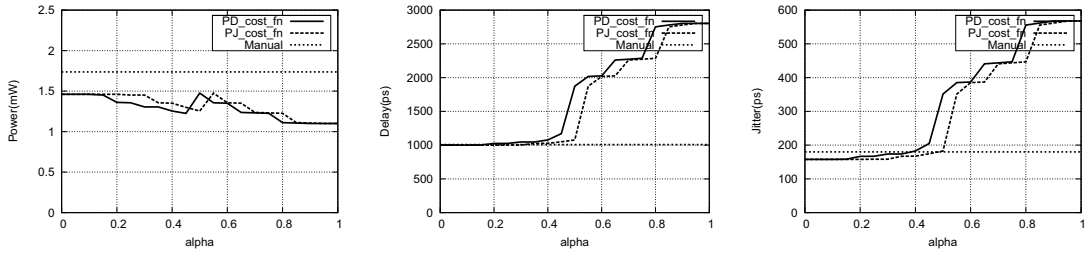
REFERENCES

[1] "The International Technology Roadmap for Semiconductors." `http://public.itrs.net/`, 2003.

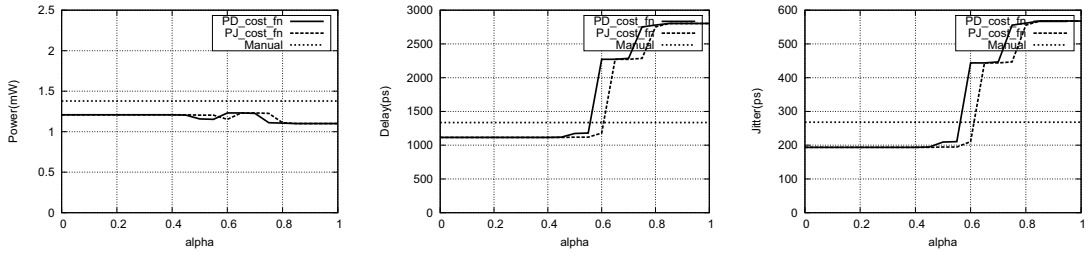
[2] E. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, pp. 665 –692, may 2001.



(a) Power (512× max buffer size) (b) Delay (512× max buffer size) (c) Jitter (512× max buffer size)



(d) Power (256× max buffer size) (e) Delay (256× max buffer size) (f) Jitter (256× max buffer size)



(g) Power (128× max buffer size) (h) Delay (128× max buffer size) (i) Jitter (128× max buffer size)

Fig. 6. DP Results (Jitter, Power, Delay) Compared with Manually Designed H-tree

		DP Estimate			Difference versus HSPICE (%)		
	Cost Func.	Delay (ps)	Jitter (ps)	Power (mW)	Delay (ps)	Jitter (ps)	Power (mW)
Up to 128×	Min. Delay	1114.7	193.5	1.21	0.59	1.88	4.58
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	1114.7	193.5	1.21	0.59	1.88	4.58
Up to 256×	Min. Delay	1000.1	157.7	1.46	0.80	2.23	4.58
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	1000.1	157.7	1.46	0.80	2.23	4.58
Up to 512×	Min. Delay	994.2	152.0	1.55	0.89	2.69	2.52
	Min. Power	1334.9	247.6	1.08	0.57	1.94	2.44
	Min. Jitter	999.3	149.0	1.66	1.11	2.17	2.92

TABLE II
COMPARISON OF DP RESULTS WITH HSPICE

[3] F. Anceau, "A synchronous approach for clocking vlsi systems," *Solid-State Circuits, IEEE Journal of*, vol. 17, pp. 51 – 56, feb 1982.

[4] Y. Chen and D. F. Wong, "An algorithm for zero-skew clock tree routing with buffer insertion," *Design and Test of Computers*, pp. 230–236, 1996.

[5] I. Liu, T. Chou, A. Aziz, and D. F. Wong, "Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion," in *Proceedings, Intl Symposium on Physical Design*, pp. 33–38, 2000.

[6] J.-L. Tsai, T.-H. Chen, and C. Chen, "Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing," *IEEE Transactions on CAD*, vol. 23, pp. 565–572, 2004.

[7] J. Cong *et al.*, "Bounded-skew clock and Steiner routing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 3, pp. 341–388, 1998.

[8] A. Rajaram and D. Pan, "Variation tolerant buffered clock network synthesis with cross links," *Proceedings, Intl Symposium on Physical Design*, pp. 157–164, 2006.

[9] R. Chaturvedi and J. Hu, "Buffered clock tree for high quality ic design," in *Proceedings, International Symposium on Quality Electronic Design*, pp. 381–386, 2004.

[10] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II*, vol. 39, pp. 779–814, Nov 1992.

[11] "PTM website." `http://www.eas.asu.edu/~ptm`.

[12] "Raphael Interconnect Analysis Tool: User's Guide." Paul Kostek, `http://www.todaysengineer.org/2008/Jun/semiconductor.asp`.

[13] W. for Blind Review, Jul 2010.

[14] I. Meta-Software, "HSPICE user's manual," Campbell, CA.