Skew-Bounded Low Swing Clock Tree Optimization

Can Sitik
Electrical and Computer Engineering
Drexel University
Philadelphia, PA, 19104 USA
E-mail: as3577@drexel.edu

Baris Taskin
Electrical and Computer Engineering
Drexel University
Philadelphia, PA, 19104 USA
E-mail: taskin@coe.drexel.edu

ABSTRACT

This paper introduces a methodology that optimizes the performance of a low swing clock tree under a skew bound. Low-swing clock trees are preferred for a reduction in the clock switching power, with an expected trade-off in clock slew and skew. In this paper, a heuristic optimization process is introduced that keeps the clock skew under the same skew budget of the originating full-swing clock tree. In this low swing clock optimization, the low power consumption property is preserved. The effect of slew on the logic timing, which is naturally degraded due to low-swing operation, is analyzed within timing slack of some paths in order to highlight the effectiveness of the low swing clock trees in lowering power consumption with limited impact on timing constraints. The experiments performed with the 4 largest ISCAS'89 benchmark circuits operating at 500 MHz, 90 nm technology and 4 different V_{dd} levels show that the optimized low swing clock tree can achieve an average of upto 11.0% reduction in the power consumption with no more than a skew degradation of 0.5% of the clock period (i.e. within the practical skew budget).

Categories and Subject Descriptors

B.7.1 [**Hardware**]: INTEGRATED CIRCUITS—*Types and Design Styles*

General Terms

Design

Keywords

VLSI, clocking, low-power, low-swing

1. INTRODUCTION

High performance designs, such as high end microprocessors, target tight performance constraints with minimum overhead whereas low-power ASIC designs target tight power constraints with maximum performance [1–4]. Clock distribution networks constitute an important part of the IC design due to their direct effect on the performance and the power consumption. Operating the clock network with low swing is one of the techniques that is explored in order to reduce the power consumption attributed to the clock network (of a microprocessor or an ASIC) [5–9].

Low-swing operation can be adopted at varying levels of a clock tree with different implications. Low swing operation on the entire clock tree without any modifications at the sink level would lead to negative impacts on clock slew, skew and logic path timing. Thus, a previous approach considers to use level shifters at the last level of the clock tree, assuming the overhead of level shifters, in order to keep the signal integrity for logic path timing at the sink level [6]. Another approach considers to use special kind of D flip-flops or buffers designed to work with low swing clock signals, proposing to use custom cells for lower power [7, 9]. All alternative approaches of low swing clock trees yield promising results. However low-swing applicability remains limited in practice due to a number of factors (based on adopted approach) including (*i*) degradation in the skew performance, (*ii*) degradation in expected power reduction, (*iii*) degradation in data timing due to slew degradation, (*iv*) necessitating level shifters of varying sizes, (*v*) necessitating low-swing FF designs. The approach in this work keeps the method highly practical for short-term applicability and integration with standard industrial-strength automation tools and design libraries. To this end, the optimization method in this work is proposed in the conversion of a full-swing clock tree built by an industrial-strength clock tree synthesis (CTS) routine to low swing clocking. The method does not utilize low-swing FFs, or depend on the existence of explicitly sized level shifters, in order to remain compliant with standard cell libraries, addressing factor (*iv*) and factor (*v*) above. The proposed optimization method also eliminates the degradation in skew performance [factor (*i*)], preserves the expected power reduction of low-swing operation [factor (*ii*)] and satisfies the data path timing requirements by controlling clock slew [factor (*iii*)]. More importantly, the optimization routine is built into a novel automation tool within the industrial-tool flow, that is the first automated routine to synthesize low swing clock network with practical slew and skew budgets.

The proposed methodology is not a straight-forward procedure due to the following challenges: 1) The effect of each upsizing/downsizing must be characterized as standard cell library models are available only at certain levels (1.2V, 0.8V etc.), 2) The slew at the clock pins of register sinks must be analyzed, because slew definition must be modified to include the effect of low swing driver buffers, 3) The change in skew and slew must be considered at all process corners for variation-awareness. The proposed methodology starts with a synthesized full-swing tree, follows a library characterization stage in order to analyze the effect of upsizing and downsizing of buffers in the given standard cell library, and finally minimizes its skew overhead by in-place upsizing/downsizing of clock buffers depending on the V_{dd} level selected and the target skew budget. To that end, this methodology can target any point on the power vs. skew trade off curve depending on the application, assuming presence of another power supply for the low swing voltage level. The methodology provides the desirable output of a low swing clock network with power savings obtained through lower V_{dd}, with (almost) the same performance of a full-swing network with a low run time, implemented as a new low-swing-CTS tool.

The rest of the paper is organized as follows. The preliminaries of the low swing clock networks are introduced in Section 2. The methodology of this work is explained in detail in Section 3. The results of the performed experiments on benchmark circuits are presented in Section 4. The paper is finalized with concluding remarks in Section 5.

2. PRELIMINARIES

In this section, the preliminaries of the low-swing clock networks are briefed. The change in the insertion delay and the power consumption depending on the V_{dd} level of the clock tree is presented in Section 2.1. The effect of lower V_{dd} and higher slew of the clock signal on the register-to-register logic path timing path is analyzed in Section 2.2.

2.1 Insertion Delay and Power in Low Swing Clock Trees

The delay and the power consumption of a clock tree depend on the supply voltage and the load capacitance including the capacitance at the sinks. Simple first order and higher order models are known that estimate the gate delay and the power consumption using the voltage and capacitance information [10, 11]. A dedicated model is needed for low-swing operation, particularly for the topology target in this paper: A low swing clock tree driving (e.g. register) sinks operating with full swing data input/output but with low swing clock input. To that end, SPICE-accurate simulations are used in this step in order to profile the delay and the power characteristics of low swing clock trees. A sample circuit, s35932 of ISCAS'89 benchmarks, is used to observe the change in the insertion delay at each sink and the total power consumption of the design with the change in the supply voltage on the clock tree at SPICE accuracy.

The profile of insertion delay at each sink vs. supply voltage is shown in Figure 1. As expected, the insertion delay at each sink increases with decreasing supply voltage due to the RC effect of the physical buffers and wires on the clock tree. The insertion delays form a monotonic behaviour at every individual sink. However, it is observed that each sink scales differently with the scaled supply voltage, which increases the difference between the maximum and the minimum insertion delay, which is also shown in Figure 2. Thus, the clock skew increases with decreasing voltage driving the (buffers of the) clock tree. This key observation leads to the fact that it is essential to monitor the skew when a low swing clock tree is used as the clock skew may not satisfy the target skew bound of the design at the low swing clock operation.

The power consumption, on the other hand, is expected to decrease with decreasing supply voltage. However, this decrease in power consumption is only until a point due to the worsening of the power profiles of the D flip-flops of the register sinks, shown in Figure 3. Although the power consumption of the clock tree scales quadratically with the supply voltage, driving D flip-flops with low swing buffer drivers increases the power consumption at the driven register sinks whose datapath is at full swing. The reason of this increase is the effect of slower switching of internal transistors of the

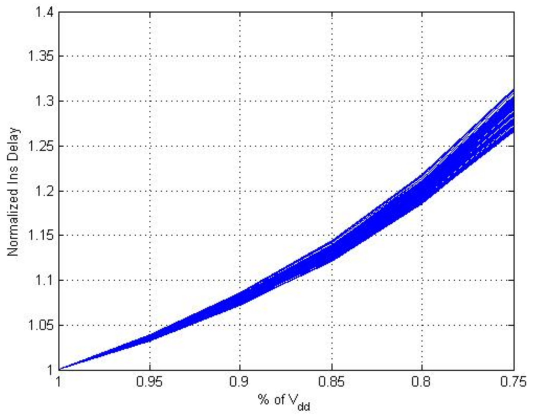

Figure 1: Normalized insertion delay of all sinks in the design, interpolated using 5 different low swing values

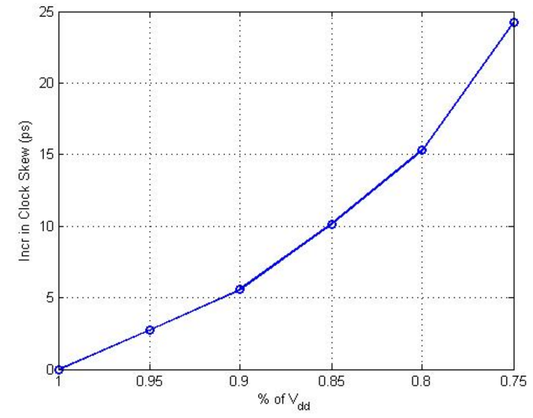

Figure 2: The increase in skew, interpolated using 5 different low swing values

D flip-flops (higher short circuit power). Thus, this increase outweighs the savings obtained at the clock tree after the 80% point. This observation implies that scaling clock tree voltage swing is not advantageous after a certain point.

2.2 Register-to-register Timing in Low Swing Clock Trees

In the standard IC Design flow, clock slew is constrained to a practical value for timing and power performance. In terms of timing performance, for instance, slew directly impacts the clock-to-output delay of the sink sequential-element (e.g. register). With this (worst-case) known impact on clock-to-output delay, a slew bound permits more accurate analysis of register-to-register timing early in the design cycle.

At low-swing clocking, the slew definition from traditional full-swing operation needs to be revised in order to define its equivalent low-swing operation form. The equivalence is defined as the identical effect on the register-to-register timing of the logic paths at full swing. Then, sufficient bounds are set for low swing voltage levels with the minimum degradation of timing slack. When a low swing clock tree drives register sinks operating at full *data* swing, the definition of slew as the time elapsed between the 10% and the

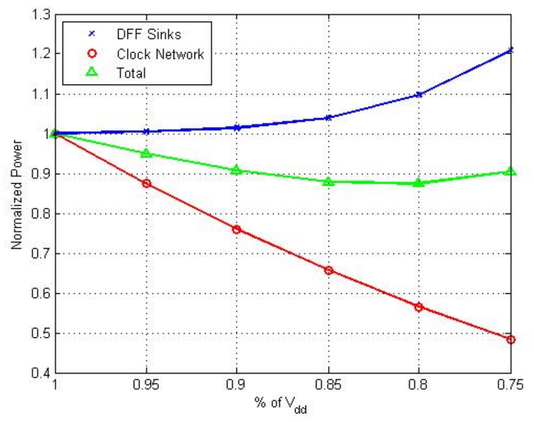

Figure 3: Normalized power for DFF sink registers, clock tree and the total power consumption, interpolated with 5 different low swing values

90% point of the *clock* signal is meaningless, because the voltage swing will likely not reach at 90% of full swing. To that end, a low swing slew must be defined considering its effect on the clock-to-output delay, in order to constraint the optimization stage with a new meaningful slew. In this work, low swing slew is defined as the time elapsed when a signal switches from 10% point to 50% point, as the voltage swing can not be scaled down to 50% of its nominal value, which is the switching point of the DFF sinks operating at full scale. It is argued in this work that this low-swing slew should be bounded so as to provide the equivalence to the original full-swing slew bound that provides similar (or constrained) clock-to-output delays at the sinks.

While the proposed definition for the equivalent low swing clock slew is sufficient to present the proposed low-swing clock optimization methodology, a complete empirical verification of the methodology necessitates standard cell library support for timing, which does not readily exist. The necessitated support is the timing data for standard cells at varying voltage levels of low swing (e.g. 0.85V, 0.9V, etc.) which typically do not exist in libraries: Instead gates are characterized only at supported voltage nodes, e.g. 0.8 V for low voltage and 1.2 V for high voltage. The static timing analysis tools are not applicable to measure the timing slack (i.e. impact of clock slew) due to the lack of these circuit models at low swing in the standard cell libraries. To fill this gap in empirical verification, 20 random paths, which are created using the available types of DFFs in the selected target library and logic cells, are analyzed at 4 different voltage levels (95%, 90%, 85%, 80% levels of the nominal V_{dd}). This analysis is performed in order to observe the relation between the low swing slew and the local timing (voltage levels beyond 80% point are excluded because they are shown to be disadvantageous in Section 2.1). Naturally, these 20 random paths do not theoretically guarantee timing but are empirically demonstrative of the effectiveness of the low-swing slew bounding methodology. It is also important to note that, this change in the clock signal only changes the clock-to-output delay of the D flip-flop, and the delay on the rest of the local path (that contributes to slack) is affected by less than 1 ps. Therefore, it is sufficient to analyze the effect of low swing slew on the clock-to-output delay of different types of DFFs in the library (there are four different DFFs in the selected SAED Library [12] of *Synopsys*, therefore 20 random paths are sufficient), independent of the logic path in the design. The

Table 1: Low swing slew values, and the effect of a low swing clock supply on local timing. The decreased slack, induced by increased clock-to-output delay, is traded off for the power savings of low swing clocks

	Low Swing Slew (ps)	Avg. Decrease in Slack (ps)	Max Decrease in Slack (ps)
V_{dd}	79.65	-	-
$0.95 \times V_{dd}$	83.83	1.26	1.78
$0.90 \times V_{dd}$	88.51	2.94	3.39
$0.85 \times V_{dd}$	93.74	5.08	5.58
$0.80 \times V_{dd}$	99.54	7.84	8.76

additional cell characterization study of standard cells, in order to develop timing models at varying voltage levels of low swing, is not presented here or used in experiments, aiming not to divert the discussion from low swing clock tree design. Without lack of generality, these studies can be performed or cell libraries with such information can be used instead, in a practical application of proposed novelties in low swing clock design.

For demonstration purposes, 200 ps is selected as the slew, which is a typical value (10% of the clock period at 500 MHz operation). The local path (logic and the DFFs) is supplied by full swing and the clock signal is applied with 4 different voltage levels of low swing, with 200 ps slew. As the low swing slew is defined as the transition between the 10% and 50% of the full swing (not the applied low swing) voltage, it is expected to have larger low swing values, and their effect on the local timing must be investigated in order to have a clear understanding of the parameters traded off against each other. The low swing slew values, the average increase and the worst case increase in the clock-to-output delay of 20 random paths at 4 voltage levels with 200 ps slew are shown in Table 1.

The measured low swing slew values, shown in Table 1 are set as the new constraints on the design, as the effect on the logical timing is limited (as low as 8.76 ps, 0.439% of the clock period, at the worst case). It is shown in Figure 3 that the power savings of the design are proportional to the scaled voltage level until 80% point, therefore it is shown here that the savings in the power consumption are successfully traded off with an insignicifant degradation in the logic timing (slack).

3. METHODOLOGY

In Section 2, it is shown that significant power improvements can be obtained through low swing clocking with a degradation in clock skew (Section 2.1) and an insignicant degradation in logic timing due to low swing slew (Section 2.2). Therefore, an optimization that minimizes the skew overhead of low swing clocks without degrading the power savings and the slew can obtain clock trees that perform (almost) the same as full swing clocks with significant power savings. To address that, a new methodology that optimizes low swing clock trees generated from a full-swing clock tree without any placement or clock topology changes, is proposed in this paper. The methodology only involves in-place buffer sizing, aiming not to alter the logic placement or full-swing clock tree topology for minimal disturbance of legacy designs. The proposed methodology has 3 mains steps:

1. Synthesizing an initial clock tree at full swing,

2. Characterizing the buffers of the library,

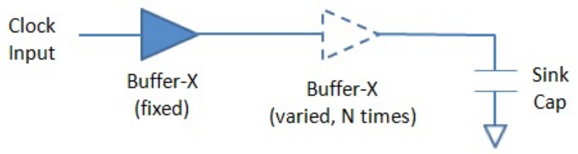

Figure 4: 2-level model for buffer characterization

3. Applying low swing optimization.

The initial CTS can be performed with any of the industrial tools at the full voltage swing. This CTS creates a clock tree that is optimized for a number of metrics such as routing congestion, tree depth, etc., therefore it is a good starting point for the proposed optimized methodology. In the second step, a buffer characterization is performed in order to observe the effect of in-place buffer sizing on the delay for the buffers available in the library, which is explained in Section 3.1. In the final step, the low swing clock tree optimization is performed at the given V_{dd} value on the clock tree that is initially synthesized for full swing, in order to decrease the skew under the skew budget without degrading the power and the slew, which is presented in Section 3.2.

3.1 Buffer Characterization

In standard cell libraries, the granularity of buffer sizes vary significantly. Convex methods for sizing of buffers are not always feasible due to highly discrete characteristic of the problem. To that end, the characterization of buffers is a necessary step, which provides the data set for a look-up table based optimization proposed in the following step (Section 3.2). Note that these look-up tables need to be generated once for each standard cell library, and the utilization of look-up tables in optimization has advantages in the run-time profiles.

In the buffer characterization step, a discrete set of candidate voltage levels for low swing operation and the discrete set of buffers in a target standard cell library are considered. The characterization analyzes the effect of discretely upsizing and downsizing of buffers on the insertion delay at each selected low swing voltage node. The challenge in this process is in maintaining the accuracy of timing of buffers with respect to those in a typical clock tree, as the sizing of one buffer affects the delay of all other buffers in the clock tree. To address this challenge, the number of sizing allowed for each buffer is defined as N, a user-specified number to be selected depending on the granularity of the buffer library so as to keep the inaccuracy of buffer characterization reasonable. The inaccuracy in this step can result in more number of iterations in the next step (Section 3.2), however this is not observed in the experiments, therefore the buffer characterization step proposed here is sufficient in practice. In order to achieve a good look-up table, a simple 2-level tree is created with the same type of buffer as the driver in the first level and a typical capacitance value that creates a slew rate of the slew constraint of the design to model a sink, shown in Figure 4. Having a 2-level tree structure enables to observe both the effect of loading to the previous level and the effect of driving a load on the insertion delay. As the number of sizings allowed is N, each buffer is simulated $2 \times N$ times, N for upsizing and N for downsizing. If there is r number of buffers in the library, this step has a complexity of $O(N \times r)$. The parameter N depends on the size of the buffer library r, therefore the complexity scales quadratically with the size of the buffer library. However, this step takes on the order of seconds, as only a 2-level model is simulated for each buffer for a small finite number of buffers.

Algorithm 1 Low Swing Clock Tree Optimization

```
Input: Buffer Characterization, Initial Clock Tree, Voltage Level
  Selected Buffers Queues, $Q_{max} = Q_{min} = \emptyset$
  Obtain the initial skew, $skew_{real}$
  while $skew_{real} < skew_{const} + skew_{margin}$ do
    $skew_{curr} = skew_{real}$
    while $skew_{curr} < skew_{const}$ do
      Find the driving buffer of the registers with min insertion delay $B_{min}$
      Find the driving buffer of the registers with max insertion delay $B_{max}$
      Select the buffer, which when removed, improves the skew the most, set empty set for the other
      $Q_{max} = Q_{max} \cup B_{max}$
      $Q_{min} = Q_{min} \cup B_{min}$
      Update $skew_{curr}$ with the remaining set
    end while
    while $Q_{max}! = \emptyset$ do
      Pop one buffer
      Analyze its neighboring buffers
      if neighboring buffers have the same delay profile then
        Size their driving buffer
      else
        Size only the corresponding buffer
      end if
    end while
    while $Q_{min}! = \emptyset$ do
      Pop one buffer
      if the neighboring buffers have the same delay profile then
        Size their driving buffer
      else
        Size only the corresponding buffer
      end if
    end while
    Run a SPICE simulation to calculate $skew_{real}$
  end while
```

3.2 Low Swing Clock Tree Optimization

Low swing clock tree optimization defines the major novelty of this paper. The objective of optimization is to preserve the power and the slew properties of low swing clock tree with a better skew in order to meet the skew budgets as closely as that of the full swing clock tree with the same topology. In this step, a low swing clock tree is optimized for skew under a skew budget via in-place buffer sizing without degrading the slew and the power consumption, with the guidance of the characteristics of the buffers in the library (Section 3.1). The proposed algorithm is outlined in Algorithm 1.

In this algorithm, the clusters of registers driven by the same clock buffer are identified and the clusters whose insertion delay is outside a window (defined by the skew budget) are marked. In order to achieve this goal, the clusters are selected one at a time from both the maximum and the minimum insertion delay paths (the algebraic different of which is the clock skew), and their driving (i.e. parent) buffer is sized up or down in order to improve the skew. In order not to disturb the rest of clock tree, the buffers that have the same driving buffer, as the (i.e. corresponding) buffer that is being analyzed, are also identified and their insertion delays are investigated. If their insertion delay have the same characteristic with the corresponding buffer, their driving buffer is sized, considering those buffers may also be in our region of interest, so as to limit the number of buffers that are modified. If not, only the corresponding buffer is sized. Once the iterations of sizing neces-

Table 2: Power, Worst Skew (WS) and Low Swing Slew (LSS) Comparison over a Full Swing Clock, and an Unoptimized Low Swing Clock Synthesized by *IC Compiler*, at $0.90 \times V_{dd}$

Circuits	Full Swing CT Synthesized by *ICC*			Low Swing CT before Optimization			Proposed Work		
	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)
s38584	30.56	60.55	63.42	28.50	72.40	75.66	28.58	61.69	75.54
s38417	34.50	62.43	59.05	31.84	73.22	70.73	31.81	61.40	70.90
s35932	28.75	83.87	62.55	26.09	101.96	74.75	26.02	82.58	74.66
b1	58.93	78.65	63.35	53.89	90.51	75.40	54.00	78.29	75.53
	Avg. Improvement Compared to Full Swing			8.1%	-13.61 ps	-12.04 ps	8.0%	0.39 ps	-12.07 ps

Table 3: Power, Worst Skew (WS) and Low Swing Slew (LSS) Comparison over a Full Swing Clock, and an Unoptimized Low Swing Clock Synthesized by *IC Compiler*, at $0.85 \times V_{dd}$

Circuits	Full Swing CT Synthesized by *ICC*			Low Swing CT before Optimization			Proposed Work		
	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)
s38584	30.56	60.55	63.42	27.87	81.02	84.20	28.03	61.83	82.79
s38417	34.50	62.43	59.05	30.95	80.61	78.87	30.92	66.54	78.84
s35932	28.75	83.87	62.55	25.25	113.94	83.23	25.23	88.85	80.76
b1	58.93	78.65	63.35	52.25	98.22	84.16	52.34	83.21	84.01
	Avg. Improvement Compared to Full Swing			10.7%	-22.54 ps	-20.52 ps	10.6%	-3.73 ps	-19.51 ps

sary buffers are completed, a SPICE simulation is run for SPICE-accurate timing verification. The clock skew can be decreased more than an allowed margin ($skew_{const} + skew_{margin}$) due to the change in the insertion delay of the registers outside of the region of interest affected by the sizing of other buffers. If clock skew degradation is not within the allowed margin, another iteration is performed. It is observed that the number of iterations is 1 in most of the cases, and no more than 3 iterations are necessary in the experiments performed for this work. The complexity of the proposed algorithm depends on the number of buffers that reside in the region of interest (defined by the neighboring buffers analyzed in optimization). In the worst case, it is $O(n)$ where n is the number of buffers at the last level of the clock tree. However, it is highly unlikely to size all the buffers at the last level (in practice, it is observed to be at most 6 buffers) therefore the complexity is approximated to be constant.

4. EXPERIMENTAL RESULTS

The proposed methodology is implemented with TCL in order to inter-operate with the existing industrial tools (e.g. Synopsys in this case) and tested on 3 largest benchmark circuits of ISCAS'89 benchmark circuits (s35932, s38417, s38584) and 1 more benchmark circuit that is created (and called b1) by combining s35932 and s38584 in order to observe the scalability. Note that ISPD'10 clock contest benchmark circuits cannot be used in this analysis as ISPD'10 clock contest benchmarks do not have data to complete the analysis in the Synopsys flow. Nonetheless, note that the largest ISCAS benchmark circuits selected here have similar number of register sinks as that in the largest ISPD'10 clock contest circuits. The RTL level designs are synthesized using *Design Compiler* of *Synopsys*, the physical placement and the synthesis of the initial full swing clock tree is performed with *IC Compiler* of *Synopsys*. The power and the skew analysis is performed using *CustomSim XA* simulator of *Synopsys* at the SPICE accuracy with 90 nm technology, operated at 500 MHz. In order to show the effectiveness of the proposed methodology, the skew, the slew and the power numbers of the full swing clock tree synthesized by *IC Compiler*, the low swing clock tree before and after the optimization are compared. Also, the comparisons are performed at 4 different low swing levels at 95%, 90%, 85% and 80% of the full swing voltage to target different levels of trade-off on the power vs. timing curve. The number of allowed buffer sizing N is set to 2, considering the number of available buffer sizes is 5 in the selected *SAED 90nm EDK Library* of *Synopsys*. The allowed degradation margin $skew_{margin}$ is set to 10 ps, 0.5% of the clock period, in order to preserve the overall skew lower than 100 ps. Note that, 100 ps is 5% of the clock period at 500 MHz, which is a typical value. In order to account for the PVT variations, the optimizations are performed at the worst case corner of the process, and the final circuit is verified at 3 different corners of process, which are:

1. Best Corner (BC): V=V_{dd}+10%, T=-40°C, fast transistors
2. Nominal Corner (NC): V=V_{dd}, T=25°C, typical transistors
3. Worst Corner (WC): V=V_{dd}-10%, T=125°C, slow transistors

The experimental results and the comparisons to the full swing clock tree and an unoptimized low swing clock tree at voltage levels of $0.90 \times V_{dd}$, $0.85 \times V_{dd}$ and $0.80 \times V_{dd}$ are shown in Table 2, Table 3 and Table 4, respectively. The comparisons at $0.95 \times V_{dd}$ are excluded because the degradation in the skew is less than the allowable margin. The experimental results show that the proposed methodology is applicable at all available voltage levels (whenever applicable to satisfy the skew and the slew constraints) in order to obtain skew values as low as the corresponding full swing clock tree while preserving the substantial decrease in the power consumption. The decrease in power consumption is achieved through the degradation in the slew, however the low swing slew constraints defined in Section 2.2 are satisfied, which guarantees an insignificant decrease in the timing slack (at 80% of supply voltage, 11.0% power savings obtained with a slack decrease of $\approx$ 8 ps, as shown in Table 1). Furthermore, it is also important to note that, the proposed methodology can obtain different power savings (8.0% to 11.0%) and different slew degradation (12.07 ps to 30.82 ps) at different low swing voltage levels with (almost) the same skew value (at most 3.73 ps increase) within the budget. Thus, the proposed methodology can target a wide range of applications by changing the applied low swing voltage level in order to budget the available timing slack for power savings.

Table 4: **Power, Worst Skew (WS) and Low Swing Slew (LSS) Comparison over a Full Swing Clock, and an Unoptimized Low Swing Clock Synthesized by *IC Compiler*, at** $0.80 \times V_{dd}$

Circuits	Full Swing CT Synthesized by *ICC*			Low Swing CT before Optimization			Proposed Work		
	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)	Power (mW)	WS (ps)	LSS (ps)
s38584	30.56	60.55	63.42	27.72	92.40	94.85	27.95	66.40	93.63
s38417	34.50	62.43	59.05	30.70	88.34	89.47	30.65	55.71	89.33
s35932	28.75	83.87	62.55	25.13	127.87	93.96	25.17	91.54	93.73
b1	58.93	78.65	63.35	51.88	108.94	95.15	51.95	77.52	94.94
	Avg. Improvement Compared to Full Swing			11.2%	-33.48 ps	-31.27 ps	11.0%	-1.42 ps	-30.82 ps

Table 5: Number of Iterations and Number of Buffers Modified for each Benchmark Circuit at All Voltage Levels

	No of Sinks	No of Clock Buffers	No of Iterations			No of Buffers Modified		
			$0.90 \times V_{dd}$	$0.85 \times V_{dd}$	$0.80 \times V_{dd}$	$0.90 \times V_{dd}$	$0.85 \times V_{dd}$	$0.80 \times V_{dd}$
s38584	1238	134	1	1	1	1	3	4
s38417	1463	150	1	1	3	1	3	2
s35932	1728	170	1	2	2	2	3	6
b1	2976	315	1	2	2	3	4	6

The run time of the proposed algorithm is also analyzed. The number of iterations (each takes about ≈ 5 mins) for each benchmark circuit at each voltage level is shown in Table 5 with the circuit size information. Even with the largest benchmark circuit with 2976 sinks, the proposed methodology converges in 2 iterations at most. The number of iterations is 1 in most of the cases, and the max number of iterations observed is 3, which occurs only once for s38417. Note that, the allowed skew degradation margin $skew_{margin}$ can be increased for better run time with less performance, or can be decreased for better performance by degrading the run time, which also shows the practicality of the algorithm for automation purposes. This phenomenon actually can be seen when the voltage level is at its 80%, shown in Figure 4, the skew is less compared to the skew at 85%, shown in Figure 3, but the number of iterations is higher at 80%. Moreover, as the delay characteristics of the neighboring buffers are considered, the clusters in the region of interest are optimized with a single sizing at the (n-1)st level and the number of buffers that are modified (sized) are limited to at most 6 buffers among the 100+ buffers of the clock trees.

5. CONCLUSION

A new methodology for optimization of low swing clock trees is proposed. Low swing clock trees are known to be effective for power savings but with a penalty in performance. In this work, a fast heuristic is proposed in order to optimize the clock skew while preserving the power savings of the low swing clock tree. Moreover, the degradation in the slew is also analyzed and constrained in order to minimize the degradation in the timing slack. Proposed methodology can achieve a skew as low as full swing clock trees with limited number of in-place buffer sizing, therefore it is highly practical. The allowed skew degradation margin can be used in order to trade-off the skew and the run time, also the applied low swing voltage level can be used to trade-off the power savings and the timing slack. Thus, the proposed methodology has a wide range of applications depending on the timing and power constraints.

6. REFERENCES

[1] G. Shamanna, N. Kurd, J. Douglas, and M. Morrise, “Scalable, sub-1w, sub-10ps clock skew, global clock distribution architecture for Intel Core i7/i5/i3 microprocessors,” in *Proceedings of the IEEE Symposium on VLSI Circuits (VLSIC)*, June 2010, pp. 83–84.

[2] N. Kurd, J. Barkarullah, R. Dizon, T. Fletcher, and P. Madland, “A multigigahertz clocking scheme for the Pentium 4 microprocessor,” *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 36, no. 11, pp. 1647–1653, Nov. 2001.

[3] D.-J. Lee, M.-C. Kim, and I. Markov, “Low-power clock trees for CPUs,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2010, pp. 444–451.

[4] J. Lu, X. Mao, and B. Taskin, “Integrated clock mesh synthesis with incremental register placement,” *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 31, no. 2, pp. 217–227, Feb. 2012.

[5] Q. Zhu and M. Zhang, “Low-voltage swing clock distribution schemes,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2001, pp. 418–421.

[6] J. Pangjun and S. Sapatnekar, “Low-power clock distribution using multiple voltages and reduced swings,” *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 10, no. 3, pp. 309–318, June 2002.

[7] F. Haj Ali Asgari and M. Sachdev, “A low-power reduced swing global clocking methodology,” *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 12, no. 5, pp. 538–545, May 2004.

[8] D. Markovic, J. Tschanz, and V. De, “Feasibility study of low-swing clocking,” in *the International Conference on Microelectronics*, vol. 2, May 2004, pp. 547–550.

[9] C. Kim and S.-M. Kang, “A low-swing clock double-edge triggered flip-flop,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 5, pp. 648–652, May 2002.

[10] S. Raja, F. Varadi, M. Becer, and J. Geada, “Transistor level gate modeling for accurate and fast timing, noise, and power analysis,” in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, June 2008, pp. 456–461.

[11] P. Li and E. Acar, “A waveform independent gate model for accurate timing analysis,” in *IEEE International Conference on Computer Design (ICCD)*, Oct. 2005, pp. 363–365.

[12] *Synopsys 90nm Generic Library*, Synopsys Inc., 2009.