Clock Power Minimization using Structured Latch Templates and Decision Tree Induction

Samuel I. Ward†, Natarajan Viswanathan‡, Nancy Y. Zhou‡,
Cliff C. N. Sze‡, Zhuo Li‡, Charles J. Alpert‡, David Z. Pan†
† ECE Dept., The University of Texas at Austin, Austin, TX 78712
‡ IBM Corporation, 11501 Burnet Rd, Austin, TX 78758
wardsi@utexas.edu, {nviswan, nancyz, csze, lizhuo, alpert}@us.ibm.com, dpan@cerc.utexas.edu

***Abstract*—This work proposes a novel latch placement methodology by computing optimized placement templates with significantly lower local clock tree capacitance at a one-time cost per standard cell library. By directly minimizing local clock tree capacitance, overall chip power is reduced. The proposed methodology first generates optimized placement solutions for a wide range of input configurations. Then, a redundancy removal approach using set-theoretic annotation is proposed demonstrating it is possible to remove over 99% of the templates with no information loss. Finally, a decision tree induction algorithm with novel impurity metric enables extremely fast template selection during the clock optimization stage of a modern physical design flow. The proposed approach reduces the local clock tree capacitance by 20-30% on average roughly equating to between a 1 and 4 watt reduction in total dynamic power on a 100-watt 22-nm microprocessor. Additionally, because of *a priori* generation, template selection during physical design is extremely fast.**

***Index Terms*—Algorithms, optimization, physical design, layout, clock placement, power**

I. Introduction

Predicted for many years, power constraints have throttled performance scaling once experienced in multi-Ghz microprocessor design. These constraints now relegate process enhancements to minor speedups with little change expected in the near future. This necessitates costly power savings techniques such as multiple supply voltage islands, multiple threshold voltages, and aggressive power gating techniques. In spite of these efforts, power persists as the greatest challenge to both modern multi-GHz designs and low-power SoCs in nanometer CMOS technologies. Complicating the issue is the increasing on-chip variation (OCV), which produces a significant drop in yield [1] [2] resulting in stricter design guide rules. These effects are particularly poignant for clock design. Clock power often contributes between 40% and 50% [3] [4] [5] of total CPU power and any improvement results in meaningful overall chip power savings. Though clock design has been heavily researched, it is still a challenging and critical aspect of physical design as shown by the recent clock synthesis contest [6].

Compounding the power problem, skew requirements of multi-Ghz design has necessitated the need for a hybrid clock routing methodology where a low-skew global clock mesh overlays the entire die area followed by locally buffered clock trees [4] [5] [7] often described as multi-source clock tree synthesis (MSCTS). An example of this methodology is shown in Fig. 1. Design rules enforce strict skew constraints on the global clock mesh, which is located on upper metal layers and shown in green. The local clock buffer (LCB) connects to the global clock mesh at specific locations providing latch placement flexibility to the physical design (PD) automation tools.

This two-tiered approach came about for a number of reasons. First, though clock trees are lower power than clock meshes, trees do not offer enough skew performance for multi-Ghz designs. Second, clock meshes are very power hungry and local clock trees (LCT)s offer a significant power savings. Third, LCTs reduce the local wire routing demands for routing the clock compared to full meshes at lower level metal layers. This methodology still faces challenges from increases in process variation and tightening design and OCV constraints making it difficult to generate correct-by-construction LCTs. Additionally, within an individual design, there could be thousands of these LCTs making accurate design time modeling and optimization difficult because of the runtime impact. In practice, overly pessimistic constraints are often applied to minimize electrical violations on the LCT, which results in extra timing closure cycles, significant redesign, or even engineering change orders (ECO)'s.

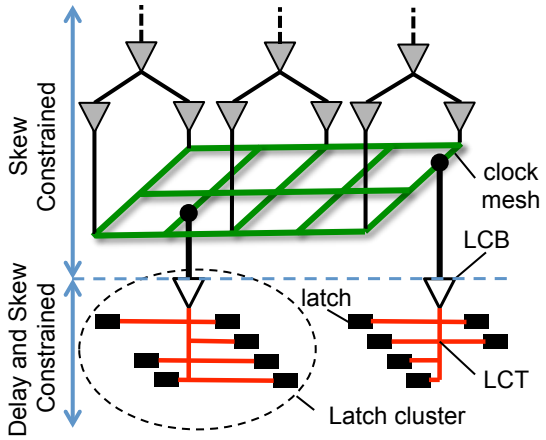


Fig. 1. State-of-the-art high performance clock mesh methodology. The global clock grid uses a clock mesh with tight skew requirements. Local clock buffers (LCB)s connect to the grid and drive local clock trees (LCT)s to each individual latch. Design guide rules maintain strict skew and nominal delay constraints for each LCT.

The key contributions of this work are as follows:

1) A methodology for significant power reduction is proposed by generating optimized latch cluster placement templates once for each technology library.
2) A correct-by-construction optimized latch cluster placement template flow is developed to meet electrical constraints reducing design time.
3) A framework is developed for reducing the required cardinality of the structured templates through set-theoretic annotation.
4) A machine learning based decision tree induction model with a novel distance metric is trained to quickly select the correct optimized template within the PD flow.

Section II outlines the background and presents a motivating example displaying the significant capacitance reduction possible through structured latch cluster placement templates. Section III presents the proposed overall template development flow and Section IV details a genetic latch placement algorithm for identifying optimized placement solutions. Structured template generation and redundancy removal is proposed in Section V and the decision tree classification with novel distance metric is described in Section VI. Section VII illustrates how the templates integrate into a modern physical design flow and lastly, experimental results are presented in Section VIII.

978-1-4799-1071-7/13/$31.00 ©2013 IEEE

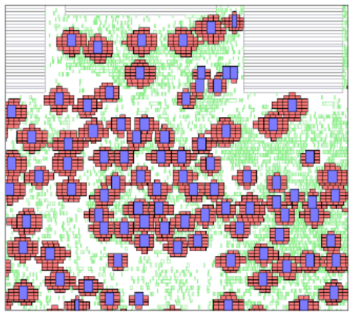


Fig. 2. Multi-Ghz design showing conventional "clustered" latches. Red cells are latches and purple cells are LCBs. One-time computation of optimized templates for a technology library make it possible to significantly decrease local clock tree (LCT) capacitance resulting in reduced total power.

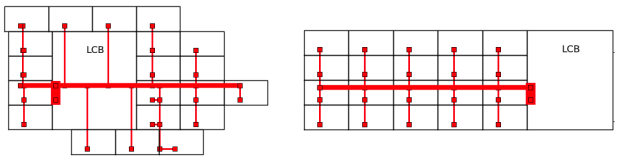
II. Background and Motivation

Clock skew is the fundamental metric for evaluating clock performance. Optimizing the clock tree in the presence of this constraint has been approached from many aspects including clock gating, multiple clock domains, reducing local clock buffers (LCB) and register (latch) placement. Modern System-on-a-chip (SoC) and multi-Ghz designs utilize many, if not all, of these techniques to build robust low skew clock networks [1] [8] [9] [10]. Recently, it has been shown that modifying the latch placement locations is an effective approach producing significant reduction in overall local clock tree (LCT) capacitance when compared to unconstrained placement [1] [2] [11] [12]. By minimizing the clock tree capacitance in this manner, clock power is directly reduced.

There are three prior latch placement modification techniques, latch shifting, latch clustering, and latch banking. *Latch shifting* is the least disruptive approach where [11] showed that incremental shifts in latch placement toward preferred locations resulted in smaller clock trees. Though latch shifting is less disruptive, the reduction in LCT capacitance is also limited. *Latch clustering* is the most common LCT reduction approach when modifying latch placement. The work in [2] showed that clustering latches around LCB's significantly improved latch power and helped to meet design rule requirements. Clustering the latches around the LCB in this manner reduced LCT capacitance up to 50% when compared to unconstrained placement. "Length-constrained latch clustering" [3] generalized this concept, where a maximum latch displacement parameter provided a tunable trade-off in LCB numbers versus layout disruption. *Latch banking* is the third approach. By automatically placing registers into fixed "banks", it is possible to reduce both clock power and skew [12]. Relative placement constraints are used to implement this approach but it is often not flexible enough to deal well with industrial challenges such as fixed obstacles and congestion [1]. Additionally, the LCT, though simplified, must still be implemented at runtime.

For all approaches, modifying the placement location of latches causes a timing degradation [3] because the clock optimization (clock opt) stage occurs after initial timing corrections have occurred. Banking is the most disruptive because it applies a fixed placement constraint without giving flexibility to the PD tool to select a more optimal structure. In spite of the disruptive nature, clustering has been widely adopted as a balanced approach because of the significant power savings. As such, modern PD flows have enhanced post clock opt steps to address the performance degradation [3] [13] [14] [15]. The result is that PD flows are robust enough to overcome disruptions caused by the modified latch placement producing significant power savings with little delay impact. Because of this, latch clustering has become the de facto methodology for multi-Ghz designs.

Figure 2 shows snapshot of a multi-Ghz design employing the conventional clustering approach. As can be seen, small groups of latches are tightly clustered around an LCB significantly reducing the total LCT length resulting in direct power savings. With local *clock trees contributing between 20-30%* of the total dynamic power [4] [5] [16], even small improvements in the LCT equate to large overall power savings.



(a) (b)

Fig. 3. Placement of 20 latches around an LCB where (a) is a conventional clustered solution with latches pulled close to the LCB producing lower skew and (b) is the proposed structured latch placement solution with higher skew but still meeting design requirements. With only 20 latches, (b) reduces total capacitance on this LCT by 33%. The red wires show the LCT routing solution.

A. Conventional verses Structured Latch Clusters

Extending the concept of conventional latch placement approaches, this work generates a set of optimized placement templates called structured latch clusters one time for the technology library. Generated *a priori*, the templates specifically reduce clock tree capacitance on the LCB while meeting the maximum skew constraint. This provides the flexibility of the clustered approach with the power savings of the banked approach. In this work, as in the 2010 ISPD Clock Contest [6], the clock tree capacitance is the primary evaluation metric for comparing multiple latch cluster placement solutions because, as Equ. 1 shows, reducing the C_{load} of the LCT directly reduces overall dynamic power. In addition to the reduction in capacitance, the placement templates are correct-by-construction resulting in reduced design iterations after clock optimization.

$$P_{avg} = \frac{1}{T} \int_0^T v(t) * i(t) dt = C_{load} * V_{DD}^2 * f \qquad (1)$$

Fig. 3 shows an example of the potential impact of structured latch placement verses a common tightly clustered solution. First, with only 20 latches, the structured latch solution reduces the total capacitance of the LCT by 33%. Second, even though the skew on 3(a) is better, both (a) and (b) meet maximum skew and delay constraints making both valid solutions. Third, 3(b) requires fewer local clock routing resources than (a). Fourth, current placement legalization techniques are much better suited at effectively legalizing solution (b) than solution (a) resulting in improved overall quality of results. Though not shown, the clock routing solution for conventional clustered approaches is also often more complex than a structured template. In the example presented, a single trunk route was required to meet the slew and skew constraints for both approaches. However, often that is not the case for conventional approaches. In cases where a horizontal and vertical trunk route is required, the capacitive savings of a structured solution can be much larger.

A *key fundamental observation of this work* is that current physical design flows are already producing packed latch cluster placements. Instead of calculating placement solutions during runtime, this work is identifying more optimal solutions *a priori* for the technology library and providing the physical design flow a runtime choice in which template to choose from. Additionally, by generating a wide range of placement templates, the PD flow has the flexibility to choose the correct placement template as opposed to a rigid latch banking approach.

B. Problem Complexity

Simply building a library of all possible structured template solutions is not practical. Modern microprocessor designs often contain more than *ten million* latches. With average latch cluster sizes between 20 and 30, that equates to over three hundred thousand local latch clusters on the microprocessor. However, any individual cluster can range between just a few or more than sixty. Additionally, twenty or more clock domains are common each with a unique set of latches. Within an individual clock domain, the standard cell library often contains latches with multiple drive strengths each resulting in different placement footprints. To optimize power, multiple LCB sizes are also needed each with different requirements for skew, slew and placement footprint. Complicating the issue, multiple placement

footprints for the latch cluster itself are required to balance tradeoffs between capacitance, skew, LCB count, placement density, fixed blockages and local routing congestion. For example, if there is significant horizontal congestion, selecting a latch cluster that has minimal horizontal routing constraints is beneficial. Combined, these requirements contribute to over one hundred thousand possible input combinations, each with a potentially different structured latch cluster solution. Additionally, latch clusters commonly contain multiple latch sizes, pushing the number of possible solutions into the millions.

Clearly selecting between millions of placement templates during run time is not practical. Three key components are required to make this tractable: first, a systematic framework to reduce the cardinality of the optimized template set, second, a generalized method for selecting the correct template, based on an unknown input combination, third, designer control because manual tuning is still a significant portion of the timing closure flow.

C. Benefits of "a priori" Optimized Placement Templates

In spite of these challenges, the proposed methodology offers five key benefits when compared against prior approaches.

1) Capacitance reduction, significantly reducing clock power, is possible through the use of optimized placement templates when compared to conventional approaches. Because of *a priori* generation, techniques to optimize the templates can sustain much longer runtimes.
2) Correct-by-construction placement solutions are possible with optimized placement templates unlike conventional approaches. This means they are almost guaranteed to legalize, to be overlap free, and meet design guide electrical and skew requirements.
3) Legalization of conventional clustered solutions can be challenging because the footprint is not guaranteed. This often causes many placement overlaps requiring further perturbations in the design closure flow. The placement footprint of the structured template solution however is very compact and known *a priori* reducing the number of timing and placement disruptions.
4) Routing aware latch placement is possible with optimized placement templates. Conventional approaches are unaware of local clock routing topology often causing localized pin accessibility issues and potentially severe congestion. With the use of structured latch cluster templates, placement solutions can be selected that mitigate congestion.
5) Runtime is a precious commodity during state-of-the-art physical design flows. By generating the optimized placement templates *a priori*, once for each technology library, compute time can be freed for other optimizations.

This work proposes a scalable solution to this problem by first generating a large set of initial placement templates. Then significantly reduces the cardinality of the solutions by using the set theoretic difference to remove redundancy. Finally, using a machine learning technique called decision tree induction, a model is developed, using a novel similarity metric for quickly selecting the correct template during design automation. Additionally, the proposed framework lends itself to be easily communicable to designers in a manner that makes it easy to interpret. As results will demonstrate in Section VIII, significant capacitance reduction on the LCT is possible through the use of the proposed approach. The next section outlines the overall flow of the proposed template design methodology.

III. Structured Template Development Flow

This work proposes a fundamental shift in latch placement methodology by proactively identifying optimized placement configurations with significantly lower capacitance one time per technology library. The proposed *a priori* template development flow is segmented into three key stages, *Template Development*, *Decision Tree Classification*, and *Classification Model Deployment* as shown in Fig. 4. *Template development*, (Section IV) is comprised of two stages, placement search and template generation. The placement search stage uses a genetic algorithm to search for a latch cluster placement solution that is as close to optimal as possible. Then, the template development

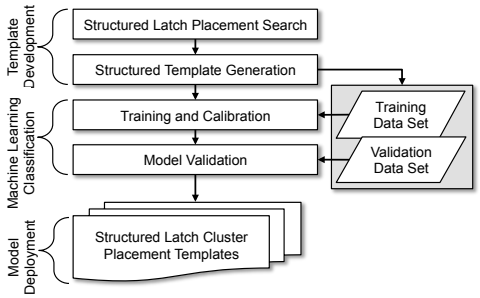


Fig. 4. Overall optimized structured template development flow design flow illustrating each stage of the proposed flow. Each is generated *a priori* per technology library at a one time cost. resulting in little runtime impact from the proposed approach.

section, (Section V), reduces the number of placement solutions by calculating the set-theoretic difference of the redundant templates and removing them. *Decision tree classification*, (Section VI), is a machine learning based classifier comprised of two stages as well. The first stage trains a supervised decision tree to map the input set to the generated templates from the prior stage. The second stage validates the decision tree on unseen patterns to verify generalization of the model. Finally, *Classification model deployment* provides the learned models for deployment during clock optimization of a normal physical design flow. All of this is generated at a one time cost resulting in little to no runtime impact from the proposed approach. The next section provides the details of the template development stage.

IV. Structured Register Placement Search

This section develops a latch cluster placement approach to implicitly reduce clock power through routed capacitance reduction while meeting design guide rules. Let F be the set of possible templates. An input vector E that results in a template $t \in F$ consists of a number of components. First is the latch cluster G consisting of two types of placeable objects, a set of latches L and an LCB, $(L, LCB) \in G$, where each latch and the LCB have a height h and width w. Second *bounding box ratio range* $\beta_{max} - \beta_{min}$ is defined as the minimum and maximum value of the placement width of the template divided by the placement height. Providing a ratio constraint gives flexibility to select a "shape". Third, the *density range* $\delta_{max} - \delta_{min}$ is defined as the structured template density. A high-density template, as displayed in Fig. 3(a), often has slightly increased capacitance requirements but is generally easier to legalize than a template that is less dense. Therefore, templates with a range of density requirements are generated. Fourth, a maximum skew constraint d_{max} for the latch cluster is provided as input. Finally, a minimum and maximum delay constraint $r_{max} - r_{min}$ is defined as an input requirement. In this work, exhaustive combinations of these input requirements are passed to the template development stage.

Given an input vector E containing G, $\beta_{min,max}$, $\delta_{min,max}$, $r_{min,max}$, and d_{max}, placement generates locations (x_i, y_i) for all placeable objects in G such that routed capacitance C_r of the local clock tree (LCT) is minimized. This is shown in Equ. 2.

$$\begin{aligned} \text{minimize:} \quad & C_r \\ \text{subject to:} \quad & d < d_{max} \\ & r_{min} \leq r < r_{max} \\ & \beta_{min} \leq \beta < \beta_{max} \\ & \delta_{min} \leq \delta < \delta_{max} \end{aligned} \tag{2}$$

A genetic search (GA) is proposed to generate the (x_i, y_i) placement locations for a optimized structured template solution. Briefly, a generic GA search begins with an initial population and applies operators to create new populations to successive generations. Reproduction is the first operator where chromosomes are copied to the next generation with some probability based on a fitness criterion.

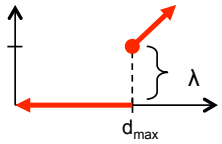


Fig. 5. Skew constraint modeled as a cost function. η_d gives preference to placement solutions meeting the skew constraint. For this work, λ is kept default across all experiments.

The second operator is crossover where weighted random pairs of chromosomes are mated creating new chromosomes. Mutation is the third operator occasionally altering a portion of the chromosome. The crossover is the most critical component for an effective GA and mutation periodically diversifies the search space. The proposed GA is summarized in Alg. 1. Further details of the GA are discussed in the following subsections with the overall algorithm presented in IV-D

A. Ordered Candidate Representation

An important requirement for utilizing a GA is the ability to represent the solution space as a set of objects, referred to as the genetic coding. In this case, each structured latch cluster (SLC) G is represented as a strictly monotonically increasing register sequence a_i, where $1 \leq i \leq |G|$. Each register $a_i \in G$ represents the tuple (x_i, y_j) where x_i and y_i are the x-axis and y-axis respectively of the placement coordinate locations for register a_i within the placement region. Strict monotonicity is maintained $\forall a_i \in G$ given by Equ. 3. This representation allows encoding each G in such a way that crossovers of feasible chromosomes result in feasible chromosomes.

$$\forall a_i \in G \begin{cases} a_i(y) \leq a_{i+1}(y) \\ a_i(x) < a_{i+1}(x) \end{cases} \iff a_i(y) = a_{i+1}(y) \tag{3}$$

B. Fitness Function

A second important requirement for utilizing a GA is an effective fitness function to score each solution. The fitness function reflects the goal to minimize the total routed capacitance C_r of G. The first component in the proposed fitness function, shown in Equ. 4, is the total routed capacitance C_r of the structured latch cluster (SLC). The second is the clock skew, modeled as the cost function shown in Equ. 5 where d is the skew of the placement solution. Figure 5 plots this cost function η_d.

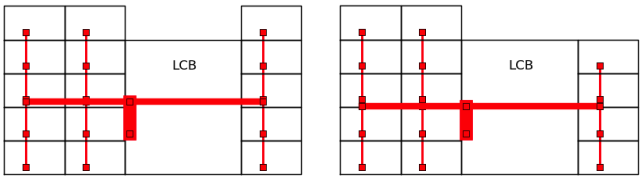
$$f(x) = \frac{1}{C_r(\eta_d + 1)} \tag{4}$$

$$\eta_d = \begin{cases} 0 & d_{slc} < d_{max} \\ \lambda \cdot (1 + d_{slc} - d_{max}) & d_{slc} \geq d_{max} \end{cases} \tag{5}$$

Maintaining the β, δ and r constraints is accomplished with an infinite cost function. Namely, chromosomes in violation of those constraints are immediately discarded. An infinite cost function for skew was evaluated but proved suboptimal compared to the proposed linear function. In the infinite case, crossing the skew boundary guaranteed the candidate solution was discarded immediately and did not mate into future valid chromosomes.

C. Order Crossover for Structured Latch Clusters

Crossover is the operator applied to a pair of parent chromosomes selected from the best solutions in the prior population. Creating new chromosomes from current ones (the crossover technique) effectively searches the solution space in a GA. In this work, that means selecting two initial parent placement solutions and generating a new placement solution based on portions of the parents. A common crossover technique [17], [18], the ordered crossover, quickly and effectively generates new candidate solutions making it an attractive option. For parent templates a_1, a_2, two child templates b_1, b_2 are generated as copies of the parents. Next, two positions are selected at random



(a) (b)

Fig. 6. Two structured latch cluster templates are shown with clock routing in red where (a), is a 15 latch template example and (b) is a 14 latch template example. Template (b) is a redundant template because (a) has more latches and the equivalent latches between (a) and (b) have identical placement.

in the interval $[1, |a|]$ and a portion of a_1 up to the first position is mapped to b_2 and a portion of a_2 from the second position is mapped onto b_1. From the respective position on, the child template is filled with the opposite parent chromosome. This approach maintains the order from the parents in positions that were mapped over.

D. Overall Placement Algorithm

The overall GA algorithm is presented in Alg. 1 and begins by randomly initializing a population. Correct-by-construction templates require legalized SLC placement solutions so the next step is overlap removal (step 2) using standard legalization techniques from [19]. Step 3, register collapsing, removes whitespace between registers within a row by collapsing them toward the LCB.

Input: G, $\beta_{min,max}$, $\delta_{min,max}$, $r_{min,max}$ and d_{max}
Output: Placement locations (x_i, y_i) for G
1: Initialize random population
2: Overlap removal
3: Register collapsing
4: Evaluate the population
while *Termination criterion is not satisfied* **do**
  5: Select chromosomes by reproduction procedure
  6: Perform Order Multiple crossover technique with probability P_c
  7: Mutate
  8: Overlap removal
  9: Register collapsing
  10: Evaluate the population
end

Algorithm 1: Proposed genetic placement algorithm for generating highly optimized structured latch clusters.

Fitness calculation is next for the entire population with the evaluation metric defined in Equ. 5. Reproduction chooses a weighted selection of the best chromosomes in the prior population. Step 6 mates the two selected chromosomes using the ordered multiple crossover technique and mutation randomly shifts a register to a new location. Steps 8 and 9 remove overlaps in the new population and collapse the registers toward the LCB. Finally, the new population is evaluated. Termination of the GA is based on iteration count with details provided in Sec. VIII.

V. Structured Template Generation

The placement search from the prior section exhaustively generated a set T of structured placement solutions for all input combinations including latch and LCB types, sizes, densities, ratios, skew and delay constraints, and latch cluster sizes. As section II-B shows, clearly it is not practical to store all placement solutions for all input combinations and in fact, many solutions are redundant.

Figure 6 illustrates such an example. As Fig. 6 shows, for all LCB and latch placement locations in (b), (a) has identical placements. By storing the difference between (a) and (b), template (b) no longer adds any additional placement information compared to the template in (a). Therefore (b) is redundant[1]. This section presents a method to significantly reduce the cardinality of T by removing redundancy.

[1]The proposed approach is a form of delta-encoding that greatly reduces data redundancy by storing information in the form of differences without loss of information.

A. Set-Theoretic Template Annotation

Given a structured latch cluster $G = (L, LCB)$ with latches L and LCB, each with placement locations (x_i, y_i) $0 \leq i \leq |L|$, Def. 1 formalizes the definition of redundancy.

Definition 1. *Given templates $(M, N) \in T$, M is redundant* $\iff$
- $|M| < |N|$,
- $\forall (x_i, y_i)$ *LCB and latch placement locations in M, there* $\exists$ *identical* (x_i, y_i) *locations in N*
- *the bounding box width of* $M = N$
- *the bounding by height of* $M = N$.

In this work, $M \lhd N$ denotes template M is made redundant by template N. By simply annotating template N with the set-theoretic difference $\xi = N \diagdown M$, it is possible to generate the placement solution for both N and M with a single template representation. Qualitatively, ξ is the set of latches in N that are not in M. Definition 2 formally defines ξ for two templates M and N.

Definition 2. *The set-theoretic difference ξ for templates $(M, N) \in T$, where $M \lhd N$ is $N \diagdown M = \{l \in N | l \notin M\}$ $\forall l \in L_{M \cup N}$*

Proving that generating both placement solutions is possible using a single template that contains ξ is trivial.

Proof: For templates $(M, N) \in T$ s.t. $M \lhd N$ and the relative compliment of N in M is ξ, let set $C = N \diagdown \xi$. The set C now contains all latches in N except those from the set ξ. But by definition 2, all latches $l \notin \xi$ is the set M. ■

By storing ξ for each redundant template in this manner, it is possible to fully reconstruct all optimized structured latch cluster placement solutions from a single annotated template. The overall reduction algorithm is presented next.

B. Proposed Redundancy Removal Algorithm

Algorithm 2 presents the proposed method for redundancy removal from T without loss of quality. Starting with an initial template set T, step (1) groups all templates into bins containing equal width and height. Then, for each group, step (2) sorts those templates by latch count. There can arise cases where multiple solutions exist with equivalent register count and total capacitance. In that case, step (3), an arbitrary solution is chosen and the others discarded. Step (4) assigns the current template being evaluated to the first element in w. The algorithm then checks for redundancy between the current template $curr_t$ and the next element in w. If template w_i is redundant, step (5) calculates the set-theoretic difference $t[i]_\xi$ and step (6) annotates $curr_t$ with $t[i]_\xi$. Finally, step (7) drops the redundant template $t[i]$ and the algorithm continues to the next template in w. If template $t[i]$ is not redundant with $curr_t$, annotation of $curr_t$ is complete. Thus, $curr_t$ is assigned the new template $t[i]$

```
Function: TemplateReduce(E, F);
Input: Initial Template Set T
Output: Reduced Template Set T annotated with ξ
1. Build template sets W of groups with equal width and height;
for each w ∈ W do
    2. ∀ templates t ∈ w, sort in descending order of latch count;
    3. Remove duplicates from w;
    4. Assign curr_t = w_0;
    for {i = 1, i < |w|, i++} do
        if t[i] ◁ curr_t then
            5. Calculate t[i]_ξ;
            6. Annoate curr_t with t[i]_ξ;
            7. Drop t[i] from T;
        else
            8. curr_t = t[i];
        end
    end
end
```

Algorithm 2: Algorithm for redundancy removal from T.

and the algorithm moves to the next template in w. The proposed approach offers three key benefits. First, Alg. 2 is a form of delta encoding offering no data loss. Second, it is very fast, running in $(O(n \cdot log(n)))$. Third, it is parallelizable on template sets W. As results will show, the proposed approach significantly reduces the cardinality of T.

VI. Decision Tree Induction for Structured Template Selection

Storing a comprehensive library mapping all inputs to a particular optimized template and ξ is neither practical nor required. Additionally, it is not possible to anticipate every situation for every latch cluster considering there are hundreds of thousands in a full design. The physical design (PD) flow needs a "decision" algorithm to quickly choose the best template given an unknown set of input requirements. Machine learning techniques offer many effective approaches including: neural networks (ANN) [20], support vector machines (SVM) [21] [22], and decision trees [23]. Additionally, many have been used to successfully solve other design automation challenges [24] [25].

Neural networks and SVMs are popular techniques because both model nonlinear relationships between the input variables and handle inter-variable interactions. However, both offer a number of drawbacks. First, both ANN and SVM cannot natively handle categorical variables with multiple classes. Arbitrary value thresholds are required for categorical discrimination (such as latch and LCB types) but decision trees handle this elegantly. Second, ANN and SVM do not present comprehensible models in a way designers will understand. Lastly, it can be difficult to incorporate ANN or SVM models into existing code without a dedicated interpreter requiring further library development. Decision trees however naturally convert to if...then...else statements leading to easy implementation or as a reference for designers. As such, this work proposes machine learning based decision tree induction for structured template selection.

A. Decision Tree Classification Overview

A decision tree classifier is a method that predicts a target class F, in this case a particular template, based on an input vector E where $(E, F) = (e_1, e_2, e_3, ..., e_k, F)$ with $(1 \leq k \leq |E|)$. For this work, E is the input parameters defined in Sec. IV. A decision tree learns by recursively partitioning the source data into subsequent subsets based on the attribute test. For this application, a decision tree classifier is particularly effective because:

1) Classifying with decision trees is a nonparametric approach meaning no prior probability distributions are required.
2) Finding an optimal decision tree is NP-complete [26] but in practice the greedy induction approaches are very effective.
3) Decisioning after training is very fast with worst case $O(\omega)$ [26], where ω is the depth of the tree.

The base decision tree induction algorithm is presented in Alg. 3. It is similar to prior techniques [23] with a critical enhancement for the splitting index. Step 1 creates a new node with either a test condition or a class label (in this case a specific template) and Step 2 and Step 3 classify and return the final decision in the case where the stopping criterion is met. In this work, a minimum number of test records within a leaf defines the stopping criterion. Let V be the set of possible templates (class labels) in node $leaf$, the leaf is labeled with the class that has the majority number of training records. Thresholding in this manner avoids over-fitting the data (avoids generalization errors). If the minimum criterion is not yet met, a new node (Step 4) called $root$ is created and Step 5 finds the best split based on the novel measure presented in VI-B. Then, for each possible outcome (class label) in $root$, Steps 7, 8, and 9 setup the recursive call for evaluating a new node.

B. Novel Placement Similarity Impurity Measure

The key enhancement to decision tree induction for structured latch templates is the observation that, inter class error rates are not consistent. In other words, if two templates are similar, the capacitance loss in choosing the wrong one is lower than choosing a template that is less similar from the correct solution. The magnitude of the impurity metric should be greater in the case where two class solutions are very different and smaller in the case where the two class solutions are very similar. Therefore, a distance measure is proposed that defines an impurity measure in terms of the similarity in placement solutions between two templates. Figure 6 will be used as an example and assume both are in the same class. Using the

Function $TreeBuild(E, F)$;
Input: Training records (E)
Output: Decision Tree Model
if $stopping\_condition(E, F) = TRUE$ **then**
    1. create a new node from the $leaf$;
    2. classify the $leaf$;
    3. return $leaf$;
else
    4. create a node $root$;
    5. find the best split and set it equal to the $root$ test condition;
    6. let $V =$ the set of possible outcome test conditions of the $root$ node;
    for *each* $v \in V$ **do**
        7. $E_v =$ training records given the $root$ test condition;
        8. $child = TreeBuild(E_v, F)$;
        9. add child as descendent of root and label;
    end
end
10. return $root$

Algorithm 3: Decision tree induction algorithm.

ordered candidate representation, the bottom left latch in both (a) and (b) will be labeled as the first latch. The second will be the next latch directly to the right of it. This continues for all latches in that row and then moves to the next row.

Once all latches are labeled in this order, it is possible to compare the latch placement at a particular location between two templates. For the example in Fig. 6, both latches have the same position for the first latch so the class average position μ would equal the position of that latch with a zero variance σ. Additionally, since template (a) has more latches than template (b), the μ for the position of the missing latch in (b) is simply the latch (a) position. This work splits each node into two groups. The average and variance for each template set for the two groups is calculated to measure the impurity. Let μ_n^i, $1 \leq n \leq |L|$ be the mean Euclidian distance belonging to class i at a given latch n at a given node. Similarly, σ_n^i is the variance at that position. Then, for each input variable E $1 \leq k \leq |E|$, Equ. 6 calculates the information gain achieved by splitting the node based on input k. By multiplying the inverse variance, classes with large variations have reduced likelihood of being selected.

$$\Delta^{i,j} = \sum_{k=0}^{K} \left(\sqrt{\sum_{n=0}^{N} (\mu_n^i - \mu_n^j)^2 * (1 + \sigma_n^i + \sigma_n^j)^{-1}} \right) \quad (6)$$

C. Supervised Learning Model Build Flow

To classify and evaluate the compact and run-time efficient template selection algorithm, the flow in Fig. 7 is proposed. In Step A, the data learning algorithms are applied over a relatively small set of input patterns with known templates. Since they are built *a priori* at a one time cost, the CPU run-time penalty is negligible.

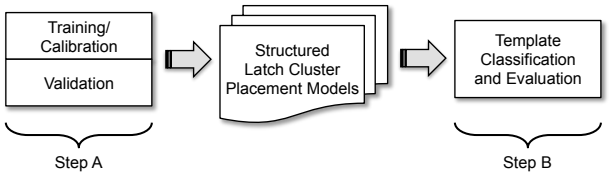


Fig. 7. Major steps to build and apply the decision-tree learning model

There are 3 major procedures involved in this step: (1) training is the process where the learning algorithms optimize structured template accuracies; (2) calibration process further improves the accuracy by adjusting the pruning rate, (3) validation process is performed over a relatively large set of known design patterns exclusive from (1) to assure the balance of learning accuracies between training data and unknown testing data, especially in Step B. Once Step A is completed, in Step B the data learning model is applied directly to classify and evaluate new unknown input patterns. All of this is done at a one time cost per technology library. To quantify the learning performance, we define the following:

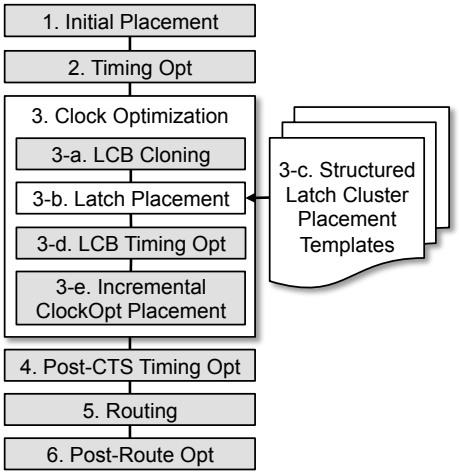


Fig. 8. Modern physical design flow illustrating each stage of the automation process. The shaded boxes are the current flow. Step 3-b displays where the optimized placement templates integrate into the flow instead of the conventional clustered approach.

Definition 3. *Template evaluation accuracy: the rate of correctly selected templates over the total number of predicted inputs.*

In the next section, an overview of how the template solutions integrate with a modern physical design flow is presented.

VII. Overall Physical Design Flow

At this point, a set of optimized placement templates have been identified and a machine learning decision model has been trained to correctly select a good template given an arbitrary input during the design flow. This is done at a one time cost per standard cell library. Before presenting the impact of the proposed work, it is worthwhile to provide a brief overview of how the model integrates into a modern state-of-the-art physical design flow. Figure 8 shows steps 1-6 in such a flow [3] [13]. This work integrates into clock optimization (Step 3) which consists of four major components: LCB cloning, latch placement, LCB timing optimization, and incremental clock optimization placement. The LCB cloning, (3-a) inserts redundant LCBs to limit the fanout and assigns latches to a single LCB. This step is important because maximum slew constraints limit the number of latches per LCB. The next stage, latch placement (3-b), modifies the position of the latches by reducing the distance between each latch and the assigned LCB. It is at this stage the latch cluster placement will be defined by the proposed structured templates. Finishing up clock optimization, steps (3-c,d) attempt to correct the timing disruption caused by latch movements within the design. The following section demonstrates the effectiveness of the proposed methodology by evaluating the capacitance ratio of the structured latch clusters verses conventional approaches. As such, Def. 4 defines the capacitance ratio used to compare the different approaches in the following section.

Definition 4. *Capacitance Ratio is defined as the routed capacitance of a structured template divided by the routed capacitance of an equivalent clustered latch cluster.*

VIII. Experimental Results

The proposed framework was implemented in C++ and the placement search, Section IV, was successively called over all input combinations. Input parameters, latch types and LCB sizes were selected based on library requirements from a state-of-the-art 22nm technology library. All resistance and capacitance values were taken from this library and clock routing and delay measurements remained consistent across all experiments. To quantify the effectiveness of the proposed methodology, results are compared against the latch clustering technique as presented in [2] and register banking in [12]. The experimental results are presented in three stages. First,

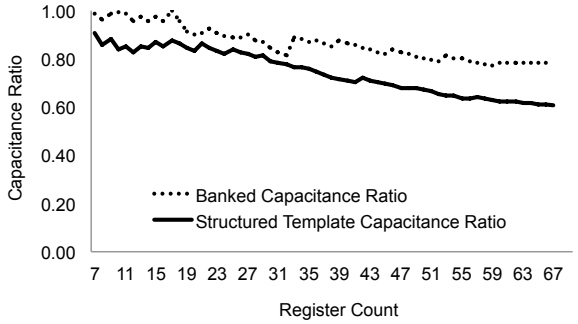


Fig. 9. Capacitance ratio of banking versus the structured template methodologies corresponding to columns 3 and 4 in Table I.

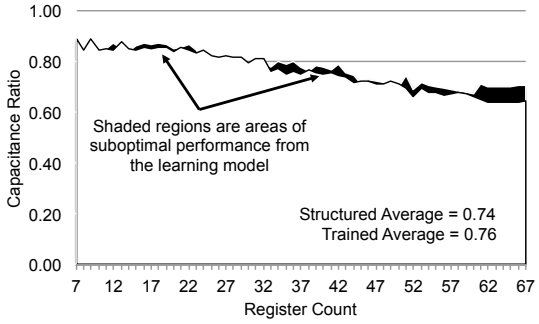


Fig. 10. Capacitance ratio of structured verses trained solutions. The shaded regions illustrate the latch cluster sizes where the learning model on average did not select the correct template.

a overview of the template generation inputs are outlined and the effectiveness of the template reduction from Section V is presented. Then, the capacitance reduction from using the generated templates is presented. Finally, the results are presented for the decision tree induction model demonstrating the effectiveness of the approach to identify good placement templates on unknown input combinations.

A. Template Generation and Redundancy Removal Results

The proposed methodology developed a set of latch placement templates optimized for minimizing local clock tree capacitance. These templates were generated over a large operating range of values expected to be encountered during a design lifecycle. Input parameters to the placement search were selected, derived from a multi-Ghz design, as follows. Latch clusters of size 7 to 67 were evaluated with four different LCB sizes. Five different bounding box ratios $[0.1 : 0.5)$, $[0.5 : 1.0)$, $[0.5 : 1.5)$, $[1.0 : 1.5)$ and $[1.5 : 10)$ and five different density ranges $[0.7 : 1.0]$, $[0.75 : 1.0]$, $[0.8 : 1.0]$, $[0.85 : 1.0]$ and $[0.9 : 1.0]$ were evaluated against four skew $4(ps)$, $5(ps)$, $6(ps)$ and $7(ps)$ and four max delay requirements $5(ps)$, $6(ps)$, $7(ps)$ and $8(ps)$. Placement solutions were generated exhaustively for *all input combinations resulting in* $96,000$ *initial solutions*. For each, the GA termination criterion was bounded at 1 million populations with 100 candidates per population and a constant mutation rate of 1%. The λ value from Equ. 5 was held constant at 0.05.

After placement solutions were generated, redundant placement cluster removal was applied as presented in Sec. V. The initial cardinality of the template set T was $96,000$. By removing invalid solutions and applying redundant template removal, the total number of templates reduced to only 534 total annotated templates. This *resulted in only 0.56% of the original possibilities* clearly demonstrating the effectiveness of the proposed approach.

B. Advantage of the Proposed Structured Latch Placement Templates

Because capacitance reduction directly reduces chip power, after generating the template set, four latch placement approaches are compared in terms of total routed local clock capacitance. The four techniques are summarized as follows:

- *Clustered*: Local clock tree (LCT) capacitances are different for each instance of a clustered solution therefore the average routed LCT of one hundred latch clusters is used as a baseline.
- *Banking*: A bank of latches is generated for all latch counts within that range of parameter requirements.
- *Structured*: This is the proposed approach with optimized structured latch templates selected based on the input parameter set.
- *Trained*: This is the template selected based on decision tree induction.

All approaches are compared relative to the conventional clustered approach using the capacitance ratio. Clearly presenting every input combination is not feasible. As such, Table I presents four sets of ratio sizes across selected latch counts with the last row displaying the average for each of the techniques.

From the selected samples, the structured approach uses 0.71 of the clock tree capacitance compared to a clustered solution with a beta ratio between 1.5 and 10. Assuming clock power consumes up to 50% of a design and the local clock tree consists of 30% of the total clock power, this results in *roughly a 4% reduction in total power* when compared to the clustered approach. When compared to the banked approach, of the same shape, it results in roughly over a 3% reduction in power. Assuming a 100-watt microprocessor, that equates to a potential 2 to 4 watt savings. At a beta ratio between 0.1 and .5, structured used 0.80 of the clustered solution resulting in up to 3% reduction in dynamic power and still produced roughly 1 watt of savings for a 100-watt microprocessor. Though templates will be technology and possibly chip specific, results clearly show the potential power savings.

To further illustrate the range of capacitance savings, figure 9 plots the capacitance ratio of both banking versus structured template methodologies corresponding to columns 3 and 4 in Table I. For local register group between 19 to 31, the two methodologies produce very similar routed capacitance. However, for latch groups of larger sizes, there is significant benefit to applying the structured latch template methodology. By increasing the number of available templates, capacitance reduces for a wider range of inputs. Next, results are presented showing it is possible to train a decision tree to effectively select the appropriate template solution.

C. Decision Tree Induction Results

This subsection presents the results from the proposed method in Section VI where a decision tree was described with a modified distance function. As the results will show, it is not necessary to record all possible input combinations and corresponding templates in a huge lookup table. Instead, it is possible to apply decision tree induction that is implemented as a series of if..and..else statements. As Fig. 7 shows, the model building process happens in two parts. First, on a small sample of data, we apply training and validation using the proposed training methodology. In this work, our training set is randomly sampled from 75% of the total data (75% training and 25% validation). Care must be made to ensure representative templates of all class labels are represented. Therefore, if there are not enough samples from a particular class, samples from an overpopulated class are discarded and a new sample is selected at random. Once training is complete, the evaluation step (Fig. 7 Step B) runs the model on untrained data samples (the remaining 25% of the population).

Using the proposed approach, *the classification accuracy was 0.918*, indicating the correct template was selected approximately 91% of the time across all validation samples. To further illustrate the effectiveness of this approach, Fig. 10 displays a graph of the average capacitance ratio of the correct template and the template selected by the learning model. The latch cluster size is presented on the x-axis. The y-axis is the average capacitance ratio of all templates from the untrained data samples. The shaded regions indicate where the learning model selects a template other than one identified as the "optimal" template.

TABLE I

TOTAL CAPACITANCE RATIO COMPARING CLUSTERED, BANKED, STRUCTURED, AND TRAINED MODEL TECHNIQUES WITH DENSITY RATIO GREATER THAN 0.7. THE BANKED AND STRUCTURED COLUMNS ARE THE RESULTS WHEN ALWAYS SELECTING THE CORRECT TEMPLATE. THE TRAINED COLUMN IS THE RESULTS PRODUCED FROM THE LEARNING MODEL.

	$1.5 < \beta < 10$				$1.0 < \beta < 1.5$				$0.5 < \beta < 1.0$				$0.1 < \beta < 0.5$			
Latch Count	Clustered	Banked	Structured	Trained	Clustered	Banked	Structured	Trained	Clustered	Banked	Structured	Trained	Clustered	Banked	Structured	Trained
7	1.00	1.99	0.85	0.85	1.00	0.99	0.87	0.90	1.00	0.99	0.89	0.89	1.00	0.96	0.86	0.86
11	1.00	0.98	0.81	0.83	1.00	0.94	0.83	0.83	1.00	0.98	0.85	0.85	1.00	0.98	0.88	0.88
15	1.00	0.97	0.83	0.84	1.00	0.96	0.84	0.86	1.00	0.97	0.84	0.88	1.00	0.99	0.88	0.90
19	1.00	0.91	0.80	0.80	1.00	0.93	0.81	0.81	1.00	0.96	0.85	0.85	1.00	0.97	0.90	0.90
23	1.00	0.91	0.79	0.79	1.00	0.98	0.79	0.81	1.00	0.96	0.85	0.85	1.00	0.95	0.89	0.89
27	1.00	0.90	0.78	0.78	1.00	0.96	0.78	0.78	1.00	0.93	0.82	0.83	1.00	0.96	0.88	0.88
31	1.00	0.83	0.75	0.83	1.00	0.88	0.77	0.77	1.00	0.88	0.81	0.81	1.00	0.88	0.82	0.82
35	1.00	0.87	0.72	0.87	1.00	0.88	0.74	0.74	1.00	0.88	0.75	0.75	1.00	0.87	0.82	0.82
39	1.00	0.87	0.68	0.87	1.00	0.87	0.73	0.73	1.00	0.87	0.75	0.75	1.00	0.87	0.81	0.81
43	1.00	0.84	0.68	0.85	1.00	0.83	0.73	0.75	1.00	0.84	0.74	0.74	1.00	0.83	0.77	0.80
47	1.00	0.83	0.65	0.65	1.00	0.82	0.71	0.71	1.00	0.83	0.71	0.75	1.00	0.82	0.76	0.76
51	1.00	0.80	0.63	0.63	1.00	0.79	0.70	0.70	1.00	0.80	0.69	0.69	1.00	0.79	0.73	0.73
55	1.00	0.80	0.61	0.61	1.00	0.79	0.69	0.69	1.00	0.81	0.68	0.68	1.00	0.79	0.70	0.70
59	1.00	0.77	0.60	0.63	1.00	0.77	0.68	0.68	1.00	0.78	0.67	0.67	1.00	0.77	0.71	0.73
63	1.00	0.78	0.59	0.64	1.00	0.77	0.67	0.69	1.00	0.78	0.64	0.67	1.00	0.76	0.70	0.70
67	1.00	0.78	0.58	0.61	1.00	0.75	0.67	0.67	1.00	0.78	0.64	0.64	1.00	0.75	0.69	0.69
Ave.	1.00	0.93	0.71	0.76	1.00	0.87	0.75	0.76	1.00	0.88	0.76	0.77	1.00	0.87	0.80	0.80

D. Runtime Results

The proposed flow is precharacterizing optimized placement templates at a one time cost for each technology library thus only template selection influences actual PD runtimes. For brevity, results are presented for all sections. Initial placement search, Section IV averaged 562 seconds(s) per template but is fully parallelizable with total iterations bounded to 1 million. Infeasible template solutions were discarded and not included in the average search time. Template redundancy removal, Section V, executed in 108(s) and model training, Section VI, completed in 419(s).

The key runtime impact is the amount of time to evaluate a decision tree. Decisioning, based on an unknown input during PD, ranges between *2 and 7 milliseconds (ms)* and averages 5.3(ms) depending on the number of branches between the root node and the leaf. Additionally, template selection is fully parallelizable per latch cluster. Thus, the proposed approach offers an extremely fast and effective method for latch cluster placement because the optimization of the template is one time for all designs.

IX. CONCLUSIONS

Microprocessor power design constraints significantly limit performance enhancements. By minimizing routed capacitance of the latch clusters, this work shows that significant power savings is possible on the local clock tree which can contribute up to 30% of the total power in a design. First, a set of structured templates is generated. Then, using machine learning based decision tree induction, a learning model is trained using a novel distance measurement. The decision tree enables the design flow to "decide", based on arbitrary input combinations, the best template to use during clock optimization of the physical design flow. Using these approaches, this work shows it is possible to reduce local clock tree capacitance by 20-30% compared to the state-of-the-art. Assuming the clock power contributes 50% of the total dynamic power, on a 100-watt microprocessor that results in roughly between a 1 and 4 watt reduction.

REFERENCES

[1] P.-H. Ho, "Industrial clock design," in *Proc. ISPD*, pp. 139–140, 2009.

[2] R. Puri, L. Stok, and S. Bhattacharya, "Keeping hot chips cool," in *Proc. DAC*, pp. 285–288, 2005.

[3] D. Papa, N. Viswanathan, Z. L. C. Sze, G.-J. Nam, C. Alpert, and I. Markov, "Physical synthesis with clock-network optimization for large systems on chips," *IEEE Micro*, vol. 31, no. 4, pp. 51–62, 2011.

[4] D. Wendel, J. Barth, D. Dreps, S. Islam, J. Pille, and J. Tierno, "Ibm power7 processor circuit design," *IBM Journal of Research and Development*, vol. 55, no. 3, pp. 1–8, 2011.

[5] J. Warnock, Y.-H. Chan, S. Carey, H. Wen, P. M. G., G. Gerwig, H. Smith, C. Yuen, J. Davis, P. Bunce, A. Pelella, D. Rodko, P. Patel, T. Strach, D. Malone, F. Malgioglio, J. Neves, D. Rude, and W. Huott, "Circuit and physical design implementation of the microprocessor chip for the zenterprise system," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 151–163, 2012.

[6] C. S. N. Sze, "Ispd 2010 high performance clock network synthesis contest: benchmark suite and results," in *Proc. ISPD*, pp. 143–143, 2010.

[7] L. Xiao, Z. Xiao, Z. Qian, Y. Jiang, T. Huang, H. Tian, and E. Young, "Local clock skew minimization using blockage-aware mixed tree-mesh clock network," in *Proc. ICCAD*, pp. 458–462, 2010.

[8] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II*, vol. 39, no. 11, pp. 799–814, 1992.

[9] R. Chaturvedi and J. Hu, "Buffered clock tree for high quality ic design," in *Proc. ISQED*, pp. 381–386, 2004.

[10] J. G. Xi and W. W.-M. Dai, "Useful-skew clock routing with gate sizing for low power design," in *Proc. DAC*, pp. 383–388, 1996.

[11] L. Huang, Y. Cai, Q. Zhou, X. Hong, J. Hu, and Y. Lu, "Clock network minimization methodology based on incremental placement," in *Proc. ASPDAC*, pp. 99–102, 2005.

[12] W. Hou, D. Liu, and P.-H. Ho, "Automatic register banking for low-power clock trees," in *Proc. ISQED*, pp. 647–652, 2009.

[13] "Rumble: An incremental, timing-driven, physical-synthesis optimization algorithm," *TCAD*, vol. 27, no. 12, pp. 2156–2168, 2008.

[14] S. I. Ward, M.-C. Kim, N. Viswanathan, Z. Li, C. Alpert, E. Swartzlander, , and D. Z. Pan, "Structure-aware placement techniques for designs with datapaths," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 2, pp. 228–241, 2013.

[15] T. Luo, H. Ren, C. Alpert, and D. Pan, "Computational geometry based placement migration," in *Proc. ICCAD*, pp. 41–47, 2005.

[16] l. Kozhaya, P. Restle, and H. Qian, "Myth busters: Microprocessor clocking is from mars, asics clocking is from venus," in *Proc. ICCAD*, pp. 271–275, 2011.

[17] M. Sehrawat and S. Singh, "Modified order crossover (ox) operator," in *Computer Science & Engineering, Intl. Journal on*, p. 2019, 2011.

[18] V. A. Cicirello, "Non-wrapping order crossover: An order preserving crossover operator that respects absolute position," in *Proc. Genetic and Evolutionary Computation Conference*, pp. 1125–1131, 2006.

[19] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proc. ASPDAC*, pp. 135–140, 2007.

[20] J. Šíma and P. Orponen, "General-purpose computation with neural networks: a survey of complexity theoretic results," *Neural Comput.*, vol. 15, no. 12, pp. 2727–2778, 2003.

[21] G. Wang, "A survey on training algorithms for support vector machine classifiers," in *Networked Computing and Advanced Information Management*, pp. 123–128, 2008.

[22] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working Set Selection Using Second Order Information for Training Support Vector Machines," vol. 6, pp. 1889–1918, 2005.

[23] S. Lomax and S. Vadera, "A survey of cost-sensitive decision tree induction algorithms," *ACM Computing Surveys*, vol. 45, no. 2, 2013.

[24] S. I. Ward, D. Ding, and D. Z. Pan, "Pade: A high-performance mixed-size placer with automatic datapath extraction and evaluation through high-dimensional data learning," in *Proc. DAC*, pp. 756–761, 2012.

[25] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "Epic: Efficient prediction of ic manufacturing hotspots with a unified meta-classification formulation," in *Proc. ASPDAC*, pp. 263–270, 2012.

[26] M. J. Moshkov, "Transactions on rough sets iii," ch. Time complexity of decision trees, pp. 244–459, 2005.