Analytical Clustering Score with Application to Post-Placement Multi-Bit Flip-Flop Merging

Chang Xu[1], Peixin Li[1], Guojie Luo[1,2], Yiyu Shi[3], and Iris Hui-Ru Jiang[4]
[1]Center for Energy-Efficient Computing and Applications, School of EECS,
Peking University, Beijing, 100871, China
[2]PKU-UCLA Joint Research Institute in Science and Engineering
[3]Dept. of ECE, Missouri University of Science and Technology, Rolla, MO 65409, USA
[4]Dept. of EE and Inst. of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan
{changxu, gluo}@pku.edu.cn, pageli328@gmail.com
yshi@mst.edu, huiru.jiang@gmail.com

ABSTRACT

Circuit clustering is usually done through discrete optimizations, with the purpose of circuit size reduction or design-specific cluster formation. Specifically, we are interested in the multi-bit flip-flop (MBFF) design technique for clock power reduction, where all previous works rely on discrete clustering optimizations. For example, INTEGRA was the only existing post-placement MBFF clustering optimizer with a sub-quadratic time complexity. However, it degrades the wirelength severely, especially for realistic designs, which may cancel out the benefits of MBFF clustering. In this paper we enable the formulation of an analytical clustering score in nonlinear programming, where the wirelength objective can be seamlessly integrated. It has sub-quadratic time complexity, reduces the clock power by about 20% as the state-of-the-art techniques, and further reduces the wirelength by about 25%. In addition, the proposed method is promising to be integrated in an in-placement MBFF clustering solver and be applied in other problems which require formulating the clustering score in the objective function.

Categories and Subject Descriptors

B.7 [**Integrated circuits**]: design aids placement and routing

Keywords

Multi-bit flip-flops, placement, clock power, timing

1. INTRODUCTION

Circuit clustering technique is a useful stage in electronic design automation. There are two categories of clustering problems: one is for circuit size reduction, where a short survey can be found in [20]; the other is for design-specific cluster formation, such as voltage island grouping [19] and register grouping [5],[6].

ISPD'15, March 29–April 1, 2015, Monterey, CA, USA.

http://dx.doi.org/10.1145/2717764.2717767.

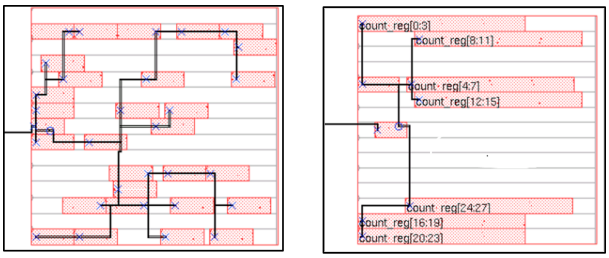


Figure 1: Clock tree synthesis for (a) conventional flow (b) MBFF flow

Table 1: The power and area of our MBFF library

Bit number	Normalized power per bit	Normalized area per bit
1	1.00	1.00
2	0.86	0.96
4	0.78	0.71

In this paper we are interested in the multi-bit flip-flop (MBFF) clustering problem, which is a promising technique for clock power reduction, due to the power savings on clock nets and flip-flops (FFs) themselves. As shown in Figure 1, the basic idea of MBFF technique is to reduce the load capacitance on the clock network, including the reduction of metal wires in the last-level of the clock tree, as well as the reduction of clock pins. In addition, the normalized per-bit power and area of a MBFF are less than a single-bit FF. Specifically, for the MBFF library we used as shown in Table 1, 4-bit MBFF is the most power-efficient and area-efficient.

Existing works explore MBFF clustering in three design stages: pre-placement stage [1][10], in-placement stage [17] [7], and post-placement stage. Because of the adequate physical information, post-placement MBFF clustering has attracted a lot of attentions, such as [21][11][18][9]. Among all these works, INTEGRA[9] delivers the best performance in both power reduction and runtime consumption and is the only one with sub-quadratic complexity. However, almost all previous works pay little attention to the signal wirelength,

either regarding it as secondary objective [21][11][18] or even ignoring it at all [9]. We demonstrate latter that for a series of realistic benchmarks, INTEGRA's clustering degenerates into a random clustering of MBFFs and induces huge degradation of signal wirelength, which might cancel out the power benefit of MBFFs.

In this paper we propose an analytical model for the clustering score. The basic idea is to first propose an exact formulation of the number of clusters based on the Dirac delta function, and smooth it using the Gaussian function to make it applicable in a nonlinear programming (NLP) framework with another objective in wirelength. With a well-designed NLP solver, the acceleration techniques such as customized fast Gauss transformation, and a further discrete refinement, our clustering flow delivers high-quality MBFF clustering results with short wirelength in sub-quadratic time. Taken INTEGRA as baseline, our method shows a comparable power reduction, reducing clock power by about 20%. Furthermore, our method optimizes signal wirelength by about 25%. What's more, the proposed method is promising to be integrated in an in-placement MBFF clustering solver, and be applied in other problems which require formulating the clustering score in the objective function.

The rest of this paper is organized as follows. Section 2 reviews the effectiveness and limitations of previous work. Section 3 introduces our optimization flow, including analytical clustering model and discrete refinement. Section 4 elaborates the details of NLP solver and the acceleration technique. Section 5 evaluates our method, and shows experimental data and comparisons. Finally, Section 6 concludes this work.

2. PRELIMINARY AND MOTIVATION

This section will focus on two concepts, the timing violation-free distance (TVFD) and the average distance between neighboring flip-flops (AFFD). The concept of TVFD will be described in Section 2.1, and the concept of AFFD and our motivations will be discussed in Section 2.2.

2.1 Post-placement MBFF Problem Formulation and Previous Solutions

Given the following inputs, **the post-placement MBFF clustering problem** will replace a few single-bit FFs with MBFF such that the power is minimized and the timing constraint and placement density constraint are satisfied.

- The number of N placed flip-flops (FFs), where each FF_i with coordinate (x_i, y_i) can either be single-bit or multi-bit.
- The upstream pin PI_i and downstream pin PO_i of FF_i, and corresponding timing slacks $\mathrm{T}_s(\mathrm{PI}_i, \mathrm{FF}_i)$ and $\mathrm{T}_s(\mathrm{PO}_i, \mathrm{FF}_i)$ between the pins and FF on the upstream and downstream timing paths, respectively.
- The MBFF library with information of power and area.
- The locations of placement blockages.

Timing constraint is proposed to prevent MBFF clustering from violating cycle delay. The basic idea is shown in Figure 2. With timing analysis, we could obtain timing slack between input (output) pins and FF. By transforming the timing slack into equivalent metal wirelength using Elmore

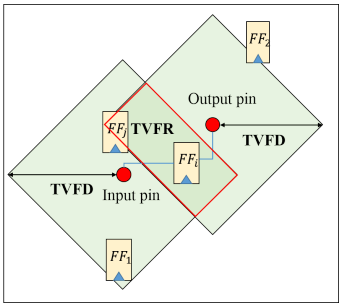


Figure 2: Concept of timing-violation-free distance (TVFD) and timing-violation-free region (TVFR)

delay model [4], we can find the timing-violation-free distance (**TVFD**) between FF and input (output) pins. Taken the input and output TVFDs into account, each FF can find a feasible region for movement without violating timing, which we label as timing-violation-free region (**TVFR**) in Figure 2. Consequently, clustering a few single-bit FFs with overlapped TVFRs can optimize power and area without violating timing constraint. Specifically, using the library in Table 1, the basic objective of MBFF clustering problem should merge FFs with overlapped TVFRs into 4-bit groups as many as possible.

Previous works certify the timing constraint with well-designed structures: intersection graph [21][11][18] or interval graph [9] and search the MBFF candidates on these graphs.

The MBFF clustering based on intersection graph has the following defects: 1) searching maximum clique is computational expensive with known best performance of $O(N^3)$. 2) The window-based optimization technique for problem size reduction limits the clustering ratio of MBFF due to the difficulty of manipulating the boundaries.

Unlike window-based technique, INTEGRA considers all FFs simultaneously, with a well designed structure: interval graph. Searching on the interval graph is more efficient, with sub-quadratic complexity. Moreover, the optimization with global information delivers better clustering ratio than the optimization with local information only. Thus, INTEGRA outperforms the previous work in terms of power reduction and runtime complexity.

However, the efficient runtime of INTEGRA partially benefits from the simple strategy of choosing MBFF candidates without measuring the impact to signal wirelength. Taking an extreme case as example, where the intervals of feasible region for each FF is as large as the placement region, INTEGRA's solution degenerates to a random solution. Although the wirelength degradation on widely used benchmarks C1-C6 [11] is acceptable (around 3%), we analyze in the following subsection that for a series of realistic designs, wirelength degradation is quite huge. Such observation motivates us to propose a new solution with great power reduction, better signal wirelength and efficient time complexity.

2.2 Previous Solutions are Inappropriate for Realistic Designs

We demonstrate in this part that 1) the widely used benchmarks C1-C6 [11] can only represent a single kind of circuit; 2) realistic designs from IWLS[2] are quite different from C1-C6 in terms of TVFR. Specifically, for the realistic de-

signs, MBFF clustering is quite easy if no signal wirelength is considered. Thus, what really matters is to find a solution with short signal wirelength while keeping the clustering ratio high.

Firstly, we define the average FF distance (AFFD) as $\sqrt{\text{ChipArea}/\#\text{FF}}$. It is obvious that the average distance between every pair of nearest FFs is approximately AFFD, assuming all the FFs are evenly placed. Then TVFD/AFFD is calculated for every FFs to roughly estimate how many FFs can be covered within the range of TVFD.

As is shown in Figure 3(a), the histogram of TVFD/AFFD for the benchmark C1 indicates that the range of TVFD is limited. The majority of FFs can only reach 3 other FFs on average within the range of TVFD.

Interestingly, all other benchmarks, C2-C6, follow exactly the same distributions as C1, even though the number of FFs is different. Moreover, if we look at the number of FFs and the placement region of these benchmarks, they scale in exactly the same ratio. These facts indicate that the benchmarks C1-C6 can only represent a single kind of circuit. They are not sufficient to evaluate the performance of MBFF clustering algorithms.

Similarly, we pick a realistic design, vga, from the IWLS benchmark suite, and synthesize it by Synopsys DC and Cadence SOC Encounter with a tight timing constraint. Specifically, the worst negative slack (WNS) reported after placement is 0.108 ns with clock cycle time of 3.5 ns and 2.5 ns for the two clock domains. The TVFD/AFFD distribution is revealed in Figure 3(b).

Comparing Figure 3(a) and Figure 3(b), we can easily find that TVFD in realistic designs is two orders of magnitude greater than TVFD in C1-C6. Thus, realistic designs have much greater freedom of choosing clustering candidates. However, INTEGRA almost has no evaluation on signal wirelength. With the analysis in Subsection 2.1 and statistics in Table 4, we see that the solution of INTEGRA damages the signal wirelength a lot for realistic designs. For example, the degradation of wirelength for ethernet is up to 20 times of original wirelength.

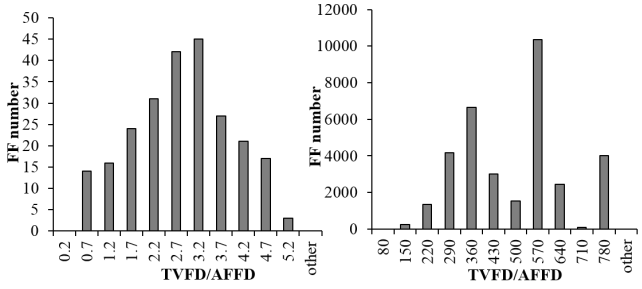


Figure 3: The histogram of TVFD/AFFD for benchmark (a) C1 (b) realistic design Vga

3. ANALYTICAL MODEL AND THE OPTIMIZATION FLOW

3.1 The Basic Idea

In this paper, we view the MBFF clustering problem as an optimization problem subject to timing constraint. The objective function optimize the number of MBFF clusters as many as possible while taking the signal wirelength into account.

Our optimization includes two steps: the analytical optimization step and the discrete refinement step. In the analytical optimization, we carefully define the formulation with a continuous and differentiable objective function and exploit an effective non-linear programming (NLP) solver [12] to make our method practical. After the analytical optimization, we can get a rough clustering. Then we invoke the discrete optimization to discretize and further improve the clustering solution. The overall flow is shown in Figure 4 and details will be illustrate in the following part.

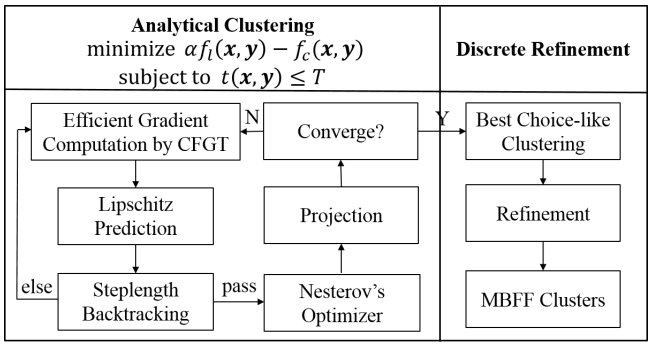


Figure 4: Optimization Flow

3.2 Analytical Optimization Step

3.2.1 Definition of Clustering Score

Our objective function trades off the signal wirelength $f_l(\boldsymbol{x},\boldsymbol{y})$ and cluster number $f_c(\boldsymbol{x},\boldsymbol{y})$ with parameter α, where $\boldsymbol{x} = (x_1,\ldots,x_N)^T$ and $\boldsymbol{y} = (y_1,\ldots,y_N)^T$ represent coordinates for N FFs. Because of the timing constraint, FFs' locations are bounded within their feasible regions. Thus, the problem is formulated as optimization problem under the timing constraint.

$$\begin{aligned} &\text{minimize} && \alpha \cdot f_l(\boldsymbol{x},\boldsymbol{y}) - f_c(\boldsymbol{x},\boldsymbol{y}) \\ &\text{subject to} && t(\boldsymbol{x},\boldsymbol{y}) \le T \end{aligned} \tag{1}$$

The signal wirelength $f_l(\boldsymbol{x},\boldsymbol{y})$ is measured by the total half-perimeter wirelength (HPWL) between locations of FFs and their upstream and downstream pins. Since HPWL is non-differentiable, we use the weighted-average approximation in [8].

The cluster number $f_c(\boldsymbol{x},\boldsymbol{y})$ is measured by the distance metric. If FF_i and FF_j are clustered eventually, their coordinates will be exactly the same. Analytically, we use the delta function to evaluate the relationship between two FFs. As shown in Equation 2 and Equation 3, the value of delta function becomes one if and only if two FFs are placed at the same location. The summation in Equation 3 calculates the total number of FFs that can be "clustered" with FF_i. For example, if $N_i(\boldsymbol{x},\boldsymbol{y})$ equals to 3, we can see FF_i is placed at the same location as three other FFs. They form a 4-bit cluster together.

$$\delta(\omega, z) = \begin{cases} 1 & (\omega = z) \\ 0 & (\omega \neq z) \end{cases} \tag{2}$$

$$N_i(\boldsymbol{x},\boldsymbol{y}) = \sum_{j\neq i} \delta(\parallel (x_i,y_i)-(x_j,y_j) \parallel, 0) \tag{3}$$

According to the library in Table 1, the 4-bit MBFF is the most efficient in power and area reduction. Thus, each FF is encouraged to cluster with three other FFs. As shown in Equation 4, minimizing the $-f_c$ term of the objective function encourages maximizing the number of FFs in 4-bit clusters. In the next subsection, we will demonstrate that our clustering function (f_c) is capable of generating attractive and repelling forces during the optimization so that it enables the formation of 4-bit clusters as many as possible.

$$\min -f_c = -\max f_c = -\max \sum_{i=1}^{N} \delta(N_i(\boldsymbol{x},\boldsymbol{y}),3) \tag{4}$$

The delta function is non-differentiable. In practice, we smooth it with Gaussian function. As Equation 5 shows, the Gaussian function can degenerate very quickly. The parameters ε and d_0^2 are used to control the degenerate speed, where ε ranges from 0 to 1 and d_0 is a distance metric. Equation 6 quantifies that the function value will decrease below ε when distance is greater than d_0. The effect of parameter d_0 will be discussed in the Subsection 4.3. With the smoothing technique, our constraint optimization problem can be solved by non-linear programming (NLP) solvers. In Subsection 4.1, we will systematically illustrate our efficient and effective NLP solver.

$$\delta(\omega,z) \approx D(\omega,z) = \exp((\omega-z)^2 \ln \varepsilon/d_0^2) \tag{5}$$

$$\begin{cases} D(\omega-z)=1 & \text{when} \quad w=z \\ D(\omega-z)<\varepsilon & \text{when} \quad \mid \omega-z \mid > d_0 \end{cases} \tag{6}$$

3.2.2 *Insights of the Clustering Function*

PROPERTY 1. *FF_i contributes a term $f_{c,i} = \delta(N_i(\boldsymbol{x},\boldsymbol{y}),3)$ to the clustering score f_c. When FF_i lies in an under-sized ($N_i < 3$) cluster, maximizing this term results in attractive forces for the neighboring FFs of FF_i; when FF_i lies in an over-sized ($N_i > 3$) cluster, maximizing this term results in repelling forces for the neighboring FFs of FF_i.*

PROOF. The force direction of FF_j resulting from $f_{c,i}$ towards can be detected by checking the sign of Equation 7. Here we only consider the x-direction without loss of generality.

$$\begin{aligned} \text{FF}_i \text{ attracts FF}_j \text{ when } (x_i-x_j)\cdot\frac{\partial f_{c,i}}{\partial x_j} > 0 \\ \text{FF}_i \text{ repels FF}_j \text{ when } (x_i-x_j)\cdot\frac{\partial f_{c,i}}{\partial x_j} < 0 \end{aligned} \tag{7}$$

When smoothing with Gaussian function, the calculation of partial derivative is shown in Equation 8, 9, 10.

$$\frac{\partial f_{c,i}}{\partial N_i} = 2\lambda_1(N_i-3)\exp((N_i-3)^2\lambda_1) \tag{8}$$

$$\frac{\partial N_i}{\partial x_j} = 2\lambda_2 \exp(\lambda_2((x_i-x_j)^2+(y_i-y_j)^2))(x_j-x_i) \tag{9}$$

$$\frac{\partial f_{c,i}}{\partial x_j} = \frac{\partial f_{c,i}}{\partial N_i}\cdot\frac{\partial N_i}{\partial x_j} \tag{10}$$

Please note that both λ_1 and λ_2 equal to $\ln \varepsilon/d_0^2$, which are negative when ε is less than 1. Thus, the force direction is only related to N_i, which is shown in Equation 11

$$\begin{cases} (x_i-x_j)\cdot\frac{\partial f_{c,i}}{\partial x_j} > 0 & \text{when } N_i < 3 \\ (x_i-x_j)\cdot\frac{\partial f_{c,i}}{\partial x_j} < 0 & \text{when } N_i > 3 \end{cases} \tag{11}$$

□

In Figure 5, we illustrate the force direction and strength from FF_i towards other FFs and the convexity of clustering function. As is shown in Figure 5(a), when FF_i is adjacent to less than 3 FFs, FF_i will attract more FFs to form a cluster of size 4; when there are more than 3 neighbors, FF_i will repel the extra FFs. Thus, it guarantees the maximization of 4-bit clusters' number. Since $f_{c,i}$ part contains Exp function, which degenerates quickly, FF_i will only affects other FFs within distance of r in Figure 5(a).

Here we take a specific case as an example to illustrate the force strength from FF_i, where FF_i locates at $(0,0)$ point, ε is set to 0.5, and d_0 is set to 100. In Figure 5(b), where the x-axis represents the change of x_j of another FF and y-axis indicates the value of $\partial f_{c,i}/\partial(x_j)$, we can see that FF_i only affects other FFs in the neighbourhood. We consider such feature to accelerate the gradient calculation with fast gauss transformation (FGT) in Subsection 4.2.

From the previous analysis, the f_l term of the objective function can pull FFs towards their "optimal locations" in terms of signal wirelength. While the f_c term of the objective function can effectively cluster FFs in the neighborhood into 4-bit groups as many as possible. Thus, our objective function achieves high-quality results in both signal wirelength and clustering ratio.

In fact, our clustering function is not convex. Supposing in the following circumstance, for FF_i, its' neighbouring FFs are FF_a and FF_b and they are aligned in horizontal. The value of $f_{c,i}$ term changes with different locations of FF_i, which has been illustrated in Figure 5(c). Although it is not convex, we will show its efficieness in Section 5.

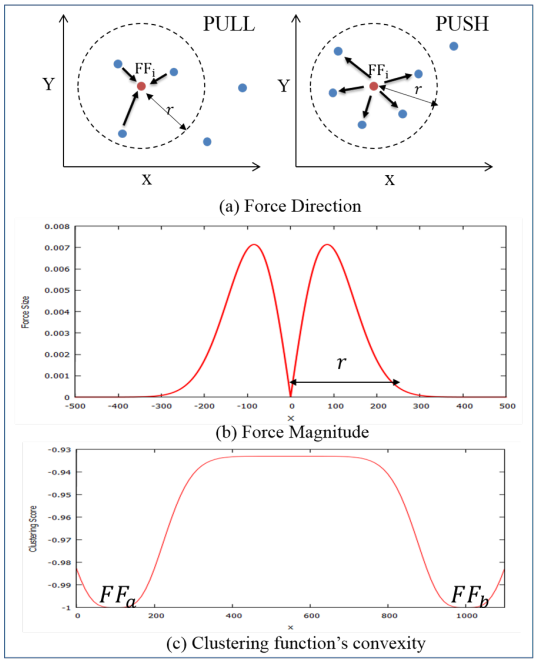


Figure 5: Analysis of force direction, magnitude and convexity of clustering function

3.3 Discrete Optimization Step

Since the analytical solution is an approximately continuous solution, additional steps are required to map the continuous solution to a final discrete solution.

We apply discrete optimization in Algorithm 1 to complete the conversion from a given analytical solution to the final solution. The discrete optimization flow looks like two-passes of best-choice clustering [3]. In the first pass (line 6-11), we extract partial clusters by bottom-up clustering based on the proximity relation after analytical solution. In this step the size of partial clusters will not exceed the maximum size (i.e. 4-bit in our library). Then we perform further refinement to improve the ratio of 4-bit clusters in the second pass (in line 12-20). It is necessary since there will be some 2-bit or 3-bit clusters after the first pass. In this step we temporally form clusters that exceed the maximum size and kick out the extra FFs immediately based on the heuristic score. Please note that we perform **timing constraint** check during cluster formation to guarantee that FFs in the same cluster have overlapped feasible regions.

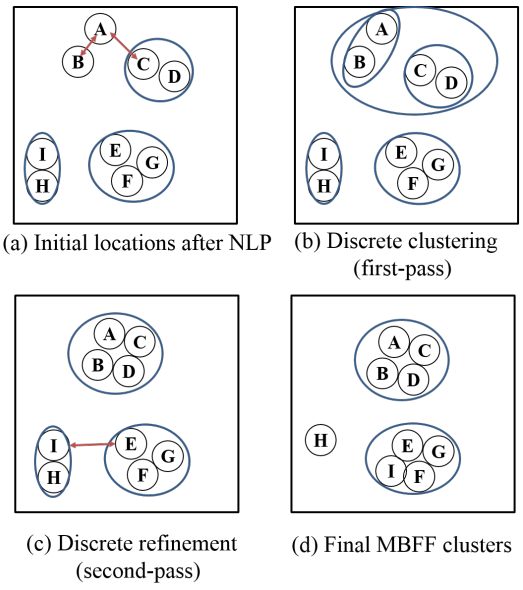
Algorithm 1 Discrete Optimization

Require:
1) The initial FFs' locations based on analytical solution
2) The feasible region for each FF
Preprocessing
1: Construct Bin-Structure for nearest neighbour searching
2: Search three nearest neighbours on Bin-Structure and Insert tuples into Priority Queue
Clustering & **Refinement**
3: $Pass \leftarrow 1$
4: **while** $Pass <= 2$ **do**
5: Pop top tuple(FF_i, FF_j, d) from PQ
6: **if** $Pass == 1$ **then**
7: **if** $|\text{Group}(\text{FF}_i) \bigcup \text{Group}(\text{FF}_j)| < 4$ **then**
8: **if** satisfy timing constraint **then**
9: Merge Group(FF_i) and Group(FF_j)
10: **end if**
11: **end if**
12: **else**
13: **if** $|\text{Group}(\text{FF}_i) \bigcup \text{Group}(\text{FF}_j)| < 4$ **then**
14: **if** satisfy timing constraint **then**
15: Merge Group(FF_i) and Group(FF_j)
16: **end if**
17: **else**
18: Merge Group(FF_i) and Group(FF_j) and Kick out excess FFs
19: **end if**
20: **end if**
21: **if** PQ.empty && Pass==1 **then**
22: Remove the 4-bit groups and insert the tuples for nearest neighbors for the remaining FFs.
23: **end if**
24: $Pass++$
25: **end while**
26: **return** MBFF clustering result

Line 1-2 extracts the proximity relation based on the analytical solution. For each FF, we calculate distances with three nearest neighbors. For the purpose of maintaining proximity relation, FF pair with small distance will be manipulated in high priority during the later cluster formation. In order to identify the globally nearest FF pair, we insert tuple$(\text{FF}_i, \text{FF}_j, d)$ into a priority-queue (PQ) with the distance d between FF_i and FF_j as the sort key. Naive searching of three nearest neighbors requires $O(N^2)$ complexity for N FFs. In order to accelerate the $O(N^2)$ searching to $O(N)$, we design an efficient bin-structure, which will be explained in Section 4.2. With our bin-structure, we can efficiently search three nearest neighbors for N FFs in $O(N)$ time, assuming that FFs are well distributed.

Line 6-11 is the clustering step (first pass). Since 4-bit MBFF is the most power-efficient and area-efficient, the cluster's capacity is limited to 4, which we call **capacity constraint**. After the preprocessing stage, we pick up the top tuple$(\text{FF}_i, \text{FF}_j, d)$ from PQ. If merging the groups that FF_i and FF_j belong to does not violate the capacity constraint and timing constraint, we commit the merge. Supposing the top two tuples are (A, B, d_1) and (A, C, d_2) (as in Figure 6(a)), after checking the capacity constraint, A and B will be firstly merged into a group of two. Then the group of AB and the group of CD will be clustered into a 4-bit group. However, group of HI and group of EFG can not be merged because of capacity constraint. The group information after clustering step is shown in Figure 6(b).

Line 12-20 is the refinement step (second pass). In this step, we allow the clustering of two groups, the sum of whose size exceeds 4, and then kick out the excess part by comparing heuristic score. The heuristic score will estimate the HPWL of signal nets if a FF candidate is stayed in the group, which is bounded by the locations of pins connected to all the FFs in the group. The less the score is, the more possibility of that FF will be stayed in the group. In Figure 6(c), we pick up the top tuple, namely (I, E, d_3), from PQ. Since merging the two groups results in a group size of 5, we kick out one FF with maximum heuristic score. The final FFs groups' formation is shown in Figure 6(d).



(a) Initial locations after NLP (b) Discrete clustering (first-pass)

(c) Discrete refinement (second-pass) (d) Final MBFF clusters

Figure 6: Discrete optimization

4. IMPLEMENTATION DETAILS

4.1 Nonlinear Programming Solver

As shown in Section 3.2, we formulate an analytical objective function in the constrained nonlinear programming model. Instead of the penalty methods, we apply a gradient projection method to handle the timing constraints.

In this work we use the Nesterov's method [12] to choose the step size to minimize the objective function efficiently. The key algorithm for the NLP solver is described in Algorithm 2.

Step size α_k is crucial to guarantee the convergence of Nesterov method. As [14] shows, $\alpha_k = L^{-1}$ satisfies steplength requirement, where L is the Lipschitz constant. Similar with [12], we approximate Lipschitz constant instead of precisely calculating. At line 1, we use backtrack method to further refine the step size.

The solution after one step of the Nesterov's method, at line 4, may violate the timing constraint. Therefore, we apply a projection step in line 5 to find a closest solution in the feasible space to replace the intermediate infeasible solution. Since the feasible solution space with respect to timing constraint is a convex space, the projection is straightforward to implement. The feasible region of a flip-flop is a circle with Manhattan distance, and it becomes a rectangle after rotating the coordinates by 45 degree. Thus, the projection can be done by rotating the coordinates by 45 degree first, and then project the location of a timing-infeasible flip-flop into its rotated rectangular timing-feasible region.

Algorithm 2 Projected Nesterov Method

Require:

$\alpha_k, \mu_k, v_k, v_k - 1, \nabla f(v_k), \nabla f(v_{k-1})$

1: $\alpha_k = BackTrack(v_k, v_k - 1, \nabla f(v_k), \nabla f(v_{k-1}))$
2: $\mu_{k+1} = v_k - \alpha_k \nabla f(v_k)$
3: $\alpha_{k+1} = (1 + \sqrt{4\alpha_k^2 + 1}/2)$
4: $v_{k+1} = \mu k + 1 + (\alpha_k - 1)(\mu k + 1 - \mu k)/\alpha k + 1$
5: $v_{k+1} = \text{Project}(v_{k+1})$
6: **return** $\mu_{k+1}, v_{k+1}, \alpha k + 1$

4.2 Customized Fast Gauss Transformation

The computation of Equation 4, in the objective function is expensive. A direct computation requires $O(N^2)$ complexity for all the N flip-flops. We use the fast Gauss transformation (FGT)[15] method to reduce the computation complexity to $O(N)$.

The basic idea of the FGT is demonstrated in Figure 7, where the evaluation of the exponential function is only needed for the neighbourhood flip-flops. Supposing FF_i is located in a square with side length $\sqrt{H}$, which equals to $\sqrt{-d_0^2/\ln(\varepsilon)}$ in our problem, as demonstrated in [15], the searching within side length of $4\sqrt{H}$ is satisfied with four digits of accuracy.

However, even using the stage-of-the-art FGT library Figtree [13], the runtime is still too slow due to the overhead of maintaining essential data structure KD-Tree, which is more suitable for high-dimensional data. Thus, we implemented a customized multithreaded FGT solver with an efficient bin structure instead of the KD-Tree structure for our two-dimensional data.

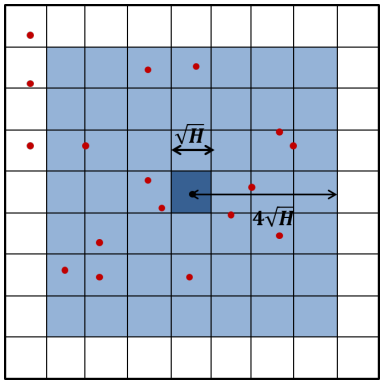


Figure 7: Explanation to fast Gauss transformation (FGT)

Table 2: Comparison of Figtree and our customized FGT

#FF	NLP solver with Figtree for FGT(s)	NLP solver with customized FGT(s)	SpeedUp
120	1.25	0.72	1.74
480	5.80	0.52	11.15
1920	31.43	1.59	19.76
5880	143.33	4.20	34.13
12000	413.45	7.92	52.20
192000	7817.23	207.68	34.64
Avg.	-	-	**25.60**

In our bin-structure, the whole chip is partitioned into a mesh of bins. And we record the amount of FFs that each bin holds. By querying adjacent bins, we can quickly obtain FFs in the neighbourhood. Time complexity is linear and the practical runtime is fast. Compared with Figtree-based implementation, our bin-structure based implementation can achieve **25.6X** speed up on average for different amount of FFs in benchmark C1-C6. Please refer to Table 2 for detailed statistics.

4.3 Parameters Tunning

The value of α (in Equation 1) is set to balance the magnitude of f_l part and f_c part. In fact, the magnitudes of f_l and f_c are positively related to $\text{Chip}_{\text{width}}$ and the total number of FFs (N), respectively. Thus, α is set as Equation 12.

$$\alpha = \frac{N}{\text{Chip}_{\text{width}}} \tag{12}$$

The value of ε and d_0 (in Equation 5) are critical for the performance. If d_0 is too small, FFs do not have enough force to affect others. However, if d_0 is too big, each FF can "see" too many FFs. Finally, FFs will reach their "balanced" states under the effects of many forces. In practice, we observe the best performance when d_0 is set as the average distance between every second nearest FF pairs in Equation 13.

$$d_0 = \frac{1}{N}\sum_{i}^{N} \parallel \text{FF}_i - \text{FF}_{\text{second\_nearest\_to\_FF}_i} \parallel \tag{13}$$

In our implementation, ε is set to 0.5, meaning that clustering score between two FFs is less than 0.5 when their distance exceeds d_0.

Table 3: Comparisons between INTEGRA and our method on C1-C6

Circuit	INTEGRA			Ours				
	PWR	*WLR*	*RT-all (s)*	*PWR*	*WLR*	*#iter-NLP*	*RT-NLP(s)*	*RT-all (s)*
C1	82.8	96	0.01	83.5	77.4	151	0.42	0.42
C2	80.9	102	0.01	82.3	76.4	208	0.96	0.96
C3	80.8	104	0.01	82.3	74.9	300	3.11	3.14
C4	81.0	104	0.02	82.4	75.6	423	10.45	10.59
C5	80.7	105	0.05	82.1	76.4	421	15.98	16.66
C6	80.7	105	1.11	82.3	82	351	197.91	217.41
Avg.	**1**	**1.33**	**1**	**1.02**	**1**	**-**	**244.88**	**251.83**

Table 4: Comparisons between INTEGRA and our method on realistic designs

Circuit	INTEGRA			Bound-INTEGRA				Ours				
	PWR	*WLR*	*RT(s)*	*factor*	*PWR*	*WLR*	*RT(s)*	*PWR*	*WLR*	*#iter-NLP*	*RT-NLP*	*RT-all(s)*
tv80	78.11	350.46	0.01	0.05	78.11	109.20	0.01	78.10	95.71	212	0.93	0.94
wbconmax	78.02	540.81	0.02	0.05	78.26	128	0.03	78.02	105	449	2.29	2.30
pairing	78.00	931.45	0.05	0.05	78.00	132.17	0.03	78.00	109	501	6.51	6.61
dma	78.03	798.29	0.06	0.05	78.04	124.71	0.05	78.02	96	501	5.39	5.43
ac97	78.02	673.28	0.09	0.05	78.02	120	0.02	78.02	95.71	351	4.74	4.88
ethernet	78.00	2038.71	1.61	0.05	78.00	216.51	0.63	78.00	87.92	501	20.81	24.51
Avg.	**1**	**9.32**	**1**	**-**	**1**	**1.43**	**0.76**	**0.99**	**1**	**-**	**82.19**	**83.52**

5. EXPERIMENTAL RESULTS

5.1 Performance Comparison

We implement our two-steps post-placement clustering method in C++, and evaluate it on an Intel Xeon machine with 16 logical threads. The MBFF clustering benchmarks include widely used C1-C6 [11] and realistic designs from IWLS-2005 suite.[2]

5.1.1 Results on the Widely Used C1-C6

We compare our results with the state-of-the-art post-placement MBFF clustering method INTEGRA in Table 3. Notations like "PWR", "WLR", "#iter-NLP", "RT-NLP" and "RT-all" represent power reduction, wirelength reduction, iterations in NLP, the runtime of NLP, and total runtime respectively. Table 3 shown that our method achieves almost the same power reduction, and further reduces the wirelength by about **33%** compared with INTEGRA. The only drawback is runtime. However, the runtime is practical for realistic circuits, and we can show that its time complexity is sub-quadratic in the next subsection.

5.1.2 Results on the Realistic Designs

We use Synopsys DC and Cadence Encounter SOC to synthesize realistic designs with tight timing constraint. Properties of these benchmarks can be found in Table 5, in which "#FF" represents the number of FFs in the circuit, "WNS" denotes worst negative timing slack after placement. Then the FFs and slacks information will feed in INTEGRA to do MBFF clustering.

Table 5: Realistic Design Property

Circuit	#FF	WNS
tv80	359	0.015
wbconmax	770	0.038
paring	1338	0.094
dma	1816	0.109
ac97	2191	0.078
ethernet	10443	0.046

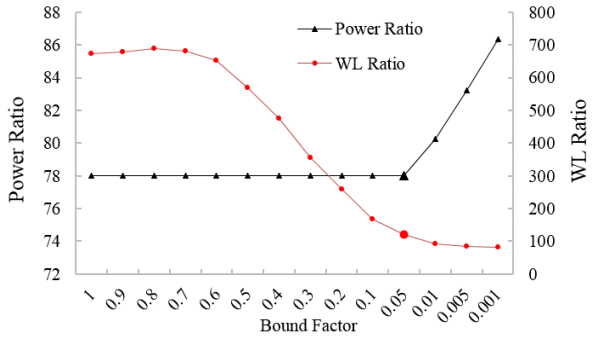


Figure 8: Effect of different bound factors to power ratio and WL ratio.

From the statistics in Table 4, we can find INTEGRA damages the wirelength about **932%** on average. According to our previous analysis, the damage of signal wirelength mainly due to the large timing-feasible region. In order to relieve this defect, we shrink the TVFD with bound factor to limit the movement of FFs for INTEGRA, which we call Bound-INTEGRA. Figure 8 compares the impacts of different bound factors to power ratio and WL ratio. We choose the one that can achieve best WL ratio when guarantee the best power reduction in our implementation. For example, in Figure 8, the best bound factor is 0.05.

As shown in Table 4, Bound-INTEGRA can achieve much better signal wirelength. Even compared with Bounded-INTEGRA, we can still obtain **43%** wirelength improvement on average. Besides, our power reduction is comparable with INTEGRA. Although our runtime is longer than INTEGRA, it is acceptable in practice.

5.2 Time Complexity Analysis

Our method consists of two steps, the analytical optimization step and the discrete optimization step.

For the analytical optimization, it requires evaluating the derivative of objective function. The derivative evaluation

of wirelength part is $O(N)$, and the derivative evaluation of clustering score is $O(N)$ with the customized FGT accelerating technique. Though it is difficult to analyze how many iterations it takes for the NLP solver, the practice in placement shows that empirical number of iterations of the placement-like NLP problem is $O(N^{1.18})$ [16]. Thus, the overall runtime for the analytical optimization is $O(N^{\alpha})$ with $\alpha < 2$.

For the discrete optimization, the most time consuming part is on the pair-wise distance calculation at line 2 and 22 in Algorithm 1. With our efficient bin-structure, the searching process can be done in $O(N)$ time.

Therefore, the empirical time complexity for the overall algorithm is sub-quadratic.

6. CONCLUSION

In this paper we propose an analytical model for the clustering objective. The basic idea of this model is to first propose a function that computes the number of clusters given a final clustering solution. The definition of such function relates to the non-differentiable Dirac delta function. In order to make it compatible with nonlinear program method, we smooth it using Gaussian function. The naive quadratic-time evaluation of the Gauss summation can be computed by the Figtree library for fast gauss transformation. However, this general implementation still does not meet the requirement of massive evaluations in an NLP solver. Thus, we implement a customized fast Gauss transformation using multithreading. The final runtime of this approach is practical, and the overall runtime is sub-quadratic. Compare to other post-placement MBFF clustering methods, our method achieves the same power reduction of about 20%, and also reduces the wirelength by 25%.

The proposed method is promising to be integrated in an in-placement MBFF clustering solver, and be applied in other problems which require formulating the clustering score in the objective function.

7. ACKNOWLEDGEMENT

The author would like to thank Tao Wang from MST and Chih-Long Chang from NCTU for their help with using Cadence and INTEGRA respectively.

This work is partly supported by National Natural Science Foundation of China (NSFC) Grant 61202073, Research Fund for the Doctoral Program of Higher Education of China (MoE/RFDP) Grant 20120001120124, and Beijing Natural Science Foundation (BJNSF) Grant 4142022.

8. REFERENCES

[1] Encounter rtl compiler advanced physical option datasheet, 2014. http://www.cadence.com/rl/resources/datasheets/rtl_physical_ds.pdf.

[2] Iwls 2005 benchmarks. http://iwls.org/iwls2005/benchmarks.html.

[3] C. Alpert, A. Kahng, G.-J. Nam, S. Reda, and P. Villarrubia. A semi-persistent clustering technique for vlsi circuit placement. In *Proceedings of the 2005 International Symposium on Physical Design*, ISPD '05, pages 200–207, New York, NY, USA, 2005. ACM.

[4] Z.-W. Chen and J.-T. Yan. Routability-driven flip-flop merging process for clock power reduction. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 203–208, Oct 2010.

[5] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang. Power-aware placement. In *Proceedings of the 42Nd Annual Design Automation Conference*, DAC '05, pages 795–800, New York, NY, USA, 2005. ACM.

[6] W. Hou, D. Liu, and P.-H. Ho. Automatic register banking for low-power clock trees. In *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pages 647–652, March 2009.

[7] C.-C. Hsu, Y.-C. Chen, and M.-H. Lin. In-placement clock-tree aware multi-bit flip-flop generation for power optimization. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 592–598, Nov 2013.

[8] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. Tsv-aware analytical placement for 3d ic designs. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 664–669, June 2011.

[9] I. H.-R. Jiang, C.-L. Chang, Y.-M. Yang, E. Y.-W. Tsai, and L. S.-F. Chen. Integra: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs. In *Proceedings of the 2011 International Symposium on Physical Design*, ISPD '11, pages 115–122, New York, NY, USA, 2011. ACM.

[10] Y. Kretchmer and L. Logic. Using multi-bit register inference to save area and power: the good, the bad, and the ugly. *EE Times Asia*, 2001.

[11] M.-H. Lin, C.-C. Hsu, and Y.-T. Chang. Post-placement power optimization with multi-bit flip-flops. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(12):1870–1882, Dec 2011.

[12] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. eplace: Electrostatics based placement using nesterov's method. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, DAC '14, pages 121:1–121:6, New York, NY, USA, 2014. ACM.

[13] V. I. Morariu, B. V. Srinivasan, V. C. Raykar, R. Duraiswami, and L. S. Davis. Automatic online tuning for fast gaussian summation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1113–1120. Curran Associates, Inc., 2009.

[14] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[15] A. M. Odlyzko and A. Schönhage. Fast algorithms for multiple evaluations of the riemann zeta function. *Transactions of the American Mathematical Society*, 309(2):797–809, 1988.

[16] P. Spindler, U. Schlichtmann, and F. Johannes. Kraftwerk2 2014;a fast force-directed quadratic placement approach using an accurate net model. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1398–1411, Aug 2008.

[17] C.-C. Tsai, Y. Shi, G. Luo, and I. H.-R. Jiang. Ff-bond: Multi-bit flip-flop bonding at placement. In *Proceedings of the 2013 ACM International Symposium on International Symposium on Physical Design*, ISPD '13, pages 147–153, New York, NY, USA, 2013. ACM.

[18] S.-H. Wang, Y.-Y. Liang, T.-Y. Kuo, and W.-K. Mak. Power-driven flip-flop merging and relocation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(2):180–191, Feb 2012.

[19] H. Wu, M. Wong, I.-M. Liu, and Y. Wang. Placement-proximity-based voltage island grouping under performance requirement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(7):1256–1269, July 2007.

[20] J. Yan, C. Chu, and W.-K. Mak. Safechoice: A novel approach to hypergraph clustering for wirelength-driven placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(7):1020–1033, July 2011.

[21] J.-T. Yan and Z.-W. Chen. Construction of constrained multi-bit flip-flops for clock power reduction. In *Green Circuits and Systems (ICGCS), 2010 International Conference on*, pages 675–678, June 2010.