Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing*

Ya-Chu Chang
Institute of Electronics
National Chiao Tung University
Hsinchu 30010, Taiwan
qwha019@yahoo.com.tw

Tung-Wei Lin
Department of Electrical Engineering
National Taiwan University
Taipei 10617, Taiwan
b04502032@ntu.edu.tw

Iris Hui-Ru Jiang
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei 10617, Taiwan
huiru.jiang@gmail.com

Gi-Joon Nam
Thomas J. Watson Research Center
IBM Research
Yorktown Heights, NY 10598, USA
gnam@us.ibm.com

ABSTRACT

As the wide adoption of FinFET technology in mass production, dynamic power becomes the bottleneck to achieving low power. Therefore, clock power reduction is crucial in modern IC design. Register clustering can effectively save clock power because of significantly reducing the number of clock sinks and register pin capacitance, clock routed wirelength, and the number of clock buffers. In this paper, we propose effective mean shift to naturally form clusters according to register distribution without placement disruption. Effective mean shift fulfills the requirements to be a good register clustering algorithm because it needs no prespecified number of clusters, is insensitive to initializations, is robust to outliers, is tolerant of various register distributions, is efficient and scalable, and balances clock power reduction against timing degradation. Experimental results show that our approach outperforms state-of-the-art work on power and timing balancing, as well as efficiency and scalability.

KEYWORDS

Register clustering; Clock power; Timing; Clustering; Mean shift

ACM Reference format:
Ya-Chu Chang, Tung-Wei Lin, Iris Hui-Ru Jiang, and Gi-Joon Nam. 2019. Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing. In *Proceedings of 2019 International Symposium on Physical Design (ISPD '19).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3299902.3309753

*This work was supported in part by Synopsys, TSMC, and MOST of Taiwan under Grant MOST 106-2628-E-002-019-MY3.

ISPD '19, April 14–17, 2019, San Francisco, CA, USA.

ACM ISBN 978-1-4503-6253-5/19/04...$15.00
https://doi.org/10.1145/3299902.3309753

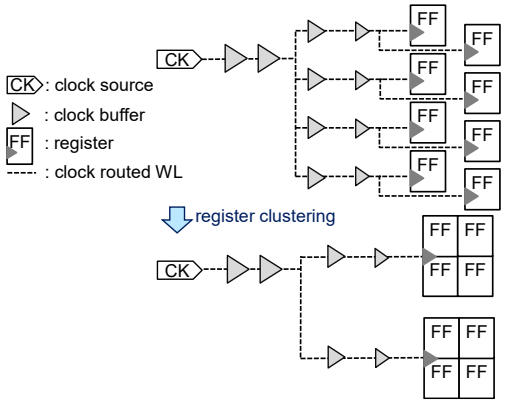


Figure 1. Register clustering reduces the switching capacitance in a clock network in all aspects.

1 INTRODUCTION

The wide adoption of FinFET technology at advanced nodes results in a dramatic drop in leakage power; therefore, dynamic power is now the bottleneck to achieving low power in modern IC design [1][2][3]. Clock power has been considered the dominant contributor to dynamic power because of its high toggle rate and large capacitive loading.

In addition to voltage scaling and clock gating, register clustering is an effective technique to save clock power. Register clustering, which gathers registers[1] into clusters, reduces the switching capacitance in a clock network in the following aspects (see Fig. 1): 1) The clock sink capacitance (in terms of register capacitance) is lowered because the number of clock sinks is greatly reduced. Furthermore, by sharing clocking circuitry within the cell, a multi-bit register has a smaller pin capacitance at the clock sinks than separate single-bit registers. 2) The clock network capacitance (in terms of clock routed

[1]A register generally means a sequential element, i.e., a flip-flop or a latch.

wirelength and clock buffers) is lessened because the depth of clock tree becomes shallow.

Recently, studies have extensively investigated register clustering, e.g., [5][6][7][8][9][10][11][12][13]. Depending on the available bit numbers, there exist two register cluster designs presented in literature: 1) Rigid cell [6][7][8][9][10][11][12][13], i.e., discrete bits. 2) Flexible template (structured latch template) [4][5], i.e., each template covers a range of bits instead of a specific number.

Most of existing works perform either in-placement or post-placement register clustering, and their solutions fall into two main categories: clique partitioning and K-means clustering. The clique partitioning approach first constructs a compatibility graph (recording the clustering compatibility between any two registers based on their timing feasible regions) and then extracts maximal cliques to form multi-bit registers without timing degradation [6][7][8][9][10][13]. The most up-to-date results were reported by Seitanidis *et al.* in [13], who enumerated all valid multi-bit registers from these cliques and then formulated an integer linear program (ILP) to generate as few feasible multi-bit registers as possible.

On the other hand, the K-means approach relaxes timing constraints to maximum displacement constraints, trying to minimize the impact on timing. K-means, however, is sensitive to initializations and outliers (distant from other registers) [14]; it starts with a prespecified number of clusters and initial cluster centers (seeds), iteratively assigns registers to nearest clusters, and finally converges to a local minimum of within-cluster total displacement [5][11][12]. Very recently, for minimizing the number of generated clusters, Wu *et al.* in [11] proposed weighted K-means, introducing a cluster size balancing weight into displacement cost. Because they intended to form large clusters (nearly maximum allowable bits of a register cell) and possibly moved outliers away, significant timing degradation cannot be avoided. Moreover, adding weights cannot guarantee elimination of oversized clusters; thus, additional processes were performed to fix over-displacement on outliers and control size overflows. Later, Kahng *et al.* in [12] adopted capacitated K-means and ILP to form feasible sized clusters. Nevertheless, due to high complexities, clique enumeration and ILP may not be applicable for large multi-bit register cells or large-scale designs.

Consider a timing-optimized placement as the input. Creating large clusters or dragging outliers far away inevitably causes large disruption to placement thus incurring significant timing degradation. The more timing degradations, the more timing ECO efforts. Besides, once registers are clustered (even few), we can save clock power. Based on these investigations, a good register clustering algorithm is desired 1) to require no prespecified number of clusters, 2) to be insensitive to initializations, 3) to be robust to outliers, 4) to be tolerant of various register distributions, 5) to be efficient and scalable, and 6) to balance power and timing.

Therefore, in this paper, we propose *effective mean shift* to perform *graceful register clustering* for reducing clock power while minimizing timing degradation (see Fig. 2). Effective mean shift augments classic mean shift with special treatments for register clustering to attain these goals.

Conceptually, clusters are expected to reside in dense regions of registers. Our idea is to direct registers towards their nearest densest spots to form clusters naturally.

In our effective mean shift algorithm, the register distribution is first mapped to a density surface; dense regions form hills. Each register climbs up (shifts) to the nearest peak in a specified search window. For register clustering, the search window (bandwidth) reflects timing criticality and local density/sparsity. Furthermore, we propose to consider effective neighbors via k-nearest neighbors (KNN) during iterative shift vector computation for efficiency and stability. Subsequently, we reassign registers and relocate clusters to further improve displacement (for timing) and refine cluster count (for power).

Effective mean shift fulfills the aforementioned requirements for being a good register clustering algorithm.

1) It requires no prespecified number of clusters. It exploits the density of registers to generate a reasonable number of clusters naturally.

2) It is insensitive to initializations. Actually, no initial seeds are needed.

3) It is robust to outliers. Our effective neighbor consideration and bandwidth setting prevent outliers in sparse regions from over-displacement.

4) It is tolerant of various register distributions. According to local density and sparsity, our clustering can tolerate uneven register distribution.

5) It is efficient and scalable. Our KNN and bandwidth setting expedites shift vector computation for each register, and our algorithm is highly parallelizable.

6) It balances power reduction against timing degradation because of graceful register clustering.

Our approach is evaluated by 2015 CAD Contest in incremental timing-driven placement benchmark suite [20], containing 768K~1932K cells with 101K~262K registers. Our approach is compared with physical design flows without performing register clustering and with the weighted K-means clustering [11]. Compared with the flow without register clustering, our method has achieved 75% reduction on clock routed wirelength, 46% decrease on clock buffer usage, and 26% savings on clock sink power with less than 2% timing degradation (in terms of total negative slack). The weighted K-means flow suffers from 11% timing degradation but with 1~2% more savings than ours on clock power. Our respective

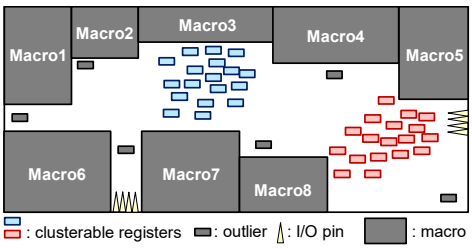


Figure 2. Graceful register clustering.

maximum register displacement and total register displacement is 19% and 43% of weighted K-means. For efficiency and scalability, our method achieves 39X (single-threaded) and 215X (multi-threaded) speedups compared with weighted K-means.

The remainder of this paper is organized as follows. Section 2 introduces preliminaries about the classic mean shift algorithm and describes the register clustering design methodology adopted in this paper. Section 3 details register clustering based on our effective mean shift algorithm. Section 4 shows experimental results. Finally, Section 5 concludes this work.

2 PRELIMINARIES

2.1 Classic Mean Shift Algorithm

Classic mean shift was introduced by Fukunaga and Hostetler [15], generalized by Cheng [16], and applied to cluster analysis in various fields, e.g., Computer Vision [17].

First, it views the data points are samples from a probability density function. Placing a kernel on each data point (Gaussian kernel is widely used [16][17]) and adding all of the individual kernels up generates a density surface (see Fig. 3). Considering a kernel k of bandwidth h, the kernel density estimator for a d-dimensional data point x is

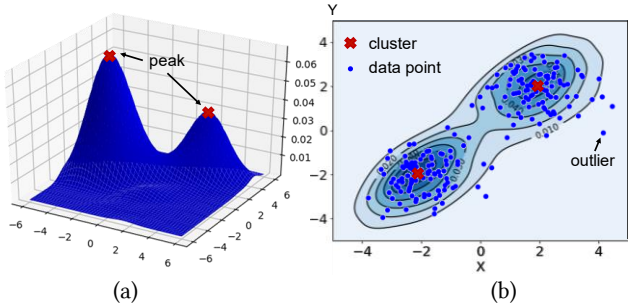
$$f(x) = \frac{1}{nh^d}\sum_{i=1}^{n} k\left(\frac{x-x_i}{h}\right). \quad (1)$$

If dense regions are present, then they correspond to local maxima of the density surface, and clusters associated with these local maxima can be identified. Classic mean shift iteratively shifts each data point uphill until it reaches the nearest peak of density surface within the bandwidth h.

The algorithm starts with making a copy of the original data points and freezing the original ones. The copied points are iteratively shifted against the original frozen points. The shift m of each point is computed by performing gradient ascent on the density function until it converges to a stationary point.

$$m(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x, \quad (2)$$

where Gaussian kernel function $k(x) = \kappa(\|x\|^2)$, $\|x\|^2$ means squared Euclidean distance, and gradient $g(x) = -\kappa'(x)$. $m(x)$ points towards the direction of maximum increase in density. Finally, all points associated with the same stationary point belong to the same cluster.



(a) (b)

Figure 3. Classic mean shift. (a) Density surface. (b) Data distribution with density contour.

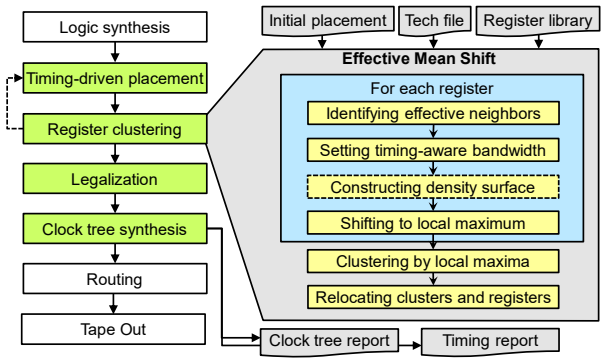


Figure 4. Register clustering methodology.

The main disadvantage of classic mean shift is its inefficiency; its time complexity is of $O(Tn^2)$, where T is the number of iterations, and n is the number of data points.

Classic mean shift shares the same kernel bandwidth h for all data points. Depending on the kernel bandwidth parameter used, the resultant density function and end clustering will vary. The bandwidth value is chosen based on domain-specific knowledge.

2.2 Problem Formulation and Methodology

In register clustering, data points represent registers in a given placement (i.e., a two-dimensional plane, $d = 2$). The induced register displacement can be approximated as the Manhattan distance between each register and the cluster center.

The inputs of the register clustering problem are a timing-optimized placement, multi-bit register library, and a user-defined maximum allowable displacement. Then, for saving clock power without placement/timing disruption, our goal is to minimize the total sum of register displacement as well as the number of clusters, while satisfying the cluster size constraint and maximum displacement constraints. The cluster size constraint is a given constant value according to the register library, while the maximum displacement for each register is set according to its timing criticality and the given maximum allowable displacement.

It can be seen that the classic mean shift algorithm cannot be directly applied due to the extra constraints and the efficiency requirement. We shall detail how we handle them by our effective mean shift in Section 3.

Fig. 4 shows the register clustering methodology adopted in this paper. Register clustering can be performed either post-placement or in-placement (if incremental placement is allowed).

3 EFFECTIVE MEAN SHIFT ALGORITHM

3.1 Overview

In this section, we propose effective mean shift to perform graceful register clustering for reducing clock power while minimizing timing degradation. We augment classic mean shift with special treatments for register clustering.

Effective mean shift naturally forms clusters according to register distribution without placement disruption. First, the

Table 1. Comparison of Classic, Adaptive, Effective Mean Shift.

Classic Mean Shift	Adaptive Mean Shift (Variable Bandwidth)	Effective Mean Shift
Density estimator $\frac{1}{nh^d}\sum_{i=1}^{n} k\left(\frac{x-x_i}{h_i}\right)$	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i{}^d} k\left(\frac{x-x_i}{h_i}\right)$	$\frac{1}{n}\sum_{i\in KNN'(x)}\frac{1}{h_i{}^d} k\left(\frac{x-x_i}{h_i}\right)$
Shift point $\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}$	$\frac{\sum_{i=1}^{n}\frac{x_i}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}$	$\frac{\sum_{i\in KNN'(x)}\frac{x_i}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i\in KNN'(x)}\frac{1}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}$

1. $k(x) = \kappa(\|x\|^2)$, Gaussian kernel 2. $g(x) = -\kappa'(x)$ 3. $d = 2$

register distribution is mapped to a density surface; dense regions form hills. Each register climbs up (shifts) to the nearest peak in a specified search window.

The search window (bandwidth) of each register varies and reflects its timing criticality and local density/sparsity. Furthermore, for efficiency and stability, we propose to consider effective neighbors via k-nearest neighbors (KNN) during iterative shift vector computation. Subsequently, we reassign registers and relocate clusters to further improve displacement (for timing) and refine cluster count (for power). Table 1 summarizes classic, adaptive, and effective mean shift.

3.2 Variable Bandwidth Selection

The kernel bandwidth parameter affects the resultant density function and clustering. As an extreme case, we use extremely tall skinny kernels (i.e., an extremely small bandwidth). The resultant density surface has a peak at each point, and thus each point forms its own cluster. In contrast, if we use an extremely short fat kernels (i.e., an extremely large bandwidth). The resultant wide smooth density surface has only one peak where all points climb up, forming one cluster. Kernels in between these two extremes lead to nicer clustering results.

Classic mean shift uses a fixed kernel bandwidth for all points. Nevertheless, the kernel bandwidth confines the search window of each point. Thus, for register clustering, each register is desired to have a variable bandwidth to reflect its timing criticality and local distribution. Then, the density function can be defined based on [18]:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i{}^d} k\left(\frac{x-x_i}{h_i}\right). \quad (3)$$

The shift vector becomes:

$$m(x) = \frac{\sum_{i=1}^{n}\frac{x_i}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)} - x. \quad (4)$$

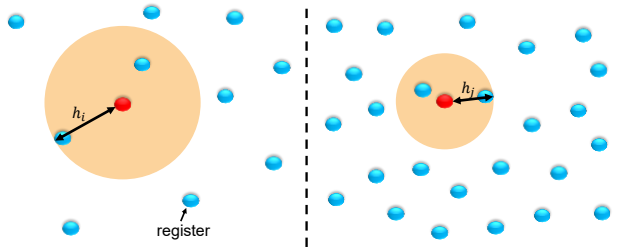


Figure 5. Bandwidth selection based on the distance to M-th nearest neighbor ($M = 2$).

For a register lying in dense regions, we select a small bandwidth, thus identifying the local maximum quickly in a narrow neighborhood and avoiding a large cluster size. Hence, considering local distribution, bandwidth is first set to as the distance to its M-th nearest neighbor (see Fig. 5). Furthermore, considering the timing criticality and maximum allowable displacement, the bandwidth of register i is

$$h_i = \min\left(h_{\max}, \alpha\|x_i - x_{i,M}\|\right), \quad (5)$$

where $h_{\max}$ denotes the maximum allowable displacement, $\|x_i - x_{i,M}\|$ means the Euclidean distance between register i and its M-th nearest neighbor ($x_{i,0} = x_i$), and α is a timing criticality coefficient; $\alpha \to 0$ for the most critical register (i.e., a very tall and skinny kernel).

3.3 Identifying Effective Neighbors

Classic mean shift considers all original data points during shift vector computation (n is usually large in practice, 101K~262K in our experiments).

However, the points that correspond to the tails of the underlying density function receive small weights in Equations (3) and (4), and thus they are almost automatically discarded. Moreover, we do not expect registers to travel far away (for minimizing disturbance to timing and placement), and try to avoid oversized clusters. Thus, we can ignore distant registers.

For achieving this goal, we identify effective neighbors via KNN, $K \ll n$. In addition, registers belonging to KNN of a register but beyond the maximum allowable displacement are also excluded (see Fig. 6). Hence, for a register at x_j, we consider the following set of registers during the computation:

$$i \in KNN(x_j) - \{x_{j,m} \mid \|x_j - x_{j,m}\| > h_{\max}, m \le K\}$$
$$= KNN'(x_j). \quad (6)$$

The density function can be rewritten as:

$$f(x) = \frac{1}{n}\sum_{i\in KNN'(x)}\frac{1}{h_i{}^d} k\left(\frac{x-x_i}{h_i}\right). \quad (7)$$

The shift vector becomes:

$$m(x) = \frac{\sum_{i\in KNN'(x)}\frac{x_i}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i\in KNN'(x)}\frac{1}{h_i{}^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)} - x. \quad (8)$$

Although the idea of effective neighbors greatly improves the efficiency of shift vector computation, when the number of iterations to convergence is large, iteratively updating effective neighbors may still be computation intensive.

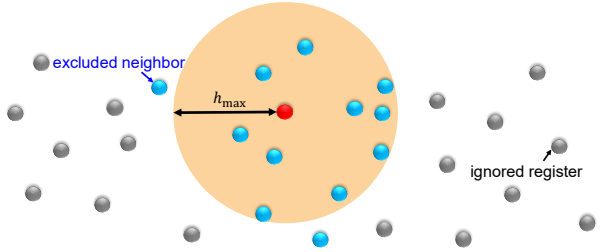


Figure 6. Effective neighbors identified by KNN ($K = 12$).

Table 2. Analysis of Distinct Neighbors (K=140).

Circuit	# of Iterations	# of Total Distinct Neighbors	# of Distinct Neighbors per Iteration
Superblue16	213	158.25	0.74
Superblue18	315	158.09	0.50
Superblue10	533	156.13	0.29

Hence, we analyze members of effective neighbors. Compared with the initial set of effective neighbors, distinct neighbors that appear throughout the entire clustering process are few. For the sample circuits, we randomly monitor 100 registers and update their effective neighbors via KNN as $K = 140$ at every iteration. Table 2 lists the statistics on average total distinct neighbors. Since neighbors barely change, effective neighbors can be identified only once (at the beginning).

3.4 Shifting to Local Density Maxima

After identifying effective neighbors and selecting a proper bandwidth for each register, we construct the density surface. We make a copy of the original register coordinates and freezing the original ones. The copied coordinates $\{y_j^t\}$ (t denotes the iteration index) are iteratively shifted against the original frozen points $\{x_j\}$. Hence, each register undergoes the following steps to seek the local density maximum.

1. Set the initial coordinates, $y_j^0 = x_j, j = 1..n$.
2. Identify effective neighbors, $KNN'(y_j^0)$; set bandwidth h_j.
3. Compute the mean shift vector $m(y_j^t)$ by Equation (8).
4. Shift each register,

$$y_j^{t+1} = y_j^t + m(y_j^t) = \frac{\sum_{i \in KNN'(y_j^0)} \frac{x_i}{h_i^{d+2}} g\left(\left\|\frac{y_j^t - x_i}{h_i}\right\|^2\right)}{\sum_{i \in KNN'(y_j^0)} \frac{1}{h_i^{d+2}} g\left(\left\|\frac{y_j^t - x_i}{h_i}\right\|^2\right)}.$$

5. Iterate steps 3 and 4 until convergence, $|y_j^{t+1} - y_j^t| < \delta$.

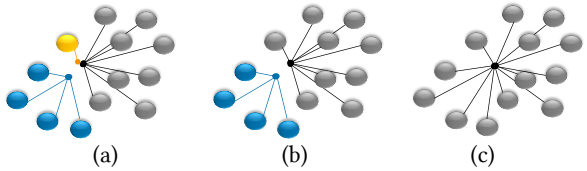
3.5 Clustering by Local Density Maxima

Classic mean shift clusters points associated with the same stationary point together. Effective mean shift considers only effective neighbors and thus induces an approximation error as computing local density maxima.

For compensating the approximation error, we further merge registers with stationary points within a threshold distance ε into a cluster (ε is very small in our experiments). As shown in Fig. 7, the greater ε, the larger cluster.

3.6 Relocation for Timing and Displacement

The previous steps in effective mean shift can be viewed as seeking the locations of clusters. We further reassign registers and relocate clusters for improving timing and displacement.



(a) (b) (c)

Figure 7. Compensating the approximation error of effective neighbors. (a) Small ε. (b) Medium ε. (c) Large ε.

First, register reassignment can be reduced to the stable matching problem. Gale and Shapley propose a stable matching algorithm [19] to map from a given set of men to the other set of women such that there exists no pair of man and woman who prefer each other to their paired partners. Due to male-optimality, in register reassignment, each register is modeled as a man, while a cluster location is modeled as a woman; the capacity of a cluster location equals the maximum allowable cluster size. The preference is ranked in non-decreasing order of displacement. Notably, the distance used during effective mean shift is measured by Euclidean distance, the displacement defined in register reassignment is the Manhattan distance from the initial location of a register to the investigated cluster location.

Second, after register reassignment, we relocate each cluster to the median coordinate of its register members for minimizing displacement and reducing timing degradation.

3.7 Complexity Analysis

Effective mean shift counts only effective neighbors by KNN and selects a proper bandwidth for each register, thus expediting the search of local density maxima.

Consider n registers, K effective neighbors for each register, T iterations to convergence, and C clusters generated. Shifting to local density maxima can be done in $O(TKn)$ time, while register reassignment and cluster relocation can be done in $O(Cn)$ time. Hence, effective mean shift is of complexity $O(TKn + Cn)$, where $K \ll n$ and $C \ll n$.

3.8 Parallelization

As shown in Fig. 4, the computation for each register is independent and thus highly parallelizable.

Fig. 8 illustrates parallel effective mean shift. First, identifying effective neighbors by KNN and setting variable bandwidth for each register can be computed in parallel. Second, shifting to local density maximum is iteratively calculated in parallel, too.

4 EXPERIMENTAL RESULTS

4.1 Experimental Setting

Our effective mean shift algorithm was implemented in the C++ programming language and compiled by G++ 4.8.5; the program

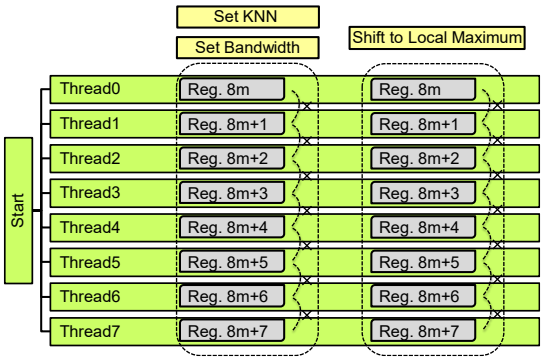


Figure 8. Parallel effective mean shift (8 threads).

Table 3. Benchmark Statistics.

Circuit	# of Cells	# of Registers
superblue16	981,559	142,543
superblue18	768,068	101,758
superblue4	796,645	167,731
superblue5	1,086,888	110,941
superblue3	1,213,253	163,107
superblue1	1,209,716	137,560
superblue7	1,931,639	262,176
superblue10	1,876,130	231,747

Table 4. Pseudo Power of Multi-bit Register Library.

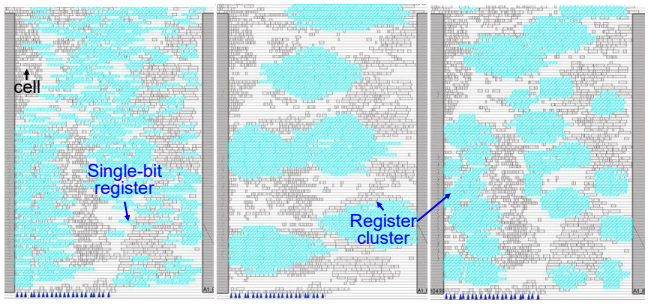
# of Bits	Normalized Pseudo Power per Bit
1	1.000
2~3	0.860
4~7	0.790
8~15	0.755
16~31	0.738
32~63	0.729
64~80	0.724

was executed on a Linux workstation with an Intel Xeon 2.6 GHz CPU and 256 GB memory. Experiments were conducted on ICCAD-2015 CAD contest in incremental timing-driven placement benchmark suite [20] as listed in Table 3, containing 768K~1932K cells with 101K~262K registers in each design. (The circuit size for [20] is far greater than that in [12] (0.5K~17K registers) and [13] (29k~50K registers).)

Our algorithm was evaluated by post-placement register clustering of the experimental flow shown in Fig. 4. We started with a timing-optimized placement and obtained the coordinates of all registers as data points in effective mean shift algorithm. After register clustering, we performed legalization and clock tree synthesis by the state-of-the-art commercial tool [21]. We analyzed the solution quality based on clock tree and timing reports. Because the cell library in [20] does not include multi-bit registers, we adopted a flexible template register library similar to the setting used in prior work [4][6][7][8][9][10] as listed in Table 4, where pseudo power is computed in a conservative way.

4.2 Comparison of Register Clustering Results

In the first experiment, we compared our results with non-clustered designs and state-of-the-art weighted K-means approach [11]. We reimplemented [11] in our flow. The maximum allowable cluster size is 80 (same setting as [11]). The maximum allowable displacement is 400 nm. For effective mean shift, K = 140 for KNN, convergence threshold δ = 0.0001 units, cluster merging threshold ε = 5000 units. (In the benchmark suite, 2000 unit length = 1 nm.)



(a) (b) (c)

Figure 9. Partial layouts (superblue16). (a) Non-clustered. (b) Weighted K-means. (c) Effective mean shift.

Table 5 compares our register clustering approach with weighted K-means ('WK') on cluster size distribution, displacement (in unit length), and runtime (in second). 'Para.' and 'Seq.' indicates the runtime of the parallel (8 threads are used in our experiment) and sequential version of effective mean shift, respectively. Weighted K-means has 2.33X average displacement of ours. We achieve 215X and 39X speedups for parallel and sequential version, respectively. Fig. 9 shows the partial layouts corresponding to the same region in superblue16 generated by non-clustering (initial timing-optimized placement), weighted K-means, and effective mean shift, where blue boxes indicate registers, and grey boxes indicate other cells. It can be seen that weighted K-means tends to generate large clusters and induce large displacement for outliers, thus incurring significant placement disruption. In contrast, effective mean shift delivers graceful register clustering. Fig. 10 shows full layouts of superblue4, where red spots indicate registers.

In addition, Fig. 10(d) shows the clustering result if the preference is computed based on the wirelength optimum site for each register instead of its initial location during register relocation (Section 3.6). The wirelength optimum site of each register is the median coordinate of its all fanin and fanout gates. Based on optimum sites, numerous registers migrate towards the regions with many obstacles, thus possibly causing severe congestions and incurring large timing degradation.

Table 6 compares the power and timing results after clock tree synthesis for non-clustered designs ('NC'), weighted K-means ('WK'), and our effective mean shift ('Ours'). 'WNS' denotes worst negative slack, 'TNS' total negative slack, 'Clock Routed WL' routed clock wirelength, '#Buffers' the number of clock buffers. 'Clock Sink Power Ratio' is computed based on Table 4. Compared with the flow without register clustering, our method achieves 75.42% reduction on clock routed wirelength, 45.97% decrease on clock buffer usage, and 25.52% savings on clock sink power, maintains the same level of WNS and induces only 1.95% timing degradation on total negative slack. TNS reflects subsequent timing ECO efforts. The weighted K-means flow suffers from 10.88% timing degradation but with 1~2% more savings than ours on clock power.

Table 5. Comparison on Cluster Size, Displacement, and Runtime with Weighted K-Means [11].

Circuit	Method	Cluster Size		Displacement	Runtime (s)	
		Min	Max	Average	Para.	Seq.
superblue16	WK	34	80	56000.54	2370	
	Ours	1	55	22353.75	35	186
superblue18	WK	35	80	60843.50	6080	
	Ours	1	70	25792.54	25	138
superblue4	WK	34	80	48129.71	8470	
	Ours	1	56	19446.86	51	311
superblue5	WK	32	80	69453.46	3590	
	Ours	1	78	29747.90	28	131
superblue3	WK	28	80	54968.00	9098	
	Ours	1	79	25696.45	45	244
superblue1	WK	42	80	64158.15	5295	
	Ours	1	62	24456.03	40	200
superblue7	WK	39	80	54761.63	37692	
	Ours	1	79	26048.28	91	513
superblue10	WK	26	80	57643.75	27474	
	Ours	1	79	27914.53	75	412
Ratio	WK/Ours			2.33	215.03	39.42

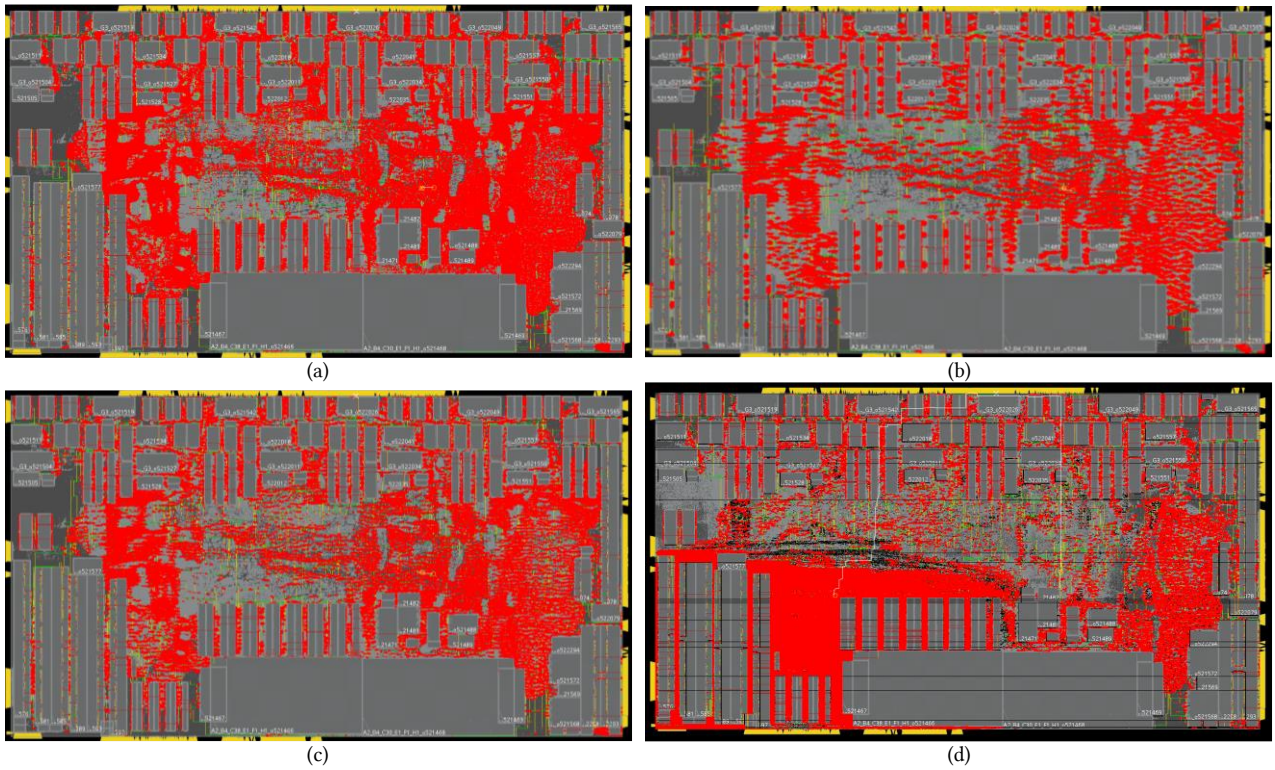


Figure 10. Full layouts (superblue4). (a) Non-clustered (b) Weighted K-means. (c) Effective mean shift. (d) Relocation using optimum sites.

4.3 Power and Timing Tradeoff

In the second experiment, we showed the power and timing tradeoff by adjusting the bandwidth to the distance to different M-th neighbors. Fig. 11 shows the corresponding results of superblue16, where timing is measured by TNS degradation, and power is measured by total clock sink power ratio. The top-left point indicates the register clustering result of weighted K-means [11]. For effective mean shift, $M = 0$ refers to the nearest neighbor (every register itself), i.e., bandwidth = 0, corresponding to non-clustered results. It can be seen that $M = 3$ brings the best power and timing tradeoff.

M	0	1	2	3	4	5
Max cluster size	1	30	35	55	78	98

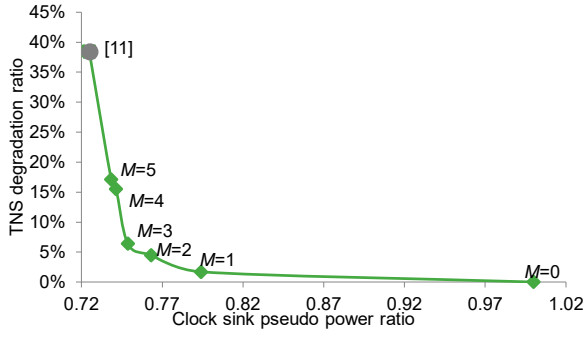


Figure 11. Clock sink power vs. TNS degradation (superblue16).

4.4 Parallelization

In the third experiment, we compared the parallel version (multi-threaded) with the sequential version (single threaded) of effective mean shift. Fig. 12 demonstrates the speedups achieved by effective mean shift on superblue18. It can be seen that effective mean shift has superior efficiency and scalability.

5 CONCLUSIONS

In this paper, we propose effective mean shift to naturally form clusters according to register distribution without placement disruption. Effective mean shift fulfills the requirements to be a good register clustering algorithm because it does not need a prespecified number of clusters, is insensitive to initializations, is

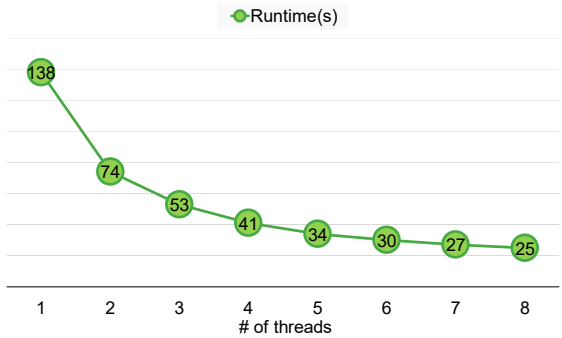


Figure 12. Speedups by parallelization (superblue18).

Table 6. Comparison on Timing and Power with Weighted K-Means (WK) [11] and Non-Clustered Design (NC).

Circuit	Method	Timing			Power				
		WNS	TNS (ns)	TNS Degradation Ratio	Clock Routed WL (um)	Ratio	#Buffers	Ratio	Clock Sink Power Ratio
superblue16	NC	-6.2	-1532.0	0.00%	934,654	100.00%	3,414	100.00%	100.00%
	WK	-6.6	-2120.9	-38.44%	196,543	21.03%	1,872	54.83%	72.47%
	Ours	-6.2	-1629.8	-6.38%	214,560	22.96%	1,873	54.86%	74.86%
superblue18	NC	-9.1	-5148.3	0.00%	629,463	100.00%	2,449	100.00%	100.00%
	WK	-9.4	-5834.8	-13.33%	143,471	22.79%	1,314	53.65%	72.47%
	Ours	-9.1	-5250.0	-1.98%	144,009	22.88%	1,228	50.14%	74.32%
superblue4	NC	-9.7	-15669.9	0.00%	1,017,709	100.00%	4,303	100.00%	100.00%
	WK	-10.1	-16738.6	-6.82%	214,560	21.08%	2,124	49.36%	72.47%
	Ours	-9.9	-15830.8	-1.03%	234,966	23.09%	2,072	48.15%	74.91%
superblue5	NC	-30.2	-19866.8	0.00%	928,619	100.00%	3,626	100.00%	100.00%
	WK	-32.3	-20607.3	-3.73%	273,496	29.45%	2,251	62.08%	72.51%
	Ours	-30.3	-19898.6	-0.16%	291,267	31.37%	2,355	64.95%	74.16%
superblue3	NC	-18.9	-7892.9	0.00%	1,047,502	100.00%	4,251	100.00%	100.00%
	WK	-19.7	-8584.5	-8.76%	266,706	25.46%	2,054	48.32%	72.48%
	Ours	-18.9	-8106.1	-2.70%	262,588	25.07%	2,133	50.18%	74.14%
superblue1	NC	-10.2	-6778.5	0.00%	1,047,502	100.00%	3,759	100.00%	100.00%
	WK	-10.5	-7825.5	-15.45%	262,261	25.04%	2,052	54.59%	72.47%
	Ours	-10.2	-7334.7	-8.21%	255,708	24.41%	2,104	55.97%	74.87%
superblue7	NC	-19.4	-12531.2	0.00%	1,702,650	100.00%	6,482	100.00%	100.00%
	WK	-20.9	-13591.3	-8.46%	362,256	21.28%	3,427	52.87%	72.48%
	Ours	-19.2	-12757.0	-1.80%	379,577	22.29%	3,341	51.54%	74.31%
superblue10	NC	-48.7	-151000.0	0.00%	1,660,396	100.00%	6,189	100.00%	100.00%
	WK	-42.7	-139000.0	7.95%	379,246	22.84%	3,210	51.87%	72.48%
	Ours	-42.3	-141000.0	6.62%	408,500	24.60%	3,495	56.47%	74.25%
Average	NC			0.00%		100.00%		100.00%	100.00%
	WK			-10.88%		23.62%		53.45%	72.48%
	Ours			-1.95%		24.58%		54.03%	74.48%

robust to outliers, is tolerant of various register distributions, is efficient and scalable, and balances clock power reduction against timing degradation. Experimental results show that our approach outperforms state-of-the-art work on power and timing balancing; we deliver similar clock power reduction with minor timing degradation. For efficiency and scalability, our method achieves 39X (sequential version) and 215X (parallel version) speedups. Future work includes the extension of effective mean shift to global placement.

REFERENCES

[1] A. Ranjan. 2015. Micro-architectural exploration for low power design. (November 2015). Semiconductor Engineering. Retrieved from https://semiengineering.com/micro-architectural-exploration-for-low-power-design/

[2] K. Brock. 2016. Six ways to exploit the advantages of finFETs. (November 2016.). Tech Design Forum. Retrieved from http://www.techdesignforums.com/practice/technique/six-ways-to-exploit-the-advantages-of-finfets/

[3] L. Rizzatti. 2015. Dynamic power estimation hits limits of SoC designs. (May 2015). Retrieved from https://www.eetimes.com/author.asp?section_id=36&doc_id=1326542

[4] S. I. Ward, N. Viswanathan, N. Y. Zhou, C. C. N. Sze, Z. Li, C. J. Alpert, and D. Z. Pan. 2013. Clock power minimization using structured latch templates and decision tree induction. In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD '13)*. IEEE, Piscataway, NJ, USA, 599-606.

[5] D. A. Papa, C. J. Alpert, C. C. N. Sze, Z. Li, N. Viswanathan, G.-J. Nam, I. L. Markov. 2011. Physical synthesis with clock-network optimization for large systems on chips. *IEEE Micro* 31, 4 (July 2011), 51–62.

[6] M. P.-H. Lin, C. C. Hsu and Y.-T. Chang. 2011. Post-placement power optimization with multi-bit flip-flops. *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)* 30, 12 (December 2011), 1870–1882.

[7] I. H.-R. Jiang, C.-L. Chang, and Y.-M. Yang. 2012. INTEGRA: Fast multibit flip-flop clustering for clock power saving. *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)* 31, 2 (February 2012), 192–204. Also see in *Proc. Int'l Symp. on Physical Design (ISPD '11)*. ACM, New York, NY, 115–121.

[8] S.-H. Wang, Y.-Y. Liang, T.-Y. Kuo, and W.-K. Mak. 2012. Power-driven flip-flop merging and relocation. *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)* 31, 2 (February 2012), 180–191. Also see in *Proc. Int'l Symp. on Physical Design (ISPD '11)*. ACM, New York, NY, 107–114.

[9] S. S.-Y. Liu, W.-T. Lo, C.-J. Lee, and H.-M. Chen. 2013. Agglomerative-based flip-flop merging and relocation for signal wirelength and clock tree optimization. *ACM Trans. Design Automation Electronic Systems (TODAES)* 18, 3, Article 40 (July 2013), 20 pages.

[10] C.-C. Tsai, Y. Shi, G. Luo, and I. H.-R. Jiang. 2013. FF-Bond: Multi-bit flip-flop bonding at placement. In *Proc. Int'l Symp. on Physical Design (ISPD '13)*. ACM, New York, NY, 147–153.

[11] G. Wu, Y. Xu, D. Wu, M. Ragupathy, Y.-Y. Mo, and C. Chu. 2016. Flip-flop clustering by weighted K-means algorithm. 2016. In *Proc. Design Automation Conf. (DAC '16)*. ACM, New York, NY, Article 82, 6 pages.

[12] A. B. Kahng, J. Li, and L. Wang. 2016. Improved flop tray-based design implementation for power reduction. In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD '16)*. ACM, New York, NY, Article 20, 8 pages.

[13] I. Seitanidis, G. Dimitrakopoulos, P. M. Mattheakis, L. Masse-Navette, D. Chinnery. 2018. Timing-driven and placement-aware multi-bit register composition. *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)*, early access. Also see in *Proc. Design Automation Conf. (DAC '17)*. ACM, New York, NY, Article 56, 6 pages.

[14] S. P. Lloyd. 1982. Least square quantization in PCM. *IEEE Trans. Information Theory* 28, 2 (March 1982), 129–137.

[15] K. Fukunaga and L.D. Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory* 21 (January 1975), 32–40.

[16] Y. Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis Machine Intelligence (TPAMI)* 17, 8 (August 1995), 790–799.

[17] D. Comaniciu and P. Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis Machine Intelligence (TPAMI)* 24, 5 (May 2002), 603–619.

[18] D. Comaniciu, V. Ramesh, and P. Meer. 2001. The variable bandwidth mean shift and data-driven scale selection. In *Proc. Int'l Conf. on Computer Vision (ICCV '01)*. IEEE, Piscataway, NJ, USA, 438-445.

[19] D. Gale and L. S. Shapley. 1962. College admissions and the stability of marriage. *American Mathematical Monthly* 69 (1962), 9–14.

[20] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan. 2015. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD '15)*. IEEE, Piscataway, NJ, USA, 921–926.

[21] Innovus, Cadence, Inc.