Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 395260, 14 pages
doi:10.1155/2012/395260

Research Article

A New Length-Based Algebraic Multigrid Clustering Algorithm

L. Rakai, A. Farshidi, L. Behjat, and D. Westwick

Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW Calgary, AB, Canada T2N 1N4

Correspondence should be addressed to L. Behjat, laleh@ucalgary.ca

Received 21 December 2011; Revised 24 February 2012; Accepted 25 February 2012

Academic Editor: Rached Tourki

Clustering algorithms have been used to improve the speed and quality of placement. Traditionally, clustering focuses on the local connections between cells. In this paper, a new clustering algorithm that is based on the estimated lengths of circuit interconnects and the connectivity is proposed. In the proposed algorithm, first an a priori length estimation technique is used to estimate the lengths of nets. Then, the estimated lengths are used in a clustering framework to modify a clustering technique based on algebraic multigrid (AMG), that finds the cells with the highest connectivity. Finally, based on the results from the AMG-based process, clusters are made. In addition, a new physical unclustering technique is proposed. The results show a significant improvement, reductions of up to 40%, in wire length can be achieved when using the proposed technique with three academic placers on industry-based circuits. Moreover, the runtime is not significantly degraded and can even be improved.

1. Introduction

Clustering is usually employed during the large-scale placement problems encountered in today's circuits, to speed up the placement process and improve the solution quality. The clustering algorithms used today are based on finding small groups of cells with high connectivity and putting each of them in a cluster. This scheme has proven effective, to a large extent, as there is a high correlation between the cells' connectivity and the lengths of the nets that connect them [1, 2]. Hence, by clustering cells that are close to one another, the lengths of the nets between them, and hence the total wire length will be reduced.

In this paper, a new perspective on how to cluster cells is proposed, where clustering decisions are made based on estimates of the lengths of nets connecting the cells. If a net is estimated to be short, then its cells are expected to be physically closer to each other, and the cells can be put in a cluster. The opposite is true for a net which has a high estimated length.

The proposed algorithm consists of three phases. In the first phase, a state-of-the-art length estimation technique is used to estimate the lengths of individual nets before placement.

Accurate length estimates are then used to direct the clustering decisions. These length estimates are used in the second phase to build a length-based proximity matrix. Then, it is proposed to use the proximity matrix in an AMG-based clustering framework to find clusters of the circuit. The AMG-based clustering finds seed cells and assigns scores based on the whole proximity matrix, not only the local connections. Hence, it can be more effective in finding a more globally optimal clustering configuration.

In the third phase, clusters are formed, and an initial placement is performed on the new clustered circuit. The initial placement solution is then refined in the unclustering phase, where a new technique for unclustering is proposed.

The major contributions of this paper are as follows:

(i) using estimated lengths when deciding which cells should be clustered;

(ii) designing and implementing a length-based AMG clustering algorithm;

(iii) proposing and implementing an unclustering refinement algorithm;

(iv) improving the runtime of the placement process;

(v) illustrating the effectiveness of multilevel length-based AMG clustering.

The rest of this paper is organized as follows. In Section 2, a literature review of existing net length estimation methods, an introduction to AMG, and a background of clustering techniques are presented. In Section 3, the proposed length-based clustering algorithm is described in detail. The proposed model efficacy is examined and validated by several experiments in Section 4. Conclusions are given in Section 5.

2. Background

In this section, a literature review of the three main components of the algorithm are given. As these components are distinct subjects not all normally used in the context of VLSI placement, the literature review is divided into three distinct sections: length estimation (2.1), AMG (2.2), and finally, clustering algorithms (2.3).

2.1. Net Length Estimation Techniques. Several a priori length estimation techniques, for example [3–6], have been proposed that try to estimate the length of individual nets. Older techniques, such as [7–12], can only estimate the average length of a set of nets or the distributions of net lengths.

In [6], a variable referred to as the intrinsic shortest path length (ISPL) is developed and used to estimate the individual net lengths. Although the estimation results obtained using this technique are exceptional, this approach is not very useful for modern mixed-size circuits since it only considers cells with unit area and nets with degree two.

Different properties of nets and cells are modeled by several variables in [3]. These variables are then used to make a third-order polynomial model for length estimation. The estimation results are well correlated for relatively small circuits that only include standard cells. However, none of these variables considers the effects of macro blocks on net lengths, and so the estimation results for mixed-size circuits are unreliable. In [5], this technique is further studied using a quadratic polynomial. Three new variables are proposed to account for the effects of macro blocks. The estimation results for modern mixed-size circuits show around 10% improvement over those obtained using [3].

Some applications of a priori net length estimation techniques are introduced in [4, 13, 14]. A variable called mutual contraction, is proposed and used for net length estimation, in [13, 14]. This variable is then used in placement. The estimated net lengths are utilized to quantify the quality of potential clusters. In [4], another application of preplacement length estimation is proposed where the negative effects of clustering are corrected based on the estimation results.

2.2. Algebraic Multigrid. A challenging and frequently encountered problem in various domains is to solve a large system of linear equations as in

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ is a large, sparse square matrix, $\mathbf{x}$ is a vector containing the unknown variables, and $\mathbf{b}$ is the known right-hand side vector. When $\mathbf{A}$ is very large, solving this system of equations directly becomes very expensive, in general. In the circuit placement domain, this equation arises from minimizing a quadratic length objective where the dimensions of $\mathbf{A}$ are on the order of millions.

The algebraic multigrid (AMG) technique uses a multilevel framework to approximately solve (1), hence reducing the computational cost. Several AMG algorithms have been developed [15–21]. Each one of these methods is suited for a specific type of problem where the matrix $\mathbf{A}$ has certain properties. General methods such as [15, 17, 20] have sufficient convergence conditions and can easily be adapted to more problems.

A typical AMG technique consists of three steps: coarsening, direct solution, and interpolation. Coarsening is used to decrease the size of the matrix $\mathbf{A}$ by keeping the essential elements of it and disregarding the nonessential elements. Once the matrix size has been adequately reduced, the second step starts. In this step, the reduced problem is solved exactly using a direct method. In the final step, interpolation, the solution obtained after the direct method step is projected back onto the sequence of larger systems of equations obtained during coarsening to achieve a final solution.

Another difference between the AMG framework and other direct methods is that in AMG, instead of solving for $\mathbf{x}$ directly, the difference or the error, $\mathbf{e}^0$, between the initial guess for $\mathbf{x}^0$ and the actual value of $\mathbf{x}$ is driven to zero. This is accomplished by solving the residual equation $\mathbf{A}^l\mathbf{e}^l = \mathbf{r}^l$ for level l, where $\mathbf{r}$ is the residual that is calculated as $\mathbf{r}^l = \mathbf{b}^l - \mathbf{A}^l\mathbf{x}^l$. A multilevel AMG scheme is illustrated in Figure 1. In this figure, in each level of coarsening, a restriction matrix, $\mathbf{W}_l^{l+1}$, is calculated. This matrix contains fewer rows than columns. To reduce the size of the equation matrix at level l, $\mathbf{A}^l$, is premultiplied by $\mathbf{W}_l^{l+1}$ and postmultiplied by $(\mathbf{W}_l^{l+1})^T$, the interpolation matrix, and the equation matrix at level $l+1$, $\mathbf{A}^{l+1} = \mathbf{W}_l^{l+1}\mathbf{A}^l(\mathbf{W}_l^{l+1})^T$ is obtained [16]. Thus, $\mathbf{A}^{l+1}$ has smaller dimensions than $\mathbf{A}^l$. At the lowest level, L, an exact solution for $\mathbf{e}^L$ is found, that is, $\mathbf{e}^L = (\mathbf{A}^L)^{-1}\mathbf{r}^L$ as shown in the figure. In the interpolation stage, the error is propagated to the higher levels using the appropriate interpolation matrix. Finally, using the calculated error in the highest level, the solution approximation $\mathbf{x}^0$ is updated: $\mathbf{x}^0 \leftarrow \mathbf{x}^0 + \mathbf{e}^0$. The relaxation steps shown in the figure refer to using a fast, iterative method to improve the solution at the current level.

2.3. Clustering Algorithms. In different stages of VLSI physical design, clustering algorithms are used to reduce the sizes of circuits. Several clustering algorithms have been proposed to handle the modern large-size circuits, for example, [1, 2, 22–26]. Most placement techniques, such as the ones that competed in the ISPD 2006 [27] placement contest, use clustering algorithms.

Clustering algorithms such as heavy-edge matching [22], Edge Coarsening [28], FirstChoice [26], and PinEC [23] are similar in how they choose and form clusters. In these algorithms, a cell is chosen randomly to become the *seed* for a cluster. A seed refers to the first cell of a cluster. Once a seed has been chosen, a cell that measures the highest in connectivity with the seed is added to the cluster. These

Algorithm AMG-LE: AMG-based clustering with length estimation
Input: Circuit net list, AMG parameters
Output: Clustered circuit
1. Pre-placement individual net length estimation
2. Proximity matrix construction using the estimated lengths
3. AMG-based coarsening on the proximity matrix
4. Cluster seed cell and score assignment
5. Final cluster formation

Algorithm 1: The high-level flow of the AMG-LE clustering algorithm.

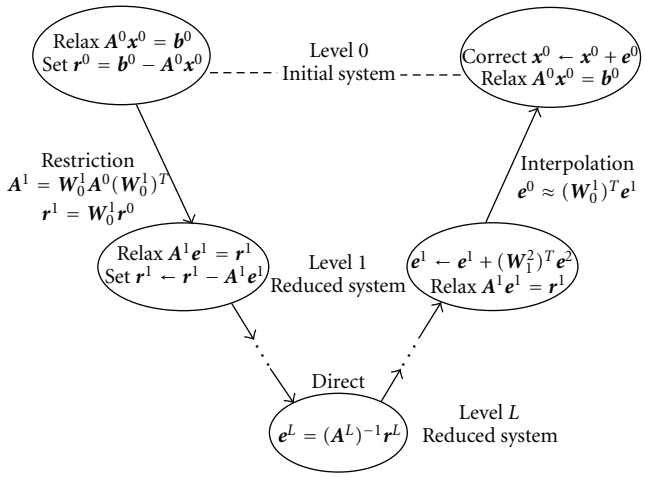


Figure 1: A multilevel AMG scheme.

algorithms are easy to implement, and the runtimes for them are typically low. However, as no comparisons between potential clusters are performed, the quality of the results can be low.

In an effort to improve results, algorithms, such as edge-separability [24], fine-granularity clustering (FGC) [25], best-choice clustering [1], and Net Cluster [2], first prepare a set of potential clusters and then either refine the potential clusters or only finalize potential clusters that have high scores.

In [29, 30], a clustering algorithm for wire length-driven placement, called SafeChoice, is proposed where a condition is used to check if potential clusters will degrade the placement solution. If a cluster passes the check, it is referred to as a safe cluster, and the cluster is finalized.

In [31], a linear-time AMG-based clustering technique is proposed. This AMG-based algorithm uses only the connectivity matrix of a circuit to form clusters.

3. The Proposed Length-Driven Multilevel Placement Framework

3.1. The Proposed Length Estimation-Based AMG Clustering Algorithm. The main objective of the placement phase in the physical design of circuits is to reduce the total wire length of the circuit. Clustering algorithms are normally used during placement to improve the total wire length and runtime. However, clustering is performed based on circuit connectivity and not the wire length. In this paper, an algorithm that performs clustering based on estimated wire lengths is proposed. The novelty of the algorithm is in using the estimated lengths instead of connectivities in determining the best cells to be clustered.

The algorithm has five main stages: preplacement length estimation, proximity matrix construction, AMG-based coarsening, cluster seed cell and score assignment, and final cluster formation, as summarized in Algorithm 1.

3.1.1. Preplacement Length Estimation. In an estimation technique, first a set of model parameters needs to be calculated. In this work, the parameters used or developed in [5] are employed as these have been shown to be a comprehensive set of parameters that are well suited to the mixed-size circuits used today. These parameters include local net characteristics, such as the half perimeter of the cells of each net and global characteristics, such as the number of degree-two nets in the design.

Once the model parameters have been selected, an estimation technique should be used to fit the parameters into a model. The model parameters in [5] are used to fit a quadratic model. Any terms of the resulting model which have small coefficients, ineffective terms, are pruned from the model. The remaining terms, effective terms, are used to make the estimation model and calculate the length estimates.

It is worth noting that any other individual net length estimation technique could be used in this step. The model in [5] is selected as it includes several factors such as macro cells but is independent of the placer to be used allowing for broader application of this work.

To better illustrate the algorithm flow, a small example circuit is presented in Figure 2. In this figure, the nets are annotated with the estimated lengths. The estimated lengths are found using a simplified version of the technique in [5]. Due to the small size of the example circuit, only the variables associated with the cell dimensions and net degree are used. These lengths will be used in the following sections to perform clustering using the proposed AMG-LE algorithm.

3.1.2. Proximity Matrix Construction. To perform circuit clustering, the circuit's net list should be represented using

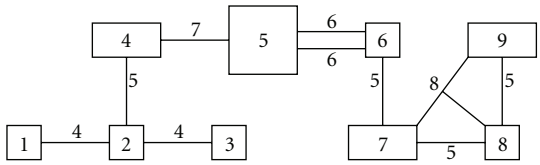


Figure 2: An example circuit with nets annotated with estimated lengths. The dimensions of all cells are 2 × 2 except cells 4, 7, and 9 that are 4 × 2, and cell 5 which is 4 × 4.

a matrix. Normally, this matrix is the connectivity matrix of the circuit which shows the total number of weighted connections between two cells. In this paper, a new matrix, called the proximity matrix, is designed which shows how close two cells are predicted to be. The proximity matrix is later used for the AMG-based clustering.

It is shown in [16] that for the best AMG performance the matrix **A** of (1), which in this case represents the net list, should be an *M-matrix*, that is, a symmetric, positive-definite matrix with positive diagonal and nonpositive offdiagonal elements. In the proposed AMG-based clustering technique, a proximity matrix $\mathbf{P}^0$, which is an M-matrix, is constructed. The rows and columns of $\mathbf{P}^0$ represent the cells of the circuit. The element $p^0_{i,j}$ shows the connectivity between cell i and cell j. Each $p^0_{i,j}$ is determined as follows: consider two cells, i and j, which are treated as points in AMG, and let $N_{i,j}$ be the set of nets, n_k, that contain both cells i and j. Then, element $p^0_{i,j}$ is calculated as

$$p^0_{i,j} = \begin{cases} \displaystyle\sum_{n_k \in N_{i,j}} -\,\mathrm{il}(n_k), & i \neq j, N_{i,j} \neq \varnothing, \\ \displaystyle\sum_{n_k \in N_{i,i}} \mathrm{il}(n_k), & i = j, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathrm{il}(\cdot)$ represents the inverse of the estimated length of a net and is defined as

$$\mathrm{il}(n_k) = \frac{1}{l_{\mathrm{est}}(n_k)}, \tag{3}$$

where $l_{\mathrm{est}}(n_k)$ denotes the estimated length of a net obtained from the model described in Section 3.1.1. The inverse of the estimate is used because nets whose estimated lengths are small have a high proximity and vice versa. Using the formulation in (2), any nonzero offdiagonal element of the proximity matrix is equal to the negative sum of the inverses of the estimated lengths of nets between i and j, and any diagonal element is equal to the sum of all the inverses of nets connected to i. This matrix is a positive-semidefinite M-matrix which is highly desirable for AMG. In addition, multiterminal nets and multiple connections between cells are handled by (2).

As an example, the proximity matrix for the example circuit in Figure 2 is given in (4). For any two cells i and j that are not connected, the $p^0_{i,j}$ entry is equal to zero. For any two cells that are connected, the $p^0_{i,j}$ entry is equal to the negative sum of the inverse of estimated lengths of the nets connecting them. As an example, cells 5 and 6 are connected by two nets each with estimated length of six. Therefore, $p^0_{5,6} = -(1/6 + 1/6) = -1/3$. Finally, the diagonal elements are the negative sum of the off-diagonal elements in the corresponding row of the matrix. For example, $p^0_{5,5} = -(p^0_{5,4} + p^0_{5,6}) = -(-1/7 - 1/3) = 10/21$.

$$\mathbf{P}^0 = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & \frac{7}{10} & -\frac{1}{4} & -\frac{1}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{5} & 0 & \frac{12}{35} & -\frac{1}{7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{7} & \frac{10}{21} & -\frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{8}{15} & -\frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{5} & \frac{13}{20} & -\frac{13}{40} & -\frac{1}{8} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{13}{40} & \frac{13}{20} & -\frac{13}{40} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{13}{40} & \frac{9}{20} \end{pmatrix} \tag{4}$$

3.1.3. AMG-Based Coarsening on the Proximity Matrix. Once the proximity matrix $\mathbf{P}^0$ has been constructed, the coarsening algorithm in [31] is used to reduce $\mathbf{P}^0$ and construct $\mathbf{P}^1$ and the associated restriction matrix $\mathbf{W}^1_0$. The coarsening can be loosely thought of as a heuristic for selecting a maximally independent subset of cells which are the most significant in $\mathbf{P}^0$. The significance of each cell is measured by the number of cells that *strongly connect* to it. A parameter, $\theta \in (0, 1]$, is used in the definition of the strength of a connection. Cell i is strongly connected to cell j if

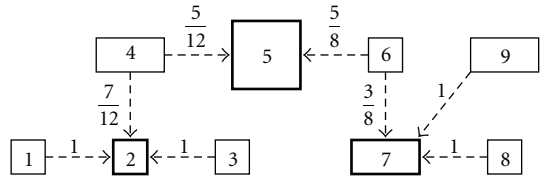
$$\left| p^l_{i,j} \right| \geq \theta \times \max_{k \neq i} \left\{ \left| p^l_{i,k} \right| \right\}, \quad k = 1, \ldots, C^l, \tag{5}$$

where, $p^l_{i,j}$ is the $(i, j)^{th}$ element of $\mathbf{P}^l$, and C^l is the number of cells in level l. The relationship between the parameter θ and the amount coarsening performed is complex. If θ is close to one, more aggressive coarsening will be performed. However, more aggressive coarsening can overly simplify the reduced matrix and the associated reduced circuit. An interpretation of the strength of connection criteria in the context of the proposed proximity matrix is that if a cell is in several nets with small estimated length it is likely to be a part of the reduced circuit.

3.1.4. Seed Cell and Cluster Score Assignment. After coarsening the proximity matrix, the cells that are selected to form the reduced circuit become seed cells used for clustering. The cells which are not selected to form the reduced circuit are considered to be clustered with the selected seed cells. This is the same technique used in [31]. In this technique, the entries of the AMG interpolation matrix $(\mathbf{W}^{l+1}_l)^T$ are used to rank each seed cell that a non-selected cell connects to. A large entry in the interpolation matrix means the seed cell is representative of the non-selected cell in the reduced circuit.

$$(\mathbf{W}_0^1)^T = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ \frac{7}{12} & \frac{5}{12} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{5}{8} & \frac{3}{8} \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

(a) interpolation Matrix $(\mathbf{W}_0^1)^T$



(b) pictorial representation

Figure 3: The interpolation matrix shown in matrix form and pictorial form for the example circuit in Figure 2. In the pictorial representation, the selected seed cells are shown in bold, and arrows are annotated with the interpolation weights of non-selected cells.

The selected seed cells for the example in Figure 2 are cells 2, 5, and 7. Therefore, the interpolation matrix has three columns, one for each seed cell with the first column corresponding to cell 2, and so forth. The number of rows is nine, which is the total number of cells and row 1 corresponds to cell 1, and so forth. The interpolation matrix is shown in Figure 3(a). Each seed cell interpolates from itself directly, so entries $w_{2,1}$, $w_{5,2}$ and $w_{7,3}$ are all one, where $w_{i,j}$ is the (i, j)th element of $(\mathbf{W}_0^1)^T$. The other non-zero entries represent the interpolation weight of non-selected cells to seed cells. As an example, cell 4 is connected to seed cells 2 and 5. The interpolation weights of cell 4 with seed cells 2 and 5 are given by entries $w_{4,1} = 7/12$ and $w_{4,2} = 5/12$, respectively. To better visualize the seed cell and cluster score assignment using the interpolation matrix, a pictorial representation of $(\mathbf{W}_0^1)^T$ is presented in Figure 3(b). In this figure, the seed cells are shown with a bold border. Each non-selected cell has an interpolation weight to the seed cells that they connect to. These interpolation weights annotate arrows pointing from non-selected cells to seed cells in Figure 3(b).

3.1.5. Final Cluster Formation. The final step of the clustering algorithm is to cluster non-selected cells to the seed cell which they interpolate from most strongly. A cluster is finalized as long as the area of the cluster is not greater than five times the average standard cell area and its interpolation weight is more than 0.9. The cluster area is restricted to prevent the formation of macro-sized clusters which require special treatment in placement algorithms. The interpolation weight threshold is set high to ensure that only the best quality clusters are formed. Each cluster meeting the constraints is given a physical shape in the clustered circuit with a height equal to the maximum height of all of its cells and a width equal to the sum of the widths of its cells.

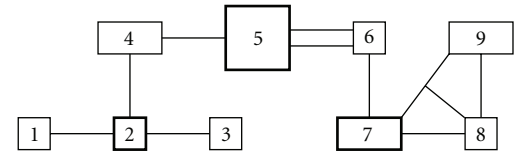
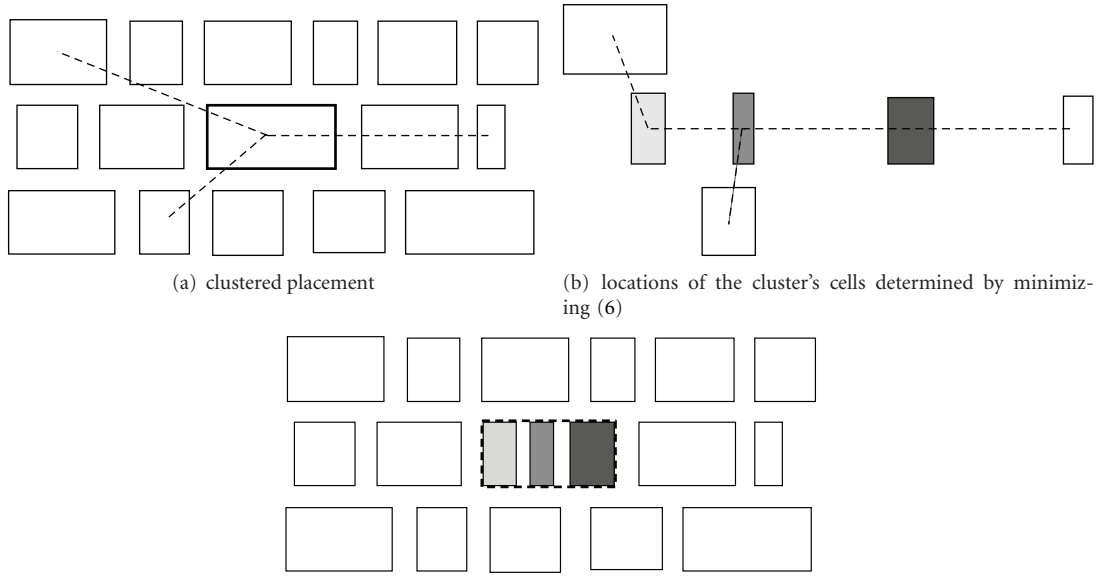


Figure 4: The final clusters for the example circuit in Figure 2 shown by dashed lines.

The final clusters for the example in Figure 2 are shown with dashed lines in Figure 4. The clusters are formed by clustering each non-selected cell with the seed cell that it has the maximum interpolation weight with. Assuming each cluster has an area of less than or equal to five times the average standard cell area, the clusters are finalized.

3.2. The Proposed Length-Driven Unclustering Technique. The description of a clustering algorithm alone is not enough to fully specify the implementation of a clustering algorithm used for placement. Just as important is the algorithm for unclustering, or determining the locations of cells within a cluster after the cluster's location has been determined. The description of this step is rarely mentioned in the literature of clustering algorithms.

In this work, a length-driven unclustering algorithm is proposed. The chief benefit of using the proposed technique is that a legal solution is preserved after unclustering.



(a) clustered placement

(b) locations of the cluster's cells determined by minimizing (6)

(c) final ordering of the cluster's cells

Figure 5: An illustration of the length-driven unclustering technique.

Global placement algorithms often produce legal solutions before performing detailed placement. A new unclustering technique which preserves the legality of the globally-placed solution improves the runtime in multilevel placement. With the legality restriction, the problem of unclustering reduces to ordering the cluster's cells. The proposed technique finds the ordering by solving a relaxed length-driven minimization and using the result to determine the ordering of the cells. In addition, the combination of preserving legality and restricting cluster areas means that the row in which the unclustered cells will be placed is also preserved. Consequently, the vertical components of the lengths can be ignored. This helps to reduce the disruption in the wire length before and after unclustering which is a significant problem for multilevel placement as discussed in [32].

The locations of cells, $\mathbf{x}_{C_i}$, in cluster C_i are determined by minimizing the quadratic matrix length objective

$$\mathbf{x}_{C_i}^T \mathbf{U} \mathbf{x}_{C_i} + \mathbf{t}^T \mathbf{x}_{C_i} + v, \tag{6}$$

where $\mathbf{U}$ is a weighted connectivity matrix between the cells of the cluster, that is, $i, j \in C_i$, and is defined as

$$u_{i,j} = \begin{cases} \sum_{n_k \in N_{i,j}} -\frac{1}{|n_k|}, & i \neq j, N_{i,j} \neq \emptyset, \\ \sum_{n_k \in N_{i,i}} \frac{1}{|n_k|}, & i = j, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $|n_k|$ is the degree of net n_k, that is, the number of cells in n_k. The vector $\mathbf{t}$ contains the weighted horizontal cell locations of cells outside of the cluster that each clustered cell connects to, that is, $i \in C_i$ and $j \notin C_i$ defined as

$$t_i = \begin{cases} \sum_{n_k \in N_{i,j}} -\frac{x_j}{|n_k|}, & i \neq j, N_{i,j} \neq \emptyset, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where x_j is the location of the cell or the cluster that a cell belongs to. The scalar v represents the length of all the nets not involving the cells in C_i and can be ignored. After minimizing (6), the cells are inserted into the area occupied by the cluster in the order of their locations in $\mathbf{x}_{C_i}$. An illustration of the length-driven unclustering is given in Figure 5. In Figure 5(a), a clustered placement is given. The highlighted cluster is the focus of the example and contains three cells (not shown in Figure 5(a)). The cluster is connected to three other cells via degree-two nets, shown by dashed lines in the illustration. The cells contained in the cluster are shown in shades of grey in Figure 5(b). The three external connections as well as the two internal connections are used to determine the non-zero values in the matrix $\mathbf{U}$. In this example all of the values for $|n_k|$ are equal to 2, the degree of each net. The locations of the cells not in the cluster, shown in white, are used in forming the vector $\mathbf{t}$. The locations of the cluster's cells in Figure 5(b) are determined by minimizing (6). These locations are used to determine the order of the cells inside of the cluster's area to complete the unclustering, illustrated in Figure 5(c). Because the unclustered cells remain inside of the cluster's area, the placement remains legal.

This technique does not take advantage of the locations of already unclustered cells as it iterates through the clusters. However, this also means that the technique is not sensitive to the order in which the clusters are visited. Furthermore, because the clusters are restricted to have an area of less

Table 1: Statistics of ICCAD04 benchmarks.

Circuit	Nets number	Cells number	$\lvert n_{\max}\rvert$	$\lvert c_{\max}\rvert$
ibm01	14,111	12,752	42	39
ibm02	19,584	19,601	134	69
ibm03	27,401	23,136	55	100
ibm04	31,970	27,507	46	425
ibm05	28,446	29,347	17	9
ibm06	34,826	32,498	35	91
ibm07	48,117	45,926	25	98
ibm08	50,513	51,309	75	1165
ibm09	60,962	53,395	39	173
ibm10	75,196	69,429	41	137
ibm11	81,454	70,558	24	174
ibm12	77,240	71,076	28	473
ibm13	99,666	84,199	24	180
ibm14	152,772	147,605	33	270
ibm15	186,608	161,570	33	306
ibm16	190,048	183,484	40	177
ibm17	189,581	185,495	36	81
ibm18	201,920	210,613	66	97

Table 2: Statistics of ISPD05 benchmarks.

Circuit	Nets number	Cells number	$\lvert n_{\max}\rvert$	$\lvert c_{\max}\rvert$
adaptec1	221,142	211,447	2,271	448
adaptec2	266,009	255,023	1,935	620
adaptec3	466,758	451,650	3,713	1224
adaptec4	515,951	496,045	3,974	416
bigblue1	284,479	278,164	2,621	388
bigblue2	577,235	557,866	11,869	119
bigblue3	1,123,170	1,096,812	7,623	1692

than five times the average standard cell area, the additional benefit of using the locations of already unclustered cells and a good ordering of the clusters is expected to be small.

4. Experimental Results

Several experiments have been carried out to verify the efficacy of the proposed clustering algorithm. These experiments are performed using the ICCAD04 benchmark circuits [33] released by IBM. The detailed statistics of these circuits are given in Table 1. In columns 2 and 3 of this table the number of nets and cells in each circuit are given, respectively. In columns 4 and 5, the maximum net degree, $\lvert n_{\max}\rvert$, and maximum cell degree, $\lvert c_{\max}\rvert$, are given.

To better show the scalability of the proposed clustering algorithm, the same experiments are also performed with the ISPD05 benchmark suite [34]. The statistics of these benchmark circuits are presented in Table 2. The benchmark circuit bigblue4 is not included in the experiments due to the memory limitations of the testing platform.

4.1. Multilevel Clustering Results Using the Proposed AMG-LE Technique. To evaluate the proposed clustering algorithm, its effectiveness in the context of multilevel circuit placement using multiple placers is illustrated. The experiments are performed using three high-quality academic placers: Capo 10.5 [35], Fastplace 3.0 [36], and mPL6 [37]. Fastplace and mPL are analytical placers and have clustering algorithms embedded inside their placement algorithms. To better evaluate the performance of the proposed technique, the internal clustering of these placers are disabled in the clustering experiments. If the internal clustering is enabled, the results will have noise and it will not be clear if the proposed clustering technique or the internal clustering is the cause of any benefits. Comparisons of the proposed technique with existing clustering techniques, including the technique used internally by Fastplace 3.0 and mPL6, are presented in Section 4.3. Capo is a partitioning-based placer and does not use internal clustering. Capo is, however, non-deterministic so the average of ten runs of each experiment are reported when results for Capo are given. The experiments are performed on a 64-bit dual core AMD opteron system running Linux with 8 GB of RAM. In addition, the parameter θ in (5) is set to 0.1 in the experiments.

Wire length and runtime results of the experiment are presented in Table 3(a) and 3(b), respectively. In both tables, Column 1 identifies the circuit that is placed. Columns 2–4, 5–7, and 8–10 show the results for Capo, Fastplace, and mPL, respectively. Each placer is used in three modes: baseline without clustering, one-level AMG-LE clustering with length-driven unclustering, and two-level AMG-LE clustering with length-driven unclustering. The columns showing one- and two-level AMG-LE are given in terms of the percentage improvement over the baseline. In this representation, positive percentages represent improvements.

The results in Table 3(a) show that AMG-LE is effective at improving wire length on average for all placers using one or two levels of clustering. A maximum improvement of 38.1% is achieved using one-level AMG-LE and Fastplace. Using one-level AMG-LE, every circuit for Capo, 16 circuits for Fastplace, and 10 circuits for mPL have improved wire length when compared to the baseline. When two-level AMG-LE is used, every circuit for Capo and Fastplace, and 12 circuits for mPL are improved.

The runtime results show that using the one-level scheme for mPL and the two-level scheme for all placers double the total runtime. For a placer, such as Fastplace, the increase in runtime may be more acceptable because it requires much less time than other placers. However, the increase for Capo and particularly for mPL are less acceptable. It should be noted that mPL does not offer an option to perform only detailed placement requiring full global and detailed placement to be performed between each level, which accounts for the significant increases compared to the other two placers. The experiment performed in the following section will illustrate how to improve the runtime results while maintaining, and even improving, the wire length results.

An interesting result arising from the experiment is that the placements resulting from using the AMG-LE framework

Table 3: Placement wire length and runtime results comparing one- and two-level AMG-LE clustering with length-driven unclustering to baseline placements without using clustering with three placers on the ICCAD04 benchmark suite.

(a) HPWL

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)
ibm01	25	**3.1**	**6.4**	24	−1.8	**1.3**	24	**4.6**	**4.2**
ibm02	51	**4.4**	**5.4**	54	**3.1**	**4.6**	52	**1.3**	**1.1**
ibm03	77	**1.8**	**9.8**	80	**5.1**	**6.8**	82	−7.0	−3.6
ibm04	93	**14.8**	**15.7**	86	**7.9**	**7.7**	109	−5.7	−4.4
ibm05	103	**1.1**	**2.1**	101	**0.7**	**0.5**	93	−5.2	−5.1
ibm06	67	**0.6**	**3.8**	99	**38.1**	**33.9**	88	**5.8**	**2.7**
ibm07	126	**9.3**	**11.9**	123	**7.4**	**12.2**	124	−3.1	−4.0
ibm08	137	**7.9**	**8.1**	147	−4.1	**10.1**	211	**4.7**	**4.2**
ibm09	145	**4.4**	**6.6**	155	**8.6**	**8.0**	189	**2.9**	**1.7**
ibm10	318	**4.0**	**4.9**	362	**10.8**	**12.2**	363	**2.0**	**1.8**
ibm11	210	**4.1**	**7.0**	225	**11.0**	**10.1**	243	−0.1	−2.6
ibm12	413	**10.9**	**14.5**	410	**11.4**	**14.4**	461	−1.1	**1.1**
ibm13	266	**3.4**	**7.6**	273	**12.1**	**11.8**	324	−0.4	**1.1**
ibm14	392	**2.6**	**4.0**	478	**14.5**	**21.7**	824	**6.5**	**6.6**
ibm15	544	**6.1**	**5.6**	577	**8.6**	**12.2**	1001	−2.9	−5.0
ibm16	629	**5.6**	**6.4**	680	**11.5**	**11.5**	931	**6.7**	**5.7**
ibm17	737	**2.7**	**3.8**	810	**9.3**	**12.2**	1144	**7.5**	**7.2**
ibm18	458	**2.1**	**3.3**	574	**19.0**	**19.8**	885	**11.0**	**10.7**
Average	—	**4.9**	**7.1**	—	**9.6**	**11.7**	—	**1.5**	**1.3**

(b) Runtime

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline (s)	1Lvl (%)	2Lvl (%)	Baseline (s)	1Lvl (%)	2Lvl (%)	Baseline (s)	1Lvl (%)	2Lvl (%)
ibm01	193	−3	−75	26	−5	−62	109	−69	−152
ibm02	329	−24	−112	46	−14	−72	236	−91	−171
ibm03	487	−21	−105	46	−20	−82	221	−108	−210
ibm04	512	−23	−110	51	−17	−90	250	−87	−195
ibm05	411	−25	−124	36	−35	−81	174	−118	−183
ibm06	580	−19	−115	96	29	−6	390	−83	−197
ibm07	920	−21	−105	96	−9	−82	387	−100	−198
ibm08	941	−20	−121	99	−31	−158	1109	−105	−197
ibm09	1198	−29	−119	122	−6	−66	898	−136	−221
ibm10	1868	−12	−120	286	4	−62	1450	−126	−210
ibm11	1834	−30	−121	135	−66	−139	1084	−84	−193
ibm12	1996	−22	−124	282	−12	−52	1517	−115	−222
ibm13	2387	−16	−120	239	−43	−118	1235	−110	−218
ibm14	3327	−29	−129	530	−42	−121	2189	−93	−213
ibm15	5781	−30	−142	624	−57	−197	3969	−101	−247
ibm16	4917	−35	−132	722	−75	−217	5457	−57	−118
ibm17	5286	−31	−135	1133	−33	−101	2788	−131	−228
ibm18	4215	−39	−142	1589	3	−57	3106	−125	−257
Average	—	−24	−119	—	−24	−98	—	−102	−202

Table 4: Placement wire length and runtime results comparing one- and two-level AMG-LE clustering with length-driven unclustering to baseline placements without using clustering with three placers on the ISPD05 benchmark suite.

(a) HPWL

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)
adaptec1	91	**1.5**	**2.3**	87	**8.6**	**8.7**	81	−1.1	−1.1
adaptec2	120	**14.2**	**14.5**	108	**4.0**	**9.6**	97	−0.1	−0.1
adaptec3	254	**6.4**	**5.7**	287	**15.4**	**17.2**	224	**0.6**	**0.6**
adaptec4	—	—	—	230	**8.6**	**12.7**	195	−0.5	−0.5
bigblue1	114	**4.0**	**3.3**	107	**5.5**	**7.8**	101	**0.0**	**0.0**
bigblue2	167	**2.7**	**2.3**	181	**8.5**	**12.3**	152	−0.3	−0.3
bigblue3	439	**3.6**	**6.4**	663	**37.0**	**40.1**	492	**1.7**	**1.7**
Average	—	**5.4**	**5.7**	—	**12.5**	**15.5**	—	**0.0**	**0.0**

(b) Runtime

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)
adaptec1	30	−24	−73	5	−79	−109	18	−80	−109
adaptec2	44	2	−49	13	−15	−4	17	−134	−211
adaptec3	81	−32	−107	28	−136	−77	65	−89	−135
adaptec4	—	—	—	21	−208	−184	45	−258	−199
bigblue1	54	11	−36	6	−158	−141	18	−128	−173
bigblue2	92	−64	−100	23	−299	−278	49	−235	−286
bigblue3	254	−24	−49	139	−45	−78	123	−323	−365
Average	—	−22	−69	—	−134	−125	—	−178	−211

are more correlated with the length estimates than the baseline placements. The correlation improvement is small, with a maximum improvement of 6% for Capo using two-level AMG-LE, but does suggest that the estimates are directing the placement.

In order to assess the scalability of the proposed clustering and unclustering algorithms, the same experiments are performed on the ISPD05 circuits. These circuits are bigger than the ICCAD04 benchmarks and have significantly larger maximum net degree. The results are tabulated in Table 4. The version of Capo used in the experiment could not place the adaptec4 benchmark on the test platform so its entries in Table 4 are not included. In general, the results resemble those of Table 3 with Capo and Fastplace obtaining significant wire length improvement for both one- and two-level AMG-LE. Meanwhile, only a negligible average improvement is achieved by mPL. The runtime increases for all placers with mPL having the largest increase in runtime, which is not surprising given that mPL must perform global placement in addition to detailed placement at each level. Capo has a more modest increase in runtime compared to the other two placers, on average. In this case, the placements for mPL are nearly identical whether one or two levels of AMG-LE clustering are performed. This means that the effects of the second level of clustering are almost entirely removed by performing global placement on the unclustered circuit.

4.2. Length-Driven Unclustering as a Detailed Placer. In this section, the same multilevel experiments are performed with the difference of not performing any placement apart from placing the bottommost clustered circuit. The circuit is placed at the bottom level using the global and detailed placement functions of each placer (if the two are distinguished). As a result, the placement at the bottom level is legal, that is, free from overlaps. To improve upon the runtime results given in the previous section, it is proposed to only use the length-driven unclustering technique as the only refinement between levels. The placement will preserve its legality using the technique mentioned in Section 3.2. The results of this experiment are given in Table 5.

The format of the Tables 5(a) and 5(b) is the same as Table 3. It can be seen that all placers' wire lengths are improved on average using either one- or two-level AMG-LE. The best wire length improvement is again for Fastplace with one-level AMG-LE on ibm06 with a 35.8% improvement. Using one-level AMG-LE 16, 15, and 14 circuits improve for Capo, Fastplace, and mPL, respectively. Improvement is observed in 14, 16, and 15 circuits for Capo, Fastplace,

Table 5: Placement wire length and runtime results comparing one- and two-level AMG-LE clustering with length-driven unclustering without detailed placement to baseline placements without using clustering with three placers on the ICCAD04 benchmark suite.

(a) HPWL

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^5$)	1Lvl (%)	2Lvl (%)
ibm01	25	−1.0	−1.7	24	−7.7	−4.9	24	−3.2	−3.4
ibm02	51	**1.6**	**1.2**	54	**1.3**	**2.5**	52	−3.8	−3.8
ibm03	77	**4.2**	**4.6**	80	**1.3**	**3.4**	82	**5.0**	**5.1**
ibm04	93	**10.6**	**10.4**	86	**4.0**	**4.7**	109	**11.3**	**15.9**
ibm05	103	**0.3**	−1.1	101	−1.9	−2.7	93	−10.1	−13.4
ibm06	67	**1.3**	**0.3**	99	**35.8**	**31.1**	88	**15.4**	**17.1**
ibm07	126	**10.2**	**9.9**	123	**3.3**	**7.0**	124	−4.4	**1.2**
ibm08	137	**2.6**	**1.9**	147	−3.0	**3.3**	211	**13.0**	**20.0**
ibm09	145	**0.6**	**1.5**	155	**4.3**	**3.9**	189	**11.9**	**15.9**
ibm10	318	−1.0	−0.5	362	**7.4**	**9.0**	363	**3.2**	**5.7**
ibm11	210	**1.9**	**1.9**	225	**7.6**	**6.3**	243	**4.3**	**4.1**
ibm12	413	**9.6**	**9.6**	410	**7.4**	**10.4**	461	**6.4**	**6.9**
ibm13	266	**3.3**	**3.4**	273	**7.7**	**7.3**	324	**9.7**	**14.3**
ibm14	392	**2.4**	**1.0**	478	**11.5**	**18.4**	824	**30.0**	**35.2**
ibm15	544	**2.6**	−0.1	577	**5.0**	**8.6**	1001	**23.3**	**23.4**
ibm16	629	**5.1**	**2.5**	680	**8.5**	**7.9**	931	**16.8**	**21.8**
ibm17	737	**1.9**	**1.2**	810	**7.0**	**9.0**	1144	**13.0**	**21.5**
ibm18	458	**0.9**	**0.0**	574	**15.3**	**15.8**	885	**18.8**	**28.9**
Average	—	**3.2**	**2.6**	—	**6.4**	**7.8**	—	**8.9**	**12.0**

(b) Runtime

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline (s)	1Lvl (%)	2Lvl (%)	Baseline (s)	1Lvl (%)	2Lvl (%)	Baseline (s)	1Lvl (%)	2Lvl (%)
ibm01	193	16	12	26	−12	−22	109	47	54
ibm02	329	−8	2	46	1	−28	236	27	38
ibm03	487	−5	5	46	−16	−36	221	10	32
ibm04	512	−1	−4	51	−13	−42	250	39	40
ibm05	411	−1	−1	36	−29	−63	174	−2	32
ibm06	580	12	5	96	34	15	390	39	51
ibm07	920	7	8	96	−8	−57	387	28	35
ibm08	941	6	−11	99	−22	−110	1109	62	67
ibm09	1198	−8	1	122	−3	−42	898	55	62
ibm10	1868	3	−3	286	8	−11	1450	8	38
ibm11	1834	−1	−5	135	−64	−94	1084	56	56
ibm12	1996	−1	−7	282	−3	−17	1517	29	39
ibm13	2387	−7	−6	239	−32	−81	1235	54	56
ibm14	3327	−2	−14	530	−51	−125	2189	27	49
ibm15	5781	0	−22	624	−38	−188	3969	32	43
ibm16	4917	−4	−11	722	−58	−167	5457	50	53
ibm17	5286	−4	−26	1133	−40	−84	2788	3	12
ibm18	4215	−8	−36	1589	2	−43	3106	2	14
Average	—	0	−6	—	−19	−66	—	32	43

Table 6: Placement wire length and runtime results comparing one- and two-level AMG-LE clustering with length-driven unclustering without detailed placement to baseline placements without using clustering with three placers on the ISPD05 benchmark suite.

(a) HPWL

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^6$)	1Lvl (%)	2Lvl (%)
adaptec1	91	−1.4	−2.7	87	**1.8**	−1.6	81	**5.3**	**3.1**
adaptec2	120	**12.6**	**10.8**	108	−0.2	**3.6**	97	**13.7**	**12.9**
adaptec3	254	**4.0**	**2.0**	287	**12.0**	**10.8**	224	**16.4**	**15.3**
adaptec4	—	—	—	230	**4.9**	**7.4**	195	**8.0**	**7.6**
bigblue1	114	**1.1**	**0.1**	107	−1.7	−2.6	101	−0.4	−2.3
bigblue2	167	**1.0**	−0.4	181	**0.8**	**2.0**	152	**3.2**	**1.6**
bigblue3	439	**4.3**	**4.6**	663	**35.7**	**37.6**	492	**30.4**	**29.1**
Average	—	**3.6**	**2.4**	—	**7.6**	**8.2**	—	**10.9**	**9.6**

(b) Runtime

Circuit	Capo			Fastplace			mPL		
		AMG-LE			AMG-LE			AMG-LE	
	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)	Baseline ($\times 10^2$ s)	1Lvl (%)	2Lvl (%)
adaptec1	30	21	17	5	−61	−65	18	32	3
adaptec2	44	34	41	13	−14	10	17	−34	−59
adaptec3	81	19	29	28	−47	−73	65	−22	16
adaptec4	—	—	—	21	−203	−195	45	−59	−52
bigblue1	54	39	42	6	−95	−74	18	−2	−36
bigblue2	92	−2	−16	23	−128	−114	49	−71	−91
bigblue3	254	11	−4	139	−29	−68	123	−90	−75
Average	—	20	18	—	−82	−83	—	−35	−42

and mPL, respectively, when using two-level AMG-LE. When compared to the results in Section 4.1, the average wire length for Capo and Fastplace degrades while mPL significantly improves. This improvement is because mPL has no option to perform just detailed placement. It performs global and detailed placement using the clustered placement as the starting point. During this process, the clusters become dispersed. Therefore, when the proposed length-driven unclustering is used as a detailed placer, the benefits of clustering and length-driven unclustering are preserved.

The runtime results show significant improvements over the results in Section 4.1. When compared to the baseline, Capo is not significantly changed while Fastplace is degraded by 19% and 66% for one- and two-level AMG-LE. The increases in runtime for Fastplace may be tolerated because of its comparatively low runtimes. However, mPL runtimes are improved significantly with more improvement seen with two-level AMG-LE, nearly cutting the average runtime by half.

In order to assess the scalability of the proposed length-driven unclustering as a detailed placer, the same experiments are performed on the ISPD05 circuits. The results are tabulated in Table 6. The results follow similar trends as those of Table 5. All placers achieve wire length improvements and significantly improved running times compared to the results of Table 4 which are obtained by using each placers' detailed placement algorithm in between clustering levels. In this case, Capo achieves an overall runtime improvement compared to the baseline with 3.6% and 2.4% average improvements in wire length when performing one and two levels of AMG-LE clustering, respectively. mPL achieves substantial improvements in wire length with modest increases in average running time. Finally, for Fastplace, significant average wire length improvements are observed for both one and two levels of AMG-LE clustering, while the runtime has degraded which is acceptable considering the relatively low runtimes of Fastplace.

In the end, the circuit designers can decide which of the placement flows presented in Sections 4.1 and 4.2 is preferred for their application. Both improve wire length but to varying degrees depending on the choice of placer. When using a placer like Capo or Fastplace, the best wire length is obtained by an increase in runtime, while mPL achieves the best results with the fast detailed placement flow. If runtime is the most urgent concern, the proposed scheme of using length-driven unclustering as a detailed placer is the best option.

Table 7: Placement wire length and runtime results comparing one-level of existing clustering algorithms and AMG-LE with and without length-driven unclustering to baseline placements without using clustering with mPL6 on the ICCAD04 benchmark suite.

(a) HPWL

Circuit					AMG-LE 1Lvl	
	Baseline ($\times 10^6$)	HEM (%)	BC (%)	AMGC (%)	Regular (%)	Fast detailed (%)
ibm01	2.4	5.9	7.1	5.7	4.6	−3.2
ibm02	5.2	−6.3	−1.6	−9.9	1.3	−3.8
ibm03	8.2	2.2	6.1	5.8	−7	5
ibm04	10.9	0.4	−1.3	−1.9	−5.7	11.3
ibm05	9.3	−5.4	−5.3	−5.4	−5.2	−10.1
ibm06	8.8	10	12.5	11.9	5.8	15.4
ibm07	12.4	1.4	−0.1	2.2	−3.1	−4.4
ibm08	21.1	−2.7	−2.9	−2.9	4.7	13
ibm09	18.9	5.1	−2	3.7	2.9	11.9
ibm10	36.3	1.9	1.7	2.1	2	3.2
ibm11	24.3	−2.6	−4.6	−1.8	−0.1	4.3
ibm12	46.1	−5	−5.9	−4.2	−1.1	6.4
ibm13	32.4	−1.2	0.3	1.7	−0.4	9.7
ibm14	82.4	6.9	7.4	7.4	6.5	30
ibm15	100.1	−7.5	−3.3	1.7	−2.9	23.3
ibm16	93.1	2.5	2.4	2.6	6.7	16.8
ibm17	114.4	5.7	5.9	4.9	7.5	13
ibm18	88.5	11.6	12	12	11	18.8
Average	—	1.2	1.5	1.9	1.5	8.9

(b) Runtime

Circuit					AMG-LE 1Lvl	
	Baseline (s)	HEM (%)	BC (%)	AMGC (%)	Regular (%)	Fast detailed (%)
ibm01	150	−20	−20	−14	−69	47
ibm02	324	−39	−41	−38	−91	27
ibm03	281	−67	−52	−85	−108	10
ibm04	344	−40	−28	−31	−87	39
ibm05	232	−45	−29	−22	−118	−2
ibm06	475	−62	−32	−49	−83	39
ibm07	488	−51	−43	−49	−100	28
ibm08	1468	−29	−44	−26	−105	62
ibm09	1134	−26	−26	−24	−136	55
ibm10	1762	−45	−49	−45	−126	8
ibm11	1501	−15	−1	−4	−84	56
ibm12	2009	−41	−57	−36	−115	29
ibm13	1456	−28	−26	−23	−110	54
ibm14	2547	−39	−33	−29	−93	27
ibm15	4208	−48	−26	−23	−101	32
ibm16	5821	−33	−35	−26	−57	50
ibm17	3025	−52	−53	−59	−131	3
ibm18	3672	−47	−49	−49	−125	2
Average	—	−40	−36	−35	−102	31

4.3. Comparison to Existing Clustering Algorithms. The proposed AMG-LE clustering algorithm with length-driven unclustering is compared to other existing clustering algorithms in this section. Each technique is evaluated in terms of after placement wire length and total runtime using the mPL6 placer [37]. The clustering algorithms compared to are heavy-edge matching (HEM) [22], best-choice (BC) [1], and the AMG clustering (AMGC) algorithm in [31]. It is worth mentioning that best-choice is the clustering algorithm used internally by Fastplace 3.0 and mPL6. Placement is first performed without any clustering to establish a baselline. Then, each of the algorithms is used to produce a clustered circuit which is placed by mPL. The clustered placement is then unclustered and detailed placement is performed by mPL or by using length-driven unclustering as a detailed placer. Results of the experiment are given in Table 7.

The percentage improvement for AMG-LE in a regular clustered placement flow, as described in Section 4.1, is given under the subheading Regular. The column labeled Fast Detailed refers to the results using only length-driven unclustering as a detailed placer, described in Section 4.2. In terms of wire length, the regular AMG-LE scheme is competitive with the existing algorithms performing better in some circuits and worse in others. On the other hand, the runtime for regular AMG-LE degrades more than the other clustering techniques. This is because the runtime includes the time to perform pre-placement length estimation and the algorithm is implemented in the MATLAB environment Therefore, the runtime can be way more competitive if the procedure is implemented in a more efficient way. However, the fast detailed variant of AMG-LE is a clear winner in terms of average wire length and runtime improvement.

5. Conclusions

This paper presents a new clustering technique by utilizing length estimates in an algebraic multigrid-based clustering technique. In addition, a physical unclustering technique is proposed with the benefits of reducing wire length and preserving legality. Because of its legality-preserving nature, the technique eliminates the need to use detailed placement between levels in the framework. These techniques are shown to be effective in reducing wire length and can also be used to benefit runtime.

Acknowledgments

This research is supported by the Natural Sciences and Engineering Research Council of Canada and Alberta Innovates Technology Futures. This research has been enabled by the use of computing resources provided by WestGrid and Compute/Calcul Canada.

References

[1] C. Alpert, A. Kahng, G. J. Nam, S. Reda, and P. Villarrubia, "A semi-persistent clustering technique for vlsi circuit placement," in *International Symposium on Physical Design (ISPD '05)*, pp. 200–207, April 2005.

[2] J. Li, L. Behjat, and A. Kennings, "Net cluster: a net-reduction-based clustering preprocessing algorithm for partitioning and placement," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 669–679, 2007.

[3] S. Bodapati and F. N. Najm, "Prelayout estimation of individual wire lengths," *IEEE Transactions on VLSI Systems*, vol. 9, no. 6, pp. 943–958, 2001.

[4] A. Farshidi, L. Behjat, L. Rakai, and B. Fathi, "A pre-placement individual net length estimation model and an application for modern circuits," *Integration*, vol. 44, no. 2, pp. 111–122, 2011.

[5] B. Fathi, L. Behjat, and L. M. Rakai, "A pre-placement net length estimation technique for mixed-size circuits," in *ACM/IEEE Workshop on System Level Interconnect Prediction (SLIP '09)*, pp. 45–52, July 2009.

[6] A. B. Kahng and S. Reda, "Intrinsic shortest path length: a new, accurate a priori wirelength estimator," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 173–180, November 2005.

[7] W. E. Donath, "Placement and average interconnection lengths of computer logic," *IEEE Trans Circuits Syst*, vol. 26, no. 4, pp. 272–277, 1979.

[8] W. E. Donath, "Wire length distribution for placements of computer logic," *Ibm Journal of Research and Development*, vol. 25, no. 2-3, pp. 152–155, 1981.

[9] M. Feuer, "Connectivity of random logic," *IEEE Transactions on Computers*, vol. C-31, no. 1, pp. 29–33, 1982.

[10] T. Hamada, C. K. Cheng, and P. M. Chau, "A wire length estimation technique utilizing neighborhood density equations," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 912–922, 1996.

[11] H. T. Heineken and W. Maly, "Standard cell interconnect length prediction from structural circuit attributes," in *IEEE Custom Integrated Circuits Conference*, pp. 167–170, May 1996.

[12] M. Pedram and B. Preas, "Interconnection length estimation for optimized standard cell layouts," in *IEEE International Conference on Computer-Aided Design (ICCAD '89)*, pp. 390–393, November 1989.

[13] B. Hu and M. Marek-Sadowska, "Wire length prediction based clustering and its application in placement," in *40th Design Automation Conference*, pp. 800–805, June 2003.

[14] Q. Liu, B. Hu, and M. Marek-Sadowska, "Wire length prediction in constraint driven placement," in *International Workshop on System Level Interconnect Prediction*, pp. 99–105, April 2003.

[15] A. Brandt, "Algebraic multigrid theory: the symmetric case," *Applied Mathematics and Computation*, vol. 19, no. 1–4, pp. 23–56, 1986.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, Pa, USA, 2nd edition, 2000.

[17] S. McCormick, Ed., *Multigrid Methods*, vol. 5 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, Pa, USA, 1987.

[18] J. Ruge and K. Stüben, *Multigrid Methods*, chapter 4, SIAM, Philadelphia, Pa, USA, 1987.

[19] K. Stüben, "Algebraic multigrid (AMG): experiences and comparisons," *Applied Mathematics and Computation*, vol. 13, no. 3-4, pp. 419–451, 1983.

[20] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.

[21] P. Wesseling, *An Introduction to Multigrid Methods*, Pure and Applied Mathematics Series, John Wiley and Sons, New York, NY, USA, 1992.

[22] C. J. Alpert, J. H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *IEEE Transactions on Computer-aided Design of*

Integrated Circuits and Systems, vol. 17, no. 8, pp. 655–667, 1998.

[23] A. Caldwell, A. Kahng, and I. Markov, "Improved algorithms for hypergraph bi-partitioning," in *Asia and South Pacific Design Automation Conference (ASP-DAC '00)*, pp. 661–666, 2000.

[24] J. Cong and S. K. Lim, "Edge separability-based circuit clustering with application to multilevel circuit partitioning," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 346–357, 2004.

[25] B. Hu and M. Marek-Sadowska, "Fine granularity clustering for large scale placement problems," in *International Symposium on Physical Design*, pp. 67–74, April 2003.

[26] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in vlsi domain," *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[27] G. J. Nam, "ISPD 2006 placement contest: benchmark suite and results," in *International Symposium on Physical Design (ISPD '06)*, p. 167, April 2006.

[28] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: application in vlsi domain," in *34th Design Automation Conference*, pp. 526–529, June 1997.

[29] J. Z. Yan, C. Chu, and W. K. Mak, "Safechoice: a novel clustering algorithm for wirelength-driven placement," in *ACM International Symposium on Physical Design (ISPD '10)*, pp. 185–192, March 2010.

[30] J. Z. Yan, C. Chu, and W. K. Mak, "Safechoice: a novel approach to hypergraph clustering for wirelength-driven placement," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 30, no. 7, pp. 1020–1033, 2011.

[31] L. Rakai, L. Behjat, S. Martin, and J. Aguado, "An algebraic multigrid-based algorithm for circuit clustering," *Applied Mathematics and Computation*, vol. 218, no. 9, pp. 5202–5216, 2012.

[32] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 24, no. 5, pp. 734–747, 2005.

[33] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of partitioning, placement and floorplanning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '04)*, pp. 550–557, November 2004.

[34] G. J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ispd2005 placement contest and benchmark suite," in *International Symposium on Physical Design (ISPD '05)*, pp. 216–220, April 2005.

[35] J. Roy and I. Markov, *Partitioning-driven Techniques for VLSI Placement. Handbook of Algorithms for VLSI Physical Design Automation*, CRC Press, 2008.

[36] N. Viswanathan, M. Pan, and C. Chu, "Fastplace 3.0: a fast multilevel quadratic placement algorithm with placement congestion control," in *Asia and South Pacific Design Automation Conference (ASP-DAC '07)*, pp. 135–140, January 2007.

[37] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "Mpl6: enhanced multilevel mixed-size placement," in *International Symposium on Physical Design (ISPD '06)*, pp. 212–214, April 2006.