Circuits and Systems, 2017, 8, 191-201
http://www.scirp.org/journal/cs
ISSN Online: 2153-1293
ISSN Print: 2153-1285

A Connectivity-Based Legalization Scheme for Standard Cell Placement

Antonios N. Dadaliaris[1], Panagiotis Oikonomou[2], Maria G. Koziri[1], Evangelia Nerantzaki[1], Thanasis Loukopoulos[3], Georgios I. Stamoulis[2]

[1]Department of Computer Science, University of Thessaly, Lamia, Greece
[2]Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece
[3]Department of Computer Science and Biomedical Informatics, University of Thessaly, Lamia, Greece
Email: dadaliaris@cs.uth.gr, paikonom@uth.gr, mkoziri@uth.gr, enerantzaki@uth.gr, luke@dib.uth.gr, georges@uth.gr

How to cite this paper: Dadaliaris, A.N., Oikonomou, P., Koziri, M.G. Nerantzaki, E., Loukopoulos, T. and Stamoulis, G.I. (2017) A Connectivity-Based Legalization Scheme for Standard Cell Placement. *Circuits and Systems*, **8**, 191-201.
https://doi.org/10.4236/cs.2017.88013

Received: July 22, 2017
Accepted: August 25, 2017
Published: August 28, 2017

Abstract

Standard cell placement algorithms have been at the forefront of academic research concerning the physical design stages of VLSI design flows. The penultimate step of a standard cell placement procedure is legalization. In this step the manufacturability of the design is directly settled, and the quality of the solution, in terms of wirelength, congestion, timing and power consumption is indirectly defined. Since the heavy lifting regarding processing is performed by global placers, fast legalization solutions are protruded in state-of-the-art design flows. In this paper we propose and evaluate a legalization scheme that surpasses in execution speed two of the most widely used legalizers, without not only corrupting the quality of the final solution in terms of interconnection wirelength but improving it in the process.

Keywords

Legalization, Standard Cell Placement, Heuristics, Interconnect Wirelength

1. Introduction

Standard cell placement can be described as the problem of defining the coordinates of each movable cell of a design, within a predefined chip area encompassing rows of specific height equal to that of the aforementioned cells, in a manner that eliminates the possibility of cell overlaps or boundary overflows. A typical placement procedure consists of three distinct steps: global placement, legalization and detailed placement.

During the global placement step, the cells are spread across the chip area so that one or more target functions are optimized, such as total interconnection wirelength, critical path length and timing. The output, in most cases, is a design where cells are placed at their optimal positions, according to the target functions, but overlaps and/or overflows occur, thus rendering the design unroutable and necessitating an additional legalization step. Figure 1 depicts the *ibm*01 benchmark circuit after its global placement with mPL6 [1]. During legalization cells are aligned in rows and overlaps are eliminated. Figure 2 illustrates a legalized version of the *ibm*01 benchmark circuit where every cell is aligned to a specific row and overlaps have been dealt with. The main performance metrics for legalization algorithms are total wirelength, overall cell displacement and runtime. The interconnection wirelength of each net is modeled as the half perimeter wirelength (HPWL) of the bounding box of its cells, while the cell displacement is the sum of the cells' displacement in each axis. Both metrics should be kept at a minimum, since wirelength directly affects the routability of the design and displacement can be interpreted as how far away we have drifted from the optimal (but unrealizable) solution. Finally, in detailed placement, heuristics are applied that lead to minor design alterations and produce better results concerning a specific metric, usually at the expense of another.

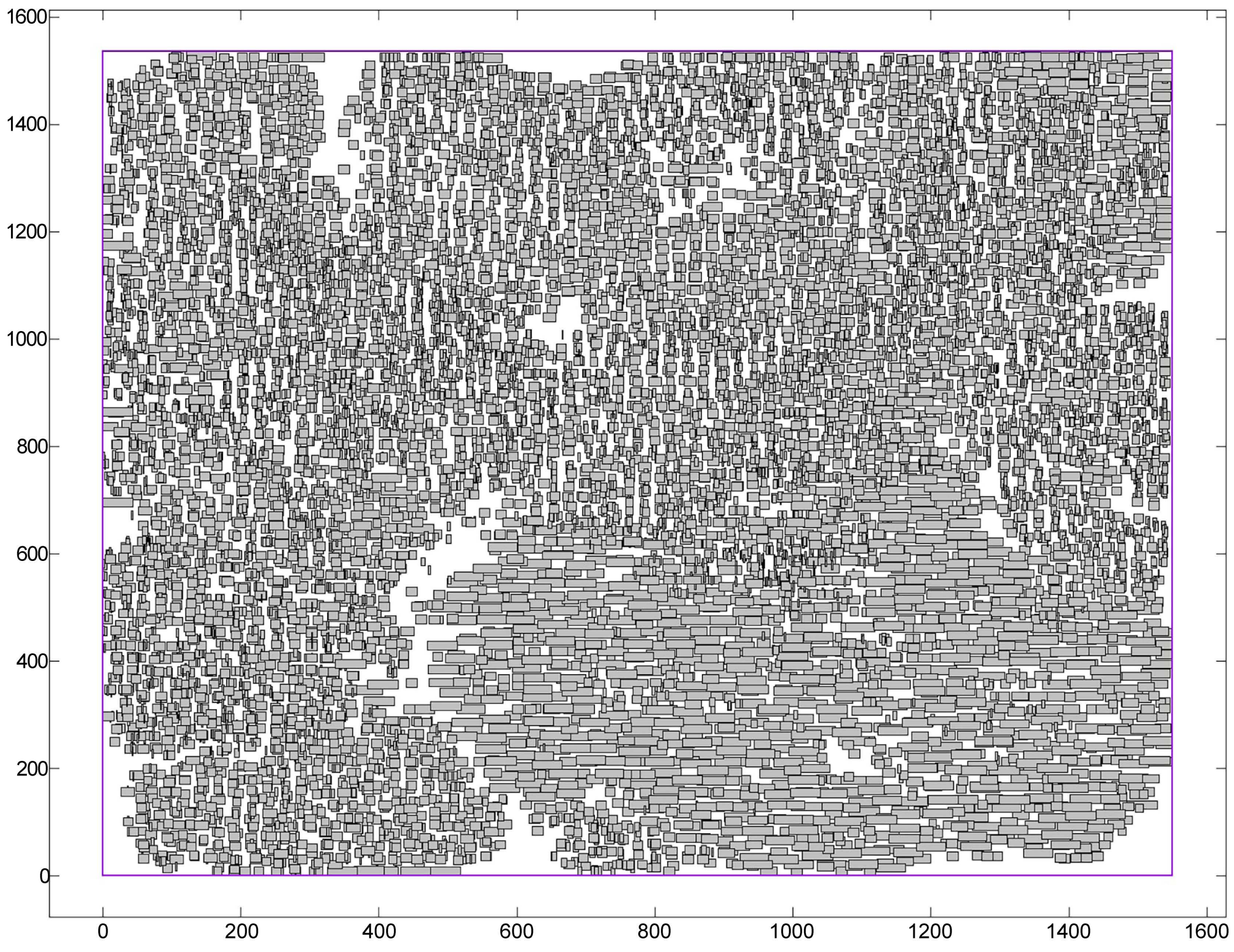

Figure 1. Cell placement by mPL6 (ibm01 benchmark circuit).

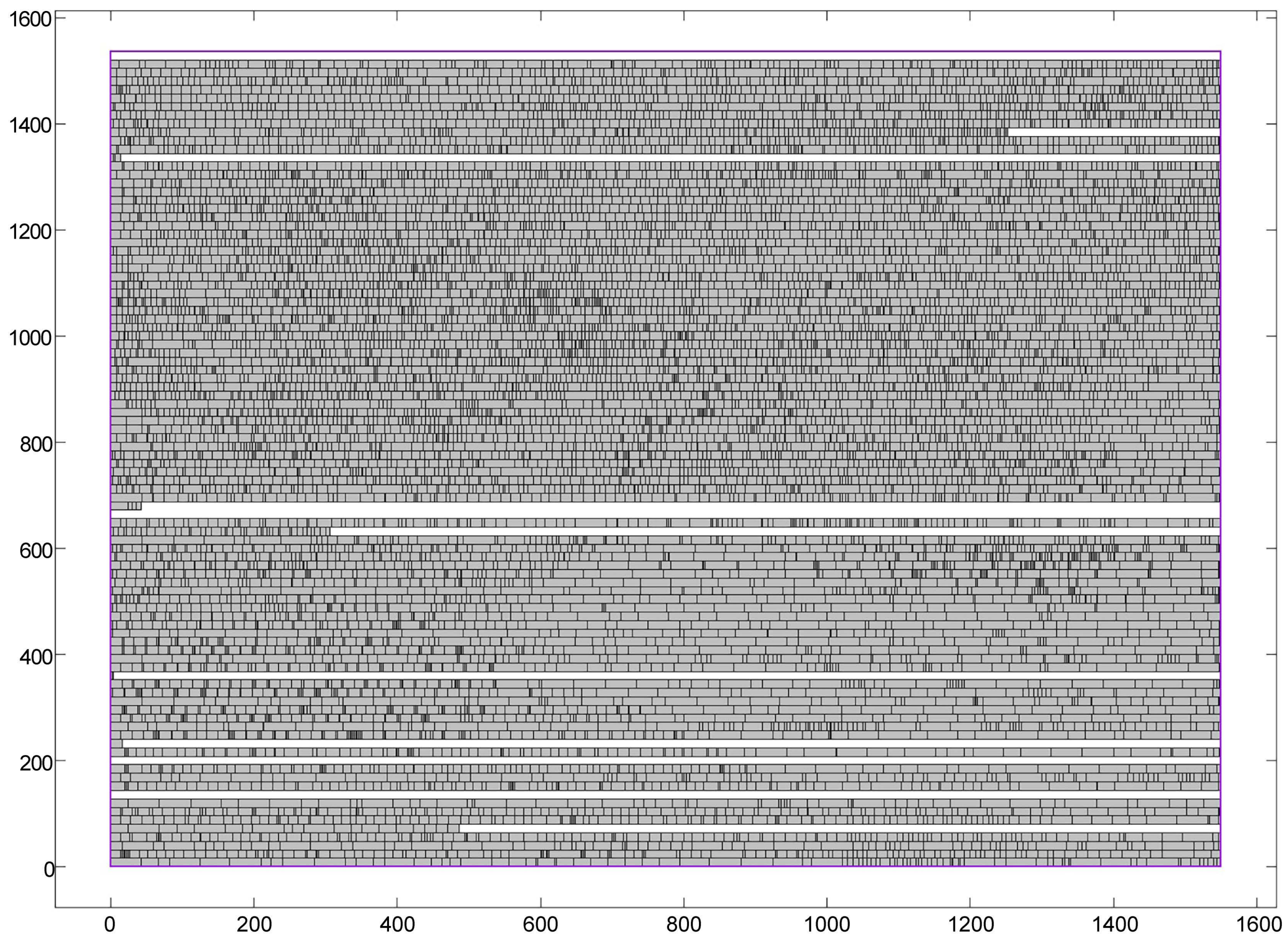

Figure 2. Cell placement by running Tetris over mPL6 (ibm01 benchmark circuit).

In this paper we introduce a scheme that improves upon the solution quality of two of the fastest existing legalization algorithms (Tetris [2] and Abacus [3]), while retaining and enhancing their advantage concerning running time against the more complex methods. The rest of the paper is organized as follows. Section 2 presents the related work on global placement and legalization. Section 3 illustrates the algorithm, which is evaluated on Section 4. Finally, Section 5 summarizes the results and provides our concluding remarks.

2. Related Work

Standard cell placement has been at the forefront of academic research throughout the last decades. Initially in the context of global placement various approaches were proposed based in simulated annealing with the most prominent being Timberwolf [4]. Timberwolf's global placement procedure is divided into two separate stages, due to the fact that it incorporates a global routing routine. In the first stage, simulated annealing is applied in order to place the cells in a way that minimizes the total wirelength, while in the second stage feed through cells are inserted (in pursuance of completing the global routing) and simulated annealing is re-applied to re-place the cells and minimize total wirelength.

Min-cut partitioning was also used to tackle the global placement problem. FengShui [5] performs min-cut multi-way partitioning (using hMetis) to spread the cells throughout the available region. Capo [6], unlike FengShui, does not explicitly use multi-way partitioning, instead the partitioning direction is dictated by placement feedback and additional cutline shifting functionality is incorporated.

A significant amount of academic global placers formulate quadratic optimization problems. Gordian [7] utilizes the connectivity between cells in the direction of formulating a series of quadratic programming problems, by iteratively partitioning the available area and adding new constraints that restrict cell movement, whilst recalibrating each cell's position to obtain the minimum wirelength. FastPlace3 [8] focuses on improving the overall runtime in the expense of the final solution quality, and POLAR [9] achieves speedup by applying parallelization techniques throughout the execution of the problem's formulation and solution procedures.

Furthermore, force-directed methods were applied in order to acquire better global placement results. Kraftwerk2 [10] and ePlace [11] suggest the creation of a set of forces that are applied to each movable node of the design. The goal is to spread the nodes at such extend that equilibrium is reached and, thus, the optimization of the metric at hand is achieved.

Significant work concerning legalization has also been published, the most important step during the placement procedure, due to its direct connection to the manufacturability of the design. In [2] Tetris, one of the most a classic approaches to the legalization problem is presented. Tetris is a simple, yet elegant, solution that is used in a plethora of placers as an add-on, given the fact that it can generate a legal solution in reasonable runtime. Its simple nature has led to the implementation of various heuristics [12] that improve significantly all legalization related metrics (total half perimeter wirelength, displacement, runtime). In [13] a two-phase overlap removal procedure is discussed. Initially, cells are moved vertically till the row capacity constraints are met, and subsequently overlaps within each row are removed through a topological shortest path calculation that achieves minimum perturbation and minimum half perimeter wirelength.

Abacus [3] shares some common characteristics with Tetris. Cells are placed in order of their x-coordinate, but in contrast to the aforementioned legalizer, in case of overlaps within a row, a cluster of the cells involved is created and its ideal position is calculated through the formulation of a quadratic problem, thus allowing the re-placement of previously placed cells. OAL [14] further extends Abacus is using a linear wirelength model instead of a quadratic for measuring cell displacement and a differentiated cell insertion policy which places cells based on their width. Domocus [15] introduces a fast and efficient legalization scheme, by exploiting Abacus. Using coarse grain parallelism, chip area is divided into equally sized zones which can be processed independently occurring minimum synchronization overhead. This parallelization scheme is also applicable in any full-fledged legalizer of this particular category (e.g. Tetris). Additionally, in [16] a fast approach to legalization is presented, that differentiates from previous algorithms by applying a fast row selection technique that is based in the k-medoid clustering approach.

BONNPLACE [17] partitions the placement area into bins and assigns cells to them. The bin assignment is balanced by an algorithm that realizes the flow between bins, therefore eliminating overflowing bins. The aforementioned algorithm is a slight modification of the successive shortest path algorithm, which ensures that only flow augmentations that can be realized are chosen and subsequently realized before the next augmentation.

Finally, in [18] a history-based legalization scheme is proposed. Legal placement solutions are generated through a min-cost problem formulation. Viable solutions are automatically translated to actual cell movements, but non-viable solutions are recorded in each iteration in a history archive, as to drop future similar flow realization attempts.

3. Our Algorithm

In this section an in-depth description of our algorithm is presented. This algorithm is designed as a legalization add-on for global placers but can additionally be applied as a detailed placement heuristic. A typical standard cell placement framework is considered, meaning that all cells and rows are of equal height and the solution consists of pinpointing the cells' optimal locations while satisfying legality constraints (elimination of overlaps and overflows).

Initially every movable node is ordered according to the x-coordinate of its lower left corner, in accordance with most academic legalizers. Subsequently, the algorithm searches for the most cost-effective row in which each cell should be placed. The insertion cost of a cell in a specific row is determined as the overall displacement of the cell.

After the cell insertion, any movable object belonging to the same nets as the aforementioned cell is moved by the same amount to the cell's direction. This policy affects only objects that are not placed out of bounds by the displacement and have not already been moved in a previous iteration of the algorithm. The process comes to a halt when all cells have been checked. **Figure 3** shows the pseudocode of the algorithm.

The main advantage of the proposed scheme is that it brings the connectivity between nodes into the decision process. **Figure 3** illustrates the placement of 6 cells in a 6 row circuit. Assume that cells 0, 1, 2, 3, 4 form net N_1 while cells 0, 4, 5, 6 form net N_2. The optimization goal of the algorithm is to reduce the total wire length which is measured for each net as follows: the minimum bounding rectangle that encloses all its cells is calculated and its half perimeter is accumulated. Red and blue box depict the HPWL for N_1 and N_2 respectively. It is straightforward to see that our methodology (**Figure 4(d)**) accounts in a minimal HPWL compared to Abacus and Tetris (**Figure 4(b)**, **Figure 4(c)**).

```
Sort cells based on their x-coordinate
foreach cell, $c_i$
    $disp_{min}$ := ∞
    foreach row, $r_j$
        Insert $c_i$ into $r_j$
        Calculate displacement cost, $disp$
        if $disp < disp_{min}$
            $r_{best} := r_j$
            $disp_{min} := disp$
        Remove $c_i$ from $r_j$
    end
    Insert $c_i$ into $r_{best}$
    foreach movable object which connects to $c_i$, $mc_i$
        Move $mc_i$ by $disp_{min}$ toward $c_i$'s new position
end
```

Figure 3. Pseudocode for the connectivity-based legalization scheme.

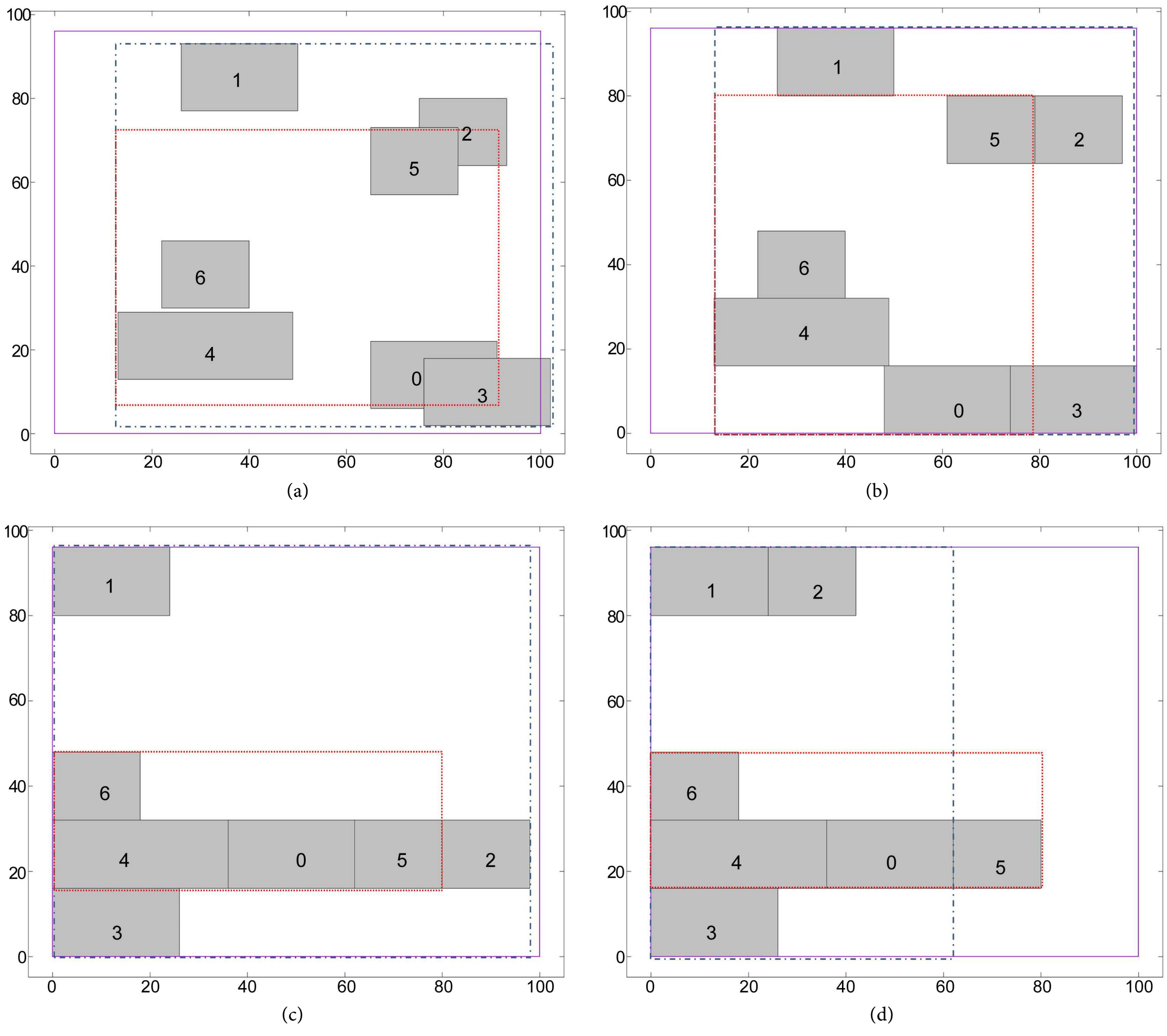

Figure 4. A legalization example for three different legalizers. (a) Initial placement; (b) Abacus legalizer; (c) Tetris legalizer; (d) Proposed legalizer.

4. Experimental Results

The ISPD05 [19] benchmark circuits were used to evaluate the performance of our approach. Global placements were obtained by four commonly used algorithms, namely, Gordian [7], NTUPlace [20], mPL6 [1] and FengShui [5]. Thereupon, the designs were legalized following our scheme and the results were compared against a classic legalization algorithm (Tetris) and a state of the art legalization algorithm (Abacus). For each circuit we recorded the percentage of performance improvement our algorithm (CB) over Tetris (T) and Abacus (A) as follows:

improvement (T) = (performance (T) − performance (CB))/performance (T)

improvement (A) = (performance (A) − performance (CB))/performance (A)

In order to characterize the total performance over the entire dataset we used the average improvement experienced over all circuits. Experiments were executed in a Linux server with two 6-core Intel Xeon E5-2630 CPUs running at 2.3 GHz.

The performance achieved by our connectivity-based approach was significant. As observed in **Tables 1-4**, applying this connectivity-based legalization scheme over the four aforementioned global placers gave results that were

Table 1. Performance improvement over tetris and abacus (Gordian global placer).

Benchmarks	Tetris			Abacus		
	HPWL	Displace	Time	HWPL	Displace	Time
ibm01	63.61%	66.64%	−50.00%	10.83%	−41.28%	2026.67%
ibm02	59.79%	78.83%	−66.67%	4.81%	−72.05%	1803.33%
ibm03	61.45%	73.95%	−76.19%	2.05%	−56.36%	1805.41%
ibm04	64.42%	85.71%	−48.39%	4.78%	−70.16%	1597.83%
ibm05	49.42%	79.16%	−35.14%	3.74%	−71.62%	1806.00%
ibm06	69.51%	74.98%	−45.45%	5.99%	−62.57%	3410.42%
ibm07	69.99%	82.92%	−15.38%	1.18%	−76.30%	3140.00%
ibm08	71.22%	82.82%	−25.71%	2.65%	−77.56%	3098.86%
ibm09	73.61%	80.46%	−12.00%	2.13%	−74.15%	3454.76%
ibm10	69.40%	80.61%	−35.58%	3.30%	−72.56%	2760.99%
ibm11	72.41%	81.20%	−33.33%	5.20%	−77.41%	4205.83%
ibm12	66.09%	80.03%	−14.29%	2.43%	−75.58%	2773.53%
ibm13	74.23%	80.82%	−36.59%	1.98%	−79.81%	4037.50%
ibm14	69.28%	79.08%	−33.20%	6.29%	−78.86%	10194.43%
ibm15	72.52%	80.20%	−26.80%	6.27%	−80.64%	10368.30%
ibm16	80.40%	82.44%	−64.78%	4.36%	−86.97%	8433.51%
ibm17	69.83%	83.77%	−49.48%	3.89%	−83.37%	3550.00%
ibm18	78.35%	82.31%	−33.17%	2.69%	−87.34%	5093.73%

Table 2. Performance improvement over tetris and abacus (NTUplace3 global placer).

Benchmarks	Tetris			Abacus		
	HPWL	Displace	Time	HWPL	Displace	Time
ibm01	69.30%	80.48%	0.00%	1.90%	−74.71%	1745.45%
ibm02	70.47%	78.77%	−33.33%	1.31%	−80.41%	1732.14%
ibm03	65.71%	80.72%	−66.67%	1.63%	−81.07%	2146.67%
ibm04	67.95%	84.96%	−29.03%	−0.64%	−73.07%	1620.00%
ibm05	59.70%	78.30%	−32.35%	3.68%	−87.20%	1982.22%
ibm06	76.11%	81.49%	−59.38%	1.13%	−85.34%	2613.73%
ibm07	78.98%	80.25%	−6.35%	2.57%	−81.96%	3338.81%
ibm08	73.40%	82.42%	−14.71%	3.08%	−90.08%	3142.31%
ibm09	75.27%	84.09%	−20.27%	4.92%	−84.97%	2676.40%
ibm10	79.56%	81.94%	−42.27%	−0.69%	−89.34%	2592.03%
ibm11	80.31%	76.18%	−33.33%	7.51%	−90.26%	4290.83%
ibm12	80.63%	82.09%	−20.72%	−1.70%	−82.28%	2766.42%
ibm13	74.38%	84.36%	−8.26%	4.67%	−87.56%	4054.96%
ibm14	85.57%	81.22%	−25.29%	−3.27%	−85.36%	5369.11%
ibm15	86.96%	77.37%	−47.65%	7.00%	−92.80%	6279.22%
ibm16	85.30%	81.57%	−14.97%	1.89%	−94.85%	5407.73%
ibm17	85.80%	80.95%	−33.85%	2.60%	−80.28%	9516.99%
ibm18	86.04%	85.52%	−24.68%	4.78%	−92.68%	4485.94%

Table 3. Performance improvement over tetris and abacus (mPL6 global placer).

Benchmarks	Tetris			Abacus		
	HPWL	Displace	Time	HWPL	Displace	Time
ibm01	71.04%	78.97%	−30.77%	3.64%	−80.80%	1717.65%
ibm02	70.40%	80.35%	−57.89%	1.66%	−84.12%	2033.33%
ibm03	73.56%	80.62%	−66.67%	1.69%	−82.36%	1977.14%
ibm04	64.63%	75.94%	−37.50%	−2.50%	−87.19%	1825.00%
ibm05	62.08%	83.56%	5.88%	−3.09%	−89.32%	2756.25%
ibm06	77.72%	82.73%	−25.71%	0.09%	−87.16%	2790.91%
ibm07	77.01%	79.05%	1.54%	0.06%	−89.75%	3521.88%
ibm08	78.45%	81.31%	−31.34%	−1.62%	−91.07%	3190.91%
ibm09	82.35%	82.21%	6.41%	3.46%	−88.76%	3887.67%
ibm10	77.81%	82.16%	−20.37%	0.40%	−88.64%	2753.08%
ibm11	81.25%	79.95%	−7.29%	3.77%	−88.82%	3634.95%
ibm12	80.53%	82.47%	−27.43%	2.33%	−89.78%	2268.75%
ibm13	79.72%	76.90%	−33.90%	4.44%	−88.91%	3328.48%
ibm14	84.15%	81.74%	−34.35%	−3.26%	−94.11%	4090.63%
ibm15	83.95%	81.77%	−38.41%	−0.72%	−94.45%	4943.19%
ibm16	85.42%	82.61%	−35.34%	−1.62%	−94.62%	4471.18%
ibm17	85.27%	83.04%	−42.01%	−1.55%	−94.41%	4127.29%
ibm18	87.63%	82.69%	−21.35%	−2.91%	−94.99%	4546.32%

constantly dominating the ones produced when applying Tetris as far as the HPWL metric is concerned. Similar observations hold for the execution time since the proposed methodology drastically cuts down time cost. Specifically, in the case of Gordian global placement, our approach achieved 68.84% in HPWL, 79.77% in displacement over Tetris and 4.14% reduction in HPWL over Abacus. It is worth mentioning that our approach dominates Abacus concerning runtime in every experiment performed.

Finally, Table 5 illustrates the average performance improvement of our approach over Tetris and Abacus.

Table 4. Performance improvement over tetris and abacus (fengShui global placer).

Benchmarks	Tetris			Abacus		
	HPWL	Displace	Time	HWPL	Displace	Time
ibm01	49.06%	95.18%	−13.33%	0.57%	9.27%	817.65%
ibm02	41.07%	95.32%	−55.56%	0.37%	10.70%	1075.00%
ibm03	36.43%	94.48%	−73.68%	0.48%	15.82%	1260.61%
ibm04	26.38%	92.67%	−26.47%	0.53%	15.19%	1188.37%
ibm05	7.90%	90.03%	−32.35%	0.58%	23.07%	1151.11%
ibm06	38.43%	93.93%	−27.03%	0.03%	−5.00%	1893.62%
ibm07	54.28%	97.27%	−6.06%	0.36%	10.78%	1735.71%
ibm08	49.31%	97.08%	−2.82%	0.13%	6.53%	2061.64%
ibm09	59.15%	97.38%	3.70%	0.38%	9.07%	2015.38%
ibm10	42.31%	96.80%	−36.08%	0.42%	19.78%	1836.36%
ibm11	45.99%	96.83%	8.77%	0.28%	9.51%	2373.08%
ibm12	39.74%	96.98%	−28.70%	0.26%	15.16%	1808.63%
ibm13	49.31%	97.05%	−30.58%	0.37%	13.38%	2201.90%
ibm14	57.25%	98.23%	−34.80%	0.44%	16.76%	2386.96%
ibm15	58.63%	98.30%	−31.02%	0.48%	17.77%	2720.65%
ibm16	61.78%	98.60%	−25.33%	0.26%	9.16%	2578.53%
ibm17	53.71%	98.55%	−29.73%	0.18%	7.55%	1785.90%
ibm18	49.06%	95.18%	−13.33%	0.57%	9.27%	817.65%

Table 5. Average performance improvement over tetris and abacus.

Global placers	Tetris			Abacus		
	HPWL	Displace	Time	HWPL	Displace	Time
Gordian	68.64%	79.77%	−39.01%	4.14%	−73.59%	4086.73%
NTUplace3	76.75%	81.26%	−28.51%	2.35%	−85.23%	3653.39%
mPL6	77.94%	81.00%	−27.58%	0.24%	−89.40%	3214.70%
fengShui	46.50%	96.30%	−25.47%	0.36%	11.85%	1885.88%

5. Conclusion

In this paper we have proposed a connectivity-based legalization scheme that produces high quality results as compared to both Tetris and Abacus in competitive execution time. More specifically, the proposed method produces Abacus-level results in Tetris-level execution time, thus providing an improvement over both legalization approaches. More specifically our approach is 67.46% better on average in HPWL from Tetris and 1.77% from Abacus while it is at least two orders of magnitude better on execution time. This is due to the fact that we have taken into account cell connectivity while others emphasize on minimizing displacement via dynamic programming.

References

[1] Chan, T.F., Sze, K., Shinnerl, J.R. and Xie, M. (2007) MPL6: Enhanced Multilevel Mixed-Size Placement with Congestion Control. In Modern Circuit Placement, Springer US, New York, 247-288. https://doi.org/10.1007/978-0-387-68739-1_10

[2] Hill, D. (2002) US Patent and Trademark Office. US Patent No. 6,370,673, Washington DC.

[3] Spindler, P., Schlichtmann, U. and Johannes, F.M. (2008) Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement. In Proceedings of the 2008 International Symposium on Physical Design, Portland, 13-16 April 2008, 47-53. https://doi.org/10.1145/1353629.1353640

[4] Sechen, C. and Sangiovanni-Vincentelli, A. (1985) The Timber Wolf Placement and Routing Package. *IEEE Journal of Solid-State Circuits*, **20**, 510-522. https://doi.org/10.1109/JSSC.1985.1052337

[5] Agnihotri, A.R., Ono, S. and Madden, P.H. (2005) Recursive Bisection Placement: Feng Shui 5.0 Implementation Details. *In Proceedings of the* 2005 *International Symposium on Physical Design*, San Francisco, 3-6 April 2005, 230-232. https://doi.org/10.1145/1055137.1055186

[6] Roy, J.A., Papa, D.A., Adya, S.N., Chan, H.H., Ng, A.N., Lu, J.F. and Markov, I.L. (2005) Capo: Robust and Scalable Open-Source Min-Cut Floorplacer. *In Proceedings of the* 2005 *International Symposium on Physical Design*, San Francisco, 3-6 April 2005, 224-226. https://doi.org/10.1145/1055137.1055184

[7] Kleinhans, J.M., Sigl, G., Johannes, F.M. and Antreich, K.J. (1991) GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **10**, 356-365. https://doi.org/10.1109/43.67789

[8] Viswanathan, N., Pan, M. and Chu, C. (2007) FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control. *In Proceedings of the* 2007 *Asia and South Pacific Design Automation Conference*, Yokohama, 23-26 January 2007, 135-140. https://doi.org/10.1109/ASPDAC.2007.357975

[9] Lin, T., Chu, C., Shinnerl, J.R., Bustany, I. and Nedelchev, I. (2013) POLAR: Placement Based on Novel Rough Legalization and Refinement. In 2013 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD*), San Jose, 18-21 November 2013, 357-362. https://doi.org/10.1109/ICCAD.2013.6691143

[10] Spindler, P., Schlichtmann, U. and Johannes, F.M. (2008) Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **27**,

1398-1411. https://doi.org/10.1109/TCAD.2008.925783

[11] Lu, J., Chen, P., Chang, C.C., Sha, L., Huang, D.J.H., Teng, C.C. and Cheng, C.K. (2014) ePlace: Electrostatics Based Placement Using Nesterov's Method. 2014 51*st ACM/EDAC/IEEE Design Automation Conference* (*DAC*), San Francisco, 1-5 June 2014, 1-6. https://doi.org/10.1145/2593069.2593133

[12] Dadaliaris, A.N., Oikonomou, P., Koziri, M.G., Nerantzaki, E., Hatzaras, Y., Garyfallou, D. and Stamoulis, G.I. (2017) Heuristics to Augment the Performance of Tetris Legalization: Making a Fast But Inferior Method Competitive. *Journal of Low Power Electronics*, **13**, 220-230. https://doi.org/10.1166/jolpe.2017.1483

[13] Kahng, A.B., Markov, I.L. and Reda, S. (2004) On Legalization of Row-Based Placements. *Proceedings of the* 14*th ACM Great Lakes Symposium on VLSI*, Boston, 26-28 April 2004, 214-219. https://doi.org/10.1145/988952.989004

[14] Chou, S. and Ho, T.Y. (2009) OAL: An Obstacle-Aware Legalization in Standard Cell Placement with Displacement Minimization. *IEEE International SOC Conference*, Belfast, 9-11 September 2009, 329-332.
https://doi.org/10.1109/SOCCON.2009.5398030

[15] Oikonomou, P., Koziri, M.G., Dadaliaris, A.N., Loukopoulos, T. and Stamoulis, G.I. (2017) Domocus: Lock Free Parallel Legalization in Standard Cell Placement. 2017 6*th International Conference on Modern Circuits and Systems Technologies* (*MOCAST*), Thessaloniki, 4-6 May 2017, 1-4.
ttps://doi.org/10.1109/MOCAST.2017.7937644

[16] Ho, T.Y. and Liu, S.H. (2010) Fast Legalization for Standard Cell Placement with Simultaneous Wirelength and Displacement Minimization. 2010 18*th IEEE/IFIP VLSI System on Chip Conference* (*VLSI-SoC*), Madrid, 27-29 September 2010, 369-374.

[17] Brenner, U. (2013) Bonnplace Legalization: Minimizing Movement by Iterative Augmentation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **32**, 1215-1227. https://doi.org/10.1109/TCAD.2013.2253834

[18] Cho, M., Ren, H., Xiang, H. and Puri, R. (2010) History-Based VLSI Legalization Using Network Flow. *Proceedings of the* 47*th Design Automation Conference*, Anaheim, 13-18 June 2010, 286-291. https://doi.org/10.1145/1837274.1837347

[19] ISPD98 Benchmark Circuits.
http://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html

[20] Chen, T.C., Hsu, T.C., Jiang, Z.W. and Chang, Y.W. (2005) NTUplace: A Ratio Partitioning Based Placement Algorithm for Large-Scale Mixed-Size Designs. *Proceedings of the* 2005 *International Symposium on Physical Design*, San Francisco, 3-6 April 2005, 236-238. https://doi.org/10.1145/1055137.1055188