DRAGON2000: STANDARD-CELL PLACEMENT TOOL FOR LARGE INDUSTRY CIRCUITS

Maogang Wang *Xiaojian Yang* *Majid Sarrafzadeh*
Department of Electrical and Computer Engineering, Northwestern University, Evan ston, IL 60208
mgwang,xjyang,majid@ece.nwu.edu

ABSTRACT

In this paper, we develop a new standard cell placement tool, *Dragon2000*, to solve large scale placement problem effectively. A top-down hierarchical approach is used in Dragon2000. State-of-the-art partitioning tools are tightly integrated with wirelength minimization techniques to achieve superior performance. We argue that net-cut minimization is a good and important shortcut to solve the large scale placement problem. Experimental results show that minimizing net-cut is more important than greedily obtain a wirelength optimal placement at intermediate hierarchical levels. We run Dragon2000 on recently released large benchmark suite ISPD98 as well as MCNC circuits. For circuits which have more than 100k cells, comparing to iTools1.4.0, Dragon2000 can produce slightly better placement results (1.4%) while spending much less amount of time (2× speedup). This is also the first published placement result on the publicly available large industrial circuits.

1. INTRODUCTION

Placement is a classical problem in VLSI physical design. A lot of effective placement tools have been proposed in the last twenty years [6, 8, 7]. Although they were quite successful at their release time, as the VLSI circuit size gets larger and the deep sub-micron (DSM) technology becomes dominant, these tools are obsolete or not effective anymore. It is time to re-think and develop a new placement tool which can handle the large industrial circuits.

Previous work on the placement problem falls into two classes: constructive and iterative. It is generally believed that placement iterative approaches can produce better results than constructive approaches but are slower. When the circuit size gets larger and larger, quality degradation is expected for the flat iterative approaches. The multi-level hierarchical technique is regarded indispensable for solving today's complex VLSI placement problem without sacrificing quality [6, 7, 10, 2].

Almost all previously published placement algorithms use MCNC benchmark suite for testing. This suite was released in 1992 with most circuits having less than 30k cells. However, circuits designed in today's industry have typically more than 100k cells. While MCNC circuits become outdated, there is no new benchmarks released for placement. Some new placement tools [2] use large circuits obtained from industry for testing. Unfortunately, these circuits can not be accessed to other researchers due to the security reason. This makes it extremely difficult to evaluate and compare with these placement tools. In 1998, Alpert modified and released 18 industrial circuits from IBM to form the ISPD98 partitioning benchmark suite [1]. ISPD98 circuits are much larger (from 10k to 200k cells) and therefore closer to the current VLSI design than MCNC circuits. Although they were originally released for the purpose of partitioning, they should be modified and used in placement as well. Howevr, there is no reported placement results on these circuits yet.

In this paper we aim at developing a placement tool which can handle large industrial circuits. We argue that the top-down hierarchical approach should be the correct way to solve the large sized placement problem. We successfully integrate the state-of-the-art partitioner and placement technique into one fast and powerful placement tool, *Dragon2000*. Comparing Dragon2000 to highly optimized commercial iTools (formerly TimberWolf), Dragon2000 can produce slightly better placement results using much less amount of runtime (2× speedup). Dragon is for wirelength minimization, it can also be used as part of a congestion minimization process [9].

The rest of the paper is organized as follows: In Section 2, we briefly describe the flow and algorithms used in Dragon2000. In Section 3, we will explain the detailed implementation of Dragon2000. Experimental results will be shown in Section 4 followed by Conclusion in Section 5.

2. OVERVIEW OF DRAGON2000

The top-down based hierarchical approach is the backbone of Dragon2000. In this section, we will have a brief overview of the general hierarchical placement approaches and algorithms used in Dragon2000.

A typical top-down hierarchical placement approach can be generalized as follows: at a given hierarchical level, the layout area is partitioned into several global bins. All cells of the circuit will be distributed into these global bins to minimize a certain placement objective. This cell distribution problem is called a hierarchical placement problem. If a cell is distributed into a particular global bin, it will be placed within the area of this bin in the final layout. As we proceed to finer levels, the number of global bins increases and the physical size of global bins decreases. Thus we can get more and more detailed information about physical locations of cells as we proceed. The top-down approach terminates when there are only a few cells in each global bin.

Dragon2000 is divided into two phases, global placement (GP) and detailed placement (DP). A top-down hierarchical approach is used in the GP phase. We recursively solve the hierarchical placement problem and quadrisect each global bin into four smaller bins at each level. Overlap between cells are allowed in the GP phase. In fact, all the cells belong to the same bin are placed at the center of the bin. The DP phase takes the output from GP and produces an overlap free layout. Then it iteratively improves the legal layout using

a greedy heuristic. Due to the computational complexity, the DP heuristic is only capable of performing optimization locally. Thus it is expected that the top-down hierarchical GP phase should finish majority of work in placement.

Wirelength and net-cut are two popularly used objectives in different hierarchical placement algorithms. It is commonly believed that partitioning tools (minimizing net-cut) are much more mature and effective than wirelength minimization tools. On the other hand, wirelength at different hierarchical levels is a more accurate estimation of the final wirelength than net-cut. In order to achieve high performance, we integrate wirelength and net-cut together in the GP phase of Dragon2000 to take advantage of both objectives. Intuitively, net-cut correlates with wirelength. By using the Rent's rule and experimental data, we theoretically proved that the wirelength obtained from a top-down approach using the net-cut objective is indeed bounded.

Theorem 1 *In a top-down quadrisectional approach, the total wirelength at the final level H is between the total net-cut and the total net-cut times $2H$: $Cut \leq WL \leq (2 \log N_c) \cdot Cut$, where N_c is the number of cells in the circuit.*

Due to the page limit, we have to omit the actual proof here. Please contact us to obtain the complete proof.

3. DETAILED IMPLEMENTATION OF DRAGON2000

The top-down hierarchical approach is used in the GP phase of Dragon2000. We integrate net-cut and wirelength together to solve the hierarchical placement problem at each level. Specifically, we start our GP from level 1 with four global bins. We go from level h to level $h+1$ by partitioning each subcircuit in a global bin at level h into four parts to reduce net-cut. Global bins at level h will be split into four smaller bins correspondingly. Thus there will be 4^{h+1} global bins and 4^{h+1} subcircuits at level $h+1$. We have a *post bin swapping stage* at the end of each level. In this stage we swap all 4^{h+1} subcircuits in level $h+1$ around to minimize the overall wirelength. GP terminates when each global bin contains less than about seven cells.

A traditional top-down hierarchical placement approach confines locations of subcircuits at level $h+1$ within the region of the global bin where the subcircuits reside in at the previous level h. This approach can greatly reduce the computational complexity. However, it can never correct wrong decisions made at higher levels. Our GP does not confine locations of subcircuits. This gives cells more freedom to move to achieve better placement results at each level. In order to reduce runtime, we limit our wirelength optimization searches in a local range.

We pick hMetis [5] as the partitioner for Dragon2000 because of its superior quality and friendly user interface. A low temperature simulated annealing is used as the base algorithm to minimize wirelength because it is very easy to implement. The DP phase of Dragon2000 uses a greedy algorithm to perform local optimization and improve the quality of final placement iteratively. We will discuss implementation details of Dragon2000 in the following subsections.

3.1. Interactions between wirelength and net-cut

At each level in GP, we quadrisect each subcircuit inside a global bin into four smaller subcircuits before we perform the post bin swapping stage. Based on previous work and intuition, we tried four approaches in the partitioning stage to improve the performance. Figure 1 illustrates these approaches. Assume we are about to partition the subcircuit within global bin B_0. Cell 1, 2, 3, 4, 5, 6, 7 are inside B_0. Cell 8, 9, 10, 11 are outside B_0 but have connections to cells inside B_0. We denote an n-terminal net by net $(c_1, c_2, ..., c_n)$, where c_1, c_2, ..., c_n are terminal cells of the net.

1. Approach A (Figure 1a): When we partition a subcircuit in a global bin at any level, nets which have terminal cells outside this global bin will be removed because these nets are alway cut no matter how we distribute the inside cells. In Figure 1a, net $(2, 6, 11)$, $(5, 6, 10)$, $(4, 7, 8, 9)$, $(7, 9)$ are removed when partitioning subcircuit in B_0.
2. Approach B(Figure 1b): When we partition a subcircuit in a global bin at any level, terminal cells which are outside this global bin are ignored. In Figure 1b, since terminal cell 8, 9, 10, 11 are ignored, original net $(2, 6, 11)$, $(5, 6, 8)$, $(4, 7, 8, 9)$, $(7, 9)$ become new net $(2, 6)$, $(5, 6)$, $(4, 7)$, respectively (net $(7, 9)$ is gone). This method encourages grouping the remaining inside terminal cells together even there are always outside terminal cells.
3. Approach C(Figure 1c): In approach A and B, we isolated the subcircuits within a global bin by removing connections between inside cells and outside cells. Intuitively, this is not good. The idea of "terminal propagation" was proposed [3] to solve this problem. It adds to the current subcircuit *dummy cells* that are fixed in the appropriate partitions. In Figure 1c, cell 8 is mapped to a fixed vertex in the lower-left part of B_0; cell 9 and 10 are mapped to two fixed vertices in the lower-right part of B_0; cell 11 is mapped to a fixed vertex in the upper-right part of B_0. This approach encourages cells be distributed in a global bin which is close to their outside neighbor cells.
4. Approach D(Figure 1d): In all above approaches, the post bin swapping stage is used to swich subcircuits around. Instead of moving the subcircuits in whole, we can also move/switch single cells around to reduce wirelength at this level. This idea was first proposed in [7]. Wirelength obtained by this approach should be better than wirelength obtained by other three approaches. However, it is not clear whether a optimal wirelength placement at any hierarchical level will produce a good final placement.

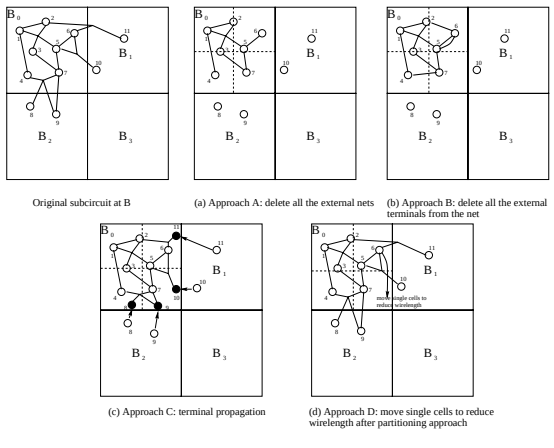

Figure 1. Illustration of Approach A, B, C and D.

In order to find out which approach performs the best, we test all four approaches on several benchmark circuits.

Table 1 shows the experimental results. The best result for each circuit among all four approaches is shown in the bold face. Quite surprisingly, approach B out-performs other approaches including approach C (terminal propagation) which is widely accepted and used in other placement tools.

In placement tools which use terminal propagation, subcircuits at each level are confined within the region of the global bin where they belong to at previous levels. We do not confine locations of subcircuits at each level because we perform the post bin swapping stage to reduce the overall wirelength after the subcircuits are formed. This post bin swapping stage is used in approach A, B and C. The use of the post bin swapping stage might be a reason why approach B works better in our GP than approach C does. To further investigate this interesting issue, we implemented the conventional min-cut scheme with and without terminal propagation. The conventional scheme with terminal propagation is basically the same as approach C except it does not perform post bin swapping. Similarly, the scheme without terminal propagation is the same as approach B without the post bin swapping stage. Table 2 shows the experimental results comparing two conventional min-cut schemes and their counter parts in our approaches (approach B and C). Indeed, the terminal propagation can improve the wirelength results over the non terminal propagation min-cut scheme. However, the post bin swapping stage can help improve performance: approach C (terminal propagation + post bin swapping) outperforms pure terminal propagation. Finally, it is very interesting to find that the widely used terminal propagation scheme actually degrade performance while the post bin swapping stage is used (approach B is better than approach C).

Another interesting fact is that approach D is not successful either. This fact suggests that conserving connecting information between cells is more important than greedily obtain a wirelength optimal placement at each level. Minimizing net-cut not only can obtain placement results fast at each level, it also helps to improve the final placement quality.

Ckts	#cells	App. A	**App. B**	App. C	App. D
ibm01	12282	4.79	**4.71**	4.98	4.81
ibm02	19321	**13.70**	13.91	14.38	13.99
ibm03	22207	13.12	**12.83**	13.02	12.93
ibm04	26633	17.66	**16.58**	17.54	17.21
ibm05	29347	38.94	**38.21**	39.32	39.12

Table 1. Comparison of four different approaches.

Ckts	mincut w/o term. prop.	mincut w/ term. prop.	App. C	**App. B**
ibm01	6.05	5.36	4.98	**4.71**
ibm02	16.77	15.04	14.38	**13.91**
ibm03	17.29	14.05	13.02	**12.83**
ibm04	22.58	18.34	17.54	**16.58**
ibm05	52.35	49.09	39.32	**38.21**

Table 2. Comparison of conventional min-cut schemes and our approaches.

3.2. The Final Stage of GP

The previous subsection shows that a minimum wirelength placement at each level does not help to produce a good final placement. However, we find that such a "single cell switching" strategy to minimize wirelength is extremely helpful in the last level of GP where there are about seven cells per global bin. After GP stops at the last level, we switch single cells locally to minimize wirelength. We use a low temperature simulated annealing algorithm in this final stage of GP. As shown in [7], since the number of possible locations for each cell is the number of global bins, the size of solution space is greatly reduced. Therefore, performing annealing at this stage is reasonably fast. Table 3 shows the comparison of the final placement wirelength using this final stage vs. not using this stage.

Ckts	w/o final stage	w/ final stage	% impr.
ibm01	4.99	4.70	5.8%
ibm02	14.71	13.76	6.5%
ibm03	13.56	12.74	6.0%
ibm04	17.07	15.79	7.5%
ibm05	42.19	38.57	8.6%
avg			6.88%

Table 3. Effect of the final GP stage.

3.3. DP Heuristics

The simulated annealing approach is the most popular DP algorithm used in other placement tools. However, due to the huge computational complexity in the DP phase, simulated annealing at this stage is very slow. For instance, the DP phase of iTools (formerly TimberWolf) consumes more than 80% of the total runtime on large circuits. In Dragon2000, since there is relatively little work left after GP is done. Instead of widely used simulated annealing, a greedy algorithm is used in the DP phase. Our DP consists of two steps. First, all the overlapping cells are spread out to produce a legal placement. Then the greedy cell exchange algorithm is used to further reduce wirelength. The algorithm randomly chooses a base cell. Then it decides whether to perform a vertical search or a horizontal search by using a adjustable parameter R_v. R_v is the ratio of vertical searches and satisfy $0 \leq R_v \leq 1$. If a vertical search is picked to perform, the cell directly above or below the base cell will be picked as the target cell. Positions of all the cells to the right of the base and the target cell might also be adjusted to remove the possible overlap and whitespace. If a horizontal search is picked to perform, all the $W-1$ cells to the left or right of the base cell will be picked as target cells. All the target cells and the base cells will be re-arranged horizontally in an exhaustive search manner to look for possible reduction in wirelength. We empirically set $W = 5$ and $R_v = 20\%$ in our DP.

4. EXPERIMENTAL RESULTS

Dragon2000 is implemented in C++. All the experiments were performed on a 500 MHz PC running under the Solaris-x86 operating system. We picked five large circuits from MCNC suite and eight large circuits from ISPD98 suite as our testing circuits. MCNC circuits are picked because they have been widely used in literature. However, all MCNC circuits except golem3 are too small (< 30k cells). The eight ISPD98 circuits we picked are relatively large (from 60k to 200k cells) in the suite . Due to the existence of very large cells, the original ISPD98 circuits are not suitable for standard cell placement. We modified ISPD98 circuits by removing very large cells. Specifically, we remove cells with the area larger than twenty times the area of the

smallest cell in the circuit. The degree of a net might decrease due to removing of large cells. Table 4. shows the characteristics of the testing circuits we use in this paper. We compare Dragon2000 with iTools1.4.0 (formerly TimberWolf, http://www.internetcad.com) on all testing circuits. Since we do not have the access to other older placement tools, we use numbers reported in literature on MCNC circuits for comparison. Since ISPD98 circuits have not been used for placement before, only Dragon2000 and iTools1.4.0 are compared on ISPD98 circuits.

Ckts	#cells	#nets	#pins	#rows
in2	12142	13419	125555	72
in3	15059	21940	176584	54
avqs	21854	22124	82601	80
avql	25114	25384	82751	86
golem3	99932	143379	336299	128
ibm11	68119	78843	248889	128
ibm12	69026	75157	301604	128
ibm13	81018	97574	311403	128
ibm14	147088	147605	547333	128
ibm15	157861	183684	653684	128
ibm16	181633	188324	762218	128
ibm17	182359	186764	834953	128
ibm18	210323	201560	817331	128

Table 4. Properties of the testing circuits.

Table 5 shows the placement results of Dragon2000, iTools1.4.0, TimberWolf7.0 and TUM [4] for MCNC circuits. Wirelength results of TimberWolf7.0 and TUM are obtained from [8] and [4], respectively. Runtime comparison is very difficult since different machines were used in literature. Therefore we do not report the runtime for TimberWolf7.0 and TUM. On small MCNC circuits (less than 30k cells), Dragon uses less time than iTools but the wirelength results are about 5% worse. However, it still outperforms other successful university tools like TimberWolf7.0 and TUM.

Ckts	TW7.0	TUM	iTools1.4.0		Dragon	
	WL	WL	WL	time	WL	time
ind2	13.53	14.6	12.30	1537	12.88	1461
in3	42.84	45.1	40.13	3154	42.33	2849
avqs	5.41	4.91	4.84	1915	5.17	1420
avql	5.86	5.38	5.19	2043	5.25	1984
golem3	90.39	-	85.44	24380	77.56	8422

Table 5. MCNC circuits comparison.

Table 6 shows the placement results of Dragon2000 and iTools1.4.0 on large testing circuits which has more than 60k cells including eight ISPD98 circuits and one MCNC circuit. On average, Dragon produces placement results with the same quality as iTools for these circuits while spending much less time (1.9× speedup). We also observed that Dragon performs better on circuits larger than 100k cells (1.4% better results and 2.1× speedup).

5. CONCLUSION

In this paper, we use a top-down hierarchical approach to develop a powerful standard cell placement tool, Dragon2000. We argue that net-cut minimization is a good and important shortcut to get high quality placement results in the shortest amount of time. In fact, experimental results show that

Ckts	iTools1.4.0		Dragon		Comparison	
	WL	time	WL	time	impr.	spdup
ibm11	39.76	18251	40.82	10301	-2.6%	1.8×
ibm12	69.56	18075	70.38	14198	-1.2%	1.3×
ibm13	49.11	22577	51.02	15456	-3.9%	1.5×
ibm14	118.8	43057	118.0	31894	0.7%	1.4×
ibm15	130.6	54262	130.8	22808	0.0%	2.4×
ibm16	163.8	70320	168.8	39001	-3.0%	1.8×
ibm17	256.6	72094	255.1	38752	0.5%	1.9×
ibm18	191.7	75363	189.6	39603	1.1%	1.9×
golem3	85.44	24380	77.56	8422	9.2%	2.9×
ave					0.1%	1.9×
ave*					1.4%	2.1×

Table 6. Placement results for 60k+ cell circuits (* average value for 100k+ cell circuits).

minimizing net-cut is more important than greedily obtain a wirelength optimal placement at intermediate hierarchical levels. We run Dragon2000 on recently released large benchmark suite ISPD98 and old MCNC suite. For circuits which have more than 100k cells, Dragon2000 can produce slightly better placement results (1.4%) while spending much less amount of time (2× speedup) than the highly optimized commercial iTools1.4.0. This is also the first published placement result on ISPD98 suite.

REFERENCES

[1] C. J. Alpert. "The ISPD98 Circuit Benchmark Suite". In *International Symposium on Physical Design*, pages 18–25. ACM, April 1998.

[2] A. E. Caldwell, A. B. Kahng, and I. L. Markov. "Can Recursive Bisection Alone Produce Routable Placements?". In *Design Automation Conference*. IEEE/ACM, 2000.

[3] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard Cell VLSI Circuits". *IEEE Transactions on Computer Aided Design*, 4(1):92–98, January 1985.

[4] H. Eisenmann and F. M. Johannes. "Generic Global Placement and Floorplanning". In *Design Automation Conference*, pages 269–274. IEEE/ACM, 1998.

[5] G. Karypis and V. Kumar. "Multilevel k-way Hypergraph Partitioning". In *Design Automation Conference*, pages 343–348, 1999.

[6] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization". *IEEE Transactions on Computer Aided Design*, 10(3):365–365, 1991.

[7] M. Sarrafzadeh and M. Wang. "NRG: Global and Detailed Placement". In *International Conference on Computer-Aided Design*. IEEE, November 1997.

[8] W. J. Sun and C. Sechen. "A Loosely Coupled Parallel Algorithm for Standard Cell Placement ". In *International Conference on Computer-Aided Design*, pages 137–144. IEEE, 1994.

[9] M. Wang, X. Yang, and M. Sarrafzadeh. "Congestion Minimization During Placement". *IEEE Transactions on Computer Aided Design*, 2000. to appear.

[10] X. Yang, M. Wang, K. Eguro, and M. Sarrafzadeh. "A Snap-On Placement Tool". In *International Symposium on Physical Design*, pages 153–158. ACM, April 2000.