Placement Benchmarks for 3-D VLSI

Stefan Thomas Obenaus
School of Computer Science
McGill University

Ted H. Szymanski
Communications Research Laboratory
McMaster University

Abstract We provide a general definition of the 3-D wirelength placement problem. This definition facilitates comparison of 3-D placement algorithms. Wirelength results using *partitioning placement* are included for the ACM/SIGDA and ISPD98 standard benchmark circuit suites. Further, a wirelength comparison between 2- and 3-D placements is made, and it is shown that larger circuits require 50%-60% less wirelength when utilising the third dimension.

Keywords: 3-D, Benchmarks, Placement, VLSI, Wirelength

1. INTRODUCTION

As research into three dimensional circuit architectures becomes more widespread (Chiricescu and Vai, 1998; Deprettere et al., 1994; Leeser et al., 1998; Leighton and Rosenberg, 1986; Ohmura, 1998; Tong and Wu, 1995), the study of techniques for placing circuits into 3-D structures is emerging. Unlike 2-D placement papers, previous 3-D placement results have been published in isolation (Leeser et al., 1998; Ohmura, 1998) and cannot be compared to each other for lack of benchmark results. In two dimensions wirelengths benchmarks have been a common means of comparing placement algorithms. However, in 3-D no tabulated results for comparison exists.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35498-9_57

L. M. Silveira et al. (eds.), *VLSI: Systems on a Chip*

Our goal is to fill the void and provide wirelength results for comparison of 3-D placement techniques. For this purpose, we have extended the established method of *partitioning placement* (Shahookar and Mazumder, 1991) to three dimensions. Other common placement methods are *simulated annealing* and *force-directed placement* (Shahookar and Mazumder, 1991). We use the currently fastest and best partitioner, hMetis (Karypis et al., 1997). The details of this placement method are described in section 2.. While using a different partitioner, the partitioning placement technique has already been used for 3-D placement for the Rothko architecture (Leeser et al., 1998).

Partitioning placement is fast and has traditionally yielded good results (Shahookar and Mazumder, 1991). For large circuits, which, as we show in section 3.2, profit most from 3-D technology, an asymptotically fast placement algorithm is essential. Asymptotically slower techniques, such as simulated annealing, are not likely to produce high quality results in an acceptable period of time for these circuits as the cooling schedule would have to be drastically shortened.

Given the high quality of the partitioner, and the fundamental soundness of the partitioning placement technique, we have confidence that the placement results presented here provide a good benchmark against which future 3-D placement techniques can be measured.

1.1 PLACEMENT MODEL

In order to provide a convenient basis for benchmark comparisons, the placement model should reflect reality, yet be general and easy to implement. In two dimensions, the *Checkerboard model* (Shahookar and Mazumder, 1991), a.k.a. the *Gate Array Model*, serves this purpose. In this model, circuit elements are of identical size and placed in checkerboard-fashion onto a grid. We extend this model to three dimensions in the natural way. While the purpose of this model is to facilitate comparison of placement algorithms, actual placement methods will have much more specific requirements for the grid size and shape, size of circuit elements, pad placements, etc.

The most common measure of quality of a placement algorithm is the total wirelength required for the circuit (Shahookar and Mazumder, 1991). The most efficient method of connecting points, eg. circuit nodes, in space along horizontal and vertical lines using a minimum of wirelength is to construct a *rectilinear Steiner tree* (Hanan, 1966).

Definition 1 *Rectilinear Steiner Tree*

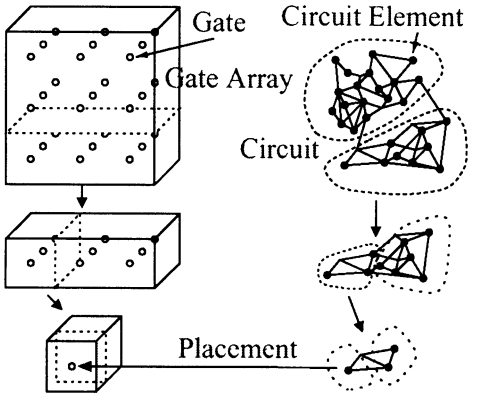

Figure 1.1 Partition Placement Process: Simultaneous Splitting of the Grid and Partitioning of the Circuit

A rectilinear Steiner tree $S(e, f)$ is the shortest tree that connects all nodes $v \in e$ at positions $f(v)$ using only orthogonal segments parallel to the coordinate axes. Its length is $|S(e, f)|$.

However, constructing a rectilinear Steiner tree and measuring its length is difficult. A common approach is to estimate the size of the Steiner tree by adding the width, height, and depth spanned by the nodes in the Steiner tree. This measure is accurate for two and three node nets. We call this approximation the semi-perimeter bounding box approximation $|S^*(e, f)| \approx |S(e, f)|$.

For placement purposes, we reduce all circuit elements to unit-size nodes, and thus we abstract the circuit into a hypergraph $G(V, E)$. $G(V, E)$ is simply the combination of the set of circuit elements, or nodes, V, with the set of nets, or hyperedges, E, where for all $e \in E$, $e \subset V$.

It remains to define the 3-D placement model, as a straight forward extension of the 2-D Checkerboard model, for comparison purposes:

Definition 2 *3-D Placement Model*
Given a circuit $G(V, E)$ find a reversible function

$$f : |V| \rightarrow \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \times \{1, \ldots, n_3\} .$$

The measure of quality of the placement function is the total wirelength estimate $\sum_{e \in E} |S^*(e, f)|$.

2. PLACEMENT METHOD

Partitioning placement is one of the fundamental placement methods (Shahookar and Mazumder, 1991). When placing a circuit into a two

Variables and predicates:

$V = \{v_1, \ldots, v_{\|V\|}\}$	set of circuit nodes
$x[v]$	coordinates of gate for circuit node v
(a_1, a_2, a_3)	coordinates of lower left front corner
(b_1, b_2, b_3)	lengths of sides of gate array box
(n_1, n_2, n_3)	initial size of gate array box

Initial Call:

`place`(V,$(0,0,0)$,(n_1, n_2, n_3))

`place`(V,(a_1, a_2, a_3),(b_1, b_2, b_3))

1: `if` $|V|$=1 `then`
2: $x[v_1] := (a_1, a_2, a_3)$
3: `else`
find largest side of box
4: $k := i$ such that $b_i = \max(b_1, b_2, b_3)$
split box b into two boxes b1 and b2
5: $(b1_1, b1_2, b1_3) := (b_1, b_2, b_3)$
6: $b1_k := \lfloor b_1/2 \rfloor$
7: $(b2_1, b2_2, b2_3) := (b_1, b_2, b_3)$
8: $b2_k := \lceil b_1/2 \rceil$
determine coordinates of lower left front corner of b1 and b2
9: $(a1_1, a1_2, a1_3) := (a_1, a_2, a_3)$
10: $(a2_1, a2_2, a2_3) := (a_1, a_2, a_3)$
11: $a2_k := a_k + b1_k$
partition V into sub-circuits V1 and V2
of sizes no more than $b1_1 \cdot b1_2 \cdot b1_3$ *and* $b2_1 \cdot b2_2 \cdot b2_3$, *respectively*
12: $(V1, V2) :=$ partition(V,$b1_1 \cdot b1_2 \cdot b1_3$,$b2_1 \cdot b2_2 \cdot b2_3$)
invoke placement routine on sub-circuits
13: `place`($V1$,$(a1_1, a1_2, a1_3)$,$(b1_1, b1_2, b1_3)$)
14: `place`($V2$,$(a2_1, a2_2, a2_3)$,$(b2_1, b2_2, b2_3)$)

Figure 1.2 Generic Partitioning Placement Algorithm

dimensional gate array, the gates in the gate array are recursively split into smaller sub-arrays. At the same time the circuit is partitioned and the sub-circuits are assigned to the sub-arrays until each circuit element can be assigned to its unique gate. This process extends to three dimensions in the obvious way, as is illustrated in figure 1.1. Figure 1.2 provides the partition placement algorithm.

Leeser et al. (Leeser et al., 1998) used a partitioning placement method for placement in the Rothko architecture. Their partition placement method was based on a 2-D variation of partitioning placement, called *quadrisection* (Shahookar and Mazumder, 1991). In quadrisection, the chip area is recursively split into four quadrants and circuits are recursively partitioned four ways. They extended this method into three dimensions by splitting the chip's volume into eight octants while concurrently partitioning the circuit eight ways. However, no placement results were published.

Circuit	Nodes	Nets	Pins	$\frac{Pins}{Node}$	$\frac{Pins}{Net}$	Circuit	Nodes	Nets	Pins	$\frac{Pins}{Node}$	$\frac{Pins}{Net}$
19ks	2844	3282	10547	3.71	3.21	s15850P	10470	10383	24712	2.36	2.38
avq.large	25178	25384	82751	3.29	3.26	s35932	18148	17828	48145	2.65	2.70
avq.small	21918	22124	76231	3.48	3.45	s38417	23949	23843	57613	2.41	2.42
baluP	801	735	2697	3.37	3.67	s38584	20995	20717	55203	2.63	2.66
biomedP	6514	5742	21040	3.23	3.66	s9234P	5866	5844	14065	2.40	2.41
golem3	103048	144949	338419	3.28	2.33	structP	1952	1920	5471	2.80	2.85
industry2	12637	13419	48158	3.81	3.59	t2	1663	1720	6134	3.69	3.57
industry3	15406	21923	65791	4.27	3.00	t3	1607	1618	5807	3.61	3.59
p1	833	902	2908	3.49	3.22	t4	1515	1658	5975	3.94	3.60
p2	3014	3029	11219	3.72	3.70	t5	2595	2750	10076	3.88	3.66
s13207P	8772	8651	20606	2.35	2.38	t6	1752	1641	6638	3.79	4.05

Table 1.1 The 1993 ACM/SIGDA Benchmark Circuits

As our partitioner, we chose the hMetis hypergraph partitioner developed by Karypis et al. (Karypis et al., 1997). This partitioner is currently the best and fastest partitioner published. Although w euse recursive tw o-way partitioning, we could easily implement 3-D quadrisection with a few modifications to the hMetis library interface. Restrictions in the current hMetis library interface made it necessary to compute a recursive balanced $(k + l)$-way partitioning to achieve a k:l split as is sometimes necessary in step 12 of the algorithm in figure 1.2 when an odd number of rows, columns, or layers needs to be split. While this increases run-time and memory requirement, it does not affect the quality of the cut (Karypis, 1999). According to Karypis, the hMetis interface could easily be adapted to allow explicit k:l cuts.

In order to compensate for the excessive memory requirement for large $(k+l)$-way partitionings, we restricted the largest dimensions of the gate arra ysfor the largest circuits to be of even length. Consequently, the largest cuts are balanced tw o-way cuts which require substantially less memory resources. F urther, to obtain accurate estimates of the runtime, assuming the hMetis interface w asadapted to allo wexplicit k:l splits, w e ran the algorithm while forcing balanced 2-way splits at all levels of recursion.

3. RESULTS

3.1 3-D RESULTS

We performed the benchmark runs on the circuits in the ACM/SIGDA circuit suite (Brglez, 1993), and the newer ISPD98 benchmark circuit suite (Alpert, 1998), cf. tables 1.1 and 1.2. While the ACM/SIGDA

Circuit	Nodes	Nets	Pins	$\frac{Pins}{Node}$	$\frac{Pins}{Net}$	Circuit	Nodes	Nets	Pins	$\frac{Pins}{Node}$	$\frac{Pins}{Net}$
ibm01	12752	14111	50566	3.97	3.58	ibm10	69429	75196	297567	4.29	3.96
ibm02	19601	19584	81199	4.14	4.15	ibm11	70558	81454	280786	3.98	3.45
ibm03	23136	27401	93573	4.04	3.41	ibm12	71076	77240	317760	4.47	4.11
ibm04	27507	31970	105859	3.85	3.31	ibm13	84199	99666	357075	4.24	3.58
ibm05	29347	28446	126308	4.30	4.44	ibm14	147605	152772	546816	3.70	3.58
ibm06	32498	34826	128182	3.94	3.68	ibm15	161570	186608	715823	4.13	3.84
ibm07	45926	48117	175639	3.82	3.65	ibm16	183484	190048	778823	4.24	4.10
ibm08	51309	50513	204890	3.99	4.06	ibm17	185495	189581	860036	4.64	4.54
ibm09	53395	60902	222088	4.16	3.65	ibm18	210613	201920	819697	3.89	4.06

Table 1.2 The 1998 ISPD Benchmark Circuits

Circuit	Grid			Average Wirelength	Standard Deviation	Minimum Wirelength	Runtime (s)
	n_1	n_2	n_3	$\sum \lvert S^* \rvert$		$\sum \lvert S^* \rvert$	
19ks	15	14	14	14493.3	1.61%	14123	37
avq.large	30	30	28	104104.1	1.22%	101693	303
avq.small	28	28	28	94688.0	0.88%	93823	277
baluP	10	9	9	3263.5	1.77%	3164	13
biomedP	19	19	19	25239.2	0.80%	24839	106
golem3	48	48	45	687104.9	0.68%	679554	1519
industry2	24	23	23	78997.7	0.92%	78052	202
industry3	26	25	24	152962.3	1.05%	149592	301
p1	10	10	9	4156.1	2.17%	4071	14
p2	15	15	14	18562.5	1.64%	18044	43
s13207P	21	21	20	26501.3	2.15%	25617	103
s15850P	22	22	22	30950.7	0.65%	30729	122
s35932	27	26	26	57926.1	1.74%	56120	218
s38417	29	29	29	73282.6	1.21%	72355	253
s38584	28	28	27	72643.9	1.10%	71560	258
s9234P	19	18	18	17670.1	0.89%	17463	86
structP	13	13	12	7064.1	1.41%	6879	23
t2	12	12	12	8501.9	1.44%	8337	22
t3	12	12	12	7828.2	1.42%	7658	22
t4	12	12	11	7375.9	1.63%	7242	22
t5	14	14	14	12568.2	0.57%	12481	36
t6	13	12	12	7968.1	1.51%	7785	23

Table 1.3 3-D Placement Results for the ACM/SIGDA Circuit Suite

Circuit	Grid			Average Wirelength	Standard Deviation	Minimum Wirelength	Runtime (s)
	n_1	n_2	n_3	$\sum \lvert S^* \rvert$		$\sum \lvert S^* \rvert$	
ibm01	24	24	23	92601.5	0.91%	90591	239
ibm02	28	28	26	202121.7	0.65%	199264	391
ibm03	30	30	27	234600.9	0.94%	231824	433
ibm04	32	30	29	301324.1	1.07%	297370	497
ibm05	32	32	30	367004.5	1.27%	360659	554
ibm06	32	32	32	333985.0	3.21%	323919	655
ibm07	36	36	36	473844.3	1.82%	458047	1084
ibm08	38	38	36	531860.7	1.38%	521203	1191
ibm09	38	38	37	617201.7	2.56%	587107	1263
ibm10	42	41	41	832125.1	3.79%	796763	1761
ibm11	42	41	41	872118.2	2.72%	842491	1661
ibm12	42	42	41	991783.9	1.63%	959567	1864
ibm13	44	44	44	1000941.8	2.05%	971330	1864
ibm14	54	54	52	1657408.1	1.12%	1630502	3064
ibm15	56	56	54	1994685.8	1.38%	1957427	3769
ibm16	58	58	57	2222138.0	1.05%	2190510	4030
ibm17	58	58	57	2745042.7	1.16%	2680986	4463
ibm18	60	60	59	2639356.6	4.29%	2504083	4285

Table 1.4 3-D Placement Results for the ISPD98 Circuit Suite

suite is more established, it lacks the larger circuits that can be found in the ISPD98 suite.

Since the hMetis partitioner is randomized, we perform 10 runs for each circuit on a Pentium II/300MHz system. The results are summarised in tables 1.3 and 1.4. The runtimes indicate the runtime for one run.

3.2 FROM 2 TO 3 DIMENSIONS

Besides having a reference platform for 3-D placement wirelengths, it is of interest to know what kind of improvement can be expected from 3-D VLSI. In table 1.5 we have compiled wirelength results for four circuits covering the dynamic range from 800 to 210,000 nodes. For each circuit we computed a placement in $d = 2$, 2 1/3, 2 2/3, and 3 dimensions. By this we mean that the base of the gate array has an edge length of $\sqrt[d]{N}$. Thus a 2-D gate array, is of size $\sqrt{N} \times \sqrt{N} \times 1$, and a 3-D gate array of size $\sqrt[3]{N} \times \sqrt[3]{N} \times \sqrt[3]{N}$. In general, for dimension d and an N-node circuit, we selected a grid size close to $\sqrt[d]{N} \times \sqrt[d]{N} \times N/(\sqrt[d]{N})^2$.

Circuit	Nodes	Grid			Dimensions	All Nets Counted		2- and 3-node Nets Only	
	$\|V\|$	n_1	n_2	n_3	d	Length $\sum\|S^*\|$	Change	Length $\sum\|S^*\|$	Change
p1	833	29	29	1	2	6751.5		3800.0	
		17	17	3	2 1/3	4584.2	-32.1%	2727.5	-28.2%
		12	12	6	2 2/3	4175.2	-38.2%	2505.5	-34.1%
		10	10	9	3	4143.5	-38.6%	2498.8	-34.2%
s9234P	5866	78	77	1	2	28353.2		17515.8	
		40	38	4	2 1/3	19192.9	-32.3%	13019.4	-25.7%
		26	26	9	2 2/3	17891.6	-36.9%	12598.2	-28.1%
		19	18	18	3	17640.6	-37.8%	12404.3	-29.2%
ibm06	32498	184	180	1	2	792670.2		224425.6	
		82	82	5	2 1/3	410851.0	-48.2%	125969.6	-43.9%
		50	50	13	2 2/3	372507.4	-53.0%	133579.7	-40.5%
		32	32	32	3	331429.9	-58.2%	116558.5	-48.1%
ibm18	210613	460	460	1	2	7705843.6		2561197.9	
		188	188	6	2 1/3	3597492.0	-53.3%	1258260.2	-50.9%
		98	98	22	2 2/3	2932640.5	-61.9%	1086833.2	-57.6%
		60	60	59	3	2641617.1	-65.7%	969398.1	-62.2%

Table 1.5 Wirelength Improvement from Two to Three Dimensions

Besides the usual total wirelength estimate, table 1.5 also shows the total wirelengths of nets with 2 and 3 nodes. For such nets, the semi-perimeter bounding box approximation of the wirelength is exact.

We notice that a substantial wirelength advantage can be achieved for large circuits even when only a few layers are employed. While small circuits exhibit only a modest wirelength improvement in 3-D, larger circuits clearly benefit from the third dimension. Small circuits `p1` and `s9234P` experience only a 30%-40% improvement in a 3-D placement, whereas `ibm18` saves approximately 50% wirelength in 2 1/3 dimensions and over 60% in 3 dimensions.

4. CONCLUSION

We presented a general definition of the 3-D placement problem. This definition can be used to compare 3-D placement algorithms to each other. On the basis of this definition, we implemented a partitioning placer and generated a first set of comprehensive wirelength results for 3-D placements of circuits in two benchmark circuits suites.

Further, we compared wirelength results of selected circuits for placements varying from two to three dimensions. These results provide empirical evidence that large circuits benefit substantially from utilising

the third dimension even if only a small number of layers in the third dimension is used.

Acknowledgments

We thank Prof. Karypis for making the hMetis 1.5.3 library available and providing support for its use.

References

Alpert, C. J. (1998). The ISPD98 circuit benchmark suite. In *Proceedings of the International Symposium on Physical Design*, pages 85–90.

Brglez, F. (1993). ACM/SIGDA design automation benchmarks: Catalyst or anathema? *IEEE Design & Test of Computers*, 10(3):87–91.

Chiricescu, S. M. S. A. and Vai, M. M. (1998). A three-dimensional FPGA with an integrated memory for in-application reconfiguration data. In *1998 IEEE International Symposium on Circuits and Systems*, volume 2, pages 232–235.

Depreitere, J., Neefs, H., Van Marck, H., Van Campenhout, J., Baets, R., Dhoedt, B., Thienpont, H., and Veretennicoff, I. (1994). An optoelectronic 3-d field programmable gate array. In *Field-programmable logic : architectures, synthesis, and applications : 4th International Workshop on Field-Programmable Logic and Applications*, volume 849 of *Lecture notes in computer science*, pages 352–360.

Hanan, M. (1966). On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265.

Karypis, G. (1999). private communication.

Karypis, G., Aggarwal, R., Kumar, V., and Shekhar, S. (1997). Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings of the 34th ACM/IEEE Design Automation Conference*, pages 526–529.

Leeser, M., Melcis, W. M., Vai, M. M., Chiricescu, S., Xu, W., and Zavracky, P. M. (1998). Rothko: A three-dimensional FPGA. *IEEE Designs and Test of Computers*, 15(1):16–23.

Leighton, F. T. and Rosenberg, A. L. (1986). Three-dimensional circuit layouts. *SIAM Journal on Computing*, 15(3):793–813.

Ohmura, M. (1998). An initial placement algorithm for 3-d VLSI. In *1998 IEEE International Symposium on Circuits and Systems*, volume 6, pages 195–198.

Shahookar, K. and Mazumder, P. (1991). VLSI cell placement techniques. *ACM Computing Surveys*, 23(2):143–220.

Tong, C. C. and Wu, C. (1995). Routing in a three-dimensional chip. *IEEE Transactions on Computers*, 44(1):106–117.