THE ISPD98 CIRCUIT BENCHMARK SUITE

Charles J. Alpert
IBM Austin Research Laboratory, Austin TX 78758
alpert@austin.ibm.com

Abstract

From 1985-1993, the MCNC regularly introduced and maintained circuit benchmarks for use by the Design Automation community. However, during the last five years, no new circuits have been introduced that can be used for developing fundamental physical design applications, such as partitioning and placement. The largest circuit in the existing set of benchmark suites has over 100,000 modules, but the second largest has just over 25,000 modules, which is small by today's standards. This paper introduces the ISPD98 benchmark suite which consists of 18 circuits with sizes ranging from 13,000 to 210,000 modules. Experimental results for three existing partitioners are presented so that future researchers in partitioning can more easily evaluate their heuristics.

1 Introduction

For over a decade, the Design Automation (DA) community has heavily relied on circuit benchmark suites to compare and validate their algorithms. Hundreds and perhaps thousands of publications have presented experimental results which use the circuits originally released by the Microelectronics Center of North Carolina (MCNC) and sponsored by ACM/SIGDA [3]. Indeed, papers in several fields, such as partitioning and placement, hardly stand a chance of being accepted into one of the major DA conferences without including experimental results that utilize these benchmarks. These benchmark suites (e.g., ISCAS85, ISCAS89, LayoutSynthesis92, Partitioning93, etc.) are currently maintained by the Collaborative Benchmarking Laboratory at North Carolina State University (www.cbl.ncsu.edu).

From 1985-1993, new suites of circuit benchmarks were regularly released; however, no new circuits have been released since. Most of these circuits are now obsolete, and do not adequately represent the complexity of modern designs. Consequently, there is a widening gap between the problems that are being solved in the academic literature and the problems that need to be solved. For example, a placer which achieves "5% improvement" on a design with 20 thousand moveable objects is not nearly as interesting or relevant as a placer which achieves "5% improvement" on a design with 200 thousand moveable objects.

One might argue that hierarchical design methodologies eliminate truly massive physical design problems. Currently, the only circuit in the existing suite of benchmarks with more than 26 thousand modules is golem3. However, given that next generation microprocessors will have between 20 and 50 million transistors, a physical design problem with just 1% of this complexity will still have between 200 and 500 thousand objects. It is not unreasonable to expect partitioning and placement problems of relatively small macros to reach this complexity. Indeed, physical design problems of this size have already been encountered within IBM. Given that golem3 is the only circuit in the public domain that can be said to represent medium to large designs, it seems unlikely that the academic community will be able to supply the algorithms that can manage the complexity expected in future designs.

circuit	#Modules	Dutt/Deng	hMetis	ML_C	LSR/MFFS
biomed	6514	83	83	83	83
s13207	8772	66	55	55	61
s15850	10470	56	42	44	43
industry2	12637	174	167	164	----
industry3	15406	241	254	243	----
s35932	18148	42	42	41	44
s38584	20995	47	47	47	47
avq.small	21918	129	130	128	127
s38417	23849	65	51	49	50
avq.large	25178	127	127	128	127

Table 1: Partitioning results for the ten largest current benchmarks (except for golem3). Solutions may have up to 10% deviation from exact bisection, and each pad and cell is assigned unit area.

The partitioning problem provides a perfect example of how both the academic and industrial community is likely to suffer from the lack of an up-to-date benchmark suite. Over the last few years, several innovative partitioning algorithms have been proposed, e.g., [1][6][8][14], and the state of the art has advanced significantly (see [2] for a survey). However, the most recent partitioners are achieving virtually identical solution quality for most of the current benchmarks. Table 1 shows the minimum cut bipartitioning results (with a 45/55 partition size balance constraint) obtained by four algorithms: Dutt/Deng[1] [8], hMetis [14], ML_C [1] and LSR/MFFS[2] [6]. Observe that there are very small differences in solution quality for almost every benchmark. Indeed, complete convergence has been obtained by several partitioners for the smaller benchmarks balu (cut=27), struct (cut=33) and s9234 (cut=40). Consequently, it appears impossible for any future partitioner to obtain more than,

1 Dutt and Deng present a general scheme for improving any iterative improvement engine. They present experiments with CLIP and CDIP on iterative improvement engines using lookahead, not using lookahead, and with probabilistic moves. Table 1 quotes the best results reported over all of the algorithms the authors proposed.
2 The results in [6] actually use non-unit area. The data for the unit area experiments quoted here was obtained directly from Sung Lim.

say, a 2% average improvement over the current best partitioner. This state of affairs hardly means that partitioning is a solved problem. Rather, with over five years of opportunities to optimize a fixed suite of benchmarks, the DA community has collectively succeeded in finding superior partitioning solutions for these benchmarks. However, virtually nothing is known about what partitioners will work best or be most efficient on designs with 150 thousand or more moveable objects. Without the introduction of new, larger circuits, the CAD literature in pure partitioning will certainly die.

Circuit	# Cells	# Pads	#Modules	# Nets	# Pins	Max%
ibm01	12506	246	12752	14111	50566	6.37
ibm02	19342	259	19601	19584	81199	11.36
ibm03	22853	283	23136	27401	93573	10.76
ibm04	27220	287	27507	31970	105859	9.16
ibm05	28146	1201	29347	28446	126308	0.00
ibm06	32332	166	32498	34826	128182	13.56
ibm07	45639	287	45926	48117	175639	4.76
ibm08	51023	286	51309	50513	204890	12.10
ibm09	53110	285	53395	60902	222088	5.42
ibm10	68685	744	69429	75196	297567	4.80
ibm11	70152	406	70558	81454	280786	4.48
ibm12	70439	637	71076	77240	317760	6.43
ibm13	83709	490	84199	99666	357075	4.22
ibm14	147088	517	147605	152772	546816	1.99
ibm15	161187	383	161570	186608	715823	11.00
ibm16	182980	504	183484	190048	778823	1.89
ibm17	184752	743	185495	189581	860036	0.94
ibm18	210341	272	210613	201920	819697	0.96

Table 2: ISPD98 circuit benchmark characteristics. Max% gives the percent of the total area occupied by the largest module in the circuit.

To offset the lack of public benchmarks, several works have studied random circuit generation. Success in this research domain could certainly offset the lack of available large circuits, yet much work remains. Early works, such as Bui et al. [5] and Garbers et al. [10], propose classes of random graphs that have natural clustering and partitioning solutions. More recent works, such as Darnauer and Dai [7] and Hutton et al. [12], generate random circuits that seek to capture such properties of real circuits as Rent parameter, circuit shape and depth, fanout distribution, reconvergence, etc. While these circuits are better than random graphs in representing real circuits, they are no substitute for actual test cases.[3]

The purpose of this work is to release a new set of circuits, called the ISPD98 benchmark suite, for physical design applications. The circuit sizes range from 13,000 to 210,000 modules and were translated from internal IBM designs. The circuits can be downloaded via the World Wide Web at vlsicad.cs.ucla.edu. In addition, some partitioning results are presented to enable easy comparisons for future work.

2. A New Set Of Circuits

Table 2 presents the characteristics of the 18 circuits in the ISPD98 benchmark suite. The circuits are all generated from IBM internal designs produced at the Austin, Burlington and Rochester sites. The designs represent many types of parts, including bus arbitrators, bus bridge chips, memory and PCI bus interfaces, communication adaptors, memory controllers, processors, and graphics adaptors. For each circuit, a cell is considered to be an internal moveable object, a pad is an external (perhaps moveable) object, and a module is either a cell or a pad. The last column, Max%, gives the percent of the total area occupied by the largest module in the design. This percentage gives some idea as to how easy it is to partition the design under tight balance constraints.

Each circuit is a translation from VIM (IBM's internal data format) into "net/are" format, a simple hypergraph representation originally proposed by Wei and Cheng [15] (see vlsicad.cs.ucla.edu for benchmarks in this format). In addition, a new format called "netD" is introduced, as described below. The circuits can be downloaded from vlsicad.cs.ucla.edu and complete descriptions of the benchmark formats can also be found there. The translation from VIM to "net/are" is performed as follows.

- All information relating to circuit functionality, timing and technology is removed. Unfortunately, this limits the direct applicability of these circuits (e.g., functional replication for partitioning); yet, the release of these circuits would have been impossible otherwise. Nevertheless, other applications besides pure partitioning can still be developed from this suite of circuits by making reasonable assumptions.
- All nets with more than 200 pins are removed from the design; most of these are likely related to clock and power distribution. The omission of these nets makes it more difficult to distinguish sequential from combinational cells. However, in modern design methodologies, layout is generally performed without the clock nets since they can bias the objective functions for partitioning and placement. For example, a placement algorithm might try to minimize the wirelength of the clock nets, forcing sequential elements to be clustered together. This may lead to an unbalanced clock distribution and misappropriation of clock resources.
- Small components that are disconnected from the largest

3 In unrelated experiments, we obtained several randomly generated circuits from the authors of [12] and ran the partitioners FM, CLIP, ML_F and ML_C [1] on these circuits. No partitioner distinguished itself as significantly superior, yet the authors of [1] clearly show that the multilevel approaches (ML_F and ML_C) significantly outperform FM and CLIP on the ACM/SIGDA benchmark suites. These experiments indicate that the randomly generated circuits are not yet adequate for benchmarking, at least for partitioning applications.

component of the circuit are removed. This helps disguise the design while having virtually no effect on the layout since the disconnected components constitute a very small percentage of the layout area. As a side benefit, optimization techniques can be applied more easily. For example, spectral methods will not compute non-degenerate eigenvectors, flow based methods only need to construct a single network, and search based methods need to start from only a single module.

- Duplicate pins are removed. If a given net is connected to multiple pins incident to the same cell, then only one of these pins is included in the translated circuit. This has no effect on the topology of the netlist, but makes it easier to write physical design tools. For example, it simplifies the updating of gain buckets in Fiduccia-Mattheyses partitioning.
- All internal cells and pads are randomly numbered. Pads are assigned a default area of 0.

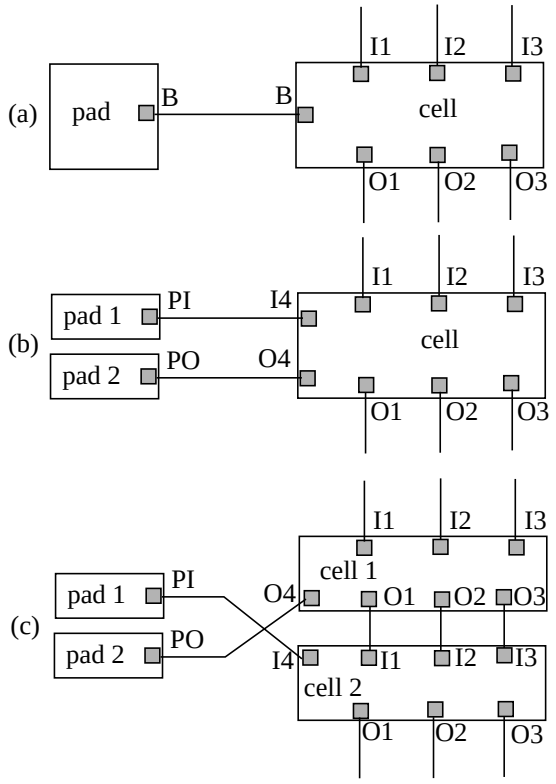

Figure 1: (a) Typical occurrences of bidirectional pads, and possible modelings by splitting (b) only the pad and (c) the pad and the cell.

One shortcoming with the original net/are format is that signal direction information is not preserved, so we propose a new format called "netD". This format is identical to net/are format except that each module in a given net is identified as either an input, output or bidirectional pin for that net. This information should enable one to apply standard directional clustering techniques such as cones and MFFCs [6]. The netD format subsumes net format, but the web site will maintain net format to ensure backward compatibility with existing tools.

A potential problem with interpreting the signal direction information lies in handling bidirectional pads. Due to strict I/O limits in many technologies, a large percentage of the pads (up to 90%) in many designs are bidirectional. This makes it difficult to perform many operations, such as computing the longest paths from primary inputs to primary outputs, or generating cones. Figure 1(a) illustrates a typical instance. Here, a 2-pin net connects a bidirectional pad to an internal cell which also has contains three inputs (I1, I2, I3) and three outputs (O1, O2, O3).

Circuit	FM			CLIP			hMetis		
	Min	Avg	CPU	Min	Avg	CPU	Min	Avg	CPU
ibm01	191	466	4.1	181	390	5.5	181	236	2.4
ibm02	266	506	6.9	265	545	10.0	262	312	5.8
ibm03	1150	2131	16.3	1068	1593	16.1	959	1068	6.8
ibm04	603	1105	14.0	563	1030	16.0	542	588	7.3
ibm05	1874	3063	24.4	2146	3016	27.0	1740	1838	9.1
ibm06	973	1384	16.7	977	1520	19.6	885	1023	10.7
ibm07	1037	2036	26.5	929	1987	30.9	848	930	17.8
ibm08	1285	2757	41.0	1261	2137	48.7	1159	1194	25.7
ibm09	912	2547	40.1	674	1770	37.3	624	685	14.9
ibm10	1490	2660	51.2	1420	2745	59.0	1265	1573	29.8
ibm11	1459	4173	52.6	1063	2657	57.7	963	1146	26.3
ibm12	2256	3791	71.6	2387	3770	67.7	1899	2123	37.8
ibm13	1181	2249	59.3	913	1955	67.7	841	979	32.5
ibm14	2963	6824	163.1	2536	4176	181.0	1928	2126	71.8
ibm15	5106	7770	123.1	3571	5689	215.4	2750	3218	99.0
ibm16	2363	5668	143.5	2638	5974	213.6	1758	2339	103.3
ibm17	3052	7212	188.6	2803	6998	210.3	2341	2430	120.6
ibm18	1706	3686	204.5	2268	5227	334.0	1528	1669	89.2

Table 3: Min-cut bipartitioning results with up to 10% deviation from exact bisection. Each cell and pad is assigned unit area.

To apply cone-based techniques, one must construct an equivalent circuit without bidirectional pads. One possibility is to split the pad into a primary input (PI) and a primary output (PO) as shown in Figure 1(b). A potential problem that arises is that the path that goes from pad 1 through the cell and then to pad 2 does not really exist. Special care would have to be taken to avoid these "false paths". Figure 1(c) shows another alternative in which both the pad and cell are replicated. All the appropriate paths are preserved, but having two distinct cells becomes problematic since both cells must always appear in the same partition. Neither (b) nor (c) may be the best way to model bidirectional pads for cone-like constructions. We leave this issue open to future researchers.

Circuit	FM			CLIP			hMetis		
	Min	Avg	CPU	Min	Avg	CPU	Min	Avg	CPU
ibm01	270	486	4.5	246	462	5.5	188	262	2.4
ibm02	313	3872	3.9	439	4163	7.3	121	228	4.7
ibm03	1624	12348	0.3	1915	9720	27.4	234	341	5.2
ibm04	554	2383	14.1	488	1232	11.5	444	525	6.0
ibm05	1874	3063	24.4	2146	3016	27.0	1744	1828	9.8
ibm06	1479	14007	36.5	1303	15658	76.9	491	685	10.3
ibm07	870	1716	24.1	748	1711	35.7	818	1030	16.1
ibm08	1411	13422	28.1	2176	15907	84.0	1178	1343	24.0
ibm09	750	3235	32.1	527	2828	31.6	573	780	15.1
ibm10	982	2244	37.1	971	2242	58.3	286	515	22.1
ibm11	1319	3562	49.9	977	2527	56.6	756	1107	24.0
ibm12	2306	10723	49.3	2713	10112	36.9	472	965	29.5
ibm13	1196	2129	48.9	1023	2075	69.4	755	1102	33.8
ibm14	3015	6558	143.0	2426	4208	157.0	1945	2161	76.6
ibm15	7197	85465	25.5	5292	62105	794.0	2143	2676	78.9
ibm16	2173	5267	13.7	2314	5975	24.5	2076	2437	90.0
ibm17	2818	6725	185.2	3634	7024	227.2	2297	2412	140.8
ibm18	1664	3539	217.2	3043	5234	363.8	1528	1650	96.3

Table 4: Min-cut bipartitioning results with up to 10% deviation from exact bisection. Cells are assigned non-unit (actual) areas.

3. Partitioning Results

We now present results for three partitioners on the new suite of circuits. The purpose is not to make a comparative evaluation of current partitioners, but rather to provide a set of data for use by future researchers. We ran three partitioning algorithms: Fiduccia-Mattheyses (FM) [9], CLIP [8], and hMetis [14]. Implementations of FM and CLIP use a LIFO bucket structure and were obtained from the authors of [1], and the hMetis executable was obtained from the authors of [14]. FM is the industry standard iterative exchange heuristic, CLIP is a modification of FM that biases cells to move in clusters, and hMetis is a multilevel partitioner. hMetis offers a choice of several different coarsening schemes, uncoarsening schemes, and V-cycle refinement schemes. We use the default schemes as described in [13].

Results are presented for two different modelings of the cells: (i) each cell and pad has unit area; (ii) each pad has area zero, and each cell has non-unit (actual) area as specified in the appropriate area file. The reasons for including both are somewhat historical. Unit areas are more prominent in the literature (partly due to the absence of area data) and is in some sense a "purer" partitioning problem. Implementation of a partitioner is much simpler with unit areas since enforcement of balance constraints is simple. However, non-unit (actual) areas affords a much more realistic problem formulation. As the following results show, there are some problems with partitioning with non-unit areas that need to be addressed.

Circuit	FM			CLIP			hMetis		
	Min	Avg	CPU	Min	Avg	CPU	Min	Avg	CPU
ibm01	203	513	4.2	207	519	5.9	203	274	2.5
ibm02	352	536	8.3	357	585	8.1	353	384	6.4
ibm03	1180	2274	15.7	1054	1578	12.0	957	1048	7.2
ibm04	820	1340	15.0	632	1167	20.5	598	660	7.3
ibm05	2017	3142	24.4	1820	3002	27.7	1738	3476	9.1
ibm06	1087	1575	22.6	1017	1561	20.3	981	1116	9.9
ibm07	1133	2429	28.8	1041	1960	37.5	983	1043	14.9
ibm08	1271	2881	57.8	1279	2589	49.8	1159	1217	25.4
ibm09	1261	2720	39.0	676	1795	39.5	629	670	15.8
ibm10	1711	2668	50.7	1540	2613	54.6	1329	1549	30.7
ibm11	1941	5063	56.7	1263	2878	70.3	1075	1307	30.4
ibm12	2507	3841	56.9	2251	3753	68.1	2014	2297	33.8
ibm13	1414	2780	54.1	1013	2231	67.9	860	1100	34.8
ibm14	3668	7926	157.1	2425	4247	215.1	1897	2185	74.5
ibm15	5328	8822	139.4	3850	5795	193.8	3007	3520	108.9
ibm16	3345	6294	153.8	2815	6219	270.4	2309	2571	120.3
ibm17	3651	8096	194.3	3859	7512	259.8	2479	2719	159.4
ibm18	1778	3836	243.1	2685	5853	374.7	1603	1818	149.2

Table 5: Min-cut bipartitioning results with up to 2% deviation from exact bisection. Each cell and pad is assigned unit area.

Table 3 presents bipartitioning results for the designs for unit cell and pad area and allowing up to 10% deviation from exact bisection, i.e., each partition must have between 45% and 55% of the total area. Both the minimum and average cuts over 100 runs of each algorithm are reported. The CPU column gives the average time required for a single run of each algorithm. Runtimes are reported for an 135 MHz IBM RS6000 S/595. Table 4 presents the same set of experiments except that the cells have non-unit areas, given in the "are" file.

Tables 5 and 6 present similar results for the three partitioners, this time allowing up to 2% deviation from exact bisection, i.e., each partition must consist of between 49% and 51% of the total area. Table 5 presents results for unit cell and pad area, while Table 6 pre-

sents results for non-unit area. Observe that some of the cut sizes for both FM and CLIP are very large in both Tables 4 and 6 for several circuits, e.g., ibm05, ibm07, ibm12 and ibm15. These large results do not necessarily reflect that FM and CLIP are poor algorithms, but rather that the implementation [1] is not particularly good at satisfying balance criteria when there are large variations in cell sizes. Indeed, the problem of even finding an exact bisection is NP-Complete when cells have non-unit areas [11]. Thus, when area constraints are fairly restricted and there are several cells with large areas, sophisticated balancing and rebalancing schemes need to be incorporated (at least in an iterative approach). This aspect of iterative partitioning has not been very actively researched. Some open questions include how to choose which partition to move a cell from, how to rebalance a solution that has become unbalanced by a given move, and how to handle designs with very large cells (e.g., more than 10% of the total area).

Circuit	FM			CLIP			hMetis		
	Min	Avg	CPU	Min	Avg	CPU	Min	Avg	CPU
ibm01	450	2701	2.1	471	2456	4.6	188	297	2.3
ibm02	648	12253	0.6	1228	12158	2.2	113	200	5.5
ibm03	2459	16944	0.5	2569	16695	0.8	427	629	5.5
ibm04	3201	20281	0.5	17782	20178	0.5	458	582	6.7
ibm05	2397	3420	26.4	1990	3156	29.9	1745	3490	9.7
ibm06	1436	16578	2.7	1499	18154	16.1	498	836	10.1
ibm07	4139	31096	2.2	14166	31326	4.1	868	1074	17.6
ibm08	2010	29962	8.3	4283	30694	22.2	1272	1426	23.4
ibm09	3246	36433	1.4	2144	37124	1.3	572	754	17.8
ibm10	3210	44262	2.8	5958	46700	3.3	629	797	22.8
ibm11	4814	44071	5.5	2269	46795	54.8	801	1202	27.6
ibm12	4761	47680	4.8	41858	49428	1.7	1297	1740	34.0
ibm13	3982	58288	3.1	2750	54160	64.9	857	1216	30.8
ibm14	3083	28618	13.2	2571	6022	12.7	1914	2239	74.8
ibm15	7221	$> 10^5$	12.9	5173	82026	418.9	2435	3202	99.6
ibm16	3416	$> 10^5$	17.8	3677	74700	103.4	2277	2652	107.4
ibm17	3634	7873	259.8	4213	6864	182.9	2389	2683	125.9
ibm18	1906	3786	252.2	3156	6113	415.5	1630	1834	146.3

Table 6: Min-cut bipartitioning results with up to 2% deviation from exact bisection. Cells are assigned non-unit (actual) areas.

.

Circuit	Unit Area			Non-Unit (Actual) Area		
	Cut	SOD	CPU	Cut	SOD	CPU
ibm01	496	1017	4.4	494	1029	4.4
ibm02	640	1351	11.2	354	740	9.3
ibm03	1737	3720	11.2	1155	2368	12.1
ibm04	1712	3596	14.0	1410	2997	13.5
ibm05	3092	6851	15.4	3103	6877	15.2
ibm06	1645	3795	16.8	1147	2451	17.1
ibm07	2179	4604	31.0	1915	4019	25.9
ibm08	2436	5231	41.6	2172	4241	36.6
ibm09	1723	3599	28.2	968	2021	30.7
ibm10	2328	4964	50.6	1531	3149	48.4
ibm11	3267	4847	45.8	2048	4220	46.8
ibm12	3784	8076	55.9	2445	5080	59.9
ibm13	1800	3893	54.8	1394	2960	56.3
ibm14	3399	7585	122.0	3342	7131	121.0
ibm15	5124	10925	169.1	4795	10241	145.2
ibm16	3944	8283	170.6	3646	7690	154.0
ibm17	5465	11373	228.8	5538	11501	200.6
ibm18	2908	6758	173.1	2919	6662	175.4

Table 7: Net cut, Sum of Degrees, and CPU times for 100 runs of hMetis 4-way partitioning for both unit and non-unit areas. Solutions were allowed to deviate up to 10% from exact quadrisection, i.e., each partition has between 22.5% and 27.5% of the total area.

Finally, Table 7 and Table 8 respectively present results for 4-way and 8-way partitioning, obtained by recursively applying hMetis. The solutions are the best recorded over 100 runs, and CPU is the amount of time for a single run. Note that hMetis first performs 100 runs of 2-way partitioning, chooses the best solution, then performs 100 runs on each of the two subpartitions. In the tables, “Cut” refers to the total number of nets cut by the solution, and “SOD” refers to the Sum of Degrees objective. Sum of Degrees is the sum over all partitions of the number of cut nets incident to the partition (see [1]). The same parameters are used as for hMetis bipartitioning, and the area of each partition can vary up to 10% from exact quadrisection or octisection. Results are given for both unit and non-unit areas. Note that for ibm03, hMetis is unable to find an 8-way partitioning solution for non-unit areas. This is most likely due to the presence of the large module which occupies 10.76% of the total area.

4. Conclusions

A new set of benchmarks is introduced for physical design applications. Results for several experiments are reported to serve as a stepping stone for future work in partitioning. It is our hope that others in industry will follow suit and make efforts to publish their data as well. Providing data in these simple formats does not compromise the intellectual property of the design, yet gives enough topological information to form real challenges to modern PD tools.

Circuit	Unit Area			Non-Unit (Actual) Area		
	Cut	SOD	CPU	Cut	SOD	CPU
ibm01	767	1642	6.4	790	1728	6.0
ibm02	1887	4002	16.2	650	1418	14.6
ibm03	2492	5858	14.5	---	---	---
ibm04	2821	6252	19.8	2576	5702	18.9
light	4482	11755	19.6	4548	11892	19.4
ibm05	2309	5837	24.7	1771	4272	24.1
ibm06	3344	7599	38.1	3061	7113	35.8
ibm07	3647	8725	51.8	3143	7549	48.1
ibm08	2663	5895	38.4	2045	4351	41.6
ibm10	3845	8454	70.6	2218	4828	67.4
ibm11	3585	7897	59.9	3137	6784	55.8
ibm12	6122	13483	76.3	4315	9013	84.8
ibm13	2972	6794	71.7	2332	5210	75.5
ibm14	5308	12025	163.7	5005	11744	156.1
ibm15	6943	15379	189.0	6967	16026	186.8
ibm16	6300	13640	223.9	5567	12121	227.1
ibm17	9052	19882	293.6	8736	18997	307.2
ibm18	5441	12663	239.3	5349	12504	249.2

Table 8: Net cut, Sum of Degrees, and CPU times for 100 runs of hMetis 8-way partitioning for both unit and non-unit areas. Solutions were allowed to deviate up to 10% from exact octisection, i.e., each partition has between 11.25% and 13.75% of the total area.

Acknowledgments

The release of these circuits would have been impossible without the help of several IBM colleagues. Many thanks are due to Steve Quay and Paul Villaruvia for helping with the translation code, to Tom Lanzoni, Steve Mercier, Mike Trick and Bruce Winter for making the design data available, and to Patrick O'Connor, George Doerre, Jim Baker, Jim Barnhart, Jon Byrn, Greg Dancker, Sumit DasGupta, Nancy Duffield, Ray Eberhard, Al McGreevy, Dan Moertl, Greg Still, Don Fuchik and Scott Smith for their support of this project. Also, thanks to University of Minnesota Professors George Karypis and Vipin Kumar for supplying the hMetis executable and for their helpful discussions, and thanks to Jason Cong, Andrew Kahng, Sung Lim, and Dongmin Xu for their assistance.

References

[1] C. J. Alpert, J.-H. Huang and A. B. Kahng, "Multilevel Circuit Partitioning", *34th IEEE/ACM Design Automation Conference*, 1997, pp. 530-533.

[2] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey", *Integration, the VLSI Journal*, 19 (1995), pp. 1-81.

[3] F. Brglez, "ACM/SIGDA Design Automation Benchmarks: Catalyst or Anathema?", *IEEE Design & Test*, 10(3), September, 1993, pp. 87-91.

[4] F. Brglez, D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *International Conference on Circuits and Systems*, 1989, pp. 1929-1934.

[5] T. Bui, S. Chaudhuri, T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior", *Combinatorica* 7(2), 1987, pp. 171-191.

[6] J. Cong, H. P. Li, S. K. Lim, T. Shibuya and D. Xu, "Large Scale Circuit Partitioning with Loose/Stable Net Removal and Signal Flow Based Clustering", *IEEE/ACM International Conference on Computer Aided Design*, 1997, pp. 441-446.

[7] J. Darnauer and W. Dai, "A Method for Generating Random Circuits and Its Application to Routability Measurement", *4th ACM/SIGDA International Symposium on FPGAs*, 1996, pp. 66-72.

[8] D. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *IEEE/ACM International Conference on Computer Aided Design*, 1996, pp. 194-200.

[9] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", *Proceedings. IEEE/ACM Design Automation Conference*, 1982, pp. 175-181.

[10] J. Garbers, H.J. Promel and A. Steger, "Finding Clusters in VLSI Circuits", *IEEE/ACM International Conference on Computer Aided Design*, 1990, pp. 520-523.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company: New York, 1979, pp. 223.

[12] M. Hutton, J. P. Grossman, J. Rose and D. Corneil, "Characterization and Parameterized Random Generation of Digital Circuits", *33rd IEEE/ACM Design Automation Conference*, 1996, pp. 94-99.

[13] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "hMetis, A Hypergraph Partitioning Package, Version 1.0", *Manuscript*, December 1997.

[14] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Circuit Partitioning", *34th IEEE/ACM Design Automation Conference*, 1997, pp. 526-529.

[15] Y.-C. Wei and C.-K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning", *IEEE/ACM International Conference on Computer Aided Design*, 1989, pp. 298-301.