Further Improve Circuit Partitioning using GBAW Logic Perturbation Techniques

Chak-Chung Cheung
The Chinese University of HK
Shatin, NT, Hong Kong

Yu-Liang Wu
The Chinese Univesity of HK
Shatin, NT, Hong Kong

David Ihsin Cheng
Ultima Interconnect Technology
Sunnyvale, CA, USA

Abstract

Efficient circuit partitioning is gaining more importance with the increasing size of modern circuits. Conventionally, circuit partitioning is solved by modeling a circuit as a hypergraph for the ease of applying graph algorithms. However, there exist rooms for further improvement on even optimum hypergraph partitioning results, if logic information can be applied for perturbation. In this paper, we present a multi-way partitioning framework which can couple any excellent hypergraph partitioner and a noval logic perturbation based (GBAW) technique for further improvement over very excellent partitioning results. Our approach can integrate with any graph partitioner. We performed experiments on 2-, 3-, 4-, and 5-way partitionings for various circuits of different sizes from MCNC benchmarks. We have chosen the state-of-the-art hMetis-Kway to obtain high quality initial solutions for the experiments. Our experiments showed that this partitioning approach can achieve a further 15% reduction in cut size for 2-way partitioning with an area penalty of only 0.33%. The good results demonstrated the effectiveness of this new partitioning technique.

1. Introduction

Traditionally, circuit partitioning is done by simply modeling the circuit as a graph (or hypergraph). Graph partitioning problems are known to be NP-hard [1]. A comprehensive survey [2] has presented the recent directions of partitioning. Commonly used partitioning algorithms can be categorized into three classes. The first class strictly abides by the modeling graph, with no attempt to change the graph. High quality results have been reported by several algorithms which include iterative improvement based [1, 3], clustering based [4], and spectrum (eigenvector) based [5, 6]. The second class of algorithms may modify the graph through node replications [7, 8]. Improvement is achieved by sacrificing some area due to node replications. These two classes both perform the partitioning task on the graph without considering the logic function of the circuit. The third class [9, 10, 11] couples the graph domain (nodes and their connections) and logic domain (function perform by each node). The tradeoff of improving the partitioning results is the expensive computational cost [10, 11].

Recently, many research works on multi-level partitioning are proposed [12, 13, 14, 15]. The general idea behind multi-level partitioning is to first cluster the whole problem by some useful algorithms to reduce the size, then apply a well-known graph domain partitioner on the coarsened graph to get a good initial solution. The graph is then unclustered and a suitable partitioning refinement algorithm is applied in order to adjust the cut edge between partitions. The quality and the runtime by multi-level partitioning are very encouraging. In particular, Karypis and Kumar [15] proposes a particular called hMETIS-Kway. It first coarsens the hypergraph, then recursively bisects the graph into k-parts, followed by uncoarsening the hypergraph with refinement algorithms. More recent research works [16, 17], in comparison with hMetis-Kway, showed that the solution by hMetis-Kway is of high quality that the cut size cannot be further reduced greatly.

Alternative wiring (rewiring) is the technique of adding single or multiple redundant wires or gates to a circuit such that other wires or gates become redundant and thus removable. This logic domain technique has been widely used for solving many logic level and physical level design problems [9, 18, 19, 20]. Circuit performance can be improved by removing a wire on the critical path and adding its alternative wire elsewhere. Circuit routability can also be improved by substituting an unroutable wire in congested area by a routable alternative wire in some other circuit part. The cut size of a partition can be reduced by replacing the wires crossing the cut line.

Figure 1 illustrates how rewiring can be used to further improve an already optimum partition result obtained by a

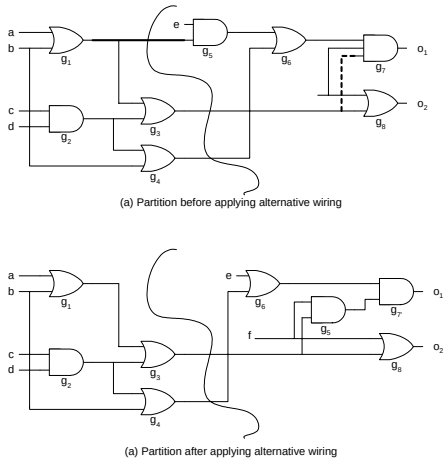

Figure 1. Circuit partitioning by rewiring

typical graph domain partition algorithm. The global optimum partition result in the graph domain, with a cut size of 3, is shown in Figure 1(a). However, if we apply the logic domain rewiring technique to replace a target wire (thick wire) crossing the cut line by its alternative wire (dotted wire), the cut size can be reduced to 2, as shown in Figure 1(b) (without injecting area increase). From this example, we can see that rewiring can be applied to partitioning to further improve upon even optimum solution in the graph domain.

To investigate the possibility of perturbing the circuit without applying any Boolean operations, minimal circuit structures yielding rewiring patterns have been studied [21]. Based on benchmark circuits, we observe that the nearest existing alternative wire is quite close to its target wire. As a result, instead of applying the ATPG-based logic implications repeatedly for a same pattern, the Graph-Based Alternative Wire (GBAW) technique [21] employs a more efficient graph pattern matching operation to locate alternative wires. The basic idea of GBAW is to match the sub-circuit with some "pre-specified" patterns. Rewiring by GBAW can be done without doing any logic implication or redundancy check, hence it runs very fast. Besides considering the alternative wire which is close to the target wire from those small "pre-specified" patterns, distant alternative wires can also be located by propagating the matchings in a cascading way.

To expand the optimization space, we applied the coupling notion between graph and logic domain into our GBAW-Partitioner (GP). In graph domain, we chose the well-known Fiduccia-Mattheyses (FM) partitioning algorithm [3] as the iterative move-based engine for its simplicity. In logic domain, we applied an augmented GBAW, which enhances the ability to locate more 2-local alternative wires, as a greedily guided perturbation engine. In our experiments, near optimum partition results were firstly obtained from the pure graph domain partitioner hMetis-Kway. Then the coupling graph and logic domain optimization by GP engine, was followed. Note that our logic perturbation process can be coupled with any powerful graph domain partitioning tool, and GBAW itself is able to handle patterns with multiple-input gates. We experimented this partition flow for 2-, 3-, 4-, and 5-way partitionings on various MCNC benchmarks ranging from small to fairly large circuits. The results showed that such a graph-logic domain coupled partitioning approach can further cut down the cut size effectively with small CPU overhead. The results also showed that it is a very promising direction for circuit partitioning.

This paper is organized as follows. The background and definition are introduced in Section 2. In Section 3, a brief introduction on Graph-based Alternative Wire technique is described. In Section 4, the details of repartitioning by rewiring is shown. In Section 5, experimental results are presented. Conclusions are drawn in Section 6.

2. Background and Definitions

A combinational circuit can be represented by a DAG where vertices correspond to the primary inputs (PI), primary outputs (PO) and the internal gates of the circuit. PI and PO are nodes which have only outgoing edges and incoming edges respectively. An internal node has at least two incoming edges and one outgoing edge and is associated with a Boolean function. Inverters are not considered as internal nodes, but as polarity of edges during logic domain perturbation. In a Boolean network, the in-degree of node y, denoted by $d^-(y)$, is defined as the number of edges entering y. The out-degree of node y, denoted by $d^+(y)$, is defined as the number of edges leaving y. We also define a node y by a triplet $(op, d^-(y), d^+(y))$, where op is the Boolean operator of y which can be AND, OR, NAND, or NOR.

We use a graph configuration D to map the logic function from a Boolean Network G. For each node n_i in sub-network S in network G, n_i is mapped to a triplet (op, i_1, i_2) in D where op denotes the operator representing the Boolean function of n_i and i_1, i_2 are non-negative integers. All edges inside S are preserved, while the edges outside S are omitted in D. In most cases, i_1 equals $d^-(n_i)$ and i_2 equals $d^+(n_i)$. The element of a triplet $(op, d^-(y), d^+(y))$ can also be don't care. For the first element, don't care means any operator. For the other elements, don't care can be any positive integers. We use a configuration to denote a minimal pattern containing both the target and its alternative wire. The mapping is illustrated in Figure 2. S is a sub-network of G. D_1 and D_2

are two mappable configurations of S. A k-local pattern denotes a minimal sub-graph with the distance between the alternative wire and its target wire being k. The distance between two wires is defined as the difference of maximum path length from any primary input to each of the wires. A wire is defined as a 2-point connection between a pair of source and sink nodes. When a larger circuit is partitioned into two sub-circuits, we define the wires crossing the partitioning cut line as cut wires. We also define a cut net as a hyperedge connecting partitions and the cut cost as the number of partitions that the hyperedge connects.

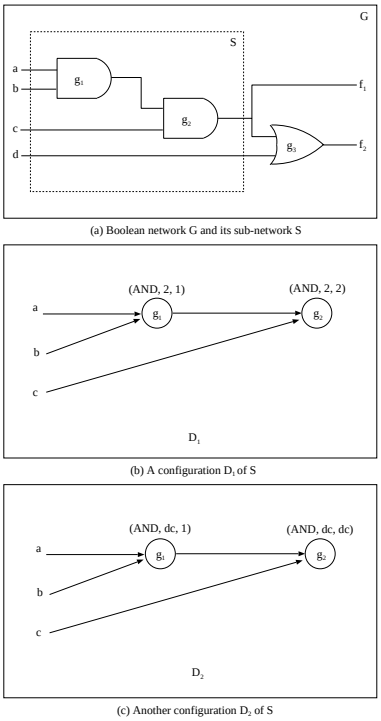

Figure 2. Configuration of a sub-network

3. Graph-Based Alternative Wire Technique (GBAW)

Graph-Based Alternative Wire (GBAW) is a newly proposed and efficient rewiring technique. It models a circuit as a directed acyclic graph (DAG) and searches alternative wires by checking graph matching between local sub-networks and the pre-specified minimal sub-graph configurations. A configuration is a minimal circuit pattern containing alternative wires within a given distance. Experiments showed that the number of all such local minimal sub-graph is limited. Most of the alternative wires are located topologically "near" to their target wires. It has been shown that about 96% of the closest alternative wires are only 2-edge distant from their target wires. When sub-network matches a pattern, GBAW can quickly determine the target wire and the corresponding alternative wires. Obviously, if w_r is an alternative wire of w_t, then w_t is also an alternative wire of w_r. Both w_t and w_r are presented in a pattern. But in a sub-network, only one of them exists. In [21], it has shown that by using GBAW as a random perturbation engine, a competitive logic optimization result is obtained when comparing to RAMBO while the runtime is greatly reduced.

There are 0-local, 1-local and 2-local patterns in GBAW. In this paper, we apply an augmented GBAW, which is a much extended scheme improved from the GBAW shown in [21], to improve the effectiveness of identifying alternative wires of a given circuit for repartitioning. Figure 3 shows some new 2-local patterns used in GBAW, with the target wire and its alternative wire shown as the thick line and dotted line respectively. The position of the target wire and alternative wire can be swapped. GBAW is able to find the alternative wire of the target wire within a limited distance, also it is able to locate distant alternative wire by waveform propagations. This paper applies the GBAW as the perturbation engine in logic domain.

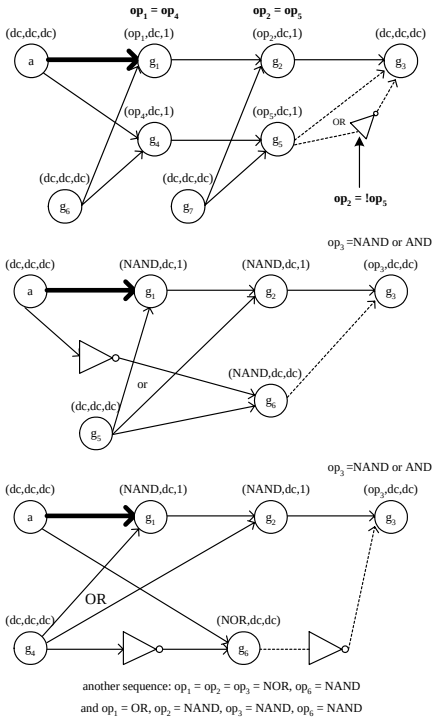

Figure 3. Some new 2-local patterns in GBAW

There are more than 40 different patterns in the implementation of GBAW. GBAW does handle the case of adding one wire and removing another one, the cases of adding one

AND, OR, NAND or NOR gate so as to remove one target wire. It also handles the cases of simultaneously adding two wires and removing two other wires as well.

4. Partitioning using Alternative Wiring

Assume that one pin is used in a partition for a net. The objective of a multi-way partition is essentially to minimize the number of pins required to connect all partitions. Since some of the wires may have alternative wires, if we replace some cut wires by their alternative wires that are not cut wires, cut size can be reduced. The rewiring process may lead to some new circuit graph, and in turn help escaping from local minima led by graph domain partitioning process.

A perturbation refers to the replacement of a target wire by its alternative wires. Figure 4 illustrates the gains regarding various perturbations in a circuit. Thick lines represent the target wires and dotted lines refer to their alternative wires. As shown in the example, we may have positive, zero and negative gains.

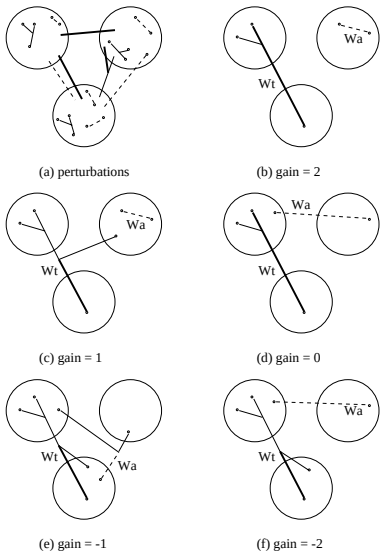

Figure 4. Perturbations and gains

We use the hMetis-Kway partitioning tool to provide a fast and near optimum solution as our initial partition. We select the well-known FM partitioning algorithm [3] as our graph domain partitioner for its simplicity and efficiency. In fact, we can apply any other graph domain partitioner. Then we apply our rewiring technique, GBAW, to perform logic perturbations aiming for further improvements. Figure 5 gives the algorithm of GP.

During the perturbation process GP, only cut wires will be selected as target wires for perturbations. We first ran-

```
Algorithm GP (best_partition, m, k, t){
    search_limit = 0;
    n_perturbations = 0;
    curr_partition = best_partition;
    last_partition = best_partition;
    for i=1 to m {
      while((n_perturbations < k) && (exit == false)){
        search_limit = 0;
        while(search_limit < t){
            search_limit ++;
            randomly select a cut wire W_t;
            use GBAW to find all alternative wires SW_a for W_t;
            if (SW_a == φ){
                search_limit ++;
                continue;
            }else
                break;
        }
        if (SW_a = φ){
            pick alternative wire W_1 with the largest gain;
            replace W_t with W_1 in curr_partition.
            curr_partition = FM(curr_partition);
            n_perturbations = n_perturbation + 1;
            if (cost(curr_partition) < cost(last_partition))
                last_partition = curr_partition;
        }
        for each wire w{
            use GBAW to find all alternative wires SW_a for w;
            do random perturbation on SW_a in curr_partition;
        }
        curr_partition = FM(curr_partition);
        if (cost(curr_partition) < cost(last_partition))
            last_partition = curr_partition;
      }
    }
}
```

Figure 5. Algorithm of GBAW-Partitioner (GP)

domly select a cut wire as the target wire. Then, GBAW is used to find the alternative wire set SWa of the target wire. Finally, among the wire set SWa, the alternative wire with the highest gain is selected for perturbation. When the SWa of the target cut wire is empty, GP may randomly select another cut wire for another trial. The number of iterations is set by m. The number of trials is limited by t times. k is the limit of perturbations. These limits serve to set some bounds for improving performance. Some (hill-climbing) perturbations with negative-gain perturbation are allowed. Therefore we can increase the chance of obtaining better solutions. By integrating GBAW to GP, our partitioner can locate nearly all the alternative wires of multi-input gate circuits.

5. Experimental Results

The algorithm GBAW-Partitioner (GP) was implemented in C and the experiments were conducted on Sun Enterprise E4500 workstation with 8 GB memory in a single-processor configuration for MCNC benchmarks of various sizes . The large benchmark circuits used in ISPD98 [22] are not applicable for our experiments due to the lack of logical domain informations. Since the rewiring engine GBAW [21] is able to locate alternative wires of multiple input gates as well as 2-input gates, thus the circuit simplification SIS [23] done by [9, 24] can be skipped.

In our experiments, we set the tolerance of area imbalance of GP to be $\pm 20\%$ of the average area in each partitioned block. Therefore the maximal ratios are 40%:60%

and 16%:24% in 2-way and 5-way partitioning respectively. In order to explore the graph domain optimization, hMetis-Kway [15] was firstly run for each circuit. As a result, a nearly optimum partition solution was obtained. The next step is to select the best solution applying GP for logic perturbation to further improve the quality of the partitioning with $k = 60$ and $t = 50$. Table 1 to 4 list the experimental results for the 2- to 5-way partitionings respectively. Column "area" lists the area of the sub-circuit in terms of the number of gates. "#lits" lists the total number of literals of the partitioned circuits. From the results, the area penalties for 2- to 5-way are 0.33%, 0.53%, 0.61% and 0.71% respectively. Column "cut cost" lists the total number of cut pins obtained for all partitioned blocks. Column "cut wire" lists the number of cut wires of the partitioning. Column "cpu" lists the cpu time (in seconds). From the results, we can see that applying logic perturbation can further cut down the cut size of the good partitionings produced by purely graph domain based partitioner. The total number of literals is slightly increased because of the area cost of the added gates during perturbation. We obtained 14.48%, 10.18%, 9.08% and 9.24% reduction in cut size for the 2-, 3-, 4- and 5-way partitionings. The last 2 columns showed that the quality and cpu time of GP are both much better than the results obtained by simply running FM for 250 times.

6. Conclusion and Future Work

In this paper, a scheme coupling the graph and logic domain partitioners to explore a larger optimization room of circuit partitioning is proposed. The scheme is shown to be very efficient in terms of CPU expenditure and is also quite capable in bringing further improvements on good partition results produced by the state-of-the-art partitioner hMetis-Kway. Without the integration with RAMBO, the input circuits is no longer limited to 2-input simple gate circuits. We conducted experiments on 29 MCNC benchmark circuits for 2- to 5-way partitionings, and obtained further reductions from 14.48% to 9.24% upon the good results produced by hMetis-Kway. Moreover, the partitions quality and CPU expenditure of GP are both better than simply running FM for 250 times. As GP can be integrated with any newly developed powerful graph partitioner, this partitioning scheme should be very practical and useful for many partition tasks.

References

[1] Y. C. Wei and C. K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10(7), pp. 911–921, July 1991.

[2] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning.," *Integration, the VLSI Journal*, vol. 19(1-2), pp. 1–81, 1995.

[3] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," *Proc. 19th ACM/IEEE Design Automation Conference*, pp. 175–181, 1982.

[4] C. W. Yeh, C. K. Cheng, and T. T. Y. Lin, "A probabilistic multicommodity-flow solution to circuit clustering problems," *Proc. of the Int. Conf. on Computer-Aided Design*, pp. 428–431, 1992.

[5] L. Hagen and A. B. Kahng, "Fast spectral methods for ratio cut partitioning and clustering," *Proc. of the Int. Conf. on Computer-Aided Design*, pp. 10–13, 1991.

[6] J. Y. Zien, M. D. F. Schlag, and P. K. Chan, "Multi-level spectral hypergraph partitioning with arbitrary vertex sizes," *Proc. of the Int. Conf. on Computer-Aided Design*, 1996.

[7] C. Kring and A. R. Newton, "A cell-replicating approach to mincut-based circuit partitioning," *Proc. of the Int. Conf. on Computer-Aided Design*, pp. 2–5, 1991.

[8] H. H. Yang and D. F. Wong, "Optimal min-area min-cut replication in partitioned circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17(11), pp. 1175–1183, November 1998.

[9] D. I. Cheng, C. C. Lin, and M. Marek-Sadowska, "Circuit partitioning with logic perturbation," *Proc. of the Int. Conf. on Computer-Aided Design*, pp. 650–655, 1995.

[10] M. Beardslee, B. Lin, and A. Sangiovanni-Vincentelli, "Communication based logic partitioning," *EURO-DAC '92. European*, pp. 32–37, 1992.

[11] D. I. Cheng, S. C. Chang, and M. Marek-Sadowska, "Parititioning combinational circuits in graph and logic domains," *Proc. SASIMI-93*, pp. 404–412, 1993.

[12] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs.," *Sandia Nat. Labs., Albuquerque, NM, Tech. Rep. SAND93-1301*, 1993.

[13] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes.," *Proc. of the 1995 Intl. Symp. on Physical Design*, pp. 113–122, 1995.

[14] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *Proc. 34th ACM/IEEE Design Automation Conference*, 1997.

[15] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning.," *Proc. 36th ACM/IEEE Design Automation Conference*, 1999.

[16] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph bipartitioning," *Proc. Asia South Pacific Design Automation Conf. 2000*, 2000.

[17] J. Cong and S. K. Lim, "Edge separability based circuit clustering with application to circuit partitioning," *Proc. Asia South Pacific Design Automation Conf. 2000*, 2000.

[18] K.-T. Cheng and L. A. Entrena, "Multi-level logic optimization by redundancy addition and removal," *Proc. EDAC-93*, pp. 373–377, Feb 1993.

[19] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng, "Perturb and simplify: Multilevel boolean network optimizer," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15(12), pp. 1494–1504, Dec 1996.

[20] S.-C. Chang, K.-T. Cheng, N.-S. Woo, and M.-S. M., "Postlayout logic restructuring using alternative wires," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 587–596, June 1997.

[21] Y.-L. Wu, W. Long, and H. Fan, "A fast graph-based alternative wiring scheme for boolean networks," *Proc. IEEE/ACM International VLSI Design*, 2000.

[22] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "The ISPD98 circuit benchmark suite," *Proc. Int. Symp. on Physical Design*, pp. 80–85, 1997.

Circuit	hMetis-Kway					GP					250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	61:71	235	30	31	0.33	73:63	239	28	39	34	28	38
9sym-hdl	67:74	232	16	8	0.36	64:77	232	10	8	14	10	42
C1355	331:269	1055	44	35	0.65	338:266	1059	36	35	74	46	266
C1908	260:256	883	82	55	0.64	271:251	889	62	55	61	62	238
C2670	516:527	1444	42	23	0.78	517:531	1449	34	23	112	126	427
C3540	615:648	2267	132	93	1.77	614:663	2280	112	100	357	180	692
C432	119:119	392	44	27	0.46	118:130	402	36	27	33	28	78
C499	231:272	854	46	47	0.46	232:272	855	36	47	59	54	214
C5315	918:1044	3282	104	71	1.98	919:1047	3286	100	69	493	234	1232
C6288	1297:1559	5195	80	270	2.33	1309:1561	5209	78	189	960	430	1708
C7552	1281:1141	4105	18	65	1.93	1286:1142	4111	18	64	727	228	1731
C880	261:222	780	54	30	0.57	260:234	791	38	30	58	38	200
alu2	190:232	777	84	93	0.81	190:236	781	80	95	115	88	150
alu4	428:357	1470	140	106	1.16	438:360	1481	120	104	223	160	322
apex6	435:473	1417	18	11	0.79	436:473	1418	16	11	109	36	370
b9_n2	87:70	208	16	13	0.32	64:93	208	12	10	16	10	41
comp	93:91	270	6	4	0.32	93:90	269	6	3	35	6	51
des	1727:2112	6655	236	221	3.99	1565:2282	6663	146	331	752	332	2338
duke2	175:211	676	80	61	0.7	162:233	684	74	72	98	82	128
f51m	72:65	244	28	32	0.37	77:65	247	26	33	35	26	41
misex3	293:245	990	80	69	0.88	301:250	1003	76	70	149	76	198
my_adder	106:106	339	4	2	0.28	107:105	339	2	2	22	2	72
pcler8	58:72	174	8	14	0.27	58:72	174	8	14	22	8	31
rot	441:383	1251	54	38	0.80	442:384	1253	46	38	95	66	322
sao2-hdl	136:114	439	26	16	0.47	128:123	440	16	16	31	16	89
term1	124:148	439	28	14	0.53	130:148	445	24	14	62	28	88
ttt2	120:107	376	10	11	0.39	102:125	376	8	11	27	8	72
x3	471:384	1334	22	16	0.84	461:396	1336	20	16	4	40	321
too_large	2161:1913	7723	318	563	5.58	2162:1917	7728	308	544	699	1210	1995
Total		45506	1850				45656	1576		5476	3658	13495
Average							+0.33%	-14.48%				

Table 1. Comparison of 2-way partitioning between hMetis-Kway & FM & GP

Circuit	hMetis-Kway					GP					250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	37:43:52	235	53	51	0.67	40:47:50	240	49	50	36	45	50
9sym-hdl	43:44:54	232	32	19	0.67	43:44:55	233	22	19	15	20	49
C1355	217:196:187	1055	84	95	1.19	217:204:189	1065	80	93	157	143	269
C1908	148:186:182	883	115	82	1.10	146:191:185	889	98	98	127	134	226
C2670	400:329:314	1444	85	59	1.34	402:337:312	1452	71	59	119	247	411
C3540	376:464:423	2267	175	172	2.88	361:506:413	2284	165	224	357	413	664
C432	67:92:79	392	56	46	0.67	72:95:83	404	54	70	67	50	91
C499	193:169:141	854	82	64	0.84	195:173:136	854	64	64	62	108	188
C5315	556:692:714	3282	117	114	3.31	521:709:739	3289	100	122	510	564	1125
C6288	810:1086:960	5195	148	429	3.98	827:1061:978	5198	165	555	704	712	1697
C7552	745:923:754	4105	94	129	3.42	750:941:739	4113	86	136	747	490	1271
C880	139:178:166	780	80	58	1.06	145:182:163	787	61	58	60	103	202
alu2	124:133:165	777	140	144	1.23	134:143:167	795	127	159	122	152	178
alu4	230:307:248	1470	214	203	2.06	218:314:268	1489	194	211	231	283	361
apex6	257:302:349	1417	75	47	1.59	243:321:358	1431	63	70	223	113	447
b9_n2	45:53:59	208	21	17	0.51	44:54:60	209	21	17	32	25	52
comp	57:62:65	270	16	8	0.59	57:62:66	271	14	7	37	20	56
des	1107:1512:1220	6655	322	342	7.06	1089:1535:1216	6656	236	628	776	657	2761
duke2	113:147:126	676	128	107	1.18	106:143:154	693	116	112	103	121	162
f51m	42:53:42	244	58	54	0.68	46:54:44	252	56	62	37	49	51
misex3	153:209:176	990	128	114	1.50	154:208:192	1006	113	162	157	139	252
my_adder	80:65:67	339	8	4	0.60	79:65:68	339	4	4	23	10	72
pcler8	41:49:40	174	17	20	0.47	42:49:39	174	15	20	25	15	37
rot	245:268:311	1251	85	68	1.44	233:272:330	1262	77	77	195	120	368
sao2-hdl	95:83:72	439	62	46	0.86	95:94:68	446	56	57	65	56	112
term1	77:88:107	439	54	38	0.88	78:91:108	443	44	38	32	54	111
ttt2	64:89:74	376	33	34	0.78	66:90:75	380	33	34	55	37	86
x3	271:262:322	1334	78	65	1.38	276:259:340	1354	67	65	106	130	419
too_large	1562:1198:1314	7723	779	976	10.46	1606:1201:1284	7740	747	1096	734	1761	2602
Total		45506	3339				45748	2999		5914	6761	14370
Average							+0.53%	-10.18%				

Table 2. Comparison of 3-way partitioning between hMetis-Kway & FM & GP

[23] E. M. Sentovich, K. J. Singh, L. Lavagno, and et. al., "SIS: A system for sequential circuit synthesis," *ERL Memorandum No. UCB/ERL*, vol. M92/41, 1992.

[24] Y. L. Wu, X. L. Yuan, and D. I. Cheng, "Circuit partitioning with coupled logic restructuring techniques," *Proc. Asia South Pacific Design Automation Conf.*, 2000.

Circuit	hMetis-Kway					GP					250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	34:27:31:40	235	66	63	0.68	38:29:29:39	239	60	78	37	58	52
9sym-hdl	36:38:31:36	232	48	26	0.80	37:40:32:32	232	29	26	16	26	52
C1355	174:158:130:138	1055	110	90	1.59	164:164:125:165	1073	96	108	166	185	305
C1908	122:125:126:143	883	137	114	1.53	126:128:121:149	890	114	126	67	184	259
C2670	262:257:262:262	1444	125	82	1.75	264:251:274:262	1452	91	114	122	274	481
C3540	324:285:356:298	2267	245	211	3.36	337:280:378:281	2278	230	286	385	458	791
C432	65:53:63:57	392	75	62	0.95	72:55:70:56	407	67	73	71	66	107
C499	122:148:116:117	854	92	77	1.26	123:150:112:120	856	81	81	131	145	218
C5315	532:511:460:459	3282	201	152	3.91	537:510:461:461	3289	192	161	523	633	1325
C6288	627:670:856:703	5195	186	582	4.52	627:678:842:716	5200	210	585	482	848	2051
C7552	537:605:647:633	4105	66	153	4.37	511:634:648:638	4114	44	153	388	732	1551
C880	108:113:136:126	780	82	58	1.33	109:123:138:123	790	69	121	124	107	230
alu2	101:128:96:97	777	178	171	1.51	99:126:103:111	794	171	193	128	190	212
alu4	162:197:204:222	1470	285	240	2.35	157:191:224:236	1493	257	308	123	352	452
apex6	233:200:261:214	1417	75	45	1.91	231:209:276:208	1433	65	77	233	139	555
b9_n2	41:48:35:33	208	36	23	0.76	46:43:36:35	209	32	23	17	31	58
comp	48:43:47:46	270	14	8	0.91	47:44:47:46	270	12	8	20	28	62
des	938:1160:947:794	6655	359	515	7.51	889:1144:953:864	6665	295	713	774	716	3322
duke2	116:95:96:79	676	159	127	1.33	113:82:115:89	698	147	137	112	158	183
f51m	37:35:36:29	244	67	67	0.75	41:39:36:28	251	65	68	40	63	55
misex3	157:138:132:111	990	188	177	1.70	160:149:143:107	1006	169	192	166	180	295
my_adder	52:54:52:54	339	12	6	0.81	52:53:52:55	339	6	6	24	23	81
pcler8	30:28:33:39	174	24	25	0.57	31:28:32:39	174	22	25	26	20	40
rot	193:191:208:232	1251	98	75	1.86	187:193:210:240	1256	93	91	205	166	457
sao2-hdl	75:60:63:52	439	93	60	1.08	69:64:73:51	446	76	60	35	74	117
term1	67:56:67:82	439	59	36	1.14	77:57:69:85	454	55	37	68	68	118
ttt2	54:66:59:48	376	51	44	0.85	55:69:60:49	382	46	46	59	48	94
x3	191:193:236:235	1334	78	52	1.86	190:196:249:244	1358	71	111	225	164	504
too_large	862:1059:964:1189	7723	1041	1345	11.58	823:1082:961:1221	7736	999	1685	756	2206	3281
Total		45506	4250				45784	3864		5523	8342	17308
Average							+0.61%	-9.08%				

Table 3. Comparison of 4-way partitioning between hMetis-Kway & FM & GP

Circuit	hMetis-Kway					GP					250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	26:21:23:31:31	235	78	71	0.89	23:22:32:29:31	237	73	77	37	69	60
9sym-hdl	32:30:24:24:31	232	57	31	0.96	32:27:28:25:30	233	33	31	16	35	55
C1355	142:113:100:130:115	1055	122	97	1.95	142:115:95:144:123	1074	109	110	175	222	369
C1908	91:111:93:117:104	883	153	111	1.70	84:118:84:123:112	888	119	131	72	206	313
C2670	221:221:174:207:220	1444	133	95	2.26	186:229:195:217:229	1457	106	135	129	314	578
C3540	214:258:223:307:261	2267	312	268	4.07	224:239:237:301:284	2288	265	341	206	551	990
C432	42:46:43:59:48	392	80	74	1.25	47:48:41:57:57	404	77	88	75	73	126
C499	107:104:85:103:104	854	107	95	1.45	111:101:81:108:105	857	93	102	138	173	245
C5315	371:315:370:497:409	3282	244	193	4.92	372:315:383:471:428	3289	232	268	558	706	1650
C6288	517:496:525:651:667	5195	222	601	5.21	508:498:532:659:669	5204	241	601	820	876	2340
C7552	568:504:396:533:421	4105	152	169	4.84	554:528:388:504:447	4095	135	220	787	836	1901
C880	104:95:82:111:91	780	104	84	1.58	107:91:86:107:100	787	94	157	131	155	293
alu2	71:78:79:95:99	777	209	198	1.92	67:93:88:89:99	791	192	228	130	227	245
alu4	150:127:143:183:182	1470	310	291	3.05	147:130:171:181:173	1487	279	375	129	399	554
apex6	205:205:148:157:193	1417	132	91	2.39	206:206:199:145:171	1436	125	112	249	165	656
b9_n2	25:30:28:39:35	208	36	27	0.84	24:29:33:39:39	214	36	32	36	39	65
comp	39:42:30:41:32	270	24	15	0.97	30:42:31:42:38	269	20	15	21	36	73
des	682:711:683:919:844	6655	484	382	10.03	667:699:667:954:889	6692	386	564	785	689	4246
duke2	73:70:71:94:78	676	175	148	1.69	88:77:65:89:89	696	164	168	115	183	217
f51m	27:21:24:35:30	244	81	78	0.93	32:30:21:27:32	249	77	86	39	76	59
misex3	86:103:111:133:105	990	216	215	2.23	95:96:127:107:129	1006	203	276	173	240	364
my_adder	52:41:41:39:39	339	16	8	1.12	51:41:42:39:39	339	10	8	26	30	87
pcler8	26:22:24:28:30	174	29	29	0.68	27:25:25:24:32	177	26	30	28	27	48
rot	145:147:150:209:173	1251	127	97	2.20	160:151:142:197:190	1266	121	123	217	184	543
sao2-hdl	63:51:45:43:48	439	109	74	1.46	61:60:46:44:48	448	97	93	66	93	145
term1	66:55:48:56:47	439	71	44	1.34	66:58:45:65:55	453	66	59	71	79	146
ttt2	46:55:35:45:46	376	59	58	1.15	47:53:36:51:45	383	56	72	62	65	112
x3	193:190:136:152:184	1334	122	91	2.23	193:195:170:139:199	1375	112	171	243	173	605
too_large	1003:851:661:782:777	7723	1221	1477	14.62	975:879:723:765:743	7734	1159	1806	802	2503	4478
Total		45506	5185				45828	4706		6309	9324	21563
Average							+0.71%	-9.24%				

Table 4. Comparison of 5-way partitioning between hMetis-Kway & FM & GP