Further Improve Circuit Partitioning Using GBAW Logic Perturbation Techniques

Yu-Liang Wu, *Member, IEEE*, Chak-Chung Cheung, David Ihsin Cheng, and Hongbing Fan

***Abstract*—Efficient circuit partitioning is becoming more and more important as the size of modern circuits keeps increasing. Conventionally, circuit partitioning is solved without altering the circuit by modeling the circuit as a hypergraph for the ease of applying graph algorithms. However, there is room for further improvement on even optimal hypergraph partitioning results, if logic information can be applied for circuit perturbation. Such logic transformation based partitioning techniques are relatively less addressed. In this paper, we present a powerful multiway partitioning technique which applies efficient logic rewiring techniques for further improvement over already superior hypergraph partitioning results. The approach can integrate with any graph partitioner. We perform experiments on two-, three-, and four-way partitionings for MCNC benchmark circuits whose physical and logical information are both available. Our experimental results show that this partitioning approach is very powerful. For example, it can achieve a further 12.3% reduction in cut size upon already excellent pure graph partitioner (hMetis) results on two-way partitioning with an area penalty of only 0.34%. The outperforming results demonstrate the usefulness of this new partitioning technique.**

***Index Terms*—Alternative wiring, partitioning.**

I. Introduction

The objective of circuit partitioning is to divide the circuit into subcircuits so that the size of each component is reasonable and the number of interconnect between the components is minimized. As design scale expands, partitioning becomes increasingly important to circuit design automation.

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 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 that include iterative improvement based [1], [3], [4], clustering based [5], and spectrum (eigenvector) based [6], [7]. The second class of algorithms may modify the graph through node replications [8]–[11]. Improvement is achieved by sacrificing areas 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 [12]–[15] 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 high computational cost [13], [14] and may only be applied to field-programmable gate array (FPGA) circuits [15].

Recently, many research studies on multilevel partitioning have been proposed [16]–[22]. The general idea behind multilevel partitioning is to first cluster the whole problem by efficient algorithms so as to reduce the size, then apply a well-known graph-domain partitioner on the coarsened graph to get a high quality 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 multilevel partitioning are very encouraging. In particular, Karypis and Kumar [22] propose a partitioner 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 [23]–[26], in comparison with hMetis-Kway, have shown that the solution by hMetis-Kway is such a 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 so 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 [12], [27]–[31]. 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 the congested area by a routable alternative wire in other circuit parts. The cut size of a partition can be reduced by replacing the wires crossing the cut line.

Fig. 1 illustrates how rewiring can be used to further improve an already optimal partitioning obtained by a typical graph-domain partition algorithm. The global optimal partitioning in the graph domain, with a cut size of 3, is shown in Fig. 1(a). However, if we apply the logic-domain rewiring technique to replace a target wire (thick line) crossing the cut line by its alternative wire (dotted line), the cut size can be further reduced to 2 as shown in Fig. 1(b) without injecting area increase. From this example, we can see that rewiring can be applied to partitioning to further improve upon even the optimal solution in the graph-domain. Binding the logic-domain rewiring technique with an efficient graph-domain partitioning tool enables a larger flexibility for obtaining better results. The rewiring technique can be used

Manuscript received April 6, 2001; revised June 8, 2002. This work was supported in part by the Hong Kong Research Grants Council under Grant CUHK4236/01E and in part by Direct Grant CUHK2050244.

Y. L. Wu and C.-C. Cheung are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: ylw@cse.cuhk.edu.hk; cccheung@cse.cuhk.edu.hk).

D. I. Cheng is with Faraday Technology, Milpitas, CA 95035 USA (e-mail: dcheng@faraday-usa.com).

H. Fan is with the Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: hfan@csr.csc.uvic.ca).

Digital Object Identifier 10.1109/TVLSI.2003.812369

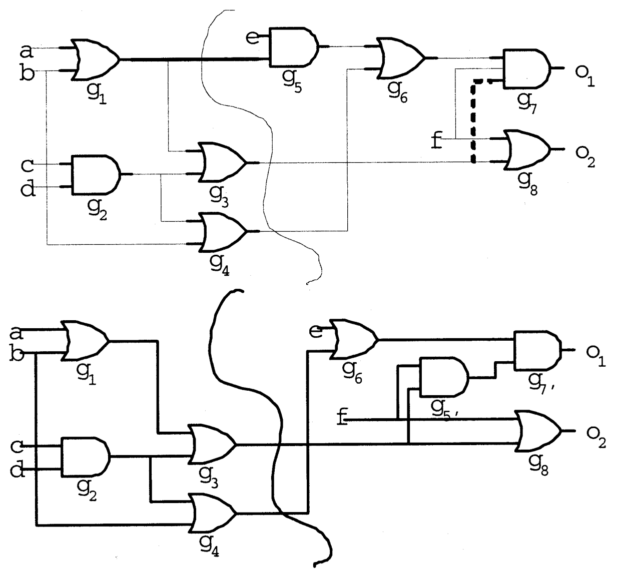


Fig. 1. Circuit partitioning by rewiring (cut net size improved from 3 to 2).

either as a greedily guided optimization tool, or as a random perturbation tool that allows for hill climbing on cut cost functions so as to release the cost out of local minima.

The well-known ATPG-based rewiring technique, redundancy addition and removal for multilevel Boolean optimization (RAMBO) [12], [14], [27], [28], [32]–[36], is a very powerful technique for identifying alternative wires of a specified target wire for a given circuit. This technique has been used for logic perturbations and has been integrated with a graph-domain partitioning algorithm to produce improved partitioning [12]. However, as it always selects a nonnegative-gain wire in replacing a target wire and it only considers simple cases of adding and replacing one single wire, it is easily trapped in local minima. Besides, the ATPG-based rewiring technique, though powerful, tends to spend much running time due to the time-consuming Boolean implication operations. Moreover, the benchmark circuits used in the Cheng's work [12] have been preprocessed by the SIS package [37] and have mapped into two-input gates.

To investigate the possibility of perturbing the circuit without applying any Boolean operations, minimal circuit structures yielding rewiring patterns have been studied [38], [39]. Based on benchmark circuits, we observe that the nearest existing alternative wire is quite close to its target wire. Therefore, these minimal patterns tend to be small and repeatedly appear in a circuit. As a result, instead of applying the ATPG-based logic implications repeatedly to a same pattern, the graph-based alternative wire (GBAW) technique [38], [39] employs a more efficient graph pattern matching operation to locate alternative wires. The basic idea of GBAW is to match the subcircuit with "prespecified" patterns. Rewiring by GBAW can be done without applying any logic implication or redundancy check, hence it runs very fast. Besides considering the alternative wire that is close to the target wire from those small "prespecified" patterns, distant alternative wires can also be located by propagating the matchings in a cascading way. By coupling RAMBO and GBAW as the perturbation engine, Wu [40] proposed the bipartitioning tool RAMBO-GBAW partitioner (RG) that also handles the two-input gates and has a larger flexibility for perturbation.

In our approach, excellent partitionings were firstly obtained using the pure graph-domain partitioner hMetis-Kway to serve as initial partitions. Then to expand the optimization space, we applied an iterative optimization process coupling both graph and logic domain partitioners. In graph domain, we chose the Fiduccia–Mattheyses (FM) partitioning algorithm [3] for its simplicity. In logic domain, we applied either the RAMBO [12], GBAW engine [38], or augmented GBAW [39], as a greedily guided perturbation engine.

Please note that the graph partitioner used in this approach is not limited to any particular one, i.e., the logic perturbation process can be coupled with any later developed more powerful graph-domain partitioning tool. We experimented this partition flow for two-, three- and four-way partitionings on various MCNC benchmarks ranging from small to fairly large circuits whose physical and logical information are both available. The results show that such a graph-logic domain coupled partitioning approach can further cut down the cut size effectively with small CPU and area overhead. Our results show that our proposed approach can further reduce the cut cost over excellent graph partitioner results by 12.3%, 11.1%, and 11.4% for two- to four-way partitionings with quite low area overhead of 0.34%, 0.49%, and 0.57% only, respectively. We observed that the experimental results amongst the three different logic perturbation engines are all significant and comparable, while the GBAW rewiring engine is the fastest one. The results seem to suggest that for partitioning objective, a simple rewiring scheme would be effective enough to produce the near best results. The encouraging results also suggest a quite promising approach for doing circuit partitioning.

This paper is organized as follows. The preliminaries and notion of alternative wiring are introduced in Section II. In Section III, a brief introduction to GBAW technique is given. In Section IV, the details of repartitioning by rewiring are shown. In Section V, experimental results are presented. Conclusions are drawn in Section VI.

II. Preliminaries

A combinational circuit can be represented by a directed acyclic graph (DAG) where vertices correspond to the primary inputs (PI), primary outputs (PO) and the internal gates of the circuit. PI and PO are nodes that 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. A Boolean network G is used to represent a system of Boolean functions with specified variables as PI and functions as PO. The functionality of a Boolean network is specified by its primary output function set. Two Boolean networks are equivalent if they have the same functionality.

A wire is defined as a two-point connection between a pair of source and sink nodes. When a larger circuit is partitioned into two subcircuits, we define the wires crossing the partitioning

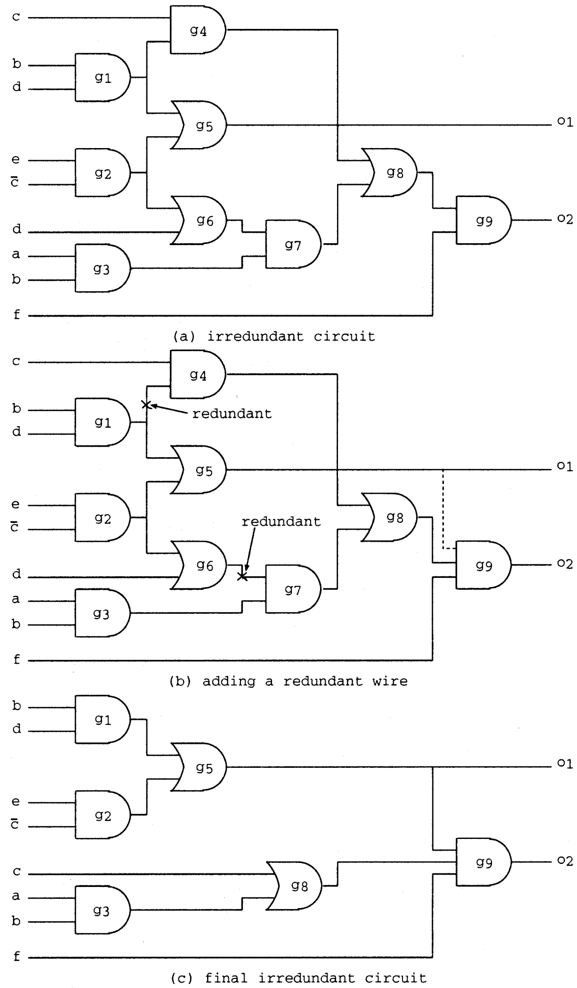


Fig. 2. Example of alternative wiring.

cut line as cut wires. We also define a cut net as a hyperedge connecting partitions and the cut pins as the total number of pins required for all partitioned blocks.

A. Rewiring

If we consider the circuit from Perturb and Simplify [28] as shown in Fig. 2(a), this circuit is irredundant because none of the wires in the circuit is removable. If we add a connection from the output of gate g_5 to the input of gate g_9 [shown as a dotted line in Fig. 2(b)], the functionality of the circuit does not change. In other words, the added connection is redundant. However, the addition of this connection causes two originally irredundant wires to become redundant as shown in Fig. 2(b). After removing these two wires and associated gates that either become floating (g_6) or have a single fanin (g_4 or g_7), the circuit can be greatly minimized as shown in Fig. 2(c). We can apply this rewiring techique to solve the logic synthesis and physical design problems.

III. GBAW Technique

A wire is replaceable if it has at least one alternative wire. We use a graph configuration D to represent a subnetwork function S in a Boolean network G. 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 that can be any associative operator like AND, OR, NAND, or NOR.

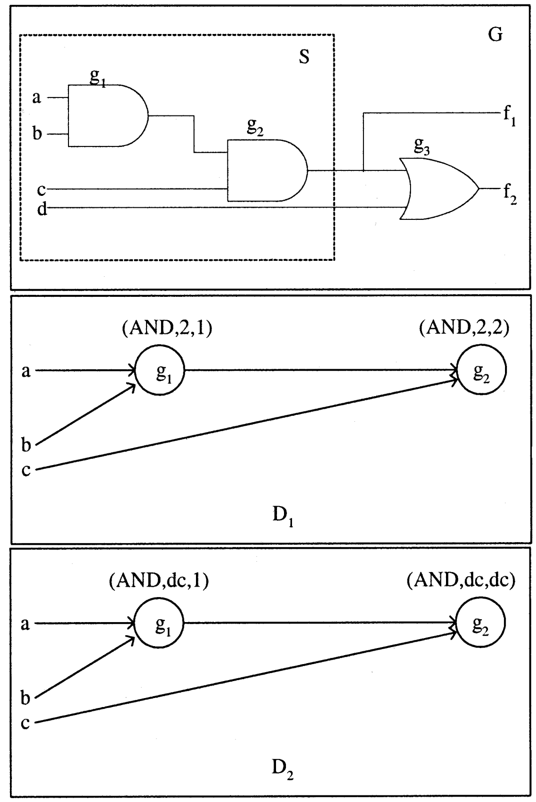


Fig. 3. Configuration of a subnetwork.

For each node n_i in subnetwork 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 nonnegative 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" (dc). For the first element, "dc" means any operator. For the other elements, "dc" can be any positive integers. We use a configuration to denote a minimal pattern containing both the target (the wire to be replaced) and its alternative (the wire to be added) wire. A minimal pattern implies that all the edges or nodes associated with the pattern cannot be removed.

The mapping is illustrated in Fig. 3. S is a subnetwork of G. D_1 and D_2 are two mappable configurations of S. They are both called subgraph as they are mapped from a subnetwork S. A subgraph is minimal if all of its nodes and edges are all essential (unremovable) to the graph. A k-local pattern denotes a minimal subgraph 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. In Fig. 4(a), gate $y3$ can be reached from $y1$ and $I4$ and the maximum path length is defined as 2.

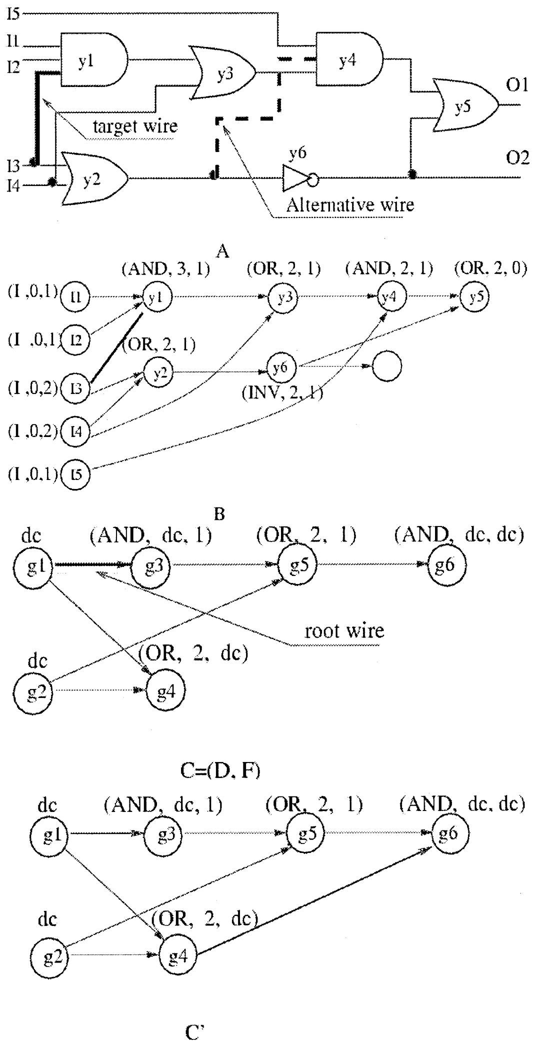


Fig. 4. A combinational network, a labeled Boolean network and two configurations.

Fig. 4 shows a combinational network (A) and its corresponding Boolean network (B) by using the configuration notation. We can further simplify this Boolean network into two configurations containing the alternative wire (C′) and without containing redundant wire (C). Finally, a pattern is constructed.

GBAW is a newly proposed and efficient rewiring technique. It models a circuit as a DAG and searches alternative wires by checking graph matchings between local subnetworks and the prespecified minimal subgraph configurations. A configuration is a minimal circuit pattern containing alternative wires within a given distance. Experiments show that the number of all such local minimal subgraphs 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 two-edge distant from their target wires. When a subnetwork 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 pescribed in a pattern. But in a subnetwork, only one of them exists.

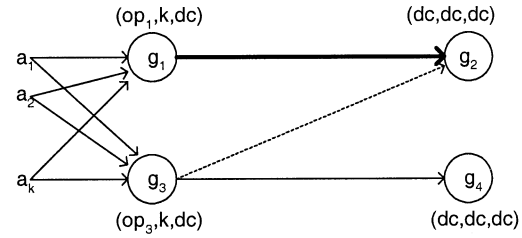


Fig. 5. 0-local pattern in GBAW.

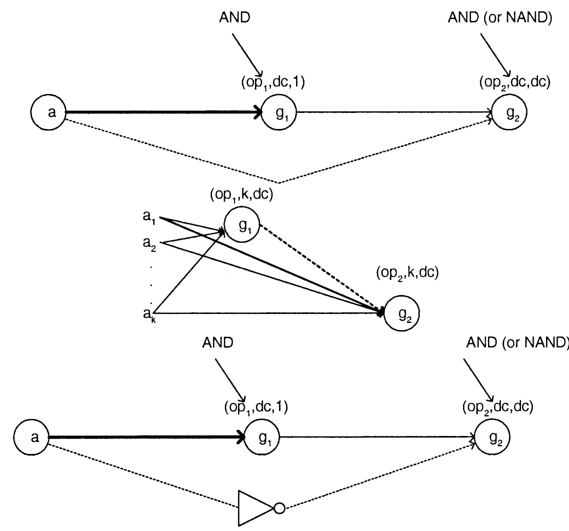


Fig. 6. 1-local patterns in GBAW.

A. Alternative Wiring Patterns

There are 0-local, 1-local, and 2-local patterns in GBAW and they are discussed briefly in the following subsections. According to our observations, the nearest alternative wire of a target wire is close to the target wire in most practical cases. In this paper, we apply an augmented GBAW scheme [39], which is a much extended scheme improved from GBAW version shown in Wu [38], to improve the effectiveness of identifying alternative wires of a given circuit for repartitioning. GBAW is able to find the alternative wire of the target wire within a limited distance; also it is able to locate a distant alternative wire by waveform propagations. This paper majorly applies GBAW as the perturbation engine in logic domain.

1) 0-Local Pattern: A 0-local pattern is a node substitution pattern such that two nodes can replace each other if they have the same logic function. As shown in Fig. 5, the target wire is $g_1 \rightarrow g_2$ and its alternative wire is $g_3 \rightarrow g_2$.

2) 1-Local Patterns: There are three basic types in 1-local patterns as shown in Fig. 6. Now, if we consider case 1-1, $a \rightarrow g_1$ can be replaced by $a \rightarrow g_2$ if $op_1 =$ AND and $op_2 =$ AND or NAND. Case 1-2 can be proved by the following. Let $op_1 =$ NOR and $op_2 =$ AND, then $g_1 = (a + x)' = a'^* x'$, where $x = b_1 + b_2 + \cdots + b_k$ and $b_1, b_2, \ldots, b_k$ are the other inputs of g_1, or $x = 0$ if g_1 has no other inputs. Then, $g_2 = (g_1{}^* y)$, where $y = c_1{}^* c_2{}^* \ldots {}^* c_l$ and $c_1, c_2, \ldots, c_l$ are the other inputs of g_2, or $y = 1$ if g_2 has no other inputs. Then g_2 can be regrouped as $g_2 = (a'^* x'^* y) = ((x')^* a'^* y) = (g_{11}{}^* a'^* y)$, where $g_{11} = x'$

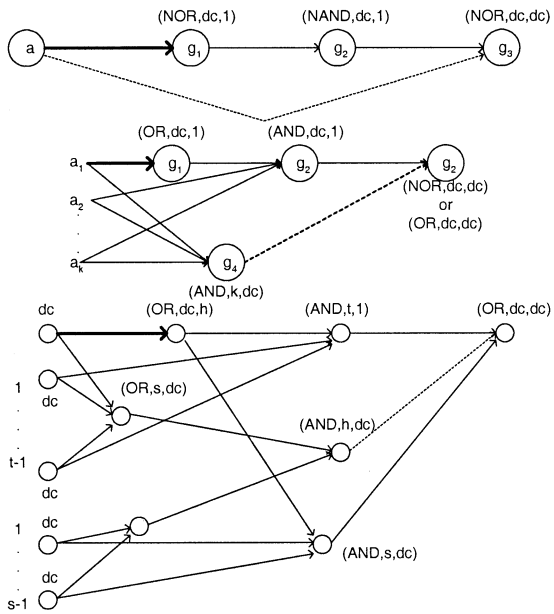


Fig. 7. 2-local patterns in GBAW.

is the logic function of g_1 after $a \rightarrow g_1$ is removed. Therefore, $a \rightarrow g_2$ is an alternative wire of $a \rightarrow g_1$.

3) 2-Local Patterns: In every 2-local patterns, the alternative wire is 2-edge far away from the target wire. In Fig. 7, three 2-local patterns are shown.

B. Functional Analyses

Fig. 8 shows four new 2-local patterns used in the augmented 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. There are more than 40 different patterns in the implementation of augmented GBAW. GBAW does handle adding one wire and removing another one, adding one AND, OR, NAND, or NOR gate so as to remove one target wire. The patterns of the original GBAW are constructed based on basic minimal pattern configurations. In augmented GBAW, more patterns extracted from benchmark circuits using RAMBO tool are included.

IV. Partitioning Using Alternative Wiring

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

A rewiring perturbation refers to the replacement of a target wire by its alternative wires. Fig. 9 illustrates the gains regarding various perturbations in a circuit. In the figure, 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.

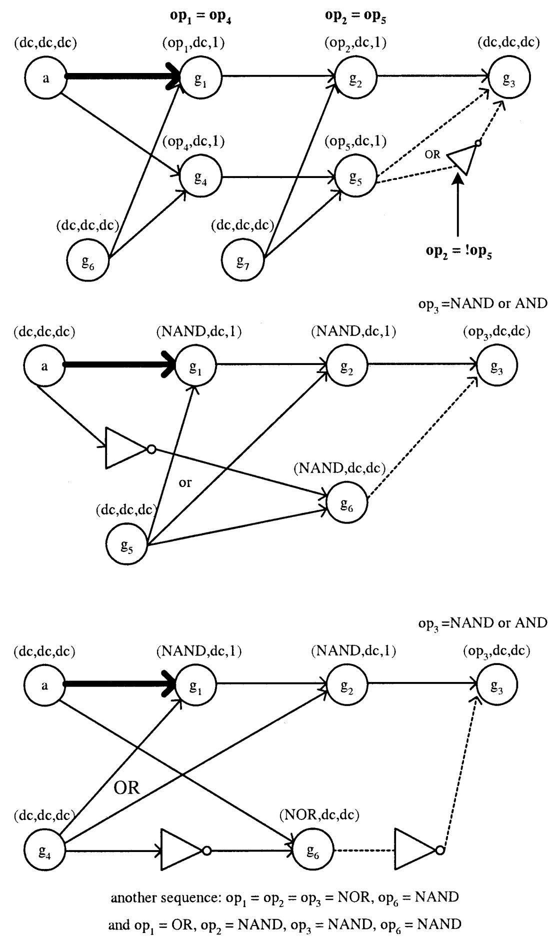


Fig. 8. Partial new 2-local patterns in GBAW.

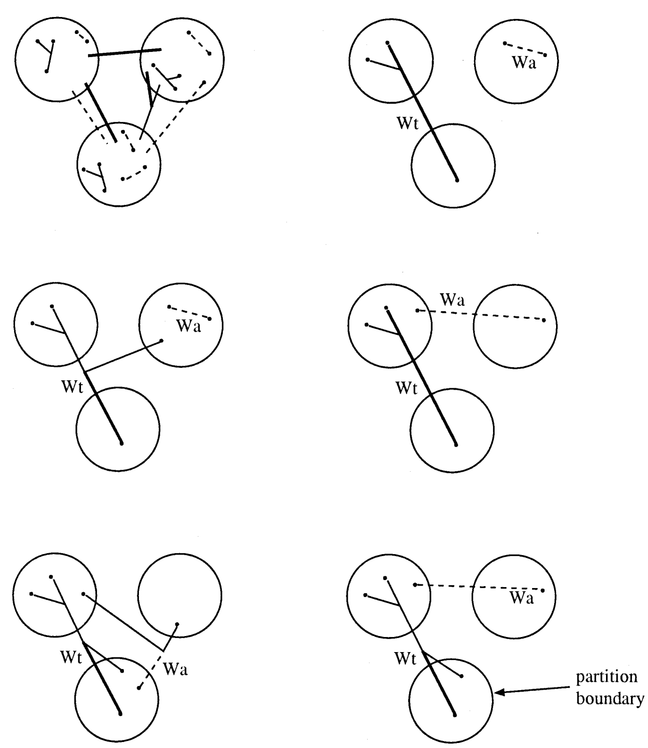


Fig. 9. Perturbations and cut pin gains for three-way partitioning.

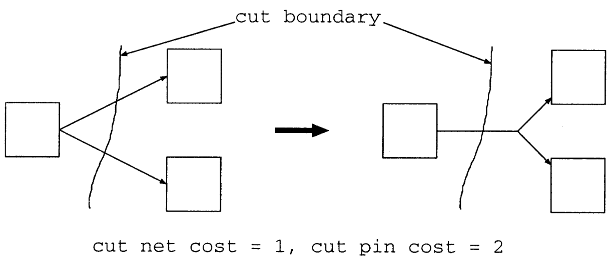


Fig. 10. Cut net cost versus cut pin cost in two-way partitioning.

```
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;
   }
  }
 }
}
```

Fig. 11. Algorithm of GBAW-partitioner (GP).

As shown in Fig. 10, it shows the cut different between cut net cost and cut pin cost of a simple bipartition example. It is important to know that the cut net cost is only 1 but not 2. It is inaccurate to measure the cut net cost in k-way partitioning, therefore, in our experiments we used the cut pin cost as the cut size in order to compare the quality of different partitioning approaches.

We use the hMetis-Kway partitioning tool to provide a fast and near optimal solution that serves as our initial partition. We adopt the well-known FM partitioning algorithm [3] as our graph-domain partitioner in our iterative graph-logic perturbation process for its simplicity and efficiency. In fact, we can apply any other graph domain partitioner for this purpose. Then we apply our rewiring technique (RAMBO for RP while GBAW for GP), to perform iterative logic perturbations aiming for further improvements. The perturbation operations include:

- substituting a wire with its alternative wire,
- adding a gate and removing several wires,
- adding one wire to remove other wires,
- adding two gates to remove other wires and so on.

Fig. 11 gives the algorithm of GP.

During the perturbation process GP, only cut wires will be selected as target wires for perturbations. We first randomly 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 bounds for unnecessary runs when the total number of alternative wires of all cut wires is zero or very small. RP is similar to GP except that RAMBO is used for rewiring. The main difference between algorithm GP and the algorithm in [12] lies in the condition of perturbations. In [12], a perturbation is performed only when the alternative wire of the selected cut wire has a nonnegative gain. However, in our experiments, hill-climbing perturbations are allowed, therefore the chance of obtaining better solutions can increase. In this paper, a negative-gain perturbation is also allowed to help escaping local minimums. On the other hand, the main difference between GP and the algorithm in Wu [40] is the perturbation engine used. In Wu [40], a coupling scheme of RAMBO and GBAW (RG) is used.

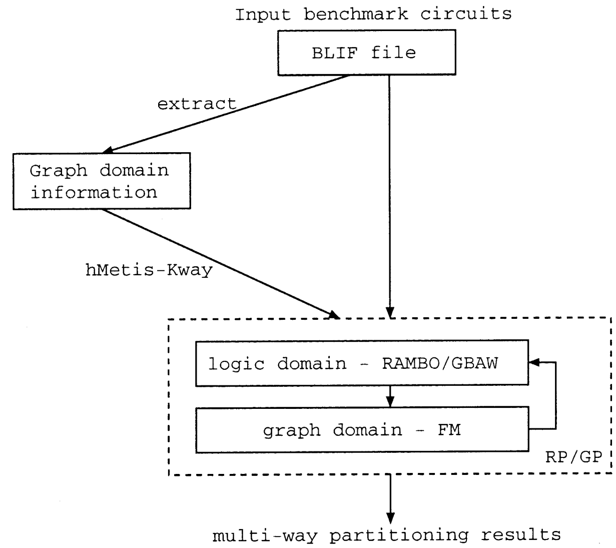


Fig. 12. Diagram shows the procedure of the experiments.

V. Experimental Results

The algorithm 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 circuits of various sizes from MCNC benchmarks. The large benchmark circuits used in ISPD98 [41] are not applicable for our experiments due to the lack of logical-domain information. The benchmark circuits are first mapped into 2-input gates by using SIS [37] package. The logic minimization by SIS [37] standard script *script.algebraic* is conducted on each benchmark circuit in the preprocess step. Fig. 12 shows the precedure of the experiments.

There are 29 MCNC benchmark circuits in our experiments. The number of nodes and wires of each circuit are shown in Table I. Column "literals," "PI," and "PO" show the literal counts, the number of primary inputs, and the number of primary ouputs, respectively. This table also lists the statistics of alternative wires on the benchmark circuits and the results are separated into three parts which are RAMBO, GBAW, and the augmented GBAW. Column "Circuit" shows the name of the circuit. Column "alt. wire" lists the number of alternative

TABLE I
BENCHMARK CIRCUITS STATISTICS

Circuit	Node	Wire	literals	PI	PO	RAMBO		GBAW		Augmented GBAW	
						alt. wires	CPU	alt. wires	CPU	alt. wires	CPU
5xp1	132	257	235	7	10	55	4.84	52	0.08	56	0.15
9sym-hdl	141	273	232	9	1	72	1.63	35	0.06	36	0.08
C1355	600	1159	1055	41	32	178	20.43	250	0.30	250	0.64
C1908	516	999	883	33	25	282	35.48	312	0.23	312	0.39
C2670	1043	1853	1444	233	140	471	81.54	502	0.43	528	0.91
C3540	1263	2476	2267	50	22	1039	289.79	1219	0.75	1228	1.49
C432	238	440	392	36	7	238	9.58	250	0.14	250	0.25
C499	503	965	854	41	32	32	10.74	34	0.22	34	0.33
C5315	1962	3746	3282	178	123	728	139.27	778	0.96	799	1.94
C6288	2856	5680	5195	32	32	2644	443.36	2214	1.54	2214	2.44
C7552	2422	4637	4105	207	108	902	285.54	680	1.81	734	2.83
C880	483	906	780	60	26	257	13.94	280	0.22	280	0.40
alu2	422	834	777	10	6	313	127.16	310	0.31	322	0.55
alu4	785	1556	1470	14	8	597	238.35	586	0.55	624	1.04
apex6	908	1681	1417	135	99	412	46.41	474	0.46	487	0.82
b9_n2	157	273	208	41	21	62	1.60	77	0.06	78	0.11
comp	184	336	270	32	3	106	4.56	88	0.08	108	0.14
des	3839	7422	6655	256	245	2643	1128.52	3334	4.81	3336	6.56
duke2	386	750	676	22	29	264	47.07	316	0.24	318	0.42
f51m	137	266	244	8	8	59	6.10	63	0.08	65	0.15
misex3	538	1062	990	14	14	414	125.51	507	0.39	507	0.65
my_adder	212	391	339	33	17	48	1.65	0	0.07	0	0.11
pcler8	130	233	174	27	17	30	1.57	30	0.03	42	0.10
rot	824	1513	1251	135	107	402	32.91	435	0.37	443	0.69
sao2-hdl	250	490	439	10	4	209	16.57	171	0.12	171	0.28
term1	272	510	439	34	10	191	11.97	197	0.12	198	0.24
ttt2	227	430	376	24	21	156	5.89	144	0.13	149	0.21
x3	855	1575	1334	135	99	385	33.63	453	0.43	459	0.79
Total						13189	3165.61	13791	14.99	14028	24.71
Normalized						1	1	1.04	0.47%	1.06	0.78%

TABLE II
COMPARISON OF 2-WAY PARTITIONING BETWEEN HMETIS-KWAY & RP AUGMENTED GP & GP

Circuit	* hMetis-Kway (Initial Part.)				RAMBO-Partitioner				Augmented-GBAW Partitioner				GBAW-Partitioner			
	area	#lits	cut pins	⋆ cpu	area	#lits	cut pins	cpu	area	#lits	cut pins	cpu	area	#lits	cut pins	cpu
5xp1	61:71	235	30	90	71:63	237	36	3	59:79	241	30	2	75:60	238	28	1
9sym-hdl	67:74	232	16	103	67:74	232	16	0	74:67	232	16	0	67:74	232	16	0
C1355	272:328	1055	48	203	324:281	1060	36	4	334:272	1061	40	3	328:273	1056	36	3
C1908	268:248	883	82	193	254:271	892	68	27	270:250	887	64	13	253:264	883	64	11
C2670	513:530	1444	42	250	520:535	1456	36	26	525:525	1451	34	26	513:535	1449	34	20
C3540	632:631	2267	134	615	615:664	2283	120	38	617:656	2276	114	43	623:650	2276	116	34
C432	119:119	392	44	133	124:123	401	36	21	119:129	402	36	6	129:119	402	36	5
C499	231:272	854	48	153	272:236	859	36	5	237:267	855	36	2	272:232	855	36	2
C5315	1046:916	3282	104	665	1053:918	3291	100	14	1047:91	3286	100	57	1047:919	3286	100	44
C6288	1455:1312	5195	82	813	1296:1575	5210	78	22	1317:1545	5201	78	111	1295:1573	5207	78	98
C7552	1281:1141	4105	18	670	1147:1281	4111	18	75	1286:1142	4111	18	77	1142:1286	4111	18	57
C880	261:222	780	50	173	259:235	791	36	8	249:245	789	36	13	262:233	789	36	11
alu2	190:232	777	82	235	197:255	807	80	64	175:252	781	78	15	179:251	785	78	11
alu4	431:354	1470	140	368	371:444	1500	128	66	434:363	1482	118	27	353:446	1484	120	20
apex6	435:473	1417	18	198	435:473	1417	18	3	435:473	1417	18	1	435:473	1417	18	1
b9_n2	87:70	208	16	95	65:93	209	12	0	63:94	208	12	0	65:92	208	12	0
comp	93:91	270	6	98	91:92	269	8	1	90:93	269	6	4	90:93	269	6	3
des	1894:1945	6655	286	1370	1989:1939	6744	276	34	1606:2233	6655	246	173	2074:1805	6684	246	150
duke2	179:207	676	82	210	232:177	699	76	10	160:231	680	78	12	167:228	684	74	9
f51m	72:65	244	28	125	81:64	252	28	8	64:76	247	28	1	76:64	247	28	1
misex3	293:245	990	80	265	250:314	1016	72	48	249:301	1002	72	18	302:250	1004	76	14
my_adder	106:106	339	4	115	106:106	339	4	0	106:106	339	4	0	106:106	339	4	0
pcler8	58:72	174	8	75	61:72	177	8	3	66:65	175	8	0	58:72	174	8	0
rot	441:383	1251	56	273	399:431	1257	52	12	384:441	1252	48	22	399:427	1253	48	17
sao2-hdl	136:114	439	26	140	114:136	418	26	0	128:123	440	16	3	129:122	440	16	2
term1	124:148	439	28	155	159:130	456	24	5	131:149	447	24	7	149:130	446	24	5
too_large	2161:1913	6387	340	140	1762:1702	6475	312	13825	1714:1664	6390	312	245	1745:1631	6388	312	219
ttt2	120:107	376	10	98	120:107	376	12	0	107:120	376	10	0	120:107	376	8	0
x3	471:384	1334	22	210	470:389	1338	12	18	469:389	1337	16	6	470:389	1338	16	5
Total		44170	1930			44572	1764	14340		44289	1696	887		44320	1692	743
Average						+0.91%	-8.6%	19.30		+0.27%	-12.1%	1.19		+0.34%	-12.3%	1

* (The result was picked from 250 runs of hMetis-Kway)
* (Total CPU times of 250 runs of hMetis-Kway)
cut size: the total number of pins required for all partitioned blocks

wires found in the circuits. Column "CPU" refers to the runtime in seconds. From the experimental results listed in Table I, we observe that GBAW and RAMBO have comparable AW searching power while GBAW uses barely 1% CPU usage of RAMBO. The augmented GBAW includes many newly formed 2-local patterns, which leads to better capability in locating AWs than the original GBAW engine.We give an example on how GP works below. Taking a C3540 benchmark as an example, the circuit is first partitioned by hMetis-Kway for 250 times to obtain an initial partition with cut pin cost of 134. The GP algorithm is then applied to further cut down the cost to 116 through the following steps.

TABLE III
COMPARISON OF THREE-WAY PARTITIONING BETWEEN HMETIS-KWAY ANDGBAW-PARTITIONER

Circuit	* hMetis-Kway (Initial Partitioning)				GBAW-Partitioner			
	area	#lits	cut pins	⋆ cpu	area	#lits	cut pins	cpu
5xp1	37:43:52	235	53	167	41:44:52	240	50	1
9sym-hdl	43:44:54	232	32	168	43:44:55	232	22	0
C1355	217:196:187	1055	84	298	217:201:190	1064	82	7
C1908	148:186:182	883	116	275	146:167:205	886	101	6
C2670	354:325:364	1444	82	335	358:310:375	1445	70	11
C3540	378:428:457	2267	185	720	376:485:404	2286	176	18
C432	67:79:92	392	56	168	70:95:77	405	54	3
C499	143:187:173	854	78	210	142:170:192	856	67	2
C5315	581:705:676	3282	119	828	522:704:741	3289	103	25
C6288	823:954:1079	5195	150	995	814:1110:934	5208	145	57
C7552	697:814:911	4105	91	855	699:943:782	4114	76	35
C880	142:159:182	780	72	265	147:168:174	790	58	6
alu2	121:164:137	777	139	308	117:139:168	793	128	6
alu4	229:248:308	1470	209	515	218:313:255	1483	195	12
apex6	276:347:285	1417	75	398	257:355:303	1431	72	10
b9_n2	45:59:53	208	21	128	44:60:54	209	21	0
comp	57:65:62	270	16	148	57:62:66	271	14	1
des	1093:1514:1232	6655	350	1765	1044:1261:1535	6658	248	87
duke2	112:124:150	676	130	295	107:135:144	691	115	5
f51m	41:53:43	244	58	170	56:48:38	248	50	2
misex3	153:209:176	990	126	375	155:209:175	1004	117	7
my_adder	80:65:67	339	8	150	67:78:67	339	8	0
pcler8	41:40:49	174	17	118	41:40:49	174	17	0
rot	235:324:265	1251	83	360	237:329:259	1255	74	9
sao2-hdl	94:77:79	439	70	215	91:76:86	446	60	3
term1	77:106:89	439	54	220	78:108:89	442	50	3
too_large	958:1330:1087	6387	749	2615	970:1058:1351	6395	683	114
ttt2	64:74:89	376	33	195	66:89:75	379	33	2
x3	243:277:335	1334	78	345	237:283:340	1354	75	10
Total		44170	3334			44387	2964	
Average						+0.49%	-11.1%	

* (The result was picked from 250 runs of hMetis-Kway)
* (Total CPU times of 250 runs of hMetis-Kway)
cut size: the total number of pins required for all partitioned blocks

- GP searches all the alternative wires of the wires which lie along the cut line and replaces them and the original graph is changed with gain 5.
- With a logically equivalent but different graph, FM reduced the cost down to 122.
- Again, GP searches along the cut line and reduces the total cost by 6.
- By switching between graph and logic domain, the cost is reduced to 116.

In our experiments, we set the tolerance of area imbalance of RP/GP to be $\pm 20\%$ of the average area in each partitioned block. Therefore, the maximal ratios are 40% : 60% and 20% : 30% in two-way and four-way partitionings, respectively. The graph-domain partitioner of RP/GP is FM and the logic-domain partitioner is RAMBO/GBAW. In order to obtain excellent initial partitions, hMetis-Kway [22] was run 250 times to pick the best result for each circuit. As hMetis-Kway is known to be quite powerful, the initial partioning results should be very excellent if not near optimum. The next step is to apply RP/GP for logic perturbation to further improve the high quality graph partitioning results with the setting of $k = 10$ and $t = 50$. Table II lists the experimental results for the two-way partitionings by four different approaches. The first one is the initial partitioning by hMetis-Kay, the second one is the ATPG-based RAMBO partitioner (RP), the third one is the augmented GBAW partitioner (GP2), and the last one, the GBAW partitioner (GP). Column "area" lists the area of the subcircuit in terms of the number of gates. "#lits" lists the total number of literals of the partitioned circuits, which is used to measure the size of the circuit. From the results, the area penalties for two-way partitionings by RP, GP2, and GP are 0.91%, 0.21%, and 0.34%, respectively. Column "cut pins" lists the total number of pins required for all partitioned blocks, which should be double of the cut net size in a two-way partitioning. Column "cpu" lists the cpu time (in seconds).

We can see that applying logic perturbation can further reduce the cut size of the good partitionings produced by the purely graph-domain partitioner significantly. The total number of literals is slightly increased because of the added gates during perturbations. Table II shows that the approach can obtain 8.6%, 12.1%, and 12.3% further reduction on cut size over the already excellent hMetis-Kway cut results in two-way partitionings using RP, GP2, and GP, respectively.

On average, RP took nearly 20 times of CPU usage than GP. In GP2, the logic-domain engine is able to locate more 2-local patterns than GP while the cut reduction is similar. Either rewiring engine applied in the logic domain can always produce further significant cut size reduction upon graph partitioner results. As GP is the fastest engine amongst the three partitioners while produces near best results, we give experimental results on three-way and four-way partitioning by using GP only in Tables III and IV. We obtained 11.1% and 11.4% reduction in cut pins for the three-way and four-way partitionings with area penalties of 0.49% and 0.57% only, respectively.

TABLE IV
COMPARISON OF 4-WAY PARTITIONING BETWEEN hMETIS-KWAY & GBAW-PARTITIONER

Circuit	* hMetis-Kway (Initial Partitioning)				GBAW-Partitioner			
	area	#lits	cut pins	⋆ cpu	area	#lits	cut pins	cpu
5xp1	40:31:34:27	235	67	170	39:31:30:37	239	61	1
9sym-hdl	38:36:36:31	232	48	200	28:35:37:40	231	26	0
C1355	141:130:177:152	1055	102	398	154:180:137:137	1077	99	8
C1908	130:131:116:139	883	134	382	125:137:118:136	886	109	6
C2670	289:238:263:253	1444	124	438	289:252:240:265	1453	84	11
C3540	281:328:357:297	2267	252	840	274:354:356:281	2280	219	22
C432	64:57:55:62	392	70	238	52:71:55:66	407	62	3
C499	140:131:114:118	854	104	315	125:146:109:125	856	87	5
C5315	460:460:520:522	3282	202	978	526:520:459:460	3288	197	25
C6288	645:674:695:842	5195	182	1130	691:847:644:674	5198	184	62
C7552	647:633:605:537	4105	66	1093	511:632:635:646	4114	44	37
C880	113:110:124:136	780	84	333	128:130:116:113	787	71	6
alu2	124:111:95:92	777	174	378	93:99:123:111	790	165	6
alu4	205:222:162:196	1470	278	587	201:156:197:231	1492	253	13
apex6	214:261:233:200	1417	75	478	209:266:233:205	1429	67	10
b9_n2	41:47:38:32	208	34	190	48:48:32:34	212	33	1
comp	43:48:47:46	270	14	228	43:48:47:46	269	12	1
des	879:1075:1035:850	6655	360	1878	986:1136:767:950	6656	277	96
duke2	95:116:96:79	676	159	333	80:115:78:113	693	142	5
f51m	35:37:30:35	244	70	188	36:40:37:29	252	67	2
misex3	111:132:138:157	990	189	425	163:136:132:108	1005	173	8
my_adder	52:54:52:54	339	12	203	54:52:52:54	339	12	0
pcler8	40:32:28:30	174	20	143	26:31:40:33	174	20	0
rot	232:208:190:194	1251	98	465	198:239:194:194	1254	91	10
sao2-hdl	60:75:52:63	439	93	270	65:63:74:49	445	73	3
term1	67:56:67:82	439	59	285	81:71:69:57	453	59	3
too_large	769:952:907:747	6387	944	2895	935:723:752:969	6391	861	160
ttt2	55:66:48:58	376	48	213	67:56:59:49	382	50	3
x3	191:193:236:235	1334	78	465	193:193:239:238	1369	70	11
Total Average		44170	4140			44421 +0.57%	3668 -11.4%	

* (The result was picked from 250 runs of hMetis-Kway)
* (Total CPU times of 250 runs of hMetis-Kway)
cut size: the total number of pins required for all partitioned blocks

VI. CONCLUSION AND FUTURE WORK

In this paper, a scheme coupling the graph-domain and logic-domain partitioners to explore a larger optimization flexibility of circuit partitioning is proposed. The scheme is shown to be very efficient in terms of CPU expenditure and is also quite capable of bringing further improvements on good partitioning produced by state-of-the-art partitioner hMetis-Kway. We conducted experiments on 29 MCNC benchmark circuits for two- to four-way partitionings and obtained further cutsize reductions, from 12.3% to 11.1%, than the high-quality results produced by hMetis-Kway. As logic-domain partitioner such as RAMBO, GBAW can be integrated with any newly developed powerful graph partitioner, this partitioning approach should be very practical and useful for various partitioning tasks.

REFERENCES

[1] Y. C. Wei and C. K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Trans.Computer-Aided Design* , vol. 10, pp. 911–921, July 1991.

[2] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning," *Integr., VLSI J*, vol. 19, no. 1–2, pp. 1–81, 1995.

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

[4] S. Dutt and W. Deng, "VLSI circuit partitioning by cluster-removal using iterative improvement techniques," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 194–200.

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

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

[7] J. Y. Zien, M. D. F. Schlag, and P. K. Chan, "Multi-level spectral hypergraph partitioning with arbitrary vertex sizes," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 201–204.

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

[9] W.-K. Mak and D. F. Wong, "Minimum replication min-cut partitioning," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 205–210.

[10] M. Enos, S. Hauck, and M. Sarrafzadeh, "Replication for logic bipartitioning," in *Proc. Int. Conf. Computer-Aided Design*, 1997, pp. 342–349.

[11] H. H. Yang and D. F. Wong, "Optimal min-area min-cut replication in partitioned circuits," *IEEE Trans. Computer-Aided Design* , vol. 17, pp. 1175–1183, Nov. 1998.

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

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

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

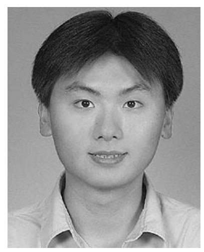
[15] R. Kuznar, F. Brglez, and B. Zajc, "Multi-way netlist partitioning into heterogeneous FPGA's and minimization of total device cost and interconnect," in *Proc. 31th ACM/IEEE Design Automation Conf.*, 1994, pp. 238–243.

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

[17] ——, "The Chaco user's guide," Sandia Nat. Labs, Albuquerque, NM, Tech. Rep. SAND93-2339, 1993.
[18] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes," in *Proc. 1995 Int. Symp. Physical Design*, 1995, pp. 113–122.
[19] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," in *Proc. 34th ACM/IEEE Design Automation Conf.*, 1997, pp. 526–529.
[20] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," in *Proc. 34th ACM/IEEE Design Automation Conf.*, 1997, pp. 530–533.
[21] S. Wichlund and E. J. Aas, "On multilevel circuit partitioning," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 505–511.
[22] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proc. 36th ACM/IEEE Design Automation Conf.*, 1999, pp. 343–348.
[23] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph bipartitioning," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 661–666.
[24] J. Cong and S. K. Lim, "Edge separability based circuit clustering with application to circuit partitioning," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 429–434.
[25] ——, "Performance driven multiway partitioning," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 441–446.
[26] M. Wang, S. K. Lim, J. Cong, and M. Sarrafzadeh, "Multi-way partitioning using bi-partition heuristics," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 667–672.
[27] K.-T. Cheng and L. A. Entrena, "Multi-level logic optimization by redundancy addition and removal," in *Proc. EDAC-93*, Feb. 1993, pp. 373–377.
[28] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng, "Perturb and simplify: Multilevel boolean network optimizer," *IEEE Trans. Computer-Aided Design* , vol. 15, pp. 1494–1504, Dec. 1996.
[29] S. C. Chang, K. T. Cheng, N. S. Woo, and M. Marek-Sadowska, "Layout driven logic synthesis for FPGA," in *Proc. 31th ACM/IEEE Design Automation Conf.*, June 1994, pp. 308–313.
[30] S.-C. Chang, K.-T. Cheng, N.-S. Woo, and M.-S. M, "Postlayout logic restructuring using alternative wires," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 587–596, June 1997.
[31] S. C. Chang and M. Marek-Sadowska, "Perturb and simplify: Multilevel boolean network optimizer," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 2–4.
[32] L. A. Entrena and K.-T. Cheng, "Sequential logic optimization by redundancy addition and removal," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 310–315.
[33] ——, "Combinational and sequential logic optimization by redundancy addition and removal," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 909–916, July 1995.
[34] L. Entrena, J. Espejo, E. Olias, and J. Uceda, "Timing optimization by an improved redundancy addition and removal technique," in *Proc. 1996 Design Automation Conf.*, 1996, pp. 342–347.
[35] S. C. Chang, L. P. V. Ginneken, and M. Marek-Sadowska, "Fast boolean optimization by rewiring," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 262–269.
[36] C.-W. Chang and M. Marek-Sadowska, "Single-pass redundancy addition and removal," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 2001, pp. 606–609.
[37] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit synthesis," in *Proc. ERL Memorandum UCB/ERL*, vol. M92/41, 1992.
[38] Y.-L. Wu, W. Long, and H. Fan, "A fast graph-based alternative wiring scheme for boolean networks," in Proc. IEEE/ACM Int. VLSI Design, 2000, pp. 268–273.
[39] Y. L. Wu, C. N. Sze, C. C. Cheung, and H. Fan, "On improved graph-based alternative wiring scheme for multi-level logic optimization," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, 2000, pp. 654–657.
[40] Y. L. Wu, X. L. Yuan, and D. I. Cheng, "Circuit partitioning with coupled logic restructuring techniques," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 655–660.
[41] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "The ISPD98 circuit benchmark suite," in *Proc. Int. Symp. Physical Design*, 1997, pp. 80–85.

Yu-Liang Wu (M'96) received the B.S. and M.S. degrees in computer science from Florida International University, Miami, FL, in 1983 and 1984 respectively, and the Ph.D. degree in electrical and computer engineering from the University of California Santa Barbara, in 1994.

In 1985, he worked with the Internet Systems Corporation as a System Programmer on network communication protocols (DARPA TCP/IP, Telnet). From 1986 to 1988, he worked with AT&T Bell Labs on the development of several telephone distributed operation systems (RMAS, MFOS). From 1988 to 1989, he worked for Amdahl Corporation on tester software designs. From 1994 to 1996, he was with Cadence Design Systems Incorporation as a senior MTS where he worked in research and development of the silicon synthesis product (PBS), targeting at binding the gap between logical and physical level optimizations for deep-submicron chip designs. In 1996, he joined The Chinese University of Hong Kong. His current research interests include optimization of logic and physical design automation of very large scale integration (VLSI) circuits and field programmable gate arrays (FPGAs) related computer-aided design (CAD) tool designs and architectural analysis/optimization.



Chak-Chung Cheung received the B.Eng. and M.Phil. degree in computer engineering and computer science and engineering from The Chinese University of Hong Kong (CUHK), in 1999 and 2001, respectively.

In 2001, he worked as a System Administrator with the Center of Large-Scale Computation (CLC), Cluster Technology, Hong Kong. He is now an Instructor with the Deparment of Computer Science and Engineering, CUHK, and is responsible for the courses on Internet and Web Programming Technologies. His current research interests include optimization of logic and physical design automation of very large scale integration (VLSI) ASIC/FPGA designs and high-level synthesis of reconfigurable computing.

David Ihsin Cheng received the B.S. degree in computer engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1986, the M.S. degree in computer science and the Ph.D. degree in computer engineering, both from the University of California, Santa Barbara, in 1990 and 1995, respectively.

He is currently with Faraday Technology Corp. Sunnyvale, CA, where he specializes in ASIC implementation. He was previously with Ultima Interconnect, Sunnyvale, CA, Mentor Graphics, Wilsonville, OR, IBM Watson Research Center, Yorktown Heights, NY and AT&T Bell Labs. His research interests include all aspects of the logical and physical implementations of ASICs.

Hongbing Fan received the B.S. degree in mathematics and the Ph.D. degree in operational research and control theory from Shandong University, Jinan, China, P.R.C., in 1982 and 1990, respectively, and received the second Ph.D. degree in Computer Science from the University of Victoria, Canada, in 2003.

In 1990, he joined the mathematics department at Shandong University as an Associate Professor. He worked with the Department of Computer Science and Engineering at The Chinese University of Hong Kong (CUHK), as an Research Associate in 1998. His research interests include combinatorial algorithm and complexity, graph theory, and various topics in very large scale integraton (VLSI) design and operations research.