Journal of the Korean Physical Society, Vol. 40, No. 4, April 2002, pp. 754∼758

New Efficient Clique Partitioning Algorithms for Register-Transfer Synthesis of Data Paths

Jong Tae KIM* and Dong Ryeol SHIN
School of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 440-746

(Received 11 February 2001)

Numerous problems can be modeled as clique partitioning problems in digital design synthesis. In this paper, we present two new polynomial time heuristic algorithms for efficient clique partitioning with or without limiting the maximum clique size. The goal of clique partitioning is to partition a graph into a minimum number of cliques. The basic approach of the new algorithm is to find small cliques by first using a fast connectivity test. The performance comparisons of the new algorithm produces better results than other existing clique partitioning algorithms. The modified algorithm, which limits the maximum clique size, becomes more efficient and can be used to solve problems in register transfer synthesis of pipelined data paths, such as interconnection sharing. The results show that the new polynomial-time heuristic algorithm finds a near optimal solution quickly.

PACS numbers: 85.40.h
Keywords: Clique, Partitioning

I. INTRODUCTION

The data path of a digital system generally consists of functional units such as storage elements and operators, and performs the storage of data in registers and memory, data transfer via a set of busses or multiplexers, and data transformation by operators. In order to complete the register-transfer (RT) data path the following tasks are needed: 1. the allocation of storage elements, 2. the allocation of data operators, 3. the allocation of interconnection units, and 4. the interconnection of all those components. The problems of data path synthesis can be formulated into the clique partitioning problem [1,2]. A clique is a complete subgraph which is contained within no other complete subgraph of a graph G [3]. The clique partitioning problem is to separate the nodes in G into a number of cliques such that every node should be in only one clique. The clique partitioning problem is NP-complete [4]. Other related research can be found in Refs. 5 and 6.

We introduce a new efficient clique artitioning heuristic algorithm, called the F_CLIQUE algorithm, to find the optimal or near optimal partitions within polynomial time. By optimal, we mean that the algorithm finds the minimum number of cliques. Then the F_CLIQUE algorithm is modified, called M_CLIQUE, to find the cliques of size L or smaller for solving the interconnection sharing problem of the pipelined data-path synthesis. We use two other algorithms, one developed by Tseng [1] and the other by Bhasker [5] , for the performance comparisons.

A detailed description and an analysis of algorithms are given in the following section. In Section 3 the performance comparisons are discussed. Section 4 contains the modified algorithms, in which the user specifies the size L of cliques and the procedure finds cliques that are equal to or smaller than L, and their performance comparisons. We conclude this work in Section 5.

II. CLIQUE PARTITIONING ALGORITHMS

There are two useful approaches in heuristic clique partitioning algorithms. First, find maximal cliques so that the number of cliques can be minimized. Tseng's algorithm uses this approach. In this procedure, the edge

```
1. Pick an edge (p,q) which has maximum number of
   common neighbors;
   Tie-breaking : select p and q such that the sum of
   node degrees is maximum;
2. - Combine p and q, and call it p;
   - Delete edges from p and q that are not connected
   to their common neighbors;
   - If the list of edges is empty, THEN STOP.
3. Pick a node, q, which has maximum number of
   common neighbors with p;
   THEN goto 1;
   ELSE goto 2;
```

Fig. 1. Tseng's algorithm.

*E-mail: jtkim@yurim.skku.ac.kr

```
1. Pick a node with smallest degree and call it p;
2. Pick a neighbor of p,q, with smallest degree;
   Tie-breaking :
   - Choose q if it has any common neighbors with p.
   - If still tie, choose q arbitrarily.
3. - Combine p and q and call it p;
   - Delete edges from p and q that are not connected
     to their common neighbors;
   - If the list of edges is empty,
     THEN STOP
     ELSE goto 1.
```

Fig. 2. Bhasker's algorithm.

```
1. Pick a node with smallest degree and call it p;
2. - Pick a neighbor of p, q, such that the number of
   common neighbors is maximum;
   Tie-breaking : select q such that the node degree
   of q is minimum;
   - Combine p and q and call it p;
   - Delete edges from p and q that are not connected
     to their common neighbors;
3. If p has any remaining neighbors THEN goto 2;
4. If the list of edges is empty,
   THEN STOP;
   ELSE goto 1;
```

Fig. 3. F_{Clique} algorithm.

with the maximum number of common neighbors is chosen first as a primary edge to initiate a search for a maximal clique. Cliques are found one at a time. Tseng's algorithm was developed for the synthesis of data paths at the register transfer level since the minimization tasks, which are the minimization of the number of storage elements, data operators, and interconnection units, can be modeled as the clique-partitioning problem. The algorithm is described in Fig. 1. The procedure applies the neighborhood property, which means the edge with the maximum number of common neighbors is chosen first as a primary edge, to search for a maximal clique, and cliques are found one at a time.

Second, find two verices such that the number of edges deleted can be minimized and the number of edges left after they are merged is always maximum. Suppose that node p and node q are merged; the number of edges deleted is $e_d = e_p + e_q - c_{pq} - 1$, where e_d is the number of edges deleted, e_p is the number of edges of node p, e_q is the number of edges of node q, and c_{pq} is the number of common neighbors between node p and node q. To minimize the number of edges to be removed, we choose two nodes with minimum e_p and e_q and maximum c_{pq}. Bhasker and Samad introduced two algorithms. The second one includes a tie-breaking rule in step 2. The procedure is presented in Fig. 2. The algorithm is very simple, and it chooses the node with smallest degree as a primary node. Unlike the case of Tseng, even though the current primary node is not isolated, a new primary node is selected, so the algorithm constructs cliques in parallel.

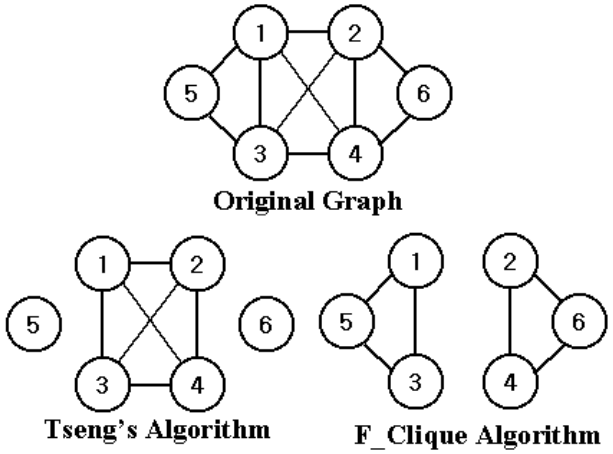

Fig. 4. Clique partitoning example 1.

Our F_CLIQUE algorithm is presented in Fig. 3. It improves major shortcomings of both Tseng's and Bhasker's algorithms. Basically F_CLIQUE and Bhasker's algorithms share a common ground; both algorithms are heuristic, and they consider only part of the requirements. Bhasker's algorithm selects the nodes with minimum e_p and e_q. Our approach chooses node p with the minimum number of edges and looks for node q which has a maximum number of common neighbors with p; *i.e.*, it considers e_p and c_{pq}. Considering the least connected node first has advantages over the neighborhood property. The reason is that the task of finding the edge with the maximum common neighbors is more time consuming than the one of finding the node with the smallest degree. Also, this time-consuming task does not guarantee finding an optimal solution.

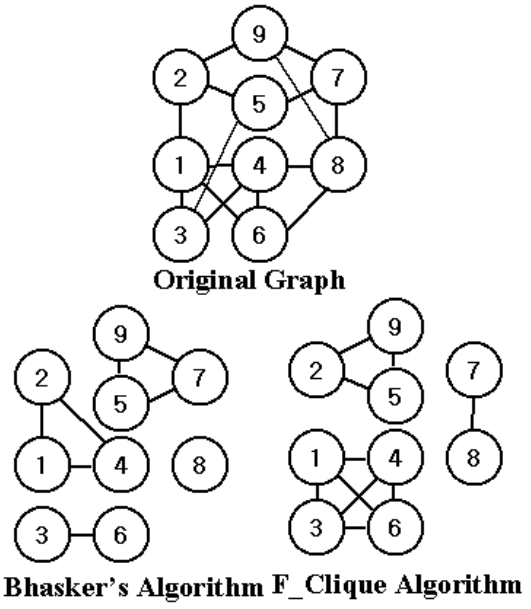

Fig. 5. Clique partitoning example 2.

Table 1. Performance comparisons of the three algorithms (30 nodes).

30 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No. of cliques	14.96	14.65	14.65
	Time	0.0054	0.0014	0.0020
30 %	No. of cliques	11.05	10.71	10.51
	Time	0.0092	0.0022	0.0032
50 %	No. of cliques	8.17	7.95	7.94
	Time	0.0116	0.0032	0.0044
70 %	No. of cliques	5.97	5.65	5.65
	Time	0.0128	0.0040	0.0060

Table 3. Performance comparisons of the three algorithms (100 nodes).

100 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No.of cliques	39.94	40.35	39.59
	Time	0.327	0.0156	0.0246
30 %	No.of cliques	26.32	26.75	25.34
	Time	0.5576	0.0262	0.0476
50 %	No.of cliques	18.73	18.55	17.79
	Time	0.633	0.0356	0.068
70 %	No.of cliques	12.44	12.00	11.82
	Time	0.5954	0.0456	0.1008

Two examples are introduced to explain the differences among the three algorithms. Fig. 4 shows the first example. In this example Tseng's algorithm first selects the edge (1 2) which has the maximum number of common neighbors in the graph. When two nodes, 1 and 2, are merged, two edges, (1 5) and (2 6), are deleted. The optimal solution cannot be found in this case. On the other hand, F_CLIQUE and Bhasker's algorithm can find the optimal solution. These examples serve only to illustrate the differences in the two approaches. The second example is shown in Fig. 5. It is worth a close look into the tie-breaking rules. Bhasker's algorithm does not guarantee the deletion of a minimum edge when tie-breaking situations occur. When the node q is selected in step 2 of Bhasker's procedure, if there several nodes have the same smallest degree of edges and also have common neighbors with node p, there is a big chance to miss the optimal set of cliques. This is not severe in the case of small graph sizes, but in large graph sizes, it is necessary to have more sophisticated tie-breaking rules. Let's look at the example in Fig. 5. Bhasker's algorithm choose vertex 1 as a primary node. Vertices 2, 3, and 6 are candidates for node q since all of them have the same minimum node degrees and the same number of common neighbors. Thus, vertex 2 is chosen since we use the lowest-index-first tie-breaking rule. This leads to the clique (1 2 4)and creates four cliques, (1 2 4), (3 6), (5 7 9), and (8). However, our algorithm considers the number of common neighbors and calculates the number of edges to be deleted when two nodes are combined. It chooses the node with the minimum number of edges to be deleted and guarantees that the minimum number of edges will be deleted in a tie situation. Our algorithm combines node 1 and 3 and produces the optimal solution with three cliques, (1 3 4 6), (2 5 9), and (7 8).

III. PERFORMANCE COMPARISONS

We have implemented three algorighms in C running on ULTRA SPARC workstations. All three programs use an adjacency matrix as a data structure. We used randomly generated graphs for comparisons. We tested algorithms on graphs of 30, 50, 100, 200, 300, and 400 vertices, varying the number of edges for each case between 10 to 70 % Of connectivity. The experimental results are presented in Table 1 through Table 6. It shows the average number of cliques and their run times obtained over 50 random graphs. For example, with 50 random graphs of 100 vertices and 50 % Connectivity, the number of cliques are 18.73, 18.55, and 17.79 on an average by using the Tseng, Bhasker, and F_CLIQUE algorithms, respectively.

Table 2. Performance comparisons of the three algorithms (50 nodes).

50 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No.of cliques	23.34	23.29	23.28
	Time	0.0298	0.0036	0.0056
30 %	No.of cliques	15.81	15.76	15.16
	Time	0.049	0.0062	0.0098
50 %	No.of cliques	11.48	11.20	11.00
	Time	0.0592	0.0088	0.0142
70 %	No.of cliques	7.97	7.63	7.61
	Time	0.0618	0.0116	0.0194

Table 4. Performance comparisons of the three algorithms (200 nodes).

200 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No.of cliques	69.43	72.31	68.28
	Time	4.106	0.0712	0.1272
30 %	No.of cliques	44.29	46.97	42.83
	Time	7.192	0.1084	0.264
50 %	No.of cliques	30.86	31.77	29.31
	Time	7.761	0.1444	0.3976
70 %	No.of cliques	20.00	19.80	18.31
	Time	6.7428	0.1848	0.5712

Table 5. Performance comparisons of the three algorithms (300 nodes).

300 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No.of cliques	95.71	103.66	96.09
	Time	18.88	0.1748	0.3504
30 %	No.of cliques	60.88	65.63	59.21
	Time	33.13	0.2508	0.7388
50 %	No.of cliques	41.50	43.38	39.78
	Time	34.92	0.3326	1.136
70 %	No.of cliques	26.18	26.86	25.10
	Time	29.22	0.4184	1.618

Table 6. Performance comparisons of the three algorithms (400 nodes).

400 NODES		TSENG	BHASKER	F_CLIQUE
10 %	No.of cliques	122.04	133.56	120.72
	Time	46.70	0.2744	0.5996
30 %	No.of cliques	76.35	83.10	74.15
	Time	80.03	0.398	1.326
50 %	No.of cliques	51.57	54.78	43.57
	Time	84.91	0.54	2.02
70 %	No.of cliques	32.30	33.40	31.15
	Time	70.26	0.7012	2.909

In general Bhasker's algorithm works fine with dense graphs, but has the worst performance with graphs with large numbers of vertices. Tseng's algorithm works well with sparse graphs, but the execution time is so long that it is not adequate for most practical applications. Since Tseng's algorithm finds the maximal clique first, its run time decreases with increasing connectivity over 50 %. We can see that the F_CLIQUE algorithm produces the best result within a reasonable time. Although it is slower than Bhasker's algorithm, F_CLIQUE algorithm is the best choice among them in the average number of cliques. For example, see Table 4, with 200 vertices and 30 % Connectivity, the F_CLIQUE algorithm takes 0.264 average run times in seconds to get 42.83 average cliques. On the other hand, Bhasker's algorithm gets 46.97 average cliques in 0.1084 seconds on average.

IV. MODIFIED CLIQUE PARTITIONING ALGORITHM

In a pipelined design [7], an initiation interval, L, is a key parameter in determining the performance of a pipeline. If the pipeline input initiation latency is one, all the operations perform useful operations at any time, except when there are hazards or conditional branches. In such a case, no operators or registers can be reused during the execution of each instruction, which makes register-transfer synthesis straightforward and simple. However, if the latency is greater than one, then each operator can be reused as many times as the latency during the execution of each instruction.

```
1. Pick an edge (p,q) which has maximum
   number of common neighbors;
   Let clique size = 1;
   Tie-breaking : select p and q such that the
   sum of node degrees is minimum;
2. - Combine p and q, and call it p;
   - Clique size is increased by 1;
   - Delete edges from p and q that are not
     connected to their common neighbors;
   - If the list of edges is empty, THEN STOP.
3. Pick a node, q, which has maximum number
   of common neighbors with p;
   Tie-breaking : select q such that the node degree
   of q is minimum;
   If node p is isolated (p has no neighbors) or
   clique size is equal to L
   THEN goto 1;
   ELSE goto 2;
```

Fig. 6. Modified Tseng's algorithm.

In register transfer synthesis of pipelined data paths, especially, the interconnect sharing task is modeled as a clique partitioning problem [8,9]. We need to get only the clique size L (L is the pipeline initiation interval and is usually small) or smaller. Also, maximum interconnect sharing is achieved by minimizing the total number of cliques and not by maximizing the size of each clique. We developed a fast and efficient polynomial-time heuristic procedure to solve this problem. This procedure finds the cliques of size L or smaller. Using this procedure, we can produce near optimal interconnect-sharing schemes in a few seconds for most practical-sized pipelined design.

Both Tseng's algorithm and the F_CLIQUE algorithm

```
1. - Pick a node with smallest degree and call it p;
   - Let clique size = 1;
2. - Pick a neighbor of p, q such that the number of
     common neighbors is maximum;
   Tie-breaking :  select q such that the node degree
   of q is minimum;
   - Combine p and q and call it p;
   - Clique size is incremented by 1;
   - Delete edges from p and q that are not connected
     to their common neighbors;
3. IF p has any remaining neighbors and clique size
    is less than L
   THEN goto 2;
4. IF the list of edges is empty,
   THEN STOP;
   ELSE goto 1;
```

Fig. 7. MCLIQUE algorithm.

Table 7. Experimental Result of Modified Algorithms(50 nodes).

50 NODES		TSENG			M_CLIQUE		
		Clique size			Clique size		
		4	3	2	4	3	2
10%	No.of cliques		23.42	26.58		23.12	25.08
	Time		0.027	0.034		0.004	0.004
30%	No.of cliques	16.70	19.25	25.85	15.40	17.25	25.00
	Time	0.050	0.059	0.077	0.008	0.008	0.008
50%	No.of cliques	14.65	18.45	25.90	13.35	17.00	25.00
	Time	0.077	0.095	0.127	0.013	0.014	0.014
70%	No.of cliq	14.60	18.25	25.60	13.40	17.00	25.00
	Time	0.100	0.122	0.165	0.019	0.019	0.017

Table 8. Experimental Result of Modified Algorithms(100 nodes).

100 NODES		TSENG			M_CLIQUE		
		Clique size			Clique size		
		4	3	2	4	3	2
10%	No.of cliques		40.80	51.33		39.87	50.07
	Time		0.345	0.459		0.022	0.022
30%	No.of cliques	29.60	35.87	51.13	26.87	34.00	50.00
	Time	0.656	0.817	1.153	0.046	0.049	0.050
50%	No.of cliques	27.67	35.07	50.87	25.93	34.00	50.00
	Time	1.03	1.301	1.841	0.080	0.084	0.080
70%	No.of cliques	26.87	34.87	50.73	25.60	34.00	50.00
	Time	1.375	1.752	2.505	0.123	0.121	0.108

can be modified to find cliques with size L or smaller without destroying the original properties since they find cliques one at a time. On the other hand, Bhasker's algorithm cannot be modified without destroying the original properties because it finds cliques in parallel. For this reason, we have implemented only Tseng's algorithm and the F_CLIQUE algorithm, and the outlines of two modified algorithms, modified Tseng and M_CLIQUE, are in Fig. 6 and Fig. 7, respectively.

We tested the algorithms on graphs of 50, 100, and 200 vertices, varying the number of edges for each case between 10 to 70 % Connectivity. Table 7 thru Table 9 show the results of experiments in terms of the average number of cliques and the run times while the clique sizes are limited in 2, 3, and 4. Our algorithm is 10 to 30 time faster than the modified Tseng while producing better results for our purpose.

V. CONCLUSION

We present two new polynomial-time heuristic algorithm for efficient clique partitioning that can be applied in RT synthesis. The goal of clique partitioning in this project is to partition a graph into a minimum number of cliques. The basic approach of our algorithm is to find small cliques first by using a fast connectivity test. The modified algorithm becomes more efficient when the maximum size of the cliques is fixed. This is very useful for the solving interconnection- sharing problem in RT synthesis of pipelined data paths. We performed an extensive experiment with new algorithms and the results show that the F_CLIQUE and the M_CLIQUE algorithms can find near optimal solutions quickly.

Table 9. Experimental Result of Modified Algorithms(200 nodes).

200 NODES		TSENG			M_CLIQUE		
		Clique size			Clique size		
		4	3	2	4	3	2
10%	No.of cliques	68.93	73.73	101.00	68.40	69.87	100.00
	Time	4.128	4.726	6.612	0.121	0.119	0.127
30%	No.of cliques	55.00	69.87	101.00	51.47	67.27	100.00
	Time	9.828	12.72	2.019	0.291	0.325	0.336
50%	No.of cliques	53.13	68.93	100.97	50.80	67.00	100.00
	Time	15.44	20.18	29.44	0.534	0.565	0.5354
70%	No.of cliques	51.93	68.20	101.07	50.60	27.00	100.00
	Time	20.62	26.71	39.59	0.834	0.832	0.727

ACKNOWLEDGMENTS

This work was supported in part by the IDEC (IC Design Education Center).

REFERENCES

[1] C. Tseng and D. Siewiorek, IEEE Trans. CAD **5**, 379 (1986).
[2] J. Jou, S. Kuang, and Chien, In *Proceedings of VLSI TSA* (Kyoto, May, 1993), p.58.
[3] E. Horowitz and S. Sahni, *Fundamental of Computer Algorithms* (Computer Science Press, Rockville, 1978).
[4] M. Gary and D. Johnson, *Computer and Intractability: A guide to the Theory of NP-completeness* (Freeman, San Francisco CA, 1979).
[5] J. Bhasker and T. Samad, Computers and Mathmatics with Applications **22**, 1 (1991).
[6] G. Mulligan and D. Corneli, J. ACM **19**, 244 (1972).
[7] G. Kim, S. Kwon, and M. Lee, J. Korean Phys. Soc. **35**, S960 (1999).
[8] N. Park and F. Kurdahi, VLSI Design **2**, 32 (1994).
[9] J. Kim and N. Park, J. Korean Phys. Soc. **37**, 1077 (2000).