Experimental Study of A Novel Variant of Fiduccia Mattheyses(FM) Partitioning Algorithm

1Mitali Sinha, 2Rakesh Mohanty, 3Prachi Tripathy
Department of Computer Science and Engineering
Veer Surendra Sai University of Technology, Burla,
Sambalpur, Odisha, India-768018
1mitalisinha.148@gmail.com, 2rakesh.iitmphd@gmail.com,
3prachitripathy@gmail.com

3Suchismita Pattanaik
Department of Electronics and Communication Engineering
Sambalpur University Institute of Information Technology,
Jyoti Vihar, Burla, Sambalpur, Odisha, India-768019
spattanaik.vlsi@gmail.com

Abstract—Partitioning is a well studied research problem in the area of VLSI physical design automation. In this problem, input is an integrated circuit and output is a set of almost equal disjoint blocks. The main objective of partitioning is to assign the components of circuit to blocks in order to minimize the numbers of inter-block connections. A partitioning algorithm using hypergraph was proposed by Fiduccia and Mattheyses with linear time complexity which has been popularly known as FM algorithm. Most of the hypergraph based partitioning algorithms proposed in the literature are variants of FM algorithm. In this paper, we have proposed a novel variant of FM algorithm. We have performed a comparative experimental study of FM algorithm and our proposed algorithm using two datasets such as ISPD98 and ISPD99. Experimental results show that performance of our proposed algorithm is better than the FM algorithm using the above datasets.

Index Terms— VLSI, Physical design automation, Partitioning problem, Hypergraph, Netcut, FM algorithm.

I. INTRODUCTION

Hypergraph partitioning is a NP-hard problem[4]. Though hypergraph partitioning has extensive applications in various fields such as data-mining, job scheduling, image processing, improving page fault and VLSI design, a number of heuristic algorithms were developed with polynomial time-complexity. Fiduccia and Mattheyses (FM) algorithm [5] is a basic hypergraph partitioning algorithm with single shift in which time-complexity is linear in nature. In this paper, we studied the FM algorithm and explored its limitations. We proposed a variant of FM algorithm and conducted experimental studies of our proposed algorithm by considering two standard data sets. Through our experiments, we have done comparative performance analysis of our proposed algorithm with the FM algorithm.

Very Large Scale Integration (VLSI) is a technique of manufacturing an Integrated Circuit(IC) by integrating thousands of connected electronic components into a single chip. The components may be transistors, resistors, capacitors and inductors etc. A group of connected components can be represented as a block. In circuit layout, the length of connections between the components of two different blocks is more than that of the length of connections between the components within the same block. Therefore we have to minimize the number of connections between the components of two different blocks to reduce the cost of wire length.

A. Hypergraph Partitioning

Circuit in the form of a graph or hypergraph is provided as the input to a partitioning algorithm. A hypergraph is a generalization of graph in which an edge connects any number of vertices and this edge is called as hyperedge. Mathematically hypergraph can be represented as \( H(V, E) \), where \( V \) is the set of vertices and \( E \) is the set of hyperedges. A circuit can be converted to a hypergraph in which a vertex of hypergraph represents component of the circuit and a hyperedge represents the set of components which share the same signal known as net. A set of nets which represent a circuit is known as netlist. Two vertices of a hypergraph are said to be neighbor if both belong to at least one common net. A circuit and its netlist representation are shown in fig. 1(a) and fig. 1(b) respectively.

Fig. 1 Input representation of a partitioning algorithm

(a) Circuit    (b) Netlist    (c) Hypergraph

The netlist in fig. 1(b) contains three nets \( n_1, n_2 \) and \( n_3 \). \( n_1 \) contains components \( c_4, c_5 \) and output of \( c_4 \) is provided as input to \( c_5 \). Similarly \( n_2 \) contains \( c_1, c_2 \) and \( n_3 \) contains \( c_1, c_2 \) and \( c_5 \). The Netlist is represented as a hypergraph \( H(V, E) \) as shown in fig.1(c) in which \( V=\{c_1, c_2, c_3, c_4, c_5\} \) and \( E=\{n_1, n_2, n_3\} \).

Fig. 2 Input and Output of a Hypergraph partitioning

We consider a hypergraph as shown in Fig.2 as the input to a partitioning algorithm and divide it into two approximately equal blocks. Here the components \( c_4, c_5, c_3 \) are present in Block-1 and \( c_1, c_2 \) are present in Block-2. The number of netcuts of this partitioning is one because the components of net \( n_3 \) are present in both the blocks.
B. Literature Review

In order to solve the partitioning problem in VLSI context, the first graph bi-partitioning algorithm was proposed by Kernighan and Lin [1], popularly known as KL algorithm. The time-complexity of KL algorithm is $O(n^2)$ where $n$ is the number of vertices of the input graph. A faster KL algorithm was introduced in [6]. As reported in [7], graph is not a proper representation of a circuit because it cannot correctly convert a net to an edge or a set of edges. The most correct representation of a circuit is hypergraph. A hypergraph partitioning algorithm was proposed by Fiduccia and Mattheyses [5] in the year 1982. The main advantage of this algorithm is its linear time-complexity with respect to the size of the circuit.

A number of variants of FM algorithm with improved performance were developed in [8]-[9]-[10]-[11]. Alpert and Kahng have done a comprehensive survey on netlist partitioning in [3]. A new class of partitioning algorithms known as 2-phase FM, has been mentioned in [12]-[13]-[14]-[15]. FM algorithm has been extended to various multi-level FM algorithms [16]-[17]-[18]-[19]-[20] for better result in terms of solution quality and run time.

C. Our Contribution

In this paper we have proposed a novel variant of FM algorithm by using the idea of pair wise swapping of vertices in hypergraph partitioning. Initially a hypergraph is partitioned in to two blocks of roughly equal size by randomly assigning the vertices of hypergraph to each of the blocks. Then vertices are selected in pair wise manner and swapped in order to reduce the total number of netcuts. We have developed a formula for reduction in netcuts due to pair-wise swapping of components in hypergraph partitioning. We have made a comparative performance analysis of our proposed variant of FM algorithm with FM algorithm using two data sets such as ISPD98 and ISPD99 benchmark circuits. Our experimental results show that our proposed algorithm outperforms FM algorithm.

D. Organization of Paper

FM algorithm and its pseudo-code are presented in section II. Section III contains our proposed variant of FM algorithm and its pseudo-code. Our experimental study and results are shown in section IV. Section V presents the conclusion and future work.

II. FM ALGORITHM

A. Notations and Definitions

Let $N_{cut}$ be the total number of nets which are cut. Cutset be the set of nets which are cut and $n(c_k)$ be the set of nets connected to $c_k$. $P$ be the maximum number of nets to which any component is connected.

**Block Size(S(B_i)):** The number of components present in a block is defined as the block size.

**Complementary block:** If partitioning of a netlist contains two disjoint blocks $B_1$, $B_2$ and a component $c_k$ is present in $B_1$, then $B_2$ is called the complementary block of $c_k$.

**Locked Component:** When a component is shifted from its current block to its complementary block, it will not be considered for further movement. So it is locked.

**Gain value (G(c_k)):** The gain value of a component $c_k$ is the number of reductions in nets from Cutset if it is moved from its current block to its complementary block. It is calculated as follow.

Let $N_{ck}$: Number of net which have only one component i.e $c_k$ in the current block of $c_k$.

Let $N'_{ck}$: Number of nets which contain component $c_k$ and completely present in the current block of $c_k$.

$$G(c_k) = N_{ck} - N'_{ck}$$

**Gain bucket:** Gain bucket is used to sort the gain values of the components present in a block. Its index ranges from $-P$ to $+P$. The $K$th index of gain bucket contains a linked list of components having gain value $K$.

**Update neighbour’s Gain of (c_k):** This function update the gain values of all unlocked components which are neighbors of $c_k$ [5] and this update will be reflected in the gain buckets.

**Make Unlock(c_k):** This function is used to unlock a component $c_k$.

**Make Lock(c_k):** This function is used to lock a component $c_k$ and delete $c_k$ from its gain bucket.

B. Pseudo Code of FM Algorithm

The first hypergraph bi-partitioning algorithm is the FM algorithm [5] with linear time complexity. It starts with a random initial partitioning of the hypergraph $H$ into two almost equal size blocks $B_1$ and $B_2$ and $N_{cut}$ is calculated. At the beginning of the process, all the components are made unlocked and the gain value of each component is calculated. Components of each block are sorted using bucket sorting according to their gain values in order avoid unnecessary search for the component having maximum gain value.

A component $c_k$ with highest gain value is selected to move from its current block to its complementary block and remains locked throughout the process. The size of $c_k$’s current block should be greater than or equal to its complimentary block. After $c_k$ is moved, the gain values of all its unlocked neighbors are updated in their respective gain bucket for next move and $N_{cut}$ is recorded at that point. This is continued until all components are locked.

This entire process is called a pass. When a component is locked, it cannot be considered for further move within that pass. At the end of a pass, the point at which the optimal $N_{cut}$ was achieved is selected and the moves of all components after that point are cancelled. The partitioning result of one pass is given as input to next pass. This process is continued till improvement in $N_{cut}$. Finally the optimal $N_{cut}$ is achieved.
After a comprehensive study and analysis of FM algorithm the following limitations are observed.

### C. Limitations of FM Algorithm

When more than one component has same gain value then FM algorithm randomly choose any one component for shifting. So it does not always provide optimal result. Component’s move operation is highly influenced by the balancing constraint of block [21]. FM algorithm uses the technique of single shifting of component instead of pair-wise swapping but pair-wise swapping provides better result than single shifting of component [6]. First limitation is addressed by many other proposed algorithms described in [8]-[10]-[11]. In our work, we have addressed the last two limitations by developing a novel variant of FM algorithm.

### III. PROPOSED VARIANT OF FM ALGORITHM

An algorithm that swaps node pairs can provide a better $N_{cut}$ improvement than one that shifts a single node at a time [6]. In this paper we have applied pair-wise swapping of components on hypergraph partitioning by proposing a novel variant of FM algorithm. In this variant of FM algorithm, two components from each block are swapped so that this pair produces the maximum reductions in nets from Cutset than any other pair as proposed in [6] for graph partitioning. Before presenting our proposed variant of FM algorithm we introduce some definitions and notations as follows.

**Critical net($n_c$):** If any component of a net is shifted from its current block to its complementary block and as a result the net is being removed from Cutset then such a net is called critical net. In fig.4 $n_3 = \{c_1, c_2, c_5\}$ is a critical net because $n_3$ is being removed from Cutset due to shifting of $c_5$ from $B_1$ to $B_2$.

![Fig.3 Pseudo-code of FM algorithm](image)

**Gain due to pair-wise swapping ($G(u_i, v_j)$):**

$$G(u_i, v_j) = G(u_i) + G(v_j) - \text{correct_term}.$$
to their gain values. Each element of $M$ corresponds to component pair $(u_i, v_j)$. In this variant, $u_i$ and $v_j$ from each block are selected so that $(u_i, v_j)$ pair provides highest gain value than any other pair. The pseudo-code of FM_Variant for best pair selection in hypergraph partitioning is described in Fig.7 as mentioned in [6] for graph. The worst case time complexity for finding the first non-neighbor component of $u_1$ is $O(d)$ i.e $b=d$ where $d$ is the maximum degree of any component if the hypergraph is visualized as a clique based graph[3]. In the worst case, the gain values of all the neighbor components of $u_1$ will be greater than gain value of non-neighbor components of $u_1$. So $b$ value will be repeated $d$ times to reach the $1$st non-neighbor component in the $1$st while loop condition in the Fig.7. After this, a for loop is continued at most $(d-1)$ times and within the for loop a while loop is continued at most $d$ times. The total time complexity is $(d+(d-1)d)$. The worst case time complexity to get best pair is $O(d^3 m)$. Hence selecting $m$ best pairs time complexity is $O(e^d m)$, $d << m$.

IV. EXPERIMENTAL STUDY

In our experimental study, we have evaluated the performance of FM algorithm and our proposed FM_variant algorithm by computing Ncut. The above two algorithms are tested using two large datasets called as ISPD98 and ISPD99 benchmark circuits. In our experiments we randomly select a component when more than one component has the same gain value.

A. Experimental Setup

The ISPD98 circuit benchmark is the largest dataset which is maintained by the Collaborative Benchmarking Laboratory. The ISPD98 circuit benchmark contains 18 types of files. These are IBM01 to IBM18. Each file comes with three formats such as .net, .are and .netD. Another version of ISPD benchmark circuit is ISPD99. This benchmark circuit contains 9 types of file. Each file having 4 version with .netD or .are format. These dataset are freely in the website http://vlsicad.ucsd.edu/UCLAWeb.html

C. Experiments Performed

We have performed the two experiments using two different datasets ISPD98 and ISPD99. For the experiments we have defined $gain(μ)$ as follow.

$$gain(μ) = \frac{\text{optimal } N_{\text{cut}}(\text{FM}) - \text{optimal } N_{\text{cut}}(\text{FM}_\text{variant})}{\text{optimal } N_{\text{cut}}(\text{FM})} \times 100$$

The Performance of FM_variant is observed to be better if the gain is higher and positive. In the first experiment we have computed the optimal Ncut of FM and our proposed variant of FM algorithm by considering ISPD98 as input dataset and compared the optimal Ncut of both the algorithms. In the second experiment, we have computed optimal Ncut of FM and our proposed variant of FM algorithm by taking ISPD99 as input dataset and compared the optimal Ncut of both the algorithms.

EXPERIMENT-1: ISPD98 AS INPUT DATASET

In this Experiment we have considered eighteen different files of ISPD98 benchmark circuit. We have computed the optimal
N_{cut} of FM and our proposed variant of FM algorithm as shown in Table I.

**EXPERIMENT-2: ISPD99 AS INPUT DATASET**

In this experiment we have taken nine different files of ISPD99 benchmark circuit and each file having 4 different versions. We have computed the optimal N_{cut} of FM and our proposed variant of FM algorithm as shown in Table II.

For experiment-1 and experiment-2 we plot the graph by considering file’s name of dataset in the X-axis and gain(μ) in Y-axis as shown in fig.8 and fig.9 for ISPD98, ISPD99 respectively.

**TABLE I. FM VS FM VARIANT OF ISPD98**

| File name | Initial N_{cut} | optimal N_{cut}(FM) | optimal N_{cut}(FM_variant) | Gain (μ) |
|-----------|-----------------|---------------------|----------------------------|----------|
| IBM01     | 9151            | 1534                | 858                        | 44.06    |
| IBM02     | 13443           | 1595                | 529                        | 66.8     |
| IBM03     | 17422           | 4013                | 2885                       | 28.1     |
| IBM04     | 20643           | 4327                | 1016                       | 76.5     |
| IBM05     | 18895           | 6881                | 3402                       | 50.5     |
| IBM06     | 22798           | 5721                | 1475                       | 74.2     |
| IBM07     | 32044           | 7028                | 2516                       | 64.2     |
| IBM08     | 33499           | 9242                | 3321                       | 64.06    |
| IBM09     | 40173           | 10438               | 2809                       | 73.08    |
| IBM10     | 50647           | 10413               | 2454                       | 76.43    |
| IBM11     | 54221           | 12893               | 4086                       | 68.3     |
| IBM12     | 52102           | 14508               | 4312                       | 70.27    |
| IBM13     | 23076           | 5275                | 1159                       | 78       |
| IBM14     | 101990          | 22990               | 11257                      | 51.04    |
| IBM15     | 125878          | 29037               | 15149                      | 47.82    |
| IBM16     | 129985          | 37057               | 7268                       | 80.4     |
| IBM17     | 131364          | 42226               | 10062                      | 76.2     |
| IBM18     | 139169          | 36949               | 3055                       | 91.7     |

**TABLE II. FM VS FM VARIANT OF ISPD99**

| File’s Name | Initial N_{cut} | optimal N_{cut}(FM) | optimal N_{cut}(FM_variant) | Gain (μ) |
|-------------|-----------------|---------------------|----------------------------|----------|
| IBM01A      | 9213            | 2148                | 364                        | 83.1     |
| IBM01B      | 4958            | 685                 | 124                        | 82       |
| IBM01C      | 4878            | 800                 | 495                        | 38.13    |
| IBM01D      | 1268            | 4985                | 295                        | 76.74    |
| IBM06A      | 5147            | 22889               | 1328                       | 74.2     |
| IBM06B      | 2565            | 9962                | 948                        | 63.41    |
| IBM06C      | 3044            | 14558               | 963                        | 68.4     |
| IBM06D      | 2360            | 8693                | 829                        | 65       |
| IBM09A      | 9966            | 40187               | 3026                       | 69.63    |
| IBM09B      | 8064            | 33104               | 3830                       | 52.51    |
| IBM09C      | 8022            | 36086               | 4706                       | 41.34    |
| IBM09D      | 8343            | 33809               | 2674                       | 67.95    |
| IBM10A      | 11989           | 50751               | 2865                       | 76.1     |
| IBM10B      | 4259            | 20021               | 2551                       | 40.1     |
| IBM10C      | 7819            | 32876               | 2322                       | 70.3     |
| IBM10D      | 4938            | 21601               | 1394                       | 71.77    |
| IBM11A      | 12833           | 54079               | 4408                       | 65.65    |
| IBM11B      | 5701            | 27379               | 3006                       | 47.27    |
| IBM11C      | 7613            | 29364               | 2744                       | 63.95    |
| IBM11D      | 5471            | 24600               | 3219                       | 41.16    |
| IBM12A      | 13339           | 51921               | 3671                       | 72.48    |
| IBM12B      | 6981            | 29498               | 2755                       | 60.53    |
| IBM12C      | 5533            | 25109               | 2018                       | 63.53    |
| IBM12D      | 6805            | 22915               | 1841                       | 72.94    |
| IBM13A      | 15645           | 66251               | 3197                       | 79.56    |
| IBM13B      | 6813            | 32990               | 499                        | 92.67    |
| IBM13C      | 4461            | 35728               | 1427                       | 68       |
| IBM13D      | 6766            | 31019               | 2019                       | 70.16    |
| IBM16A      | 36869           | 129950              | 6844                       | 81.44    |
| IBM16B      | 19083           | 72220               | 2653                       | 86.1     |
| IBM16C      | 18269           | 61793               | 3047                       | 83.32    |
| IBM16D      | 14341           | 48717               | 3084                       | 78.5     |
| IBM17A      | 43544           | 131753              | 9609                       | 77.93    |
| IBM17B      | 24782           | 74174               | 2186                       | 91.17    |
| IBM17C      | 18965           | 62567               | 4157                       | 78.1     |
| IBM17D      | 11707           | 41472               | 3802                       | 67.52    |

**Fig.8 FM VS FM VARIANT OF ISPD98**

**Fig.9 FM VS FM VARIANT OF ISPD99**
V. CONCLUSION AND FUTURE WORK

In this work we have proposed the variant of FM algorithm using a Pair-wise Swapping technique. We have conducted an experimental study to evaluate the performance of our proposed algorithm and FM algorithm by considering two input datasets such as ISPD98 and ISPD99 benchmark circuits. From experimental result, we observed that our proposed algorithm outperforms FM algorithm.

In future work, we can consider and apply FILO technique for selections of components from gain bucket in our proposed algorithm and compare its performance with FM-LIFO [13]. As reported in [13], FILO technique provides better result than random and FIFO technique. Our proposed variant of FM algorithm can also be enhanced by using multi-level technique.

REFERENCES

[1] B.W Kernighan and S. Lin, “An Efficient Heuristic procedure for partitioning Graphs”, The Bell system technical journal, Vol.49, Feb. 1970, pp. 291-307.
[2] N. A. Sherwani, “Algorithms for VLSI Physical Design Automation”, Springer, 3rd Ed., 1999.
[3] C.J. Alpert, A. B. Kahng, “Recent directions in netlist partitioning: a survey”, Elsevier the VLSI Integration Journal, Vol.19, 1995, pp. 1-81.
[4] M. R. Garey and D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NPCompleteness,”San Francisco, CA: Freeman, 1979.
[5] C.M. Fiduccia and R.M. Mattheyses, “A Linear Time Heuristic for improving Network Partition” 19th Design Automation Conference, 1982, pp. 175-181.
[6] S. Dutt, “A New Faster Kernighan-Lin Type Graph Partitioning Algorithm”, In proc. IEEE Intel Conf. computer Aided Design, 1993, pp. 370-377.
[7] B.W Kernighan and D.G schweikert, “A proper Model for Partitioning the electrical circuit” proc. 9th Design Automation workshop, Dallas, TX, June 1979, pp. 57-62.
[8] B. Krishnamurthy, “An Improved Min-cut Algorithm For Partitioning VLSI Network”, IEEE Transactions on computers, Vol. C-33, May 1984, pp. 438-446.
[9] L. A. Sanchis, “Multi-way Network Partitioning”, IEEE Transaction on Computers, Vol. 38, No. 1, January 1989, pp. 62-81.
[10] A. G. Hoffmann, “The Dynamic Locking Heuristic-A new Graph Partitioning algorithm”, In Proc. IEEE Intel Symp. Circuit and Systems, 1994, pp. 173-176.
[11] L.W. Hagen , D.J. Haung, A.B Kahng, “On Implementation choices for iterative improvement partitioning algorithms”, European Design Automation Conference, 1995, pp. 144-149.
[12] T. Bui’, C. Heigham, C. Jones and T. Leighton, “Improving the performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithm”, 26th ACM/IEEE Design Automation Conference, 1989 pp. 775-778.
[13] L. Hagen and A.B. Kahng, “A new approach to effective circuit clustering”, Proc. of IEEE International Conference on Computer Aided Design, 1992, pp. 422-427.
[14] H. Shin, and C. Kim, “A Simple Yet effective technique for partitioning”, IEEE Transactions On Very Large Scale Integration (VLSI) SYSTEMS, Vol. 1, No. 3, September 1993, pp. 380-386.
[15] Cong and smith, “A parallel bottom-up clustering algorithm with application to VLSI partitioning”, In Proc. ACM/IEEE Design Automation Conference, 1993, pp. 755-760.
[16] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, “Multi-level hyper graph partitioning algorithm : Application in VLSI domain“, 34th Design Automation Conference, Vol. 7, No.1, 1997.
[17] C.J. Alpert, J.H. Huang, and A.B. Kahng, “Multi-level circuit partitioning” IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems, Vol. 17, No. 8, 1998.
[18] Y. Saab, “A new 2-way multi-level partitioning algorithm”, VLSI Design Overseas Publishers Association, Vol. 2, No. 3, 2000, pp. 301-310.
[19] G. Karypis, V. Kumar, “Multi-level K-way hyper graph partitioning”, VLSI Design Overseas Publishers Association, 2000, pp. 1–16.
[20] J. Cong and S.K. Lim, “Edge separability based circuit clustering with application to multi-level circuit partitioning”, IEEE Transactions On Computer-Aided Design Of Integrated Circuits and Systems, Vol. 23, No. 3, 2004.
[21] Chan-Ik-Park and Yun-Bo Park, “An Efficient Algorithm for VLSI Network Partitioning Problem using a cost function with balancing factor”, Computer Aided Design of Integrated Circuits and Systems, IEEE Transactions, Vol. 12, No. 11, August 2002, pp 1686-1694.