Hierarchical Partitioning for Field-Programmable Systems[†]

Vi Chi Chan[*] and David Lewis
Department of Electrical and Computer Engineering, University of Toronto
billy@lsil.com, lewis@eecg.toronto.edu

Abstract

This paper presents a new recursive bipartitioning algorithms targeted for a hierarchical field-programmable system. It draws new insights into relating the quality of bipartitioning algorithm to circuit structures by the use of the partitioning tree [11]. The final algorithm proposed not only forms the basis for the partitioning solution of a 1-million gate Field Programmable System [1] but can also be applied to general VLSI or multiple-FPGA partitioning problems.

1.0 Introduction

The reprogrammability of FPGAs has made possible a number of systems for rapid prototyping and emulation. These multiple-FPGA designs, primarily aimed at ASIC applications, tend to be severely pin limited. Since the pin constraints of the FPGA are hard limits, and low in terms of the number of pins per device, high quality partitioning is necessary to achieve acceptable utilization of the FPGA. Most common methods, based on Fiduccia-Mattheyses (FM) [2] cannot achieve reasonable utilization.

This paper addresses hierarchical partitioning for Field Programmable (FP) systems, with particular regard to the Transmogrifier-2 (TM-2) [1]. This paper focuses on bipartitioning with special attention to the quality of overall partitioning result across all levels of recursive bi-partitioning, rather than a single bi-partitioning. The algorithm devised is also a high quality bi-partitioner in its own right. The specific techniques introduced in this paper are:

(1) A new adaptive FM iterative partitioning procedure.

(2) The use of spectral partitioning methods to prune the network to exploit circuit structure and remove nets that would otherwise distort the partitioning.

(3) The use of the partitioning tree to explain the effectiveness of FM, KL-FM [3] in terms of circuit structure.

† This work was supported by Micronet, Canada.

* Now with the Toronto Design Centre, LSI Logic Corp. of Canada.

Our implementation of this algorithm is 10-50% more effective than previously published bipartitioners.

Section 3 of this paper discusses background information of this research. Section 4, 5 and 6 present the above mentioned research while Section 7 presents the experimental results. Section 8 concludes with summary and suggestion for future work.

2.0 Previous Work

Traditional partitioning schemes can be divided into two major categories: iterative or analytical.

The most commonly used iterative method is Fiduccia-Mattheyses (FM) [2] partitioning. This method consists of multiple *passes*. In each pass, random partitions are improved iteratively by cell movements from one partition to the other until no further improvement in cutset size is observed.

Many passes (each begin with a unique random starting point) are usually needed to produce good results, especially when circuits are large (as random partition assignment cannot sufficiently cover the solution space). Out of the many passes, only the best pass results are saved. Information captured by all the other passes is lost.

Many modifications/improvements of the FM method are available. Krishnamurthy [3] suggested the use of gain vectors instead of single gain values to reduce ambiguity when choosing cells to move; Hagen et. al [4] introduced locality in cell selection by using LIFO queue as data structure instead of simple linked list. Although each has its own benefit, neither of them improves partitioning result significantly, largely because they do not introduce any means for the algorithm to climb over steep hills of valleys where local minima reside.

Another partitioning algorithm suggested by Kernighan and Lin [5] is more successful in improving the results of FM[R]. The KLFM algorithm divides the partitions $\{A, B\}$ generated by FM into 4 sub-partitions

$\{A_0, A_1, B_0, B_1\}$ by applying FM partitioning to A and B. These 4 sub-partitions are swapped an merged to form new initial partitions $\{C, D\}$ for another pass of FM, i.e. $C = A_0 \cup B_0$, $D = A_1 \cup B_1$ or $C = A_0 \cup B_1$, $D = A_1 \cup B_0$. This procedure of sub-dividing, swapping and re-partitioning is continued until no further improvement on the cutset is observed.

This method produces much better results per iteration because it explores different combinations of sub-partitions to form initial partitions, instead of just a single trial of random partition assignment. By doing so, this method is effectively exploring different way to construct the partitioning tree at the top level (as can be seen later in this paper).

The other major partitioning technique is analytical partitioning. Among different methods in this area, the spectral method (eigenvector embedding [6]) has received much attention recently. In this method, the circuit is represented in an hypergraph from which a symmetric non-negative definite matrix (the Laplacian $Q = D - A$) is derived (after transforming multi-point nets into cliques). The eigenvalue of Q can be expressed as $\lambda = \frac{x'Qx}{|x|^2}$, where x is the corresponding eigenvector. If we consider x as a one-dimensional mapping of the cells in the circuits, the quantity $x'Qx$ is equivalent to $\sum_{(v_i, v_j) \in E_H} (x_i - x_j)^2$, the sum of the quadratic distances between connected cells. For a normalized eigenvector x, $\lambda = x'Qx$. Therefore, the smaller the λ, the smaller the sum of the quadratic distances between connected cells. In a connected graph, this quantity can be used as a heuristic measure of placement or partitioning quality, i.e. by minimizing this sum of quadratic distances, the distances between heavily connected cells should also be minimized, yielding a natural clustering of cells from which good quality partitions can be determined. Since Q has 0 as its smallest eigenvalue, the first non-trivial solution to this minimization problem is thus the second smallest eigenvalue of Q and its corresponding eigenvector.

Although popular, the spectral method has been shown [10][12] to be sensitive to changes in circuit structures. This result, however, also implies that if only we can choose the information processed by the analytical method (selecting certain connections between cells and mask out others), we can target specific area of a circuit for partitioning.

Our interest in partitioning is motivated in particular by a field-programmable rapid-prototyping called the Transmogrifier-2 (TM-2)[1]. This system contains a number of FPGAs, up to 32 in a full sized system, and a collection of partial crossbars providing interconnected between the FPGAs. At a conceptual level, there are up to 5 levels of crossbar, where level i provides interconnect between a collection of 2^i FPGAs. An overview of the architecture is given in Fig. 2.1, and for full details consult ref. [1].

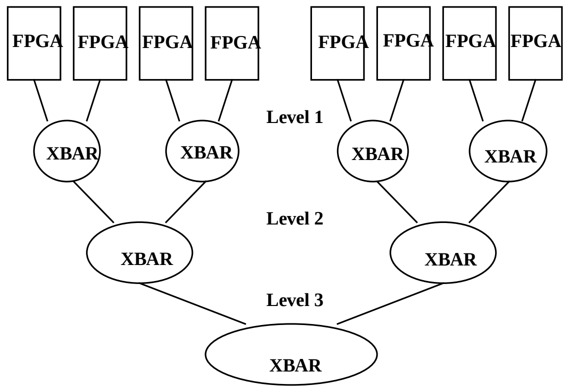

Figure 2.1 Hierarchical model of the TM-2 routing architecture

3.0 Algorithms

This section of the paper describes our algorithm development. There are three main features of our algorithm.

3.1 Adaptive FM

In the traditional FM procedure a large number of passes is usually required to obtain a good result; several hundred passes can be required for large circuits.

To avoid this disadvantage, a new method of retaining information from each of the partitions generated by FM is proposed. This method adaptively produces new initial partitions using previous good partitioning results. For each pass, an initial placement vector is constructed, using a weighted placement value that depends on all previous partitioning results. Each gate is initially assigned to the left or right side depending on the weighted average of its final position at the end of previous passes. The weighting factor is based on the quality of the final result in each previous pass, so that good results are weighted higher than poor results. The result of each pass is only used if the cutset size is within 20% of the best cutset size seen so far. A minor complication occurs in that left and right are arbitrary partitions (i.e. the partition with each gate swapped from left to right is identical in cost, but appears to be the exact opposite partition.) The algorithm must optionally swap the partitioning to determine the orientation that is most similar to the previous partition. This is done by comparing the number of cells that have the same orientation as in the result of the best pass so far. If this is less than half, the left and right partitions are swapped before they are used to calculate their contribution to the initial weighting of the next pass.

(1) Generate initial partitions $\{A_0, B_0\}$ by a random procedure followed by FM, with cutset size c_0 and scaling factor $s_0 = 1/c_0$. Generate the orientation vector $P_0 = \{p_0, p_1, \ldots, p_n\}$ where $p_i = -1$ iff $v_i \in A_0$, and $p_i = 1$ iff $v_i \in B_0$. From P_0 and s_0, derive a weighted placement vector: $X = \{x_0, x_1, \ldots, x_n\}$, where $x_i = p_i \times s_0$. Record the best cutset and the best orientation vector (partition assignments) as c_{best} and P_{best}.

(2) Set $k = 1$. Generate a new partition $\{A_k, B_k\}$ as in (1).

(3) If $0.8 \le \frac{c_k}{c_{best}} \le 1.2$, goto step (4) else (5).

(4) Compare the orientation vector P_k with P_{best}. If the majority of partition assignments disagree between P_k and P_{best} (i.e. the sum of $xor(P_k, P_{best}) > n/2$) then flip the orientation of P_k. Update the placement vector X: $x_i = x_i + (s_1 \times p_i)$, where p_i is the i-th element of the adjusted P_k. If $c_k < c_{best}$, update $P_{best} = P_k$ and $c_{best} = c_k$, goto step (6).

(5) $k = k + 1$, if $k > k_{max}$, goto (7); else produce new partition $\{A_k, B_k\}$ as in (1), goto step (3).

(6) Sort the placement vector X to produce a one-dimensional embedding of cells. The optimal cut point is determined within the cutratio constrain, forming partitions $\{A_{k+1}, B_{k+1}\}$. FM is then applied to give $\{A_{k+1}, B_{k+1}\}$, Set $k = k + 1$. if $k > k_{max}$, goto step (7), else goto step (3).

(7) The best partition result P_{best} is restored as the output. End.

This algorithm is graphically represented in Figure 3.1.

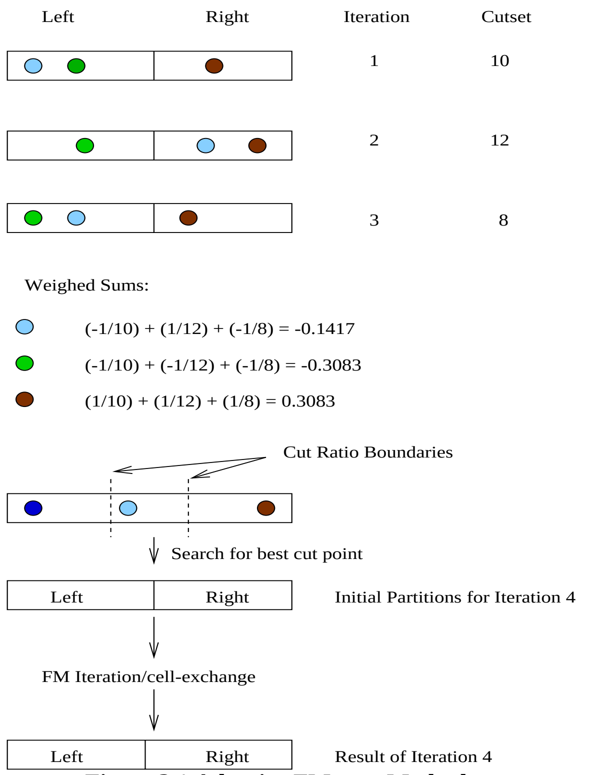

Figure 3.1 Adaptive FM pass Method

Experiments were carried out to compare this new method with that of traditional random initial placement scheme. These experiments estimated the number of passes required to achieve an acceptable result, defined as within 5% of the best that can be achieved. Since the best possible is not known, we used FM for 200 passes and adopted this as an estimate of the best possible. The first acceptable pass is define as the first pass of FM that comes within 5% of the final result. On average, it takes 83 passes of FM to come within 5% the 200 pass result. The adaptive method achieves cut counts 23% smaller, and takes only 35 passes on average. The reduction in CPU time is much more dramatic because later FM passes converge much more quickly to their final result. CPU time is reduced 92%. The results are summarized in Table 1.

3.2 Partitioning Tree Based Placement

The adaptive FM method shows that partitioning results can be significantly improved if more information about the circuit is gathered. In fact, as circuits grow larger and more complex, circuit structure information will be more important to the success of partitioning of the circuit.

The partitioning tree (Figure 3.2) is a graphical repre-

Table 3.1 Adaptive FM vs. Conventional FM

Test Circuits	Size - # of Logic Gates	Conventional FM			Adaptive FM			% Improvement on Cutset Size
		First Acceptable Pass	Best Cutset Size	CPU Time (sec.)	First Acceptable Pass	Best Cutset Size	CPU Time (sec.)	
balu	801	183	84	19.33	183	84	19.45	0.00%
primary1	833	24	58	2.87	24	58	3.90	0.00%
bm1	882	166	60	31.63	9	76	0.97	-26.67%
test4	1515	18	132	5.80	98	140	29.63	-6.06%
test3	1607	22	109	10.23	60	115	27.64	-5.50%
test2	1663	66	186	23.02	40	183	13.28	1.61%
test6	1752	150	108	65.51	150	108	61.53	0.00%
struct	1952	18	162	8.85	10	46	3.73	71.60%
test5	2595	38	273	31.82	9	251	7.97	8.06%
19ks	2844	78	244	145.79	6	160	9.42	34.43%
primary2	3024	66	314	162.11	9	173	10.42	44.90%
s9234	5866	36	312	213.29	7	212	34.58	32.05%
biomed	6514	179	364	1303.78	6	182	32.40	50.00%
s13207	8772	79	311	1069.42	8	163	121.06	47.59%
s15850	10470	30	481	526.38	7	316	111.63	34.30%
industry2	12637	177	609	7415.62	8	224	169.56	63.22%
s35932	18148	7	1103	669.27	4	871	239.64	21.03%
s38584	20995	100	1751	8812.52	30	1327	1841.39	24.21%
s38417	23949	147	1082	23493.47	8	524	899.39	51.57%
avg.		83	7743	2316.35	35	5213	191.45	23.49%

sentation of circuit structure as affected by recursive bipartitioning process. It has been used in placement algorithm or the estimation of circuit parameters [11], but seldom has it been used to motivate the partitioning algorithm before.

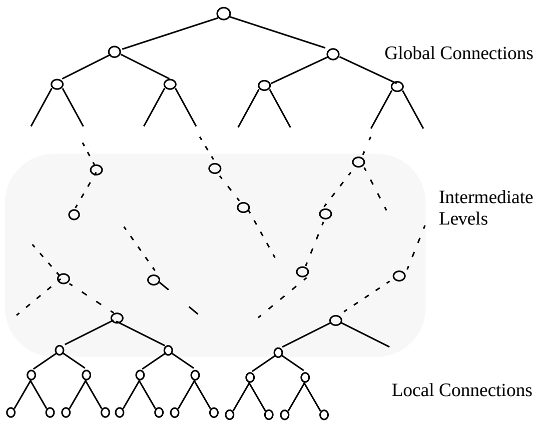

Figure 3.2 Partitioning Tree

When considering partitioning algorithms, especially algorithms which involve exchanging cells or groups of cells, the partitioning tree can be used to demonstrate the effectiveness in exploring circuit structure by such algorithms, e.g.:

(1) KL-partition exchange scheme exchanges sub-partitions at the top of the tree. Branches at the top 2 levels are affected the most.

(2) FM cell exchanges swap cells individually. The effect of cell exchanges propagate up from the leaves of the tree.

The GORDIANL [9] based bipartitioning algorithm PARABOLI [8] recursively bipartitions down to single cell, effectively traversing the entire partitioning tree and building it from bottom up to find the best bipartition point for the circuit.

The traditional partitioning scheme of KLFM (KL-partition exchange followed by FM passes) attempts to explore different combination of the upper level branches while allow the cell movement at the bottom to refine the results. For small circuits, such an approach can sufficiently cover the entire partitioning tree structure. For large circuits, a considerable subset of the middle portion of the circuits is not explored (since the effect of FM cell movement weaken as it propagates upward, affecting fewer upper level branches).

3.3 New Bipartitioning Algorithm

From the above section, it is clear that a good partitioning algorithm must be able to extract information from the entire partitioning tree to be effective. However, little is known about how the middle portion of the partitioning tree can be explored.

Considering the spectral method, we realize that since the entire circuit is transformed into the Laplacian matrix Q, any connectivity information must also be stored in the matrix and partitions generated by the spectral method will have information related to the middle portion of the partitioning tree too. Therefore, our first attempt is to combine the KLFM algorithm with the spectral method, forming a new algorithm called KLEFM.

The KLEFM algorithm uses the spectral method to produce an initial partition (by determining cut point in the one-dimensional embedding generated by the eigenvector) which is then processed by one pass of KLFM. The results are encouraging (Table 3.2): on average, it is 55% better than 20 passes of FM and 40% better than EIG1 [6].

Table 3.2 Comparison of Adaptive FM to Eigenvalue and KLFM

	Cut Count		
Circuit	**20 Adaptive FM Passes**	**EIG1**	**KLEFM**
19ks	251	179	141
bm1	64	75	61
primary1	59	75	52
primary2	314	254	161
test02	186	196	109
test03	109	85	60
test04	132	207	57
test05	279	167	107
test06	134	295	67
balu	89	110	39
struct	162	49	54
biomed	394	286	153
s13207	311	110	89
s15850	481	125	72
industry2	700	525	384
Total	**3665**	**2738**	**1655**

However, straight-forward application of the spectral method is not desirable because of it is sensitive to circuit structure. In another experiment [12], a small number (< 5%) of nets were removed from test circuits. The spectral method shows large variance in the partition results over small changes in circuit structure. Because we are only interested in using the spectral method to extract the information related to medium connectivities, feeding the spectral method with the entire circuit seem unnecessary. In light of the fact that spectral method is sensitive to changes in circuit structure, it is also possible that by giving the spectral method the entire netlist, some connectivity information at the global or local levels would interfere with our objective. If we can select connectivities related only to the middle portion of the partitioning tree, we can use the spectral technique to produce partition that only reflects the circuit information in that area of the partitioning tree.

The middle portion of the partitioning tree represents cells that are connected in medium connectivities (i.e. not too locally nor globally connected). The spectral technique produces a linear embedding of cells related to the quadratic distances between connected cells. It is a heuristic measure of how the cells should be placed to minimize the sum of the quadratic distances between connected cells. Thus, it is reasonable to assume that cells that are placed close together have local connectivities and that cells placed far apart have global connectivities.

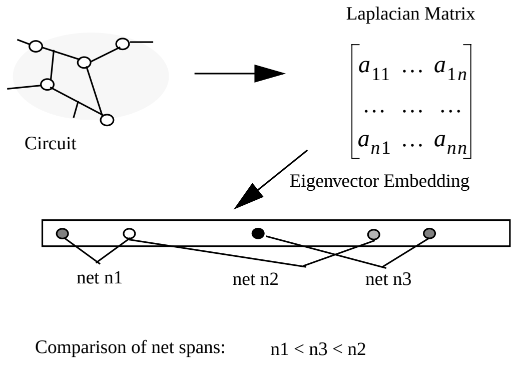
Laplacian Matrix
$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$
Circuit
Eigenvector Embedding
net n1
net n2
net n3
Comparison of net spans: n1 < n3 < n2

Figure 3.3 Eigenvector Embedding and Net Spans

We define the *span* of a net to be the distance (in the 1-dimensional embedding) between the farthest cells connected by the net. The assumption made earlier allows us to remove nets with extremely large or extremely small spans, in order to extract nets with medium connectivities. By reconstructing a new graph with only those nets and partition it with the spectral method, we screen out the global and local portions of the nets from the circuit, allowing the spectral method to just focus on the medium connectivity information. The algorithm is thus as follows:

(1) Transform circuit into graph $G(V, E)$ using a clique model.

(2) Generate the Laplacian matrix Q and extract the 2nd smallest eigenvalue/vector pair (λ, x).

(3) For each net n_i, calculate the span $s_i = max(x_{i0}, \dots, x_{ik}) - min(x_{i0}, \dots, x_{ik})$, where $x_{i0}, \dots, x_{ik}$ are coordinates of cells connected to n_i.

(4) Sort vector $S = \{s_o, \dots, s_N\}$.

(5) Remove the top 5% and bottom 5% of the sorted list, which represent the global and local connections of the original circuit.

(6) Construct a new graph $G'(V', E')$ with the remaining nets. Repeat step (2) for this new graph.

(7) Partition the eigenvector embedding as in the spectral method.

(8) The partition is then processed by KLFM.

(9) 20 passes of Adaptive FM is the followed to refine the result.

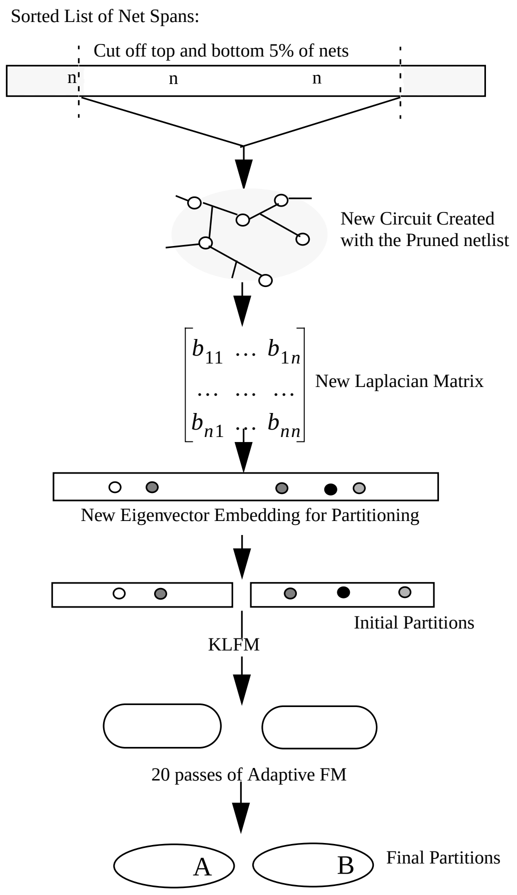

Figure 3.4 Hierarchical Partitioning Algorithm

Figure 3.4 shows the algorithm flow while Table 3.3 presents the experimental results. On average, the proposed algorithm out-performs EIG1, PARABOLI [8] and MELO [7] by 50%, 17% and 10% respectively.

4.0 TM-2 specific algorithms

Several other changes were also implemented to tailor the proposed algorithm to the TM-2 routing structure:

(1) The original FM cost function (simple cutset), was replaced by a pin usage cost function. In multiple FPGA systems, a wire between 2 FPGAs introduces two pins (one for each FPGA) while a primary IO introduces one pin. When nets divided at the top level of the hierarchy is further divided in the lower level, more and more pins are introduced. At the lower level, the simple cutset cost function no longer reflects the wire resources usage accurately. To accurately model the FPGA pin cost, a pin usage gain function was developed for the FM procedure, outlined as follows. An appropriate gain update function is also necessary, and is described in [12].

```
Pinuage Gain Calculation:
Given: $c$ - any given cell in the circuit
    $N = \{n_0, \dots, n_k\}$ - nets connected to $c$
    $P = \{p_0, \dots, p_m\}$, I/O pins connected to $c$
    $A$ - partition contains $c$, $B$ - the opposite partition
    $U_A$ and $U_B$ are number of pins in $A$ and $B$
    $\wp_{max}$ - set of partitions that have the highest pin usage
Output: $g$ - the gain of cell c
begin
    $g \leftarrow 0$
  Use original FM gain calculation to calculate $g$
  for all $p_i$ in $P$, begin
    group cells connected to $p_i$ into $\{\wp_A, \wp_B\}$, where $\wp_A$
    contains cells in $A$, $\wp_B$ is defined similarly.
    if ( $|\wp_A| = 1$ and $A \in \wp_{max}$ ) then
      if ( $|\wp_B| > 0$ or ( $|\wp_B| = 0$ and $U_A > U_B + 1$)) then
         $g \leftarrow g + 1$
      end if
    end if
    if ( $|\wp_B| = 0$ and $B \in \wp_{max}$ ) then
       $g \leftarrow g - 1$
    end if
  end for
end
```

Table 3.3 Comparison between EIG1, PARABOLI, MELO and the Proposed Algorithm

	Cut Count				% Improvement Compare to:		
Circuit	**EIG1**	**PAR**	**MELO**	**Our Algo.**	**EIG1**	**PAR**	**MELO**
19ks	179	-	119	129	27.93%	-	-8.40%
bm1	75	-	48	47	37.33%	-	2.08%
prim1	75	53	64	49	34.67%	7.50%	23.44%
prim2	254	146	169	160	37.01%	-9.50%	5.33%
test02	196	-	106	100	48.98%	-	5.67%
test03	85	-	60	66	22.35%	-	-10.00%
test04	207	-	61	69	66.67%	-	-13.11%
test05	167	-	102	90	46.11%	-	11.76%
test06	295	-	90	66	77.63%	-	26.67%
balu	110	41	28	27	75.46%	34.15%	3.57%
struct	49	40	38	36	26.53%	10.00%	5.26%
biomed	286	135	115	83	70.98%	38.52%	27.83%
s9234	166	74	79	49	70.48%	33.78%	37.97%
s13207	110	91	104	75	31.82%	17.59%	27.88%
s15850	125	91	52	55	56.00%	39.56%	-5.77%
ind2	525	193	319	224	57.33%	-16.06%	29.78%
TOTAL	**2904**		**1554**	**1324**	**avg. 49.30%**	**avg. 17.28%**	**avg. 10.62%**

(2) A post-processing stage was also introduced to route the crossbar structure to remove congestion in any overly crowded crossbars (see [12]). Because of the implementation of the TM-2 [1], higher level crossbars can be used to route lower level connections, but not vice versa.

(3) A balancing scheme was devised to determine the appropriate cut-ratios at each level of bi-partition. This scheme dynamically determines the cut-ratio to use at each level by considering the cutset produced and the number of levels below the hierarchy, trying to avoid congestion at the lower level induced by unbalanced top level partitions. The detail of this scheme can also be obtained from ref. [12].

5.0 Experimental Results

The new hierarchical partitioning algorithm was applied to all the large (>1500 4-LUTs) circuits in the MCNC LGSynth93 benchmark. Since there are multiple levels of wiring resources, the results have been graphed and compared on each of the levels. Figure 4.1 shows a comparison of the routing resources required the top level of an 8-FPGA system. It compares recursive FM to our algorithm since no direct comparison could be made with other published results. Because it is difficult to find circuits of sufficient size to fill a significant proportion of the TM-2, we used a scaled version for the experiments. This defines a boundary of feasible systems based on hypothetical FPGAs with pin counts that scale as $\sqrt{n}$, where n is the number of gates per FPGA. The partitioner is also capable of using the capacity of the FPGAs to determine the utilization necessary at each stage in partitioning. As the required utilization of the FPGAs decreases, the imbalance of partition sizes increases, causing lower wire counts, but requiring larger FPGAs. Each curve in the plot shows the cut count as the utilization of the FPGAs is varied from 80% down to 20%. The solid lines show results for our algorithms and the dashed for recursive FM. The solid curve at the lower portion of the graph shows the TM-2 feasible boundary (the $\sqrt{n}$ curve). As FPGA sizes increases, the partitioner has more freedom to perform better partitioning, thus the solid lines drop down below the feasible boundary, indicating that the design is implementable with the TM-2 structure. From this graph, we can see clearly that FM fails to produce partitions that are feasible under the TM-2 architecture. 10 out of the 11 test circuits could be mapped into a scaled 8-FPGA TM-2 system with utilization ranging from 20 to 80%. If recursive FM is used, only 2 out of the 11 circuits could fit into the scaled system. Comparisons are also made in 4, 16 and 32-FPGA system and the results are presented in [12].

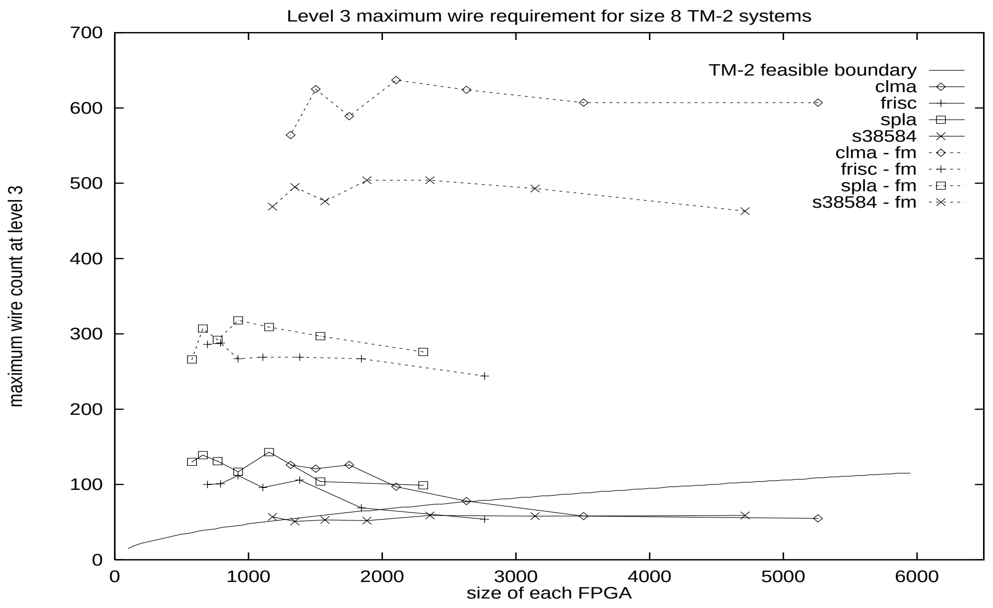

Figure 4.1 Proposed Algorithm vs. FM (8-FPGA System)

6.0 Conclusions

This paper presents a top-to-bottom partitioning scheme for a hierarchical FPGA system. The bi-partitioning procedure, when applied alone, produces results that are 10-50% better than previously published bi-partitioning methods. The effective use and combination of the spectral method and KLFM allows us to explore circuit information at different level of the partitioning tree. When applied to hierarchical system, the proposed algorithm produces more fits (10 vs. 2) than recursive FM procedure, allowing logic utilization to increase while maintaining a feasible wiring requirement.

As circuit designs become more complex and highly integrated, circuit partitioning must look for ways to improve the partition result for placement and routing. This research along with other recent partitioning research [13] show that one of the way to improve circuit partitioning is by extracting information from circuit structure, be it through clustering or through heuristic approach of the partitioning tree. This research also provides insights into partitioning problem relating to circuit structure and hierarchical partitioning.

7.0 References

[1] D.M. Lewis, D.R. Galloway, M. van Ierrsel, J. Rose and P. Chow, “ The Transmogrifier-2: A 1-Million Gate Rapid Prototyping System”, FPGA 97.

[2] C.M. Fiduccia and R.M. Mattheyses, “A Linear-time Heuristic for Improving Network Partitioning”, 19th Design Automation Conference, 1982, pp. 175-181.

[3] B. Krishnamurthy “An Improved Min-Cut Algorithm For Partitioning VLSI Networks”, IEEE Transactions On Computers, Vol. C-33, No. 5., 1984, pp 438-446.

[4] L. Hagen, D. Huang and A. Kahng, “On Implementation Choices for Iterative Improvement Partitioning Algorithms”, 1995.

[5] B.W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs”, Bell System Technical Journal, Vol. 49, No. 2, 1970, pp. 291-307.

[6] L.Hagen and A. Kahng “Fast Spectral Methods for Ratio Cut Partitioning and Clustering”, ICCAD, 1991, pp. 10-13.

[7] C.J. Alpert and S. Yao “Spectral Partitioning: The More Eigenvectors, The Better”, 32nd Design Automation Conference, 1995, pp. 195-200.

[8] B.M. Riess, K. Doll and F.M. Johannes “Partitioning Very Large Circuits Using Analytical Placement Techniques”, 31st Design Automation Conference, 1994, pp 646-651.

[9] G. Sigl, K. Doll, F.M. Johannes “Analytical Placement: A Linear or a Quadratic Objective Function?”, 28th Design Automation Conference, 1991, pp. 427-431.

[10] S. Guattery and G. Miller, “On The Performance of Spectral Graph Partitioning Methods”, 6th Annual ACM/SIAM Symposium on Discrete Algorithms 1995.

[11] L. Hagen, A. Kahng, F. Kurdahi and C. Ramachandran, “On the Intrinsic Rent Parameter and Spectral-based Partitioning Methodologies”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13. No. 1. 1994, pp. 27-37

[12] Vi Chi Chan, “M.A.Sc. Thesis: A Partitioning Algorithm for the Transmogrifier-2”, University of Toronto, 1997.

[13] S. Dutt and W. Deng, “VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques”, ICCAD, 1996, pp. 194-200.