ENISLE: An Intuitive Heuristic Nearly Optimal Solution for Mincut and Ratio Mincut Partitioning

Shun-Wen Cheng and *Kuo-Hsing Cheng*

Department of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien 25137, TAIWAN

Abstract Mincut partitioning is to minimize the total cuts of the edges by the partitioning of nodes into two sets. The proposed method, Edge-Node Interleaved Sort for Leaching and Envelop (ENISLE) algorithm, is not only using node information but also using edge information. It is simple, but works effectively, and never appears in any earlier literature. It can soon get an intuitive heuristic nearly optimal solution for mincut and ratio mincut partitioning at the same time, is very suitable for EDA usage.

Index Term – min-cut partitioning, ratio min-cut partitioning, clustering effect, iterative-improvement, radix sort, IC CAD, EDA, System-on-a-Chip (SoC), VLSI circuit.

I. Introduction

Circuit partitioning plays a key role in the design automation of VLSI chips, multichip systems, and system-on-a-chip (SoC). It is used to reduce VLSI chip area, reduce the component count and the number of interconnects in multiple FPGA implementations of large circuits or systems. Circuit partitioning facilitates efficient parallel simulation of circuits, facilitates design of tests for digital circuits and reduces timing delays, and facilitates the various combinations of sub-system layouts [18].

Iterative-improvement two-way min-cut partitioning is an important stage in most circuit placement tools. Recursive two-way partitioning is an efficient and popular approach to obtaining k-way partitions for k > 2 [5], [11], [20]. So many papers are concern here with the two-way mincut partitioning problem. Since two-way mincut partitioning is NP-complete problem [10], a number of approximate schemes have been proposed. These include iterative improvement methods [7], [8], [13], [15], [16], simulated annealing [17] and clustering-based techniques [3], [11], [20]. Iterative improvement methods are sometimes also used as a preprocessing phase for clustering as in [19], [20].

Most partitioning techniques for circuit netlists like the Fiduccia–Mattheyses (FM) method, computes the gains of nodes using local netlist information that is only concerned with the immediate improvement in the cutset. This can lead to misleading gain information. This matter let us turn back the origin of the mincut partitioning problems and solving them by global viewpoints. The proposed new method in the paper is a totally different method.

Let a circuit be represented by a hyper-graph or netlist $G = (V, E)$ where V is the set of nodes that represent components of the circuit and E the set of hyperedges that represent the nets of the circuit. Each hyperedge or net connects two or more nodes together; generally the output of a node is connected to the inputs of several other nodes by a net.

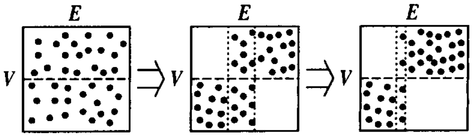


Figure 1. The min-cut partitioning.

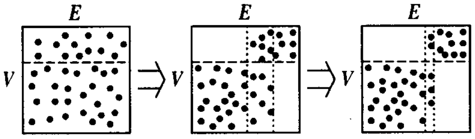


Figure 2. The ratio min-cut partitioning.

The goals of min-cut and ratio min-cut partitioning display on the V-E plain are shown in Fig. 1 and Fig. 2. Our goal is partitioning, not only clustering [1], [9]. Notice when (V, E) pairs approach to the uniformly distribution on the V-E plain, and if we can find the min-cut, the "outline areas" of the (V, E) pairs will from the product VE approach to VE /2, like vapor compression behavior.

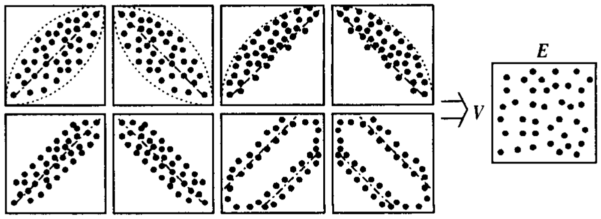


Figure 3. Some special (V, E) distribution cases.

Fig. 3 shows some special (V, E) distribution cases. It may often be seen when describe the row-based placements in sequence. The "outline areas" of (V, E) pairs occupied on the V-E plain are far less than the VE product. Under this condition, the probability that we can get the min-cut solution is relatively small. This is unlike vapor compression behavior, is like "melting" the material. So first we need "heating" it, adding "entropy" to it. And notice these cases ease to be randomized. If it exists local cluster blocks, we only need to resolve the blocks. And some small blocks will be automatically resolved in the proposed method.

This paper is organized as follows. In Section 2, describe the proposed method and demonstrate by examples. In Section 3 we then reveal the importance of the distributed conditions. Finally conclude the major findings and outline the future work.

0-7803-6685-9/01/$10.00©2001 IEEE

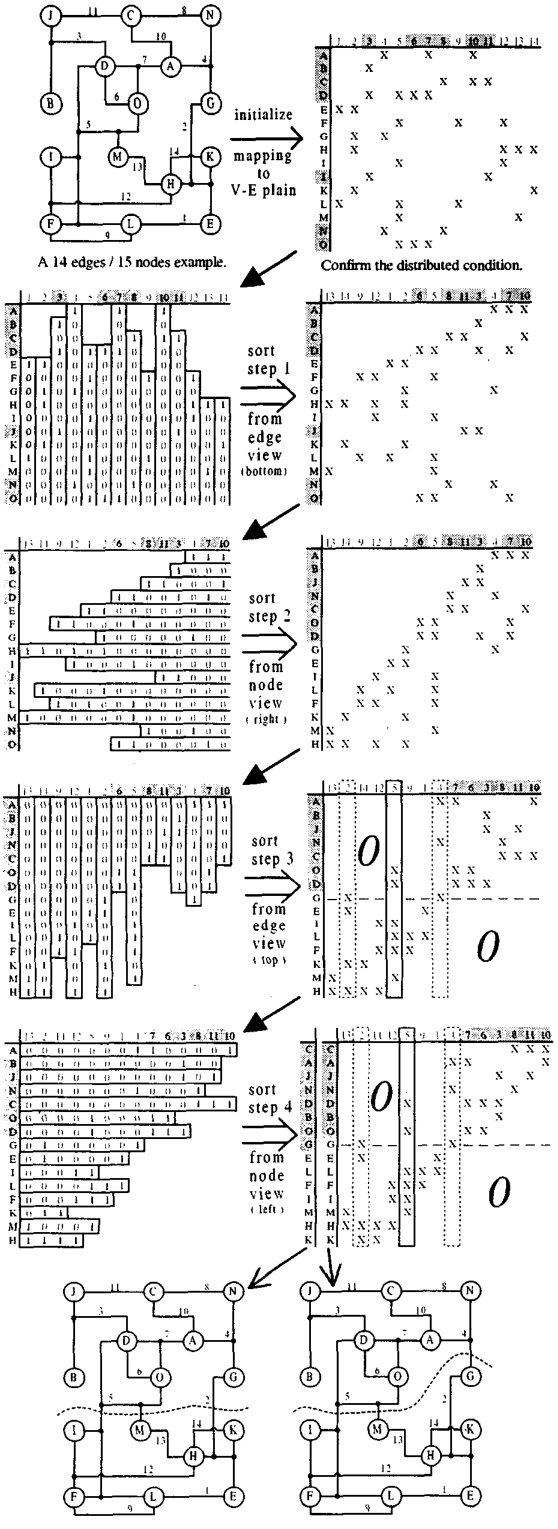
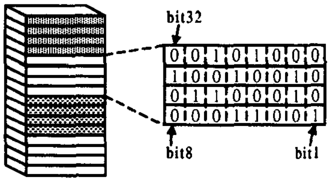
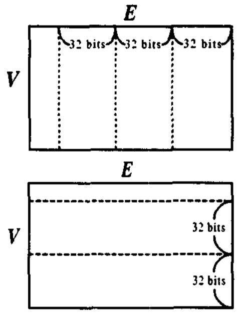


Figure 4. The first four steps (phase one) of the ENISLE algorithm are shown by this example in detail.



```
struct bitfield32 {
  bit32 :1;
  bit31 :1;
  bit30 :1;
  .......
  bit2 :1;
  bit1 :1;
} radix_sort_unit;
```

Figure 5. Using bit field structure to reduce the memory requirement, and accelerate the sort process.



Radix Sorting (LSD) Example:

232, 321, 213, 231, 111, 112, 132, 123, 221
1S → 321, 231, 111, 221
2S → 232, 112, 132
3S → 213, 123

321, 231, 111, 221, 232, 112, 132, 213, 123
10S → 111, 112, 213
20S → 321, 221, 123
30S → 231, 232, 132

111, 112, 213, 321, 221, 123, 231, 232, 132
100S → 111, 112, 123, 132
200S → 213, 221, 231, 232
300S → 321
o/p:111, 112, 123, 132, 213, 221, 231, 232, 321

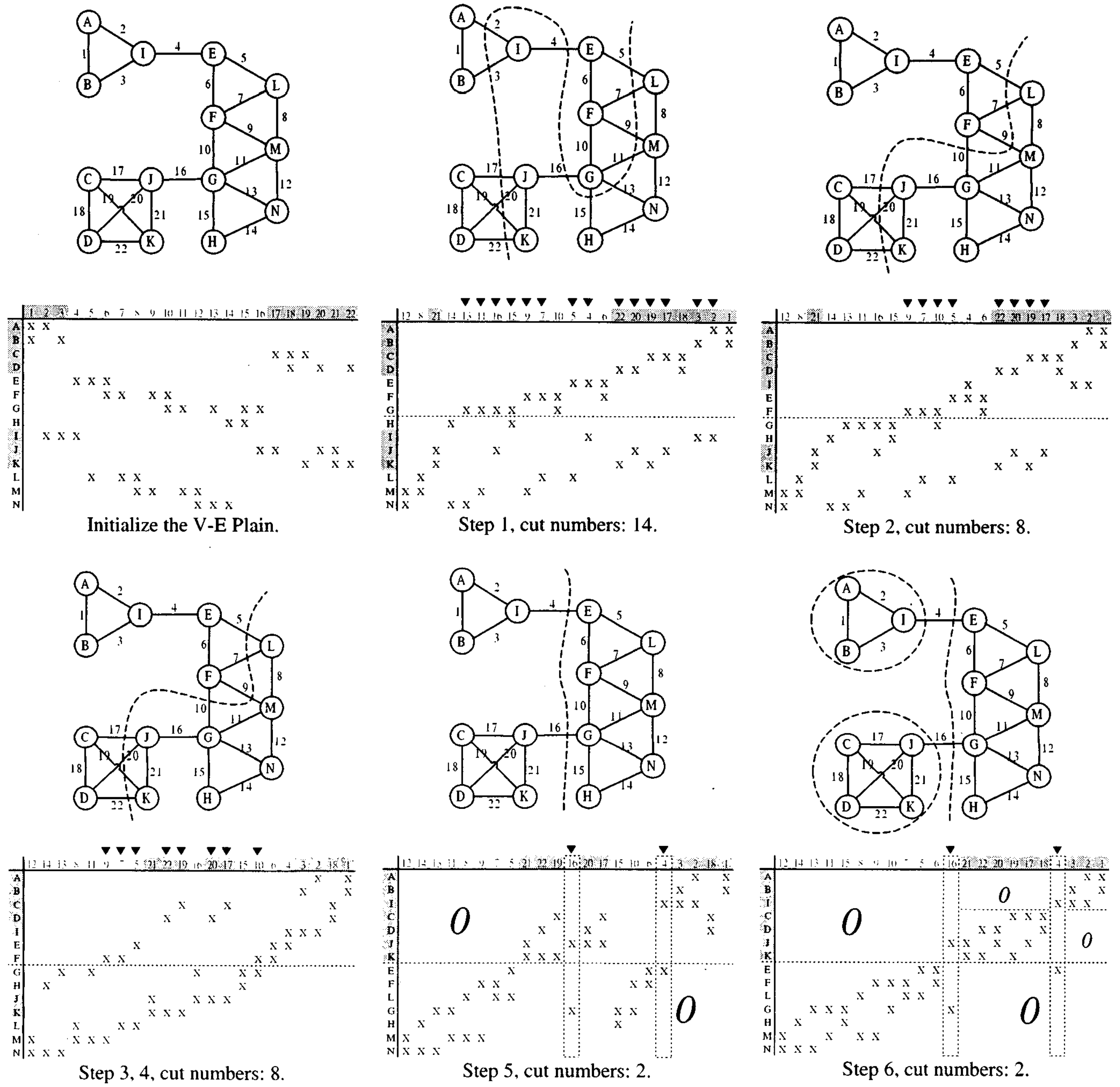
Figure 6. Using radix-sorting technique can effectively handle the sorting job of mass numbers.

II. The Edge-Node Interleaved Sort for Leaching and Envelop (ENISLE) Algorithm

The proposed method in this work is called Edge-Node Interleaved Sort for Leaching and Envelop (ENISLE) algorithm. It is briefly shown as the following:

```
Algorithm ENISLE
begin
  (* Initialize the V-E plain & randomize (V, E) pairs *)
  if ( CLUSTERED_DISTRIBUTION() ) then
    RANDOMIZE();
  endif          (* Phase One: Basic four steps, Edge interleave 1 *)
  SORT-EDGE();        (* From Bottom-side *);
  SORT-NODE();        (* From Right-side *);
  SORT-EDGE();        (* From Top-side *);
  SORT-NODE();        (* From Left-side *);
  INIT_RECORD_NODE-SET();
  repeat   (* Phase Two Begins: Additional steps, Edge interleave 2 *)
    SORT-NODE();          (* From Right-side *);
    SORT-EDGE();          (* From Top / Bottom side *);
    if ( not NODE-SET_CHANGED() ) then
      break;
    endif
    SORT-NODE();           (* from Left-side *);
  until ( NODE-SET_CHANGED() );
end.
```

Notice it can intuitively determine distributed uniformly or not by the diagram clearly, not need additional computing about correlation coefficients or co-variances. If (V, E) pairs are not uniformly distributed on V-E plain, and if we do not randomize it, then directly issue the converge procedures, may get a worse cut solution and leave the loop. No non-determined / infinite loops occur. In Fig. 4, first four steps of the ENISLE algorithm are demonstrated by an example in detail.


Initialize the V-E Plain.

Step 1, cut numbers: 14.

Step 2, cut numbers: 8.

Step 3, 4, cut numbers: 8.

Step 5, cut numbers: 2.

Step 6, cut numbers: 2.

Figure 7. It shows the proposed ENISLE algorithm effectively solves the min-cut and the ratio min-cut partitioning at the same time.

In the ENISLE algorithm, carefully arrange the memory requirement is necessary. As shown in Fig. 5, using bit field structures reduce to one-eighth-memory space. If it has 100K nodes and 500K edges, the program will need about 6.4GB virtual memory space. A powerful sorting engine decides the performance of this method -- we need sort very mass numbers! Using radix sort [14] can handle this problem effectively, as shown in Fig. 6. If the circuits are more enormous, multilevel methods [2] can be considered.

On a 1280 × 1024 pixels × 24 bits true color display monitor, assume 1280 × 16 bits edges / 1024 ×8 bits nodes = 20480 edges / 8192 nodes per screen, or 1024×24 bits edges / 1280 bits nodes = 24576 edges / 1280 nodes per screen. We can scroll the screen, as scroll the spreadsheets. It can directly observe every iterative improvement, get useful information, or decide to manually halt the procedures or not, if necessary. This is very suitable for IC EDA industrial certain cut constraints under non-uniformly distributed cases.

In Fig. 7, this example demonstrates the cut interchange conditions in every step. The ENISLE algorithm effectively solves the min-cut partitioning and the ratio min-cut partitioning at the same time. Fig. 8. shows another successful example.

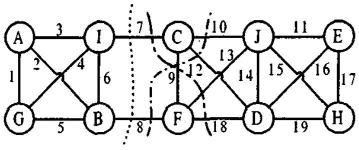


Figure 8. Another successful example.

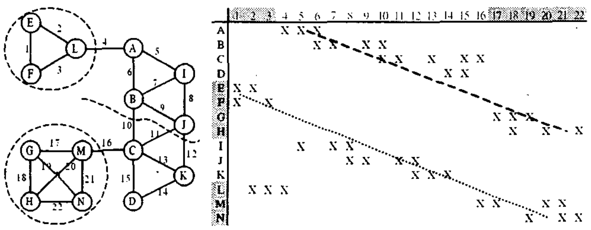


Figure 9. Non-uniformly distributed condition.

III. The Qualitative Analysis of (V, E) Pairs Distributed Conditions

In Fig. 9, (V, E) pairs are not uniformly distributed on V-E plain. And if we do not randomize it, then directly issue the convergent procedure. It shows that we cannot get the optimal solution (two cuts), but get a nearly optimal solution (three cuts). It can intuitively determine distributed uniformly or not by this diagram clearly.

As mentioned in Fig. 3, it shows (V, E) pairs non-uniformly distribution lead the probability that can get the min-cut solution is relatively small. The work finds the relationship between cut numbers and initial (V, E) pairs distributed condition/entropy. The relationship is a very important issue. It is shown as Fig. 10.

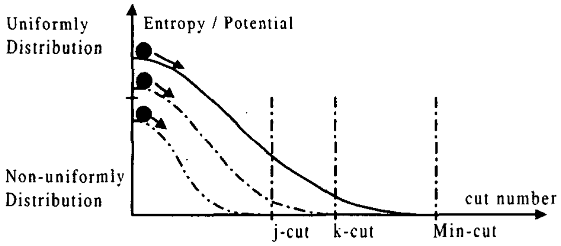


Figure 10. The relationship between cut numbers and initial (V, E) pairs distributed condition/ entropy. Notice the cut number j > k > min-cut, k is second optimal cut, and j is third optimal cut. The higher initial potential, the more probability aims the min-cut.

IV. Conclusion and Further Research

Due to the proposed new method ENISLE is different with any other min-cut partitioning methods, not improves or modifies other min-cut partitioning methods. So this paper does not concentrate on the comparisons with them, mainly focus on the demonstration of the proposed new method. We will completely finish further into the work.

The paper indicates that the proposed ENISLE method can effectively solve the mincut partitioning and the ratio mincut partitioning at the same time by global viewpoints. It is not only using node information but also edge information. Hundreds of netlists experiments have ever been processed and found if we can let (V, E) pairs approach to uniformly distribution on the V-E plain, it can soon get the optimal solution, no more NPC problem. If we can't, or just require certain cut constraint, not min-cut, the proposed method can provide an intuitive heuristic nearly optimal solution. It is very suitable for EDA industrial usage.

V. References

[1] S. B. Akers, "Clustering Techniques for VLSI," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 1982, pp. 472–476.

[2] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," in *Proc. Design Automation Conf.*, 1997, pp. 530–533.

[3] C. J. Alpert and S.-Z. Yao, "Spectral partitioning: The more eigenvectors the better," in *Proc. IEEE/ACM Design Automation Conf.*, 1995, pp. 195–200.

[4] J. Cong et al., "Large scale circuit partitioning with loose/stable net removal and signal flow based clustering," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1997, pp. 441–446.

[5] J. Cong and S. K. Lim, "Multiway Partitioning with Pairwise Movement," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1998, pp. 512–516.

[6] S. Dutt and W. Deng, "A probability-based approach to VLSI circuit partitioning," in *Proc. IEEE/ACM Design Automation Conf.*, June 1996, Best-Paper Award, pp. 100–105.

[7] S. Dutt, "New faster Kernighan-Lin-type graph-partitioning algorithms," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1993.

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

[9] J. Garbers, H. J. Promel, and A. Steger, "Finding clusters in VLSI circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1990, pp. 520–523.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: W. H. Freeman, pp. 209–210.

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

[12] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "A fast algorithm for performance driven placement," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1990, pp. 328–331.

[13] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Tech. Journal*, vol. 49, pp. 291–307, Feb. 1970.

[14] D. E. Knuth, *Sorting and Searching*. Addison-Wesley, 1973.

[15] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks," *IEEE Trans. Computers*, vol. C-33, pp. 438–446, May 1984.

[16] Y. G. Saab, "A fast and robust network bisection algorithm," *IEEE Trans. Computers*, pp. 903–913, 1995.

[17] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*, B. V. Deventer, Ed. Amsterdam, Netherlands: Kluwer.

[18] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation. 3rd Ed.*, Boston, MA: Kluwer, 1999.

[19] Y. C. Wei and C. K. Cheng, "An improved two-way partitioning algorithm with stable performance," *IEEE Trans. Computer-Aided Design*, pp. 1502–1511, 1990.

[20] Y. C. Wei and C. K. Cheng, "Toward efficient hierarchical designs by ratio cut partitioning," in *Proc. Int. Conf. Computer-Aided Design*, 1989, pp. 298–301.