Circuit Bipartition Using Simulated Annealing

Siddhartha Banerjee, Bibek Ranjan Ghosh
Assistant Professor, Ramakrihna Mission Residential College (Autonomous), Narendrapur, Kolkata - 700103, West Bengal, India

Sayan Jana, Atanu Pramanik, Sudip Sarkar
B.Sc. student, Ramakrihna Mission Residential College (Autonomous), Narendrapur, Kolkata - 700103, West Bengal, India

Abstract – Circuit partitioning is the most critical step in the physical design of various circuits in VLSI design. In this paper, simulated annealing algorithm for circuit Bi-partitioning has been attempted. In the partitioning main objective is to minimize the number of cuts. This chapter addresses the problem of partitioning and particular the use of the simulated annealing algorithms for circuit partitioning. The objects to be partitioned in VLSI design are typically logic gates or instances of standard cell. Here the circuits are considered as graph where the nodes represent logic gates or cells and edges represent the connection between these gates or cells. Hence circuit partitioning problem becomes as graph partitioning problem. The algorithm can partition circuit into two sub-circuits. Our method calculates the fitness value and discards solution with low fitness value. The result of the simulated annealing is compared with Kernighan-Lin (KL) algorithm and the result is satisfactory.

Keywords – Partitioning, PCB, Simulated Annealing Algorithm, Net list, global optimum, local optimum, cut size.

I. INTRODUCTION

Efficient designing of any complex system necessitates decomposition of the same into a set of smaller subsystems. Subsequently, each subsystem can be designed independently and simultaneously to speed up the design process. The process of decomposition is called partitioning [1].

A VLSI system is partitioned into several levels due to its complexity. At the highest level, a system is divided into a set of PCBs. This is called system level partitioning. The partitioning of a PCB into chips is called board level partitioning while the partitioning of a chip into smaller subcircuits is called chip level partitioning.

Due to the limitations of memory space and computation power available, it may not be possible to layout the entire chip in the same step. Therefore, the chip is normally partitioned into sub-chips (called blocks). Each block has terminals located at the periphery that are used to connect the blocks. The connection is specified by a netlist, which is a collection of nets. The net is a set of terminals which have to be made electrically equivalent. Figure 1 (a) shows a circuit, which has been partitioned into three subcircuits. Note that the number of interconnections between any two partitions is four (as shown in Figure 1(b)).

If the circuit assigned to a PCB remains too large to be fabricated as a single unit, it is further partitioned into subcircuits such that each subcircuit can be fabricated as a VLSI chip. However, the layout process can be simplified and expedited by partitioning the circuit assigned to a chip into even smaller subcircuits. The partitioning process of a process into PCBs and a PCB into VLSI chips is physical in nature. That is, this partitioning is mandated by the limitations of fabrication process. In contrast, the partitioning of the circuit on a chip is carried out to reduce the computational complexity arising due to the sheer number of components on the chip. The partitioning is a hierarchical procedure of a computer system.
A. Problem formulation

The partitioning problem can be expressed more naturally in graph theoretic terms. A hypergraph $G = (V, E)$ representing a partitioning problem can be constructed as follows. Let $V$ be a set of vertices and $E$ be a set of hyperedges. Each vertex represents a component. There is a hyperedge joining the vertices whenever the components corresponding to these vertices are to be connected. Thus, each hyperedge is a subset of the vertex set i.e., in other words, each net is represented by a hyperedge. The modeling of partitioning problem into hypergraphs allows us to represent the circuit partitioning problem completely as a hypergraph partitioning problem. The partitioning problem is to partition $V$ into set of subsets $V_1, V_2, ..., V_k$ such that

$$V_i \cap V_j = \emptyset \quad \text{for} \quad i \neq j$$

$$\bigcup_{i=1}^{k} V_i = V$$

B. Classification of Partitioning Algorithms

The mincut problem is NP-complete, it follows that general partitioning problem is also NP-complete. As a result, variety of heuristic algorithms for partitioning has been developed. Partitioning algorithms can be classified in three ways. The first method of classification depends on availability of initial partitioning. There are two classes of partitioning algorithms under this classification scheme:

- Constructive algorithms: The input to constructive algorithms is the circuit components and the netlist. The output is a set of partitions and the new netlist. Constructive algorithms are typically used to form some initial partitions which can be improved by using other algorithms. In that sense, constructive algorithms are used as preprocessing algorithms for partitioning. They are usually fast, but the partitions generated by these algorithms may be far from optimal.

- Iterative algorithms: Iterative algorithms, on the other hand, accept a set of partitions and the netlist as input and generate an improved set of partitions with the modified netlist. These algorithms iterate continuously until the partitions cannot be improved further.

The partitioning algorithms can also be classified based on the nature of the algorithms. There are two types of algorithms:

- Deterministic algorithms: Deterministic algorithms produce repeatable or deterministic solutions. For example, an algorithm which makes use of deterministic functions, will always generate the same solution for a given problem.

- Probabilistic algorithms: The probabilistic algorithms are capable of producing a different solution for the same problem each time they are used, as they make use of some random functions.

The partitioning algorithms can also be classified on the basis of the process used for partitioning. Thus we have the following categories:

- Group Migration algorithms: The group migration algorithms start with some partitions, usually generated randomly, and then move components between partitions to improve the partitioning. The group migration algorithms are quite efficient. However, the number of partitions has to be specified which is usually not known when the partitioning process starts. In addition, the partitioning of an entire system is a multi-level operation and the evaluation of the partitions obtained by the partitioning depends on the final integration of partitions at all levels, from the basic subcircuits to the whole system. An algorithm used to find a minimum cut at one level may sacrifice the quality of cuts for the following levels. The group migration
method is a deterministic method which is often trapped at a local optimum and can not proceed further.

- Simulated Annealing and Evolution based algorithms: The *simulated annealing/evolution* algorithms carry out the partitioning process by using a cost function, which classifies any feasible solution, and a set of moves, which allows movement from solution to solution. Unlike deterministic algorithms, these algorithms accept moves which may adversely effect the solution. The algorithm starts with a random solution and as it progresses, the proportion of adverse moves decreases. These degenerate moves act as a safeguard against entrapment in local minima. These algorithms are computationally intensive as compared to group migration and other methods.

- Other partitioning algorithms.

Among all the partitioning algorithms, the group migration and simulated annealing or evolution have been the most successful heuristics for partitioning problems. The use of both these types of algorithms is ubiquitous and extensive research has been carried out on them. In this paper we introduce a simulated annealing based bi-partitioning algorithm and compare it with a well known group migration algorithm (KL algorithm).

In section II we give some previous work done in this field and in section III we discuss about Simulated Annealing Algorithm. In section IV we introduced the proposed method. Section V lists the experimental result. Finally the conclusive remarks are given in section VI.

II. RELATED WORKS

Many approaches have been proposed for the circuit partitioning problem. They include group swapping [3], [4], simulated annealing [2], network flow [5], eigenvector decomposition [6], etc.

Kernighan and Lin proposed a two-way partitioning algorithm with constraints on the subset size. This algorithm randomly starts with two subsets, and pairwise swapping is iteratively applied on all pairs of nodes. Subsequently, many improvements have been made to this method. Schweikert and Kernighan proposed the use of a net cut model so that the algorithm can handle multipin net cases [3].

Fiduccia and Mattheyses improved this algorithm by reducing time complexity to $O(P)$ with respect to the number of pins $P$, and Krishnamurthy [4] further added in lookahead ability. The Kernighan-Lin based algorithm is quite efficient but it needs a predefined subset size to start with.

Simulated annealing [2] is another method based on iterative improvement. The objective function in simulated annealing is analogous to energy in a physical system, and each move is analogous to changes in the energy of the system. The Metropolis Monte Carlo method is used to decide whether a move is accepted. Simulated annealing usually produces good results at the expense of very long running time.

The maximum-flow-minimum-cut algorithm was presented by Ford and Fulkerson. They transformed the minimum cut problem into the maximum flow problem [5]. In order to separate a pair of nodes into two subsets, the minimum number of crossing edges is equal to the maximum amount of flow from one node to the other. Although this algorithm can find the optimum solution between any pair of nodes in a network, there is no constraint on the sizes of resultant subsets. In practice, the result is not useful if two very unevenly sized subsets are generated.

In eigenvector decomposition [6], connections are represented in a matrix. The eigenvectors of the matrix define the locations of all components and thus derive partitioning results. This method requires the transformation of every multipin net into several two-pin nets in real circuits before establishing the matrix.

S. Gurjot Singh et al. [7] described a comparative analysis of KL algorithm and Simulated Annealing based partitioning algorithm. As observed from the results, SA algorithm is showing better results only in certain cases or circuits. Otherwise, cut-set becomes larger than the previous cut-set taken. This is because the parameters taken are constant for all the 15 VLSI NET circuits. But, the results may be different and may be better if different parameters are selected for the different circuit configurations.

S. S. Gill et al [8] present a two way partitioning of a circuit represented as a graph using simulated annealing procedure and minimization of delay between the partitions. The various parameters used in the annealing process like initial temperature, cooling rate, and the threshold, given as a number of calculations , are changed and its influence on the delay between the partitions is discussed.

III. SIMULATED ANNEALING ALGORITHM

Simulated annealing (SA) is a generic probabilistic, heuristic algorithm for the global optimization problem of locating a good approximation to the global optimum of a given function.
in a large search space. It is often used when the search space is discrete. For certain problems, simulated annealing may be more efficient than exhaustive enumeration – provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution. The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.

In order to apply the SA method to a specific problem, one must specify the following parameters:

- Representation of possible solutions
- A generator of random changes in solutions
- A means of evaluating the problem functions
- An annealing schedule - an initial temperature and rules for lowering it as the search progresses.

The parameters of the simulated annealing algorithm are discussed in the following subsections:

A. Solution representation and generation

When attempting to solve an optimization problem using the SA algorithm, the most obvious representation of the control variables is usually appropriate. However, the way in which new solutions are generated may need some thought. The solution generator should

1. Introduce small random changes, and
2. Allow all possible solutions to be reached.

B. Solution Evaluation

The SA algorithm does not require or deduce derivative information; it merely needs to be supplied with an objective function for each trial solution it generates. Thus, the evaluation of the problem functions is essentially a "black box" operation as far as the optimization algorithm is concerned. Obviously, in the interests of overall computational efficiency, it is important that the problem function evaluations should be performed efficiently;
especially as in many applications these function evaluations are by far the most computationally intensive activity.

Some thought needs to be given to the handling of constraints when using the SA algorithm. In many cases the routine can simply be programmed to reject any proposed changes which result in constraint violation, so that a search of feasible space only is executed.

However, there are two important circumstances in which this approach cannot be followed:
1. If there is any equality constraints defined on the system,
2. If the feasible space defined by the constraints is (suspected to be) disjoint, so that it is not possible to move between all feasible solutions without passing through infeasible space.

C. Annealing schedule

The annealing schedule determines the degree of uphill movement permitted during the search and is thus critical to the algorithm's performance. The principle underlying the choice of a suitable annealing schedule is easily stated: the initial temperature should be high enough to "melt" the system completely and should be reduced towards its "freezing point" as the search progresses.

Choosing an annealing schedule for practical purposes is something of an art. The standard implementation of the SA algorithm is one in which homogeneous Markov chains of finite length are generated at decreasing temperatures. The following parameters should therefore be specified:

1. An initial temperature $T_0$
2. A final temperature $T_f$ or a stopping criterion
3. A length for the Markov chains and
4. A rule for decrementing the temperature.

1. Initial Temperature

A suitable initial temperature $T_0$ is one that results in an average increase of acceptance probability $P_0$ of about 0.8. In other words, there is an 80% chance that a change which increases the objective function will be accepted. The value of $T_0$ will clearly depend on the scaling of $f$ and, hence, be problem-specific.

2. Final Temperature

In some simple implementations of the SA algorithm the final temperature is determined by fixing
- The number of temperature values to be used, or
- The total number of solutions to be generated.

Alternatively, the search can be halted when it ceases to make progress. Lack of progress can be defined in a number of ways, but a useful basic definition is no improvement (i.e. no new best solution) being found in an entire Markov chain at one temperature, combined with the acceptance ratio falling below a given (small) value.

3. Length of Markov Chains

An obvious choice for $L_k$, the length of the $k$-th Markov chain, is a value that depends on the size of the problem, so $L_k$ is independent of $k$. Alternatively, it can be argued that a minimum number of transitions $N_{\text{min}}$ should be accepted at each temperature. However, as $T_k$ approaches 0, transitions are accepted with decreasing probability so the number of trials required to achieve $N_{\text{min}}$ acceptances approaches 1. Thus, in practice, an algorithm in which each Markov chain is terminated after
- $L_k$ transitions or
- $N_{\text{min}}$ acceptances,
whichever comes first, is a suitable compromise.

4. Decrementing the Temperature

The simplest and most common temperature decrement rule is:

$$T_{k+1} = aT_k$$

where $a$ is a constant close to, but smaller than, 1. This exponential cooling scheme (ECS) was first proposed with $a = 0.95$. In a linear cooling scheme (LCS) in which $T$ is reduced every $L$ trials:

$$T_{k+1} = T_k - T$$

The reductions achieved using the two schemes have been found to be comparable, and the final value of $f$ is, in general, improved with slower cooling rates, at the expense of greater computational effort.

IV. PROPOSED METHODOLOGY

In this paper we introduced a Simulated Annealing Algorithm for partitioning. Simulated annealing process starts with a random initial partitioning. An altered partitioning is generated by exchanging some elements between partitions. The resulting change in score, $ds$, is calculated. If $ds < 0$ (representing lower energy), then the move is accepted. If $ds = 0$ then the move is accepted with probability $p$. The probability of accepting an increased score decreases with the increase in temperature $t$. This allows the simulated annealing
algorithm to climb out of local optima in search for a global minimum.

The Group Migration algorithm [1] is, however, quite robust. The complexity of these types of algorithms is considered too high even for moderate size problems. We have done a comparative study of these algorithms and our proposed method on several moderate size problems and find a satisfactory result.

A. Solution representation

For bi-partitioning a circuit represented as graph, we first create two partition LEFT and RIGHT by using the following algorithm Generate_Partition () as in Figure 3. This algorithm randomly generates N numbers between 1 to N, where N be the number of nodes in the circuit. First \( \lfloor \frac{N}{2} \rfloor \) numbers are stored in LEFT and remaining numbers are stored in RIGHT. Since the LEFT and RIGHT are two disjoint set of nodes we check for repetition of number in the set. The numbers are temporarily stored in an array A [1...N]. An example initial partition is shown in Figure 4.

B. Evaluation

Since we want to bi-partition the circuit such a way that the interconnection between these two partitions is minimized, we fixed up objective function of a partition as the cut size i.e. number of interconnection between two partitions. If we generate a partition with minimum fitness value we get a better partition.

Objective function value of the Partition \( P \) i.e. \( F(P) = \) cut size of the partition \( P \).

The partition \( P \) which is defined by two set of nodes LEFT and RIGHT, the cut size of the partition is calculated as follows:

\[
\text{Cut size} = \text{Number of edges between vertex pair } u \text{ and } v \text{ such that } u \in \text{LEFT} \text{ and } v \in \text{RIGHT}.
\]

For example if the circuit represented as graph as shown in Figure 5, and the partition \( P \) which is defined by two sets LEFT = \{1, 2, 3, 4\} and RIGHT = \{5, 6, 7, 8\} then the cut size is 9. Because there exist edges between \((1, 5), (1, 6), (2, 5), (2, 6), (3, 6), (3, 7), (3, 8), (4, 7)\) and \((4, 8)\).

The objective function value of the initial partition of Figure 4 is shown in Figure 6.
C. New solution generation

New solution is generated using following algorithm New_solution ( ) in Figure 7.

The New_solution ( ) algorithm is illustrated by the following example on the partition of Figure 6.

New_solution ( )
{
    Step 1: C1 = Chose a random number between 1 and \(\lfloor N/2 \rfloor\)
    Step 2: C2 = Chose another random number between 1 and \(\lfloor N/2 \rfloor\)
    Step 3: A new Partition P\(_{\text{new}}\) is created by interchanging the C1 and C2 column of the LEFT and RIGHT sets of Partition P.
}

D. Selection of new solution

After generating the new partition and its function value the new solution is selected depending on the changes in score ds.

\[ ds = \text{Function value of new partition} - \text{Function value of old partition} \]

If \(ds < 0\) we accept the new solution and discard the previous one, since the function value i.e. cut size of new solution is less than old one. If \(ds \geq 0\) we chose the new solution with probability \(p\). The probability is set as

\[ p = e^{-(ds/T)} \]

where \(T\) is temperature which is initially set by a large value and is decreased with time. As \(T\) decreases with time, the probability to accept the new solution with large cut size also decreases.

V. RESULT

We apply this algorithm on different benchmark circuits for bi-partition. In our experiment we set the initial temperature \(T\) by 1000 and decrease the temperature by a cooling rate of .25. Number of iteration in each temperature is considered as 500. In all cases the algorithm is successful to provide solutions with minimum interconnection between two partitions. We compare our algorithm with a well known group migration algorithm i.e. KERNIGHAN-LIN ALGORITHM (KL algorithm) and we find that our algorithm finds solution with less time than KL algorithm. The solution for different benchmark problem is given in Table 1.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Nodes</th>
<th>KL algorithm</th>
<th>Proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial Cutsize</td>
<td>Final Cutsize</td>
<td>Initial Cutsize</td>
</tr>
<tr>
<td>Actlow</td>
<td>18</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Moore</td>
<td>21</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Regfb</td>
<td>25</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>Mealy</td>
<td>37</td>
<td>34</td>
<td>12</td>
</tr>
<tr>
<td>Sequence</td>
<td>49</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>Decade</td>
<td>71</td>
<td>393</td>
<td>510</td>
</tr>
<tr>
<td>Binbcd</td>
<td>101</td>
<td>866</td>
<td>97</td>
</tr>
</tbody>
</table>

TABLE 1: COMPARISON OF KL ALGORITHM WITH PROPOSED METHOD

VI. CONCLUSION

In this paper we provide a simulated annealing based algorithm for circuit bi-partitioning that provide result with less interconnection between partitions in less time. This
algorithm can also be extended for multi-way partitioning of the circuit.

REFERENCES


