A GENETIC ALGORITHM BASED APPROACH TO SOLVE VLSI FLOORPLANNING PROBLEM

Leena Jain
Associate Professor Global Institute of Management and Emerging Technologies, Amritsar, Punjab

Amarbir Singh
Research Scholar, I. K. Gujral Punjab Technical University, Kapurthala, Punjab

ABSTRACT

Definition – VLSI (Very Large Scale Integration) floor planning is known to be NP-hard (Non-Deterministic Polynomial-time Hard) problem. There is a need to develop automated algorithms to decide the relative positions of circuits on a VLSI chip. During the floor planning phase, the major objective is to minimize area (dead-space) in a chip. In this paper, in order to find the near to optimal solution for floor planning problem a genetic algorithm has been applied for outline-free floor plans. A heuristic placement strategy is used in our work which helps in deciding the positions of rectangular modules on a chip. Microelectronics Centre of North Carolina (MCNC) benchmark circuits are used to test the performance of our algorithm and experimental results show that genetic algorithm and heuristic placement strategy are able to produce near to optimal solutions which are comparable with various techniques proposed in literature for outline-free floor plans.

Key words: VLSI Floor planning, Floor plan, Genetic Algorithm, Chip


http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=6

1. INTRODUCTION

During VLSI floor planning relative positions of circuits are decided on a chip Floor planning helps to determine the topology of layout and this problem is known to be NP-hard (Non-Deterministic Polynomial-time Hard). The solution space of this problem increases exponentially with the increase in number of modules in a circuit. Due to this it becomes very difficult to find the optimal solution by exploring the global solution space [1]. This problem has received much attention of researchers in recent years due to its usefulness and complexity. As the number modules are increasing day by day due to the improvement in integration technology, floor planning becomes a critical step in the process of VLSI physical
A Genetic Algorithm Based Approach to Solve VLSI Floor planning Problem

design process. Floor planning consists of finding the optimal order of placement of modules so that dead space and wire length can be minimized and there are no module overlaps [2].

Recently, various stochastic optimization algorithms, such as Simulated Annealing (SA), Genetic Algorithm (GA), Artificial Neural Networks (ANN) and tabu search have been used to solve complex problems. It has been proved that GA is an effective method for solving NP-hard optimization problems, so due to this reason GA has been applied on VLSI Floor planning problem in our research. Genetic algorithm (GA) is a commonly used global search technique based on the survival of the fittest strategy. GA has been used effectively in the past to solve non-slicing VLSI Floor planning problem [4]. In our research work we have purposed an algorithm based on heuristic placement strategy and Genetic Algorithm (GA) to solve the VLSI floor planning problem with the objective of minimizing area (dead-space) for non-slicing floor plans with hard modules. The proposed Simple Genetic Algorithm (SGA) is tested for outline-free floor planning.

Rest of the paper is organized as follows. Floor plan Representations used in VLSI floor planning problem are discussed in section 2. In section 3, problem description and related work done by different researchers in past is given. Heuristic placement strategy and details about the proposed Simple Genetic Algorithm (SGA) are discussed in section 4. Experimental results for outline free floor planning benchmark circuits are presented in section 5. Finally, some concluding remarks are given in section 6.

2. FLOORPLAN REPRESENTATION

There are basically two floor plan layout structures slicing and non-slicing. Any floor plan is sliceable if its rectangular dissection can be obtained by recursively dividing rectangles into smaller rectangles until each non-overlapping rectangle is invisible. So such floor plans which are not sliceable are called non-sliceable floor plans. The slicing floor plan is less useful than non-slicing floor plan, as it can be used to solve the floor planning problems with rather limited scope [5]. Different representations of non-slicing floor plan have been proposed in recent years: Corner Block List (CBL) representation [6], Sequence Pair (SP) representation [7], Bounded Slicing Grid (BSG) representation [8], O-tree representation [9], Transitive Closure Graph (TCG) representation [10], B* tree representation [11] [12] etc. Further details of the above mentioned representations can be found in literature [13]. Literature [14] introduces integer coding representation, which is simple, effective and quite suitable for floor planning problem, due to these advantages the integer coding representation has been adopted in this research work. According to this representation modules can be represented in the format of \(< V_1, V_2, \ldots, V_i, \ldots, V_n >\), where \(1 \leq V_i \leq n\). Now \(V_i = j\) implies that the i-th module is to be placed at j-th position. After that a heuristic approach is used to adjust the modules while taking both their shapes and directions into consideration. In this paper, an integer coding representation is used but it only contains the order of modules. Information about shape, position and direction of the modules is not included in the representation.

3. PROBLEM DESCRIPTION AND RELATED WORK

Given a rectangular region with width \(W\) and height \(H\), a set of modules \(M = \{ r_1, r_2, r_3, \ldots, r_m \}\), in which module \(r_i\) is a rectangular block with fixed width \(w_i\) and height \(h_i\) (hard module) and given a net list \(N\) specifying interconnections between the modules in \(M\), the problem is to find a packing of all the modules into the rectangular region, such that no module can be aligned in a diagonal position, it must be placed parallel to the coordinate axis. Overlapping of one module with any other module is not allowed and a module may be rotated by 90° before its placement. Given \(M\) and \(N\), goal of the floor planning problem is to find a floor plan \(F\) such that a cost function is minimized [15]. The cost of a floor plan \(F\), cost
Leena Jain and Amarbir Singh

\( F \), consists of two parts, one is the area, \( area \ (F) \), which is measured by the smallest rectangle that encloses the floor plan and the other is \( wire \ length \ (F) \), which is the wire length of all the nets specified by \( N \) or the interconnection cost between various modules. Now the cost function can be defined as follows:

\[
\text{cost}(F) = (w) \times \frac{\text{area}(F)}{\text{norm.area}} + (1 - w) \times \frac{\text{wire.length}(F)}{\text{norm.wire}} 
\]

Where \( w \) is a weight assigned for primary objective related to area, \( w \in [0, 1] \). As we have to minimize area, value of \( w \) is taken as 1. The \( \text{norm.area} \) and \( \text{norm.wire} \) are the minimal area and the minimal wire length cost of the problem respectively. This kind of cost function has been adopted from [16]. Since we do not know \( \text{norm.area} \) and \( \text{norm.wire} \) in practice, estimated values are used.

3.1. Related Work

In the context of circuit design, floor planning is the process of placing circuit modules of arbitrary sizes and dimensions on a given layout area with an objective of minimizing area and wire length between the modules and this problem has been tackled by using various approaches in literature. Iterative approaches are most popular for optimizing VLSI floor plans. Commonly used iterative approaches are Simulated Annealing (SA), Genetic algorithm (GA) and Particle Swarm Optimization (PSO). A brief literature review of these popular approaches that have been used to address VLSI floor planning problem is given here.

Chen et al. (2006) [17] studied two types of modern floor planning problems: 1) fixed-outline floor planning and 2) bus-driven floor planning (BDF). This floor planner used B*-tree floor plan representation based on fast three-stage simulated annealing (SA) scheme called Fast-SA in order to solve multi objective VLSI Floor planning problem. Maolin Tang et al. (2007) [18] proposed a memetic algorithm (MA) for a non-slicing and hard-module VLSI floor planning problem. This MA is a hybrid genetic algorithm that uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region.

Further the work on optimization using genetic algorithm was carried out by Pradeep Fernando et al. (2008) [19] they proposed a multi-objective genetic algorithm for floor planning that simultaneously minimizes area and total wire length. The proposed genetic floor planner is the first to use non-domination concepts to rank solutions and Jianli Chen et al. (2010) [20] described that hybrid genetic algorithm (HGA) uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region.

Particle Swarm Optimization (PSO) [1] has been used apart from Genetic Algorithm and Simulated Annealing for optimization of VLSI Floor planning problem.

4. FRAMEWORK OF OUR APPROACH

4.1. Heuristic Placement Strategy

The module sequences for placement will be generated during the creation of population for genetic algorithm, and some rules are needed to convert modules sequence to two dimensional floor plan. For the placement of modules on a chip, a heuristic placement strategy was proposed in literature [16]. This strategy can be represented in the form of an algorithm, which is given below.

\text{Begin}

generate the population of ordered list of modules using genetic algorithm;

\text{End}
for each orientation of module length-wise and width-wise
place first ordered module from ordered list at bottom left corner;
reduce the requirement for that ordered module by one;
initialize list of pivot points;
while not (all ordered modules are considered) do
  while not(requirement of ordered module met) or (no more module can be placed at any pivot point) ) do
    select a pivot point;
    initialize enclosing area= 1.0E34;
    while not( all pivot points have been checked) do
      if placed module is spreading over the boundaries of outline floor plan;
      if placed module is overlapping with any already placed ordered module then skip to next pivot point
      compute enclosing area of placed ordered modules;
      if ( computed enclosing area) ≤ ( previous enclosing area)
        then save this information and newly generated pivot points;
      end while
      if not( placed any selected ordered module) then skip to next ordered module;
    delete used pivot point from the list of pivot points and insert newly created pivot points in
the list of pivot points;
  end while
end while

4.2. Simple Genetic Algorithm
Genetic algorithm does not guarantee to find an optimal solution, but the past experience shows that genetic evolution based approach is very useful in solving VLSI problems in an efficient manner [21]. Modified one point ordered crossover operator is used in our work [16]. Various steps used to optimize the floor planning problem in Simple Genetic Algorithm (SGA) are given below.

begin
Generate N random sequences of rectangular modules
Calculate the dead space produced by each sequence by using the heuristic placement strategy
Save the sequence producing the least dead space
repeat
for each member of the population
  Select two parents randomly using the tournament selection method
  Apply modified one point ordered crossover and mutation to produce a new offspring
  Calculate the dead space produced by new offspring by using the heuristic placement strategy
Best individuals from previous generation are passed to new generation
end for
5. EXPERIMENTAL RESULTS
The experiment is carried out for outline-free floor planning using hard modules with the objective of minimizing the wastage area (dead-space) in the floor plan. MCNC benchmark circuits have been used to test and compare the results of proposed algorithm with different approaches proposed in literature. Characteristics of MCNC benchmark circuits are shown in Table 1. MATLAB R2015a is used to implement and run the programs on a machine with an intel CPU running at 1.70 GHz and 4 GB RAM. Algorithm is run for 50 times in order to obtain best possible values for area and time. The fitness function is described in Eq. (1) and the parameters of the SGA are set as follows: The value of ‘w’ in Eq. (1) is set to 1, population size for genetic algorithm varies for different benchmark circuits, crossover probability \((C_p)\) are set as 0.8 and mutation probability \((M_p)\) is 0.1.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Circuit</th>
<th>#Cells</th>
<th>Cell area (mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCNC</td>
<td>Apte</td>
<td>9</td>
<td>46.56</td>
</tr>
<tr>
<td></td>
<td>Xerox</td>
<td>10</td>
<td>19.35</td>
</tr>
<tr>
<td></td>
<td>Hp</td>
<td>11</td>
<td>8.83</td>
</tr>
<tr>
<td></td>
<td>ami33</td>
<td>33</td>
<td>1.156</td>
</tr>
<tr>
<td></td>
<td>ami49</td>
<td>49</td>
<td>35.445</td>
</tr>
<tr>
<td>GSRC</td>
<td>n100</td>
<td>100</td>
<td>0.1795</td>
</tr>
</tbody>
</table>

Table 2 contains the best and average values produced by our algorithm during the minimization of area (dead-space) for MCNC benchmark circuits. In order to show the efficiency of the algorithm for outline-free floor planning, the results produced by SGA are compared with SP Parquet [3], Fast SP [25], Corner Block List (CBL) [6], Transitive Closure Graph (TCG) [22], B* Tree [11], Enhanced O-tree [24], Genetic Algorithm (GA) [26], Evolutionary Search GA [21], Mimetic Algorithm (MA) [18], Fast SA [23], PSO [1] in Table 3. Results of these floor planning algorithms are taken from their published reports in literature. SGA produces very competitive and comparable results in comparison with other algorithms proposed in literature by different researchers. It has been observed that proposed algorithm performs better for problems with large number of modules. For ami49 benchmark circuit SGA produces better results than almost 65 per cent of other algorithms. Comparison of dead-space percentage is also given in Table 3. Generally the running time of the algorithms implementing NP-hard problems is high. SGA is quite efficient in this regard as it produces better runtimes than one fourth of other algorithms. Normalization of results is used to compare our results with other benchmark algorithms in terms of runtime. The normalized Run Time (NRT) is a ratio of the results of other placement algorithms to the results of our algorithm. Details of runtime and NRT are provided in Table 4.

<table>
<thead>
<tr>
<th>MCNC Benchmark</th>
<th>Area (mm²)</th>
<th>DS (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Average</td>
<td>Best</td>
</tr>
<tr>
<td>Apte</td>
<td>46.92</td>
<td>46.92</td>
<td>0.77</td>
</tr>
</tbody>
</table>
A Genetic Algorithm Based Approach to Solve VLSI Floor Planning Problem

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Xerox</th>
<th>20.18</th>
<th>20.52</th>
<th>4.11</th>
<th>5.69</th>
<th>1.55</th>
<th>1.76</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hp</td>
<td>9.20</td>
<td>9.66</td>
<td>4.02</td>
<td>8.45</td>
<td>7.92</td>
<td>8.27</td>
<td></td>
</tr>
<tr>
<td>ami33</td>
<td>1.23</td>
<td>1.28</td>
<td>6.02</td>
<td>9.80</td>
<td>109.7</td>
<td>112.68</td>
<td></td>
</tr>
<tr>
<td>ami49</td>
<td>37.25</td>
<td>38.58</td>
<td>4.85</td>
<td>8.08</td>
<td>182.9</td>
<td>188.28</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3** Performance Comparison for Classical (outline-free) floor planning with the objective of minimizing area (dead-space) on MCNC benchmark circuits (w=1)

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Apte</th>
<th>Xerox</th>
<th>hp</th>
<th>ami33</th>
<th>ami49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Area (mm²)</td>
<td>DS (%)</td>
<td>Area (mm²)</td>
<td>DS (%)</td>
<td>Area (mm²)</td>
</tr>
<tr>
<td>SGA</td>
<td>46.92</td>
<td>0.77</td>
<td>20.18</td>
<td>4.11</td>
<td>9.20</td>
</tr>
<tr>
<td>SP Parquet</td>
<td>47.07</td>
<td>1.08</td>
<td>19.83</td>
<td>2.42</td>
<td>9.14</td>
</tr>
<tr>
<td>Fast SP</td>
<td>46.92</td>
<td>0.77</td>
<td>19.80</td>
<td>2.27</td>
<td>8.94</td>
</tr>
<tr>
<td>CBL</td>
<td>NA</td>
<td>NA</td>
<td>20.96</td>
<td>7.68</td>
<td>NA</td>
</tr>
<tr>
<td>TCG</td>
<td>46.92</td>
<td>0.77</td>
<td>19.83</td>
<td>2.42</td>
<td>8.94</td>
</tr>
<tr>
<td>B*-Tree</td>
<td>46.92</td>
<td>0.77</td>
<td>19.83</td>
<td>2.42</td>
<td>8.95</td>
</tr>
<tr>
<td>Enhanced O-tree</td>
<td>46.92</td>
<td>0.77</td>
<td>20.21</td>
<td>4.26</td>
<td>9.16</td>
</tr>
<tr>
<td>GA</td>
<td>46.90</td>
<td>0.72</td>
<td>20</td>
<td>3.25</td>
<td>9.03</td>
</tr>
<tr>
<td>Evolutionary Search GA</td>
<td>46.92</td>
<td>0.77</td>
<td>20.26</td>
<td>4.49</td>
<td>9.15</td>
</tr>
<tr>
<td>MA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Fast SA</td>
<td>47.07</td>
<td>1.08</td>
<td>20.11</td>
<td>3.78</td>
<td>9.11</td>
</tr>
<tr>
<td>PSO</td>
<td>46.92</td>
<td>0.77</td>
<td>19.55</td>
<td>1.02</td>
<td>NA</td>
</tr>
</tbody>
</table>

**Table 4** Running Time Comparison for Classical (outline-free) floor planning with the objective of minimizing area on MCNC benchmark circuits (w=1)

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Apte</th>
<th>Xerox</th>
<th>hp</th>
<th>ami33</th>
<th>ami49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Time (sec)</td>
<td>NRT</td>
<td>Time (sec)</td>
<td>NRT</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>SGA</td>
<td>0.32</td>
<td>1</td>
<td>1.55</td>
<td>1</td>
<td>7.92</td>
</tr>
<tr>
<td>SP Parquet</td>
<td>4</td>
<td>12.5</td>
<td>3</td>
<td>1.94</td>
<td>4</td>
</tr>
<tr>
<td>Fast SP</td>
<td>1</td>
<td>3.13</td>
<td>14</td>
<td>9.03</td>
<td>6</td>
</tr>
<tr>
<td>CBL</td>
<td>NA</td>
<td>NA</td>
<td>30</td>
<td>19.35</td>
<td>NA</td>
</tr>
<tr>
<td>TCG</td>
<td>1</td>
<td>3.13</td>
<td>18</td>
<td>11.61</td>
<td>20</td>
</tr>
<tr>
<td>Enhanced O-tree</td>
<td>11</td>
<td>34.38</td>
<td>38</td>
<td>24.52</td>
<td>19</td>
</tr>
<tr>
<td>GA</td>
<td>8.80</td>
<td>27.5</td>
<td>20.8</td>
<td>13.42</td>
<td>16.80</td>
</tr>
<tr>
<td>Evolutionary Search GA</td>
<td>0.066</td>
<td>0.21</td>
<td>0.1</td>
<td>0.06</td>
<td>0.7</td>
</tr>
<tr>
<td>Fast SA</td>
<td>0.09</td>
<td>0.28</td>
<td>0.06</td>
<td>0.04</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Figure 1 shows the simulation result for xerox benchmark circuit produced by our algorithm, it is the best result produced during the execution of SGA for fifty times. Figure 4 shows the convergence graph for the xerox benchmark circuit and it describes the minimization of area values during the execution of 50 generations of SGA.
6. CONCLUSION

In this paper, a Simple Genetic Algorithm (SGA) is presented for minimizing area (dead-space) for VLSI floor planning problem. The proposed algorithm made use of heuristic placement strategy and a genetic algorithm to explore global search space in an efficient way. Different experiments on MCNC benchmark circuits show that our method can produce very competitive results in comparison with other algorithms for outline-free floor planning. Results for SGA show that various benchmark circuits can be processed in a reasonable time. It has been observed that, to further enhance the work in future more attention can be given to reduce the runtime of algorithm while optimizing area and wire length simultaneously.

ACKNOWLEDGEMENT

Authors are thankful to the I.K. Gujral Punjab Technical University, Kapurthala (Punjab) for the support and motivation for research.
REFERENCES


