CHAPTER 4

VLSI FLOORPLANNING USING SEQUENCE PAIR AND DIFFERENTIAL EVOLUTION ALGORITHM

4.1 INTRODUCTION

This chapter presents the application of Differential Evolution (DE) Algorithm for the VLSI floorplanning problems. DE Algorithm has been successfully applied to VLSI sequence pair representation. The important advantages of DE are its simplicity, fast convergence, and easy for implementation. DE algorithm is a population based algorithm like GA using similar operators such as crossover, mutation and selection. Simulation results on MCNC benchmark circuits show that, convergence speed of DE is significantly better than SA, PSO and HPSO algorithms. These results demonstrate that the DE offers a viable tool to VLSI floorplanning problems.

4.2 DIFFERENTIAL EVOLUTION ALGORITHM

GA is a stochastic optimization algorithm employed for combinatorial and continuous optimization problems. It has three main operators: crossover, mutation operators from genetic science, and a selection operator simulating natural selection phenomena. GA can efficiently search large solution spaces due to its parallel structure and the probabilistic transition rules employed in the operators. However, a basic GA has two main drawbacks; lack of good local search ability and premature convergence. In order to overcome these disadvantages, the DE algorithm has been introduced
by Storn and Price (Storn and Price 1995). It has shown a good performance in finding optimal solutions in many cases.

DE algorithm is a stochastic optimization method, minimizing an objective function that can model the problem’s objectives while incorporating constraints. The algorithm mainly has three advantages: finding the true global minimum regardless of the initial parameter values, fast convergence, and using a few control parameters (Storn and Price 1997). Being simple, fast, easy to use, very easily adaptive for integer and discrete optimization, quite effective in nonlinear constraint optimization including penalty functions and useful for optimizing multi-modal search spaces are the other important features of DE.

The main difference between DE and GA, lies in constructing better solutions, is that genetic algorithms rely on crossover while DE relies on mutation operation. This main operation is based on the differences of randomly sampled pairs of solutions in the population. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space.

The major difference between DE and PSO is caused by the selection schemes. Using DE, vectors those yield a smaller objective function value than the respective target vector will be admitted for the following generation. This is called a greedy selection scheme. In contrast, the PSO algorithm accepts all evolved particles, regardless of their objective function value.
4.2.1 DE Optimization Process

An optimization task consisting of D parameters can be represented by a D-dimensional vector. In DE, a population of Np solution vectors is randomly created at the start. This population is successfully improved by applying mutation, crossover and selection operators. The main steps are given below (Storn and Price 1995, 1997).

1. Initialization

The first step of the DE routine is to generate an initial population of candidate solutions by assigning random values to each decision parameter of each individual of the population. Such values must lie inside the feasible bounds of the decision variable and can be generated by the equation (4.1)

\[ X_{i,j}^{(0)} = X_{j}^{\text{min}} + \text{rand}_j(0,1) \cdot (X_{j}^{\text{max}} - X_{j}^{\text{min}}) \]  \hspace{1cm} (4.1)

\[ i = 1,2, \ldots, Np, \ \ j = 1,2, \ldots, D \]

where \( X_{j}^{\text{min}} \) and \( X_{j}^{\text{max}} \) are the lower and upper bound of the j\textsuperscript{th} decision parameter respectively and \( \text{rand}_j(0,1) \) is a uniformly distributed random number within \([0, 1]\) generated a new for each value of j. D represents the number of parameters of the objective function.

2. Mutation

After the population is initialized, this evolves through the operators of mutation, crossover and selection. For mutation and crossover, different types of strategies are in use. Basic scheme is explained here elaborately. The mutation operator is in charge of introducing new parameters
into the population. To achieve this, mutation operator creates mutant vectors as given in equation (4.2) by perturbing a randomly selected vector \( X^{r3} \) with the difference of two other randomly selected vectors \( X^{r2} \text{ and } X^{r1} \). All of these vectors must be different from each other, requiring the population to be of at least four individuals to satisfy this condition. To control the perturbation and improve convergence, the difference vector is scaled by a user defined constant in the range \([0, 1.2]\) (Babu and Rakesh Angira 2003, and Babu and Sastry 1999). This constant is commonly known as the scaling factor (F).

\[
i U_{j,G} = X_{j,G} + F(X_{j,G} - X_{j,G}) \tag{4.2}
\]

where \( i = 1, 2, \ldots, N_p \), \( X^{r1}, X^{r2}, X^{r3} \), are randomly chosen vectors \( \in \{i = 1, 2, \ldots, N_p\} \) and \( r1 \neq r2 \neq r3 \neq i \). \( X^{r1}, X^{r2}, X^{r3} \) are generated a new for each parent vector.

3. Crossover

The crossover operator creates the trial vectors, as given in equation (4.3) (Storn and Price 1995, 1997) which are used in the selection process. A trial vector is a combination of a mutant vector and a parent (target) vector based on different distributions like uniform distribution, binomial distribution, exponential distribution generated in the range \([0,1]\) and compared against a user defined constant referred to as the crossover constant. If the value of the random number is less or equal than the value of the crossover constant, the parameter will come from the mutant vector, otherwise the parameter comes from the parent vector.
The crossover operation maintains diversity in the population, preventing local minima convergence. The crossover constant must be in the range of \([0, 1]\). A crossover constant of one means the trial vector will be composed entirely of mutant vector parameters. A crossover constant near zero results in more probability of having parameters from the target vector in the trial vector. A randomly chosen parameter from the mutant vector is always selected to ensure that the trial vector gets at least one parameter from the mutant vector even if the crossover constant is set to zero.

\[
U_{j,G+1}^i = \begin{cases} 
U_{j,G}^i & \text{if } ( \text{rand}_j(0,1) \leq CR \text{ or } j = j_{\text{rand}} ) \\
X_{j,G}^i & \text{otherwise}
\end{cases}
\]  

(4.3)

where \( i = 1, 2, \ldots, N_p \), \( j = 1, 2, \ldots, D \)

\( j_{\text{rand}} \) is a randomly chosen index \( \in \{ 1, 2, \ldots, D \} \) generated (prior to the construction of each trial vector) that guarantees that the trial vector gets at least one parameter from the mutant vector. \( X_{j,G}^i \) is the target vector, \( U_{j,G}^i \) the mutant vector and \( U_{j,G+1}^i \) the trial vector.

4. Selection

The selection operator chooses the vectors as given in equation (4.4) (Storn and Price 1995, 1997) that are going to compose the population in the next generation. This operator compares the fitness of the trial vector and fitness of the corresponding target vector and selects the one that performs better.
\[ X_{G+1}^i = \begin{cases} U_{G+1}^i & \text{if } f(U_{G+1}^i) \leq f(X_G^i) \\ X_G^i & \text{otherwise} \end{cases} \] 

(4.4)

\[ i = 1, 2, \ldots, N_p. \]

5. Repetition

Steps 2–4 are repeated until a stopping criterion is met. Furthermore, the whole procedure is repeated a number of times. This ensures that the random starting values do not affect the outcome.

4.2.2 Choice of DE Parameters

There are four parameters in DE which are used for fine tuning the heuristic: \( N_p, F, CR, \) and DE strategy.

1. Population Size (\( N_p \))

If the population is larger, the probability of finding the global minimum for the functions is also larger. However, a larger population implies a larger number of function evaluations.

2. Scaling Factor (\( F \))

Scaling Factor (\( F \)) should not be chosen too small to prevent premature convergence. On the other hand, \( F \) should not be chosen too large because the number of function evaluation increases as \( F \) increases. This value depends on the cost function and on the other strategy parameters. A larger \( F \) increases the probability for escaping a local optimum (Figure 4.1). However, for \( F > 1 \), the convergence speed decreases. It is more difficult to
converge for a population when the perturbation is larger than the distance between two members. The optimal value of F lies in the range of 0.4 to 1.0 (Storn and Price 1997).

![Figure 4.1 Effect of scaling factor (F) on DE algorithm](image)

3. Crossover Constant (CR)

A crossover constant (CR) should not be chosen too large to avoid that the perturbations get too high and the convergence speed decreases. However, a small CR decreases diversity and might cause the strategy to get stuck. This value depends on the cost function and is located in the region CR = 0.2……1.0. A good choice for the crossover constant is a value between CR = 0.3 and CR = 0.9.

4. Different Strategies of Differential Evolution

Price et al (2005) suggested ten different strategies of DE which are given below. The strategies can vary based on the vector to be perturbed, number of difference vectors considered for perturbation, and finally the type of crossover used.
1. DE/ best / 1/ exp.
2. DE/ rand / 1/ exp.
3. DE/ rand-to-best / 1/ exp.
4. DE/ best / 2/ exp.
5. DE/ rand / 2/ exp.
6. DE/ best / 1/ bin.
7. DE/ rand / 1/ bin.
8. DE/ rand-to-best / 1/ bin.
9. DE/ best / 2 /bin.
10. DE/ rand / 2/ bin.

The general convention used above is DE/x/y/z. DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp: exponential; bin: binomial). The perturbation can be either in the best vector of the previous generation or in any randomly chosen vector. Similarly for perturbation either single or two vector differences can be used. For perturbation with a single vector difference, out of the three distinct randomly chosen vectors, the weighted vector differential of any two vectors is added to the third one. Similarly for perturbation with two vector differences, five distinct vectors, other than the target vector are chosen randomly from the current population. Out of these, the weighted vector difference of each pair of any four vectors is added to the fifth one for perturbation. In exponential crossover, the crossover is performed on the D variables in one loop until it is within the CR bound. If a randomly picked number between 0 or 1 goes beyond the CR value, no crossover is performed and the remaining D variables are left intact. In binomial crossover, the crossover is performed on each of the D variables whenever a randomly picked number between 0 and 1 is within the CR value.
So for high values of CR, the exponential and binomial crossovers yield similar results. The strategy to be adopted for each problem is to be determined separately by trial and error.

### 4.3 VLSI FLOORPLANNING WITH DE ALGORITHM

The floorplanner based on sequence-pair representation first extracted the sequence-pair from the given initial block placement. Then the floorplanner set the extracted initial sequence-pair as an initial state, and performed the heuristic search method such as DE to search for the optimal sequence-pair. The value of D, Np, CR, and F are initialized. The maximum number of generation and strategy are chosen. All the population vectors are initialized randomly. Since the upper bounds are all different for each variable in the problem, the variables are all normalized. Hence a random number between 0 and 1 is generated for all the variables for initialization. The cost (area of each block) of each vector is evaluated. The best vector found so far is taken and mutation, and crossover, are performed. The objective function is evaluated and vector with lower cost is selected for the next generation. This is repeated for a specified number of generations and the best cost (area) is displayed. The flowchart of DE algorithm for floorplanning using sequence pair representation is given in Figure 4.2.
4.4 **DE SETTINGS**

The DE algorithm has few control parameters; number of population Np, scaling factor F, and crossover rate CR. The problem specific parameters of the DE algorithm are the maximum generation numbers $G_{\text{max}}$ and the number parameters defining the problem dimension D. The parameters are $N_p = 30$, $F = 0.5$, $CR = 0.7$, $G_{\text{max}} = 100$.

4.5 **SIMULATION RESULTS**

The performance of DE algorithm is tested on MCNC benchmark circuits. Simulations were carried out on a Pentium 4, 1.8GHz and 256 MB
RAM. A seed value for the pseudo random number generator should be selected by trial and error. In principle any positive integer can be taken. Integers 3 to 12 are tried and found that seed 5 produces optimal results. The different parameters of DE are evaluated permutatively to find the optimal input values $CR=\{0.1,0.2,\ldots,1\}$ $F=\{0.1,0.2,\ldots,0.9\}$ and strategy number $=\{1,2,3,\ldots,10\}$. The different values were iteratively evaluated on the hp circuit. The lowest average value was produced by $CR=0.7$ and $F=0.5$. Using these values, the best strategy is found out. The results are presented in Table 4.1. As observed, strategy 7, on average performs better than the other strategies and was selected. The strategy $DE/rand/1/bin$ yielded the best results.

The chip area and the runtime produced by DE floorplanner are compared with those of SA and shown in Table 4.2. Simulation results show that DE based floorplanner generates significantly better floorplans for all the benchmark circuits.

### Table 4.1 Area/Runtime of hp circuit for different strategies

<table>
<thead>
<tr>
<th>S.NO</th>
<th>Strategy</th>
<th>Area(mm$^2$)</th>
<th>Run time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$DE/\text{best} / 1/ \exp$</td>
<td>9.08</td>
<td>1.9</td>
</tr>
<tr>
<td>2</td>
<td>$DE/\text{rand} / 1/ \exp$</td>
<td>9.05</td>
<td>1.8</td>
</tr>
<tr>
<td>3</td>
<td>$DE/\text{rand-to-best} / 1/ \exp$</td>
<td>9.24</td>
<td>2.2</td>
</tr>
<tr>
<td>4</td>
<td>$DE/\text{best} / 2/ \exp$</td>
<td>9.21</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>$DE/\text{rand} / 2/ \exp$</td>
<td>9.05</td>
<td>2.1</td>
</tr>
<tr>
<td>6</td>
<td>$DE/\text{best} / 1/ \bin$</td>
<td>8.95</td>
<td>1.8</td>
</tr>
<tr>
<td>7</td>
<td>$DE/\text{rand} / 1/ \bin$</td>
<td>8.90</td>
<td>1.6</td>
</tr>
<tr>
<td>8</td>
<td>$DE/\text{rand-to-best} / 1/ \bin$</td>
<td>9.24</td>
<td>2.2</td>
</tr>
<tr>
<td>9</td>
<td>$DE/\text{best} / 2/ \bin$</td>
<td>9.03</td>
<td>2.1</td>
</tr>
<tr>
<td>10</td>
<td>$DE/\text{rand} / 2/ \bin$</td>
<td>9.05</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Table 4.2 Area/Runtime comparisons for the SP based SA and DE

<table>
<thead>
<tr>
<th>MCNC Circuit</th>
<th>No. of modules</th>
<th>SA</th>
<th></th>
<th>DE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Area (mm²)</td>
<td>Run time (sec)</td>
<td>Area (mm²)</td>
<td>Run time (sec)</td>
</tr>
<tr>
<td>apte</td>
<td>9</td>
<td>48.12</td>
<td>13</td>
<td>46.87</td>
<td>1.1</td>
</tr>
<tr>
<td>xerox</td>
<td>10</td>
<td>20.69</td>
<td>15</td>
<td>19.70</td>
<td>1.4</td>
</tr>
<tr>
<td>hp</td>
<td>11</td>
<td>9.93</td>
<td>25</td>
<td>8.90</td>
<td>1.6</td>
</tr>
<tr>
<td>ami33</td>
<td>33</td>
<td>1.31</td>
<td>676</td>
<td>1.21</td>
<td>4</td>
</tr>
<tr>
<td>ami49</td>
<td>49</td>
<td>38.84</td>
<td>1580</td>
<td>36.50</td>
<td>10</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>23.78</td>
<td>461.8</td>
<td>22.63</td>
<td>3.62</td>
</tr>
<tr>
<td>Comp.</td>
<td></td>
<td>1.050</td>
<td>127.570</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

From Table 4.2, it clearly shows that DE reduced the average area by 5% compared with SA. It also achieved average speedup of 127.57 over SA. Figures 4.3 to 4.7 show the convergence graphs of various MCNC benchmark circuits for area and Figures 4.8 to 4.12 show the convergence graphs for wirelength respectively. From Figure 4.3 it is clear that DE converges to the minimum area of 1.21 for ami33 circuit at 20th iteration whereas SA converges to 1.31 at 72nd iteration.

![ami33](attachment:ami33.png)

Figure 4.3 Convergence graph of ami33 circuit for chip area
Figure 4.4 Convergence graph of ami49 circuit for chip area

Figure 4.5 Convergence graph of apte circuit for chip area
Figure 4.6 Convergence graph of hp circuit for chip area

Figure 4.7 Convergence graph of xerox circuit for chip area
Figure 4.8 Convergence graph of ami33 circuit for wirelength

Figure 4.9 Convergence graph of ami49 circuit for wirelength
Figure 4.10 Convergence graph of apte circuit for wirelength

Figure 4.11 Convergence graph of hp circuit for wirelength
In order to study the impact of population size on the performance of DE, the population size is varied from 5 to 80 in steps of 5. From the Figure 4.13 it is clear that DE reduced to the minimum of 1.21 for the ami33 circuit for the population size of 30 and remained the same as the population size was increased further. Thus optimal population size is identified as 30. Figures 4.13 and 4.14 show the chip area vs. population size for the ami33 circuit and runtime vs. population size for the ami33 circuit for DE respectively.

To study the tradeoff between the two optimization objectives, i.e. the minimal area and minimal interconnection, the algorithms are tested on the MCNC benchmark circuits with different values of $\alpha$ and $\beta$. For each of the configuration, the tests are conducted for 50 independent runs and recorded the minimal and best results. For the values of $\alpha = 1$ and $\beta = 0$, the
Figure 4.13 Chip area vs. population size of ami33 circuit

Figure 4.14 Runtime vs. population size of ami33 circuit
objective function area is optimized and for \( \alpha = 0 \) and \( \beta = 1 \), the wire length is optimized considerably. Table 4.3 shows minimum / average distribution with different weights for SP based DE algorithm. DE algorithm generated compact layouts for each benchmark circuit and the constraint graphs and layouts are shown from Figures 4.15 to 4.28.

Table 4.3  Minimum / average distribution with different weights for SP based DE algorithm

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Area weight =1.0</th>
<th>Area weight = 0.6</th>
<th>Wire weight = 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area ( (\text{mm}^2) )</td>
<td>wire ( (\text{mm}) )</td>
<td>Area ( (\text{mm}^2) )</td>
</tr>
<tr>
<td>hp</td>
<td>8.83/8.9</td>
<td>103/140</td>
<td>9.3/10.5</td>
</tr>
<tr>
<td>ami33</td>
<td>1.18/1.21</td>
<td>55.5/68</td>
<td>1.25/1.32</td>
</tr>
<tr>
<td>ami49</td>
<td>36.0/36.4</td>
<td>689/890</td>
<td>37.6/40.5</td>
</tr>
</tbody>
</table>

Figure 4.15 Constraint graph generated for apte circuit
Figure 4.16 Floorplan of apte circuit

Figure 4.17 Floorplan of apte circuit with nets
Figure 4.18 Constraint graph generated for xerox circuit
Figure 4.19 Floorplan of xerox circuit
Figure 4.20 Floorplan of xerox circuit with nets
Figure 4.21 Constraint graph generated for hp circuit
Figure 4.22 Floorplan of hp circuit

Figure 4.23 Constraint graph generated for ami33 circuit
Figure 4.24 Floorplan of ami33 circuit
Figure 4.25 Floorplan of ami33 circuit with nets
Figure 4.26 Constraint graph generated for ami49 circuit
Figure 4.27 Floorplan of ami49 circuit
4.6 CONCLUSION

From this study the following points are observed:

1. The DE is a population-based algorithm, providing significant improvement in the optimal results by achieving the global optimum, compared to SA.

2. For the floorplanning problem with the sequence pair representation, from the ‘speed’ point of view, DE/rand/… strategies are better than DE/best/… for the selected seed values of 5.
3. DE/rand/1/bin strategy is found to be the best out of the presently available strategies of DE.

4. DE is robust. It is able to reproduce the same results consistently over many runs.

5. DE, a simple evolution strategy is significantly faster compared to SA. Binomial crossover finds to be better than that of exponential, as it yields the global minimum in less number of generations. DE has more strategies to choose from, which is an advantage over other algorithms such as GA and PSO. From the results, it is evident that DE has shown remarkable performance within 20\textsuperscript{th} generations and yielding optimal results, which PSO could achieve only in 38\textsuperscript{th} generations for ami33 circuit. Hence it is concluded that DE is significantly faster than PSO.