A Modified SA Algorithm for VLSI floorplan

Aravindaraj P, Sujatha K
Dept of Electronics & Communication
BMSCE Bangalore, 560019

DOI: 10.29322/IJSRP.9.09.2019.p9335
http://dx.doi.org/10.29322/IJSRP.9.09.2019.p9335

***Abstract*--In recent year as the complexity of Integrated circuits increases the physical design complexity also increases. As the technology advances the Integrated circuits are expected to be low area as possible. To reduce area of total chip, we use some efficient algorithms for floorplan. A floorplan optimization is usually about placing all modules inside a partition so that the total area of block should be less as possible, to get the solution for this NP hard problem we use some meta-heuristic algorithms such as Simulated annealing(SA), Particle swarm optimization(PSO), Ant colony optimization(ACO), Genetic algorithm(GA) etc. Basically, all these algorithms try to reduce the total area required for the whole partition by placing modules in certain way. Different algorithm gives different solution for floorplan problem, but it is important to get the better solution for floorplan with less run time of algorithm. Since the complexity of Integrated circuits are increasing the number of modules in partition also increases, hence it may increase the run time of algorithm to get the better solution. In this work, the proposed modified Simulated Annealing algorithm can get the better solution in less run time compared to older Simulated Annealing algorithm implemented.**

***Index Terms*—VLSI floorplanning, B*tree, Simulated Annealing, slicing and non-slicing, Dead space etc.**

INTRODUCTION

Floorplanning is one of the initial stages in Physical Design flow, this involves the process of identifying structures that should be placed close together, and allocating space for them in such a manner as to meet the sometimes conflicting goals of available space (cost of the chip), required performance, and the desire is to have everything close to each other so that the effective area of floorplan should reduce. Floorplanning is basically an NP- hard problem. It is important to get an effective solution for this problem since the resulting floorplan affects all the subsequent steps in physical design, such as placement and routing. For solving this problem there is no known polynomial algorithm can solve it in a polynomial time. For solving floorplanning problem, most commonly used algorithms are SA (Simulated Annealing), PSO (Particle Swarm Optimization), GA(Genetic algorithm), ACO(Ant Colony Optimization) etc. Simulated Annealing is one of the popular algorithms used to solve floorplan problem.

Since floorplanning is a NP -hard problem, the search space increases exponentially as the number of modules increases, therefore it is difficult to get an effective solution in reduced run time. The quality of the floorplanning depends on the fact that how it is represented.

The representation of floorplan determines the size of search space and the complexity of problem. There are 2 types of floorplans sliceable and non-sliceable and there are various representations methods such as Bounded sliced Grid (BSG), Corner Block List (CBL), Transitive Closure Graph (TCG), B*tree, O-tree, Sequence pair etc.

PROBLEM STATEMETNT

The input data of floorplanning and placement problem is the information of each module with length, width, area of all modules and connection information between these modules. A rectangular floorplanning region P is given as shown below Fig 1.1, with W width and H height, a set of modules P= {b_1, b_2, b_3, ……..b_n}, in which mi stands for rectangular block with fixed width(w_i) and height(h_i).

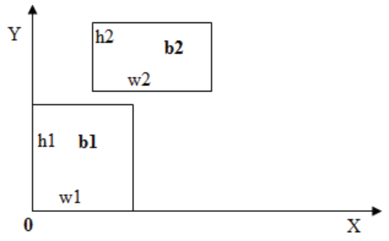

Fig1 Floorplan area

N is the netlist which specifies the interconnection information between the modules in floorplan. The main problem of this floorplan and its placement is to arrange or pack all modules into rectangular floorplan area P. The input modules are specified in Netlist N with its fixed w_i and h_i. To achieve the goal of optimization, the following condition should meet

i. There should not be change in input values and their interconnection.

ii. There must be no overlap between modules with each other.

iii. Each module must be lie in the rectangular shape with w_i and h_i.

iv. All module must be parallel to the coordinator axis of floorplan region.

FLOORPLAN REPRESENTATION

For optimization of floorplan F, each module can be represented by B* tree, as shown in fig 2(a). A B* tree is known as binary tree whose rood corresponds to the bottom left corner module. We design tree using depth first search procedure for an admissible floorplan F and optimized placement in recursive manner.

For representing in B* tree, first construct the left sub-tree. The B*-tree is having the direct geometric relationship between all modules with respect to nodes of its tree. As shown in fig 2(b) node rd_0 is in directly relation with b_0 root node of nd_i, is placed first in floorplan area layout, whereas right node of nd_i is placed in F at the left side root module b_0 and vice versa.

The construction of a B* tree begins from the root, the left child of node rd_i corresponds to the lowest, unvisited block in floorplan. The right child of rd_i represents the lowest block located above and with its x-axis equal to that of b_i(i^{th} module) and its y-axis less than that of top boundary of the module of the left -hand side and adjacent to b_i. Each node combination of a B*Tree corresponds to a floorplan, so the solution space consists of all combinations of B*Tree obtained by perturbing the Tree by three movements namely: Node Movement (deleting a node and inserting in some other locations), Node Swap (swapping two blocks) and Block Rotation (No change in tree structure).

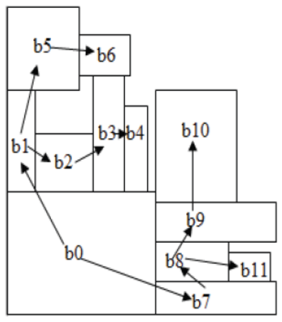

Fig 2(a) Floorplan

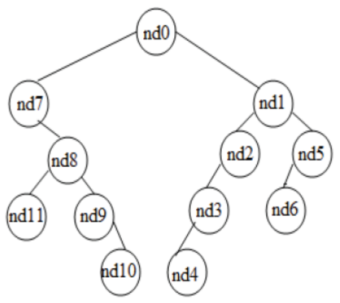

Fig 2(b) B*-tree Representation

FLOORPLAN ALGORITHM

In this paper the classical simulated annealing is modified in order to get an effective solution for floorplan problem in reduced run time. Simulated annealing is inspired by an analogy between the physical annealing of solid and combinatorial enhancement. In physical annealing process a solid is melted first and then cooled slowly, spending a long time at low temperatures, to obtain a perfect lattice structure at minimum energy state. In SA this analogy is transferred to local search algorithms for combinatorial enhancement problems. It does so by associating the set of solutions of the problem with the states of the physical system, the objective function with the physical energy of the solid, and the optimal solutions with the minimum energy states.

In classical SA at very low variations of temperature parameters and when the solution search for each temperature can reach an equilibrium condition, have very high chances of finding the global optimal solution. But it requires very slow temperature variations and thus increase the required computational effort. So, in this paper we implemented a modified SA algorithm to get an effective solution in reduced run time, thereby reducing the complexity. So, in this work we are defining different expression for Temperature variable 'T' to reduce the runtime of the algorithm and getting optimum solution for floorplan problem. We initially randomly perturb to get the initial solution and continued. The idea is to reduce uphill moves in the beginning, because most of the uphill at this stage lead to inferior solution.

Algorithm for Modified Simulated Annealing:

```
Start
if local ==0
avg=est_avg;
assign local_T=avg/log(P);
get initial solution as best;
    do {
    count=count+1;
    for (; uphill<N && MT<2*N; MT++)
    {get solution using perturb ();
    get new cost as cost;
    get difference between pre-cost and new cost as d_cost;
    calculate probability value p;
    if (d_cost <=0 or rand () <p)
    {keep obtained solution;
    assign pre_cost=cost;
    if(d_cost>0)
    uphill++, bad_num++;
    else
    good_num++;
    if (cost < best)
    assign cost as best;
    }
    else
    reject++, recover_sol();
    }
   Calculate standard_variance and variable r_t;
    assign local_T = r_t*local_T
    if(count==local)
    local_T= estimate_avg/ log(P);
    local_T= T * 0.9local;
    actual_T= exp(estimate_avg/local_T);
    if(count>local)
    actual_T = exp(estimate_avg/local_T);
    assign conv_rate as 0.9;
    reject_rate =reject_rate/MT;
} while (reject_rate < conv_rate && actual_T > term_T) ;
```

A. *Calculations for Modified SA*

The Probability function for SA is

$$\text{Probability} = e^{\left(\frac{\Delta c}{T}\right)}$$

The temperature T at initial stage is given by,

T= avg/ log(P)

where 'avg' is a variable.

And get an initial solution.

For further iterations temperature is defined as,

T= r_t*T

Where, $r\_t = e^{\left(lamda * \frac{T}{sv}\right)}$

standard variance,

$$sv = \sum_{i=1}^{N} \sqrt{\left(\frac{Si}{N-1}\right)}$$

$$si = (ai - m)^2$$

where, mean m = Total area/ N

B. *Cost Function*

Area and wire-length are perhaps the two most commonly used costs for floorplan design. We can adopt the following cost function for simulated annealing:

Cost function used is, $Cost = \frac{\alpha A}{Anorm} + \frac{(1-\alpha)W}{Wnorm}$

where A is the floorplan area, A_{norm} is the average area, W is the total wirelength, W_{norm} is the average wirelength, and a control the weight between area and wirelength. To compute A_{norm} and W_{norm}, we can perturb the initial normalized Polish expression by use of the three operations for m times to obtain m floorplans and compute the average area A_{norm} and the average wirelength W_{norm} of these floorplans. The value m is proportional to the problem size.

IV.PERFORMANCE AND RESULT ANALYSIS

In this paper, we have used MCNC benchmark problems to study the performance of modified SA algorithm. This algorithm is implemented by C++ programming language with Linux on virtual machine environment.

TABLE 1. CHARCTERISTIC OF BENCHMARKS

Benchmarks	No of Modules	Area(mm^2)
ami33	33	1.15
ami49	49	35.44
apte	9	4.65
hp	11	8.83
xerox	10	19.35

TABLE 2: RESULTS OF MODIFIED SA on MCNC netlists

Bench mark circuit	Number of modules	Used Area(mm^2)	Total Area (mm^2)	Wirelength (mm)	Dead space (%)	Run time (sec)
ami33	33	1.1961	1.1564	128.629	3.32	1.38
ami49	49	36.794	35.445	1894.74	3.67	0.29
apte	9	47.31	46.561	755.87	1.59	0.23
hp	11	9.363	8.8305	239.862	5.70	0.05
xerox	10	20.106	19.350	779.72	3.76	0.07

The experimental result of Modified SA using MCNC benchmark shows better results in terms of area reduction and reduced run time compared to other previous algorithms.

TABLE 3: Comparison of areas with different algorithms

MCNC Benchmarks	SA with Sequence pair	SA with B* tree	PSO with sequence pair	PSO with B* tree	Hybrid PSO	Modified SA
ami33	1.31	1.36	1.31	1.28	1.28	1.19
ami49	38.84	43.34	38.84	41.01	42.30	36.79
apte	48.12	47.30	48.12	46.92	51.59	47.31
xerox	20.69	20.47	20.69	19.55	21.70	20.10

CONCLUSION

The Modified Simulated Annealing algorithm is implemented using C++ coding language and tested with MCNC benchmark netlists. This algorithm is efficient compared to earlier algorithms, which is shown in table 4.2. The prosed Modified Simulated Annealing algorithm can get reduced area for floorplanning comparatively and the reduced run time. Hence this algorithm can give effective solution for area for floorplanning like classic SA algorithm, but it takes very less runtime to find the better solution (since in classic SA algorithm, the runtime will be more and so as the computation complexity). This algorithm is well suited since, as the technology grows the transistors count in design increases so as the complexity of design increases, hence there is a need to reduce the total area of the chip, and the run time for EDA tools also important. This work mainly concentrated on reducing area and runtime to get the solution for the floorplan, but wirelength results in this algorithm are comparatively degraded.

References:

1. Rajendra Bahadur Singh & Dr. Anurag Singh Baghel “Dead Space Reduction of Floorplan using Simulated Annealing Algorithm” International Conference on Energy, Communication, Data Analytics and Soft Computing, 2017 IEEE.
2. Dalfard VM. Adjustment of the Primitive Parameters of the Simulated Annealing Heuristic. Indian Journal of Science and Technology. 2011 Jun; 4(6):1-5.
3. Chen TC, Chang YW. Modern floorplanning based on B-tree and fast simulated annealing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.2006 IEEE
4. J. Jenifer, S. Anand and Y. Levingstan “Simulated Annealing algorithm for modern vlsi floorplanning problem” ICTACT JOURNAL ON MICROELECTRONICS, APRIL 2016 IEEE
5. Amarjot Kaur & Dr. Sandeep Singh “Hybrid swarm inetelligence for vlsi floorplan” International Conference on Computing, Communication and Automation (ICCCA2016), 2016 IEEE.
6. Xiaodong Yang, Junying Niu & Zefan Cai “Chaotic Simulated Annealing Particle Swarm Optimization Algorithm” 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference(IMCEC 2018).
7. Leena Jain & Amarbir Singh “Non Slicing Floorplan Representations in VLSI Floorplanning: A Summary” International Journal of Computer Applications (0975 – 8887) Volume 71– No.15, June 2013
8. Naushad Manzoor Laskar, Rahul Sen, P.K.Paul & K.L.Baishnab “A Survey on VLSI Floorplanning: Its Representationand Modern Approaches of Optimization”
IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems.
9. AP Engelbrecht “Fitness Function Evaluations: A Fair Stopping Condition?” 2014, IEEE.
10. Matthew H. Jones ,K. Preston White, Jr. “Stochastic Approximation With Simulated Annealing As An Approach To Global Discrete-event Simulation Optimization” 2004, IEEE.