Improving Placement in VLSI Design Process via Hybridization of Simulated Annealing and Genetic Algorithms

Nada Adel*, Nader AbdAlGhani†, Nourhan Gamal‡ and Sarah Raafat§
Computer Engineering Dept., Faculty of Engineering, Cairo University
Cairo, Egypt
*nadoza.98@gmail.com, †nader_abdelghani@hotmail.com, ‡nourgamal1498@gmail.com, §sararaafat1@yahoo.com

***Abstract*—Vestibulum rhoncus est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat vivamus at augue eget arcu dictum varius duis at consectetur lorem donec massa sapien faucibus et molestie ac feugiat sed lectus vestibulum mattis ullamcorper velit sed ullamcorper morbi tincidunt ornare massa eget egestas purus viverra accumsan in nisl nisi scelerisque eu ultrices vitae auctor eu augue ut lectus arcu bibendum at varius vel pharetra vel turpis nunc eget lorem dolor sed viverra ipsum nunc aliquet bibendum enim facilisis gravida neque convallis a cras semper auctor neque vitae tempus quam pellentesque nec nam aliquam sem et tortor consequat id porta nibh venenatis cras sed felis eget velit aliquet sagittis id consectetur purus ut faucibus pulvinar elementum integer enim neque volutpat ac tincidunt vitae semper quis lectus nulla at volutpat diam ut venenatis tellus in metus vulputate eu scelerisque felis imperdiet proin fermentum leo vel orci porta non pulvinar neque laoreet**

***Index Terms*—Very-large-scale integration (VLSI), Standard cell placement, Computer-aided design (CAD), Simulated annealing, Genetic algorithms, Integrated circuits (ICs).**

I. Introduction

The result of the astronomical impact of the very large scale integration (VLSI) industry on the global economy and the extreme widespread use of electronic devices and equipment in various fields across the globe have led to the exponential increase in demand and sophistication of electronic devices. This, of course, goes along with Moore's law which refers to Gordon Moore's projection in 1965 that the number of transistors crammed in integrated circuits (ICs) doubles every two years, though their cost is halved. This increase in the number of transistors put in an IC comes at a cost which are the problems that arise during both the design process and the fabrication process of such intricate devices. This paper addresses the role of placement in the physical design process. Placement tools are used mainly for two things, the first is to place cells in a suitable way for the consequent routing process, and the second is to guide both synthesis and floorplanning processes. Over the years, placement has no longer become a fine-tunning tool used in physical design, but for that matter, it has become a notable contributor to the timing closure process results as well as being an overall major step in physical design. Conventionally, placement is divided into two subprocesses, global placement and detailed placement, however, the scope of this paper will focus exclusively on the former. Global placement aims to evenly and legally distribute the cells over the available placement region putting in mind cells relative positions to each other globally regardless of local problems that may arise, hence the name. This goal is achieved in favour of certain cost metrics such as overall wirelength and the time taken to achieve such a goal. The placement problem is to place the cells in such a way to achieve the best possible cost metrics. Much like the well-known travelling salesman problem and most, if not all, physical design problems, it is an NP-complete problem in combinatorial optimization. Most of the attempts at solving this problem use heuristic algorithms instead of the ineffective brute force trivial approach. Among those metaheuristics are simulated annealing which is popularly used in the optimization of this particular problem, and genetic algorithm which is used for optimization problems in general. In this paper, we implemented the best of both worlds to achieve overall better optimization in terms of wirelength and time.

II. Related Work

As already stated, the placement problem is an NP-complete problem where its optimum solution can be found by trying all possible cells configurations, but due to the tremendous number of cells that exist in almost all VLSI designs which range from millions to billions of cells, it would be utterly impractical to compute every possible configuration in order to find the fittest one. Because of this nature, previous literature made significant improvements in terms of solution fitness and the amount of time consumed through using heuristic techniques such as simulated annealing used in [7], [10], [11], [12], [13] and [14], min-cut placement algorithm used in [1] and [15] and genetic algorithms used in [2], [3] and [7]. However, optimizing placement algorithms isn't limited to heuristic approaches only as others have used different approaches to achieve solid results. For instance, Natarajan et al [8] used an analytical placement algorithm that is based on the quadratic placement approach.

A. Simulated Annealing

Simulated annealing is one of the popular and most effective algorithms among all placement algorithms, in fact, it is generally used as an optimization technique for combinatorial problems. The basic idea of this method comes from the

annealing treatment in metallurgy. Annealing in metallurgy is used to accomplish a crystalline lattice structure of the particles of solid metal as it reflects a minimum energy state for the solid. This is achieved by heating the metal in a heat bath until it reaches a temperature named the initial temperature of the annealing process at which all particles are randomly arranged, then whenever the metal reaches thermal equilibrium the temperature is reduced gradually according to a cooling schedule otherwise the resulting crystal will not mirror a minimum energy state. This analogy applies to the simulated annealing metaheuristic where an initial value of temperature T is set and at each iteration, the positions of the cells get scrambled around legal regions then the cost function C is measured, a result is accepted whenever it leads to a decreasing cost function, however, if a result leads to an increasing cost function, it is accepted if and only if the move has a probability greater than a random value between 0 and 1. The probability function is defined as follows:

$$P = e^{-\Delta C/T} \tag{1}$$

At the end of each iteration, the temperature value decreases by multiplying it by a cooling factor which is less than 1. This process iterates until the temperature goes below 1. It is clear that the performance of this algorithm is based upon four parameters/steps:

1) Initial Configuration: In this step, the individual cells are extracted from the circuit and for each cell, the inputs and outputs are determined. The following diagram and table show the circuit is decomposed.

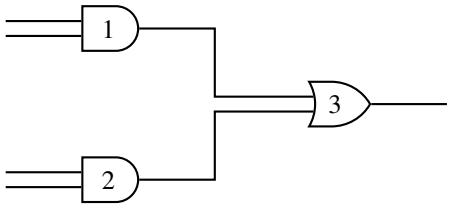

Fig. 1. Sample circuit diagram

TABLE I
SAMPLE CELLS DATA

Cell	*Input*	*Output*
1	-	3
2	-	3
3	1	-
3	2	-

At this step, the total wirelength of cells interconnections will most likely be large due to them being placed randomly. Hence the presence of the rest of the subsequent procedures which aim to achieve the optimal placement of the cells.

2) Move Generation Function: At each iteration, cells are rearranged to generate a new placement configuration of the cells. To do so, there are two possible strategies used:

- Move cells randomly to new legal positions
- Swap the positions of each two cells

3) Cost Function: The cost function consists of two terms as follows:

$$C = C_1 + C_2 \tag{2}$$

Where C_1 is the measure of the total wirelength and C_2 is the measure of total overlap of the chip which is sometimes neglected as in our case but it is used in other papers [16] nevertheless.

Let d_h^{ij} d_v^{ij} be the horizontal and the vertical distance between cell i and its j^{th} out cell respectively, the total wire length of the chip can be expressed as the following equation

$$C_1 = \sum_{i=cell}^{n} \sum_{j=outcell}^{n} (d_h^{ij} + d_v^{ij}) \tag{3}$$

Where n is the total number of cells in the chip. In case of the second term C_2, whenever two cells are swapped, they might overlap, therefore, O_{ij} indicates the overlap between cell i and cell j, this overlap is undesirable and it negatively affects the cost function by increasing its value by the term C_2 which is the summation of each overlap value squared.

$$C_2 = \sum_{i!=j} (O_{ij}^2) \tag{4}$$

The cost function is calculated at each iteration after generating a new cells layout.

4) Annealing Schedule: At first, the temperature T is initialized to a very large value, and at every iteration, the temperature is reduced using the following equation:

$$T = \alpha * T \tag{5}$$

Where α is the cooling rate. In our case, it has a constant value of 0.995. Yet, it can be set dynamically in the following fashion: Initially, it is set to a small value which contributes to the rapid decrease in temperature, then in the middle of the annealing process the temperature is reduced slowly via a larger cooling rate value. When the temperature is relatively low, the temperature decreases rapidly again until the stopping condition which is when the temperature falls below 1 is met.

Algorithm 1 Simulated Annealing

Output: Optimized solution S

Create an initial solution x
Set $\alpha \in (0, 1)$, temperature T
Set $S = x$
while Stopping criterion is NOT satisfied **do**
    $x^{'}$ = a neighbouring solution
    Calculate $C(x^{'})$
    Calculate $\Delta C = C(x^{'}) - C(x)$
    if $\Delta C < 0$ **then**
        $x = x^{'}$
        if $C(x) < C(S)$ **then**
            $S = x$
        end if
    else if $rand(0, 1) < e^{-\Delta C/T}$ **then**
        $x = x^{'}$
    end if
    $T = T * \alpha$
end while
return S

There is no doubt that this widely-used algorithm yields excellent results, as a matter of fact, it can produce the global optimum result given enough time which makes it a solid contender for solving the end-case placement problem. Though it still has its set of drawbacks. For instance, it is extremely slow especially if rooting for the global optimum solution, not to mention that it suffers hugely if there is a scarce number of local minima.

B. Genetic Algorithm

Genetic algorithms are metaheuristics based on the concepts of Charles Darwin's theory of biological evolution. Its idea is powered by the natural principle known as survival of the fittest where the fittest individuals have the highest probability of survival and to pass their genes to their offspring for further reproduction. On the other hand, the less fit individuals have the highest probability to die out. This resemblance is commonly used for creating optimization algorithms for combinatorial problems. But given the characteristics of the VLSI design placement problem, a genetic algorithm can also be employed to solve it as in [2]. Genetic algorithms effectiveness is defined by the following parameters:

1) Initial Population: The initial population individuals are a subset of the solution space. They represent the first generation and they are usually created randomly. However, by intuition, the better the first generation is the lesser time consumed by the algorithm to find better individuals. Each individual is represented by a chromosome which reflects its information in an appropriate encoding.

2) Crossover: Crossover is quite similar to sexual reproduction in biology as it is used for combining the genetic information of two individuals "parents" of the current population to generate a new solution. Solutions are usually mutated before being added to the new population.

3) Mutation: The mutation process operates on the output of the crossover process. It introduces new variations in their chromosomes in an attempt to span the whole solution space.

4) Selection: Selection is the criteria with which the selection of the fittest individuals of the generated offspring for later reproduction occurs. It is usually implemented as a fitness function that is evaluated for each individual.

Algorithm 2 Genetic Algorithm Pseudo Code

Output: The fittest Individual according to a certain criteria

Randomly generate an initial population
Evaluate the fitness of each individual of the population
while The fittest Individual NOT found **do**
    Select the fittest individuals
    Generate new individuals using the crossover operator
    Mutate the new individuals
    Evaluate the fitness of the new individuals
    Replace the worst individuals of the population with the best new individuals
end while
return The fittest individual

From [7], we can deduce that as much as genetic algorithms offer lots of advantages like having relatively easy implementations, being faster than simulated annealing to finding fit solutions and being able to span the solution space more rapidly. But despite those benefits, they have some hindering limitations. For example, they cannot find the exact minima value despite spanning most of the solution space quickly and they may also be trapped at local minima.

III. Proposed Approach

As already introduced, both the simulated annealing and genetic algorithms have their set of benefits and shortcomings. This paper approach tries to compensate each algorithm drawback with the advantage of the other. The role of the simulated annealing algorithm in our approach is to generate the initial population of the genetic algorithm used, as having an already-improved initial population will prevent the genetic algorithm from getting trapped in local minima. On the other side of the coin, the genetic algorithm spans the solution space in parallel via its population, which is much faster than the sequential spanning done by the simulated annealing algorithm. The proposed algorithm runs the simulated annealing algorithm on random placement configurations with a temperature value directly proportional to the number of cells in the input circuit to generate output configurations equal to the number of individuals in the initial population of the genetic algorithm. At the receiving end, the genetic algorithm evaluates each individual of the initial population according to the fitness function which sorts them with respect to their total wirelength, the shorter, the better. The crossover process is done by creating a new child placement configuration where each cell in this new placement is taken either from the first or the second parent while making sure that the cell taken at

each crossover iteration does not coincide with another cell at the same coordinates. After the crossover process is executed between all the available cells, the generated configuration is mutated by swapping two randomly-chosen cells.

Algorithm 3 Proposed Algorithm

Input: Circuit C_o
Output: Optimized placement configuration η

Generate an initial random cells placement P
while Population σ is NOT filled completely **do**
    Set $\alpha = 0.995$
    Set temperature $T = 2 * number\ of\ cells\ of\ C_o$
    Set optimized circuit configuration $\Phi = P$
    while $T > 1$ **do**
        $P^{'}$ = randomly generated cells placement
        $\Delta C = wirelength(P^{'}) - wirelength(P)$
        if $\Delta C < 0$ **then**
            $P = P^{'}$
            if $wirelength(P) < wirelength(\Phi)$ **then**
                $\Phi = P$
            end if
        else if $rand(0, 1) < e^{-\Delta C/T}$ **then**
            $P = P^{'}$
        end if
        $T = T * \alpha$
    end while
    Append Φ to σ
end while
while The termination condition is NOT satisfied **do**
    Sort σ in descending order by the fitness level
    $\sigma^* = crossover(\sigma)$
    $\sigma^* = mutation(\sigma^*)$
    foreach $\sigma_i^* \in \sigma^*$ **do**
        Evaluate σ_i^* fitness level
    end foreach
    $\sigma = \sigma^*$
end while
η = the fittest individual of σ
return η

IV. Experimental Analysis

A. Assumptions

Insert assumptions here

B. Experimental Scheme

Insert I/O here

C. Performance Metrics

Insert performance metrics here
Insert simulation output here

D. Performance Comparisons

Insert experiments comparisons here

E. Observation

Insert observation here

V. Conclusion and Future Work

Insert conclusion and future work here

References

[1] Saab, Youssef, "A fast clustering-based min-cut placement algorithm with simulated-annealing performance", *VLSI Design*, vol. 5, no. 1, 1996, pp. 37-48.

[2] H. Esbensen, "A genetic algorithm for macro cell placement", *European Design Automation Conference*, Hamburg, Germany, 1992, pp. 52-57.

[3] K. Shahookar and P. Mazumder, "GASP-a genetic algorithm for standard cell placement", *Proceedings of the European Design Automation Conference*, Glasgow, UK, 1990, pp. 660-664.

[4] J. Vygen, "Algorithms for detailed placement of standard cells", *Proceedings Design, Automation and Test in Europe*, Paris, France, 1998, pp. 321-324.

[5] Min Pan, N. Viswanathan and C. Chu, "An efficient and effective detailed placement algorithm", *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005*, San Jose, CA, USA, 2005, pp. 48-55.

[6] Z. Jiang, H. Chen, T. Chen and Y. Chang, "Challenges and Solutions in Modern VLSI Placement", *2007 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, Hsinchu, 2007, pp. 1-5.

[7] Elhaddad, Younis, "Combined simulated annealing and genetic algorithm to solve optimization problems", *World Academy of Science, Engineering and Technology*, 2012.

[8] N. Viswanathan and C. C. -. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 5, May 2005, pp. 722-733.

[9] D. Zaporozhets, D. Zaruba and N. Kulieva, "Hybrid heuristic algorithm for VLSI placement", *2019 International Russian Automation Conference (RusAutoCon)*, Sochi, Russia, 2019, pp. 1-5.

[10] R. F. Hentschke and R. A. D. L. Reis, "Improving simulated annealing placement by applying random and greedy mixed perturbations [IC layout]", *16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings*, Sao Paulo, Brazil, 2003, pp. 267-272.

[11] K. Vorwerk, A. Kennings and J. W. Greene, "Improving Simulated Annealing-Based FPGA Placement With Directed Moves", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, Feb. 2009, pp. 179-192.

[12] Atanu Roy, Karthik Ganesan Pillai, "Parallel simulated annealing for VLSI cell placement problem", 2009.

[13] J. A. Chandy and P. Banerjee, "Parallel simulated annealing strategies for VLSI cell placement", *Proceedings of 9th International Conference on VLSI Design*, Bangalore, India, 1996, pp. 37-42.

[14] J. W. Greene and K. J. Supowit, "Simulated annealing without rejected moves", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, no. 1, January 1986, pp. 221-228.

[15] A. B. Kahng and S. Reda, "Wirelength minimization for min-cut placements via placement feedback", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, July 2006, pp. 1301-1312.

[16] Shahookar, K. and Mazumder, "VLSI cell placement techniques", *ACM Computing Surveys*, vol. 23, no. 2, pp. 145-220.