An Educational Tool For Teaching Simulated Annealing And Placement

Tania Mara Ferla, Guilherme Flach, Ricardo Reis
{tmferla, gaflach, reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul (UFRGS)
PGMICRO / PPGC - Instituto de Informática
Av. Bento Gonçalves 9500
Porto Alegre - Brasil

Abstract

This paper presents an educational tool which can be used for teaching the Simulated Annealing (SA) algorithm. The SA is applied to solve integrated circuit placement. SA is an effective method to solve NP-complete problems as circuit placement. The interface developed in this work provides a visualization of the execution steps of the SA, which makes the tool more iterative and didactic. The tool is developed using web standards (HTML 5, CSS and JavaScript) and hence it is accessible across multiple platforms.

1. Introduction

In an integrated circuit design some goals, such as area, speed, power dissipation, design time, testability of the circuit and the total cost must be taken into account. To achieve these goals, some steps of the synthesis of the circuit layout are performed, such as partitioning, placement and routing.

The circuit placement step defines the cell positions inside the circuit area, without overlap, trying to reduce the length of the connections between the cells [1]. The placement methods can be divided into three main categories: (1) simulated annealing, (2) partitioning and (3) analytical.

The SA [4] is known to be a method capable of providing near-optimal solutions, but at a high time. Therefore it can only be fully applied to relatively small circuits (tens of thousands of cells) in feasible time. However, the SA can be successfully applied in hybrid solutions such as the placer Dragon [5] which mixes partitioning and SA. SA can be used to solve minor problems derived from the complete problem.

In this work, we present an interactive and configurable placement tool based on Simulated Annealing. Our intent with the implementation of this placer is to provide an interactive teaching tool that can be used as teaching material for SA and placement, enabling a better understanding of the functioning of the SA algorithm.

Our tool runs on any Internet browser that supports HTML5, enabling it to be used in various electronic devices such as computers and tablets without requiring installation.

This paper is divided as follows: section 2 presents tools for teaching VLSI; of section 3 presents the basic concept of SA. The implementation of the algorithm is detailed in section 4. Section 5 will describe the interface of SA. Section 6 shows the results of tests performed in the algorithm. And Section 7 shows the conclusions of this work.

2. Teaching tools for VLSI

With the spread of distance learning are being increasingly developed Web tools that allow access to various types of content dynamically via the internet. This is no different in microelectronics, where web technologies can be used not only for distance learning, but to improve and support learning in the classroom. Today there are various applets, which allow the visualization of algorithms often used in VLSI designs, such as in [6]. Thus enabling the visualization step by step how they work, allowing us to understand more quickly what is its dynamics. There are online resources, where various resources are available for the area of VLSI, physics and mathematics, for example [6] [7] [8] and [9]. Some educational tools aiming simulated annealing have already been developed as the Blue Macaw [14] created in our research group GME.

3. Simulated Annealing

The SA is a metaheuristic optimization method proposed by in 1983 by Kirkpatrick et al. [11] based on a search method proposed by Metropolis et al. [12] in 1953. SA is used to solve several NP-complete problems in the area of EDA, but also in many other areas such as telecommunications, biology, geology, computer science and medicine [13].

The SA has been based on an analogy to a process known as annealing from thermodynamics, used to melt metal in metallurgy. Basically, the process has the following steps: increasing of the metal temperature to a maximum value and then performing a cooling slowly back to room temperature. High temperatures allow

metal atoms to freely move. As the temperature is decreased slowly the atoms tend to organize themselves in a cristal structure increasing the metal properties such as strength and hardness. Following these principles, the SA algorithm can be defined primarily by three main functions: the random perturbation function, the cost calculation function, and the function that accepts a perturbation or not, as shown in the code below [3].

```
While (temp*=0.99 > final value){
        perturbation();
        If (new_cost < current_cost || rand() < e^(-deltacost / temp) )
        Accepted ();
                    else
        Rejected ();
}
```

Where:

- *temp*: temperature or number of iterations
- *perturbation()*: move the selected cell to a randomly selected location
- *new_cost*: Recalculates the cost, with the updated values of the HPWL and overflow
- *current_cost*: previous perturbation cost
- *rand*: random number between 0 and 1
- *deltacost*: calculates the delta cost, that is the difference between the new cost and the old cost
- *Accepted():*accepted the disturbance and *Rejected():*rejected the disturbance and discards the movement

If the solution generated by the perturbation is better than the previous one, the perturbation is maintained. Otherwise, a random number in the interval [0,1] is generated and if it is less than (1) the perturbation is maintained.

$$e^{-delta/T} \quad (1)$$

The acceptance function (1) depends on two variables: delta and T. The delta is the cost obtained by variation of the disturbance. And T is the temperature which must be adapted according to the size of the circuit - the higher the circuit the greater the temperature.

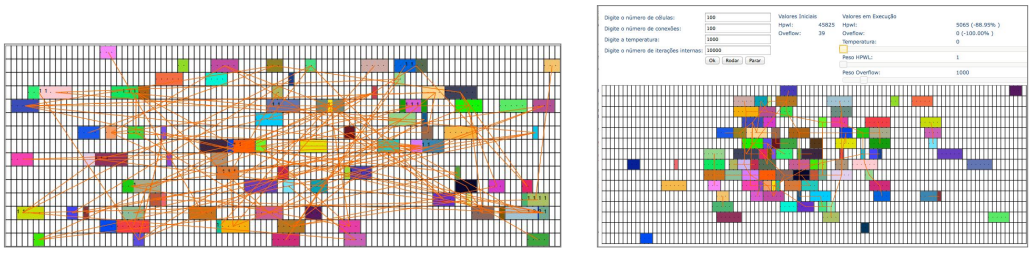
The higher the temperature the greater the value generated by (1) and consequently the greater the chance of a disturbance to be accepted. Similarly, the higher the delta (i.e. the greater the increase in the cost), the lower the likelihood of acceptance.

SA at high temperatures behaves as a random algorithm, (i.e. random perturbations and does accept almost all of them). But at low temperatures behaves as a greedy algorithm, (i.e. only accepts perturbations that improve the current state of the solution). And the more greedy, more likely to get stuck in a local minimum, because only accepts the best results.

In the case of placement implemented, perturbation function randomly selects a cell positioned and also at random. The function that calculates the cost, has the half-perimeter wire length (HPWL) [10] as the cost.

The HPWL is an estimate of the length of the net required to connect the points of a network. It is calculated by the semi-perimeter of the minimum rectangle that encompasses all net nodes.

Figure 1 (a) shows the initial distribution of the cells and their respective connections and (b) the final result, after the simulation. One can observe the initial HPWL (a) before the placement and SA was as placement, and the overflow HPWL after placement (b).



(a) (b)

Fig. 1 – (a) HPWL initial=45825 e (b) final=5065

It can be observed in figure 1 that in order to find the lowest HPWL, SA put close cells that connect to each other. In this simulation depicted in Figure 1, the overflow (overlapping cells) began with 39 but reduced to 0.

4. Implementation

The SA of this work was developed to run on regular web browsers using JavaScript and HTML5. Also, the jQuery frameword was used to draw the bars values of parameters that can be dynamically changed as the HPWL, overflow and temperature. The main function of the SA can be basically defined by figure 2. The steps are of the function (according to figure 2):

- Randomly selects a cell
- Calculates the current cost that is equal to:
  *weightHPWL*hpwl_current+ weightOverflow*overflow_current*
- Perturbates the current solution: move the selected cell to a randomly selected location
- Recalculates the cost, with the updated values of the HPWL and overflow
- Calculates the delta cost, that is the difference between the new cost and the old cost

And finally, if the delta cost is less than 0 (meaning an improvement in the result) or if it is a random number between 0 and 1, it accepts the perturbation, otherwise it rejects and discards the movement.

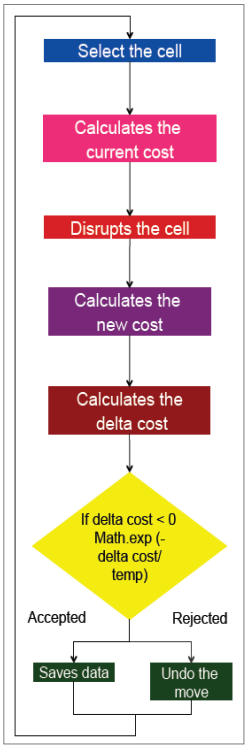
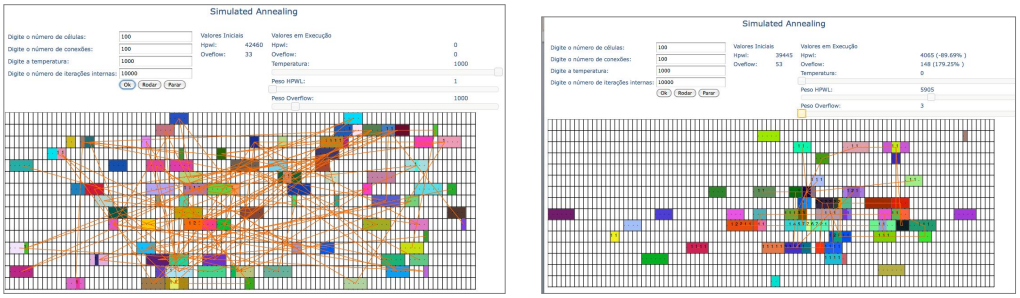


Fig. 2 – Flowchart of the main function of the SA

5. Interface

One of the objectives of this study was to develop an interface which would enable to observe the SA algorithm execution (http://www.inf.ufrgs.br/~tmferla/teste/sa.html), as shown in Figure 3 (a).



(a) (b)

Fig. 3 (a) Interface of SA and (b) Weight change of the overflow of cells during the execution

It can be seen in Figure 3 (a) that several parameter were set to default values, for example, temperature, number of iterations internal HPWL and the weight of the overflow. But the interface allows the user to modify these defaults by entering the number of cells, connections, temperature and number of inner iterations. The number of inner iterations is the number of times the algorithm executes before redrawing the screen, for example if one enter 1, you can track the position of cells by clicking the "Parar" and "Rodar", (i.e. the SA running step-by-step). In a simulation with small number of cells one can see exactly what are the movements of each cells, the current overflow and the current HPWL.

The interface allows to visualize the effects of changing some parameters (HPWL, overflow and temperature), even while running, for example, by changing the weight of the overflow (overlap) to a minimum value cells tend to concentrate, as shown in figure 3 (b).

6. Results

Tests were performed comparing the initial HPWL and final HPWLand comparisons between the initial and overflow resulting from the execution of the algorithm 10 circuits were defined randomly. The temperature, the weight of the overflow, weight of the HPWL were defined in all the tests as "1000", "1" and "1000" respectively. The results are shown in Table 1.

TABLE 1 - Comparison hpwl and overflow before and after execution

	# cells	# lines	HPWL initial	HPWL final	Overflow initial	Overflow final
C1	100	100	47915	5120 (-89,3%)	51	0
C2	350	400	182595	49195(-72,06%)	407	0
C3	700	535	236925	26150 (-88.96%)	1273	946 (-25.69%)
C4	805	638	274955	27080 (-90.15%)	1572	1323 (-15.84%)
C5	599	579	258905	46540 (-82.02%)	978	574 (-41.31%)
C6	458	685	302030	139390 (-53.85%)	637	155 (-75.67%)
C7	235	123	56470	4850 (-91.41%)	193	0
C8	522	352	155320	26235 (-83.11%)	815	373 (-54.23%)
C9	255	300	130520	27145 (-79.20%)	225	0
C10	315	600	268185	92820 (-65.39%)	329	0

Based on the analysis results of the tests, it can be concluded that this algorithm has been more successful with smaller circuits, for example up to 400 cells. The drawback of SA developed in this work is the execution time, which is very slow. However the developed tool is intended to be an educational tool. For the educational point of view our interface allows manually changes in the parameters as HPWL, overflow and temperature, allowing the visualization of the effects that they cause in the whole process, for example, the trade-off that occurs between the HPWL and overflow.

7. Conclusions

The SA developed in this project allows the user to have a good view of how the algorithm works and is dynamic, allowing the SA parameter to be changed during execution. Another feature is that the algorithm can be run step by step and allowing a better understanding of some basic concepts of placement, such as HPWL. The SA is also more accessible, since it is designed to be run via the web and can be accessed on page http://www.inf.ufrgs.br/ tmferla ~ / test / sa.html.

Even now there are many tools for teaching about various topics within the area of integrated circuit design, it is clear that much remains to be developed, as this area is very wide, especially Web, which allows the creation of materials more accessible.

8. References

[1] Johann, M. Algoritmos para Síntese Física. EMICRO 2004 http://lcr.uns.edu.ar/electronica/Posgrado/EAMTA/Documents/CAD/Part3/EAMTAmjem2k4final.ppt

[2] Reis, Ricardo A. L.. Concepção de Circuitos Integrados. Porto Alegre: II da UFRGS, 2000. 252 p.

[3] Reis R. Johann, M. Notas de Aula. http://www.inf.ufrgs.br/~johann/cmp241/aula07.partplacefloor.ppt.pdf

[4] S. Kirkpatrick, C.D. Gelatt, Jr., e M.P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671–680, Maio de 1983.

[5] Dragon. http://er.cs.ucla.edu/Dragon/

[6] CADAPPLETS Animations of VLSI CAD Algorithms.http://workbench.lafayette.edu/~nestorj/cadapplets

[7] Falstad http://www.falstad.com/mathphysics.html

[8] Formation of a PN Junction Diode and its Band Diagram http://fiselect2.fceia.unr.edu.ar/fisica4/simbuffalo/education/pn/pnformation_B/index.html#

[9] Visualgos: N Queens http://yuval.bar-or.org/index.php?item=9

[10] A. A. Kennings and I. L. Markov, "Smoothening Max-terms and Analytical Minimization of Half-Perimeter Wirelength" *VLSI Design*, vol. 14, no. 3, 2002, pp. 229-237.

[11] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi, Optimization by simulated annealing, Science, Vol220, No 4598, May 1983, p. 671-680

[12] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, (1953), "Equations of State Calculations by Fast Computing Machines," Journal of Chemical Physics, 21, 1087-1092.

[13] Simulated annealing: theory and applications Kluwer Academic Publishers Norwell, MA, USA ©1987 ISBN:9-027-72513-6

[14] Hentschke, Renato Fernandes ; Johann, M. O. ; Reis, Ricardo. Blue Macaw: A Didactic Placement Tool Using Simulated Annealing. In: Achim Rettberg; Christophe Bobda. (Org.). New Trends and Technologies in Computed-Aided Learning for Computer-Aided Design. Boston: Springer, 2005, v. , p. 37-47.