The American University in Cairo
School of Sciences and Engineering

A Simple Simulated-Annealing Cell Placement Tool

CSCE 3304-01

Maya Hussein & Mariam Abdelaziz

Fall 2022

GitHub Repository: https://github.com/mariamussama/Digital_Design2

Algorithm & Implementation

```
readfile() -
   reads data from netlist file
   Initialize the core array
   Initialize the cells array
   Does the initial placement of the cells
   Save the netlist in an array
   Returns N (number of cells), Core array, Cells array,
   n_con(number of connections), rows, columns
Cells array
     First Index = component number
     Second index = X or Y axis
     Value = X, Y value
Core array
     Index = location of the site
     Value = component number
```

This function Initially initializes the dimension of the 2D core array using the values of rows and columns provided in the file, all sites are initialized by -1 (empty site)

Then, Each component is randomly assigned an x and y value in the core (if not empty) and this data is saved in a 2D-array (cells). The sites are now containing the values of the components assigned to. If they are not assigned to any component, they have a value of -1.

```
HPWL() -
     Finds the max & min length and width within the net
     Computes the half-parameter
          half-parameter  = (l_max - l_min) + (w_max - w_min)
     sums all computed half-parameter to total
```

The Half-Perimeter Wire Length is calculated by finding the min and max lengths through every iteration and then calculates the half-parameter by deducting the max - min lengths and the max- min width. This value is then added to the total length.

```
annealing() -
     T = T_initial = 500 * HPWL of initial core
     T_final = (5 * 0.000001 * HPWL_initial)/no.of nets
     while T > T_final
          pick 2 random core positions
          swap core_pos1 & core_pos2
          Update value of X,Y in the cells array
          Calculate deltaL = HPWL(after swap)-HPWL(before swap
          if deltaL > 0
               update core and cells
          else
               If rand > (1 - e^-deltaL/T)
                    update core and cells
          T = new_Temp (0.95 * T)
```

This function declares the initial and final temperatures according to the provided formulas, where the initial cost is the HPWL of the initial representation. The wire length is updated as long as the current temperature is less than the final temperature. In each loop we swap two random cells in the core and calculate the new HPWL. We compare if the new HPWL is smaller than the previous one, if so, we update the current core. If not we reject by a probability.

```
schedule_temp() -
     T = T * 0.95
swap_core() - swaps 2 components from core grid


swap() - swaps 2 cells with one another


equation() -
     return 1 - e^-deltaL/T
print_core() - prints core


window() -
     for the row in core rows
          for the col in core cols
               if core[row][col] != 1
                    add core components to widgets
               else
                    add "--" to widgets
     Add to widgets new wire length
```

This function creates a grid and adds the core values to its corresponding places in the grid. It also displays the wire length.

Cooling Rate Experiment Results

The total wire length got decremented compared by the initial random placement in all test cases:
The total wire length was decrimented from 88 to 37 for the netlist in d0.txt file
The total wire length was decrimented from 166 to 67 for the netlist in d1.txt file
The total wire length was decrimented from 3832 to 1189 for the netlist in d1.txt file
The total wire length was decrimented from 3854 to 1065 for the netlist in d1.txt file

Bonus Feature:

We implemented a GUI to display the final placement where the grid and the total wire length will be displayed.