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 dictionary,
   n_con(number of connections), rows, columns
Cells dictionary
     key  = component number
     Value = (X, Y)
Core dictionary
     Key = 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-perimeter
          half-perimeter  = (l_max - l_min) + (w_max - w_min)
     sums all computed half-perimeter to total
     Add the half-perimeter of every net into intit_net
dictionary
Init_net dictionary
     Key = index of the net
     Value = half-perimeter
```

The Half-Perimeter Wire Length is calculated by finding the min and max lengths through every iteration and then calculates the half-perimeter by deducting the max - min lengths and the max- min width. This value is then added to the total length. Also, the half-perimeter of each net is calculated and stored in the dictionary init_net.

```
mod() -
     Loops over the cell oriented net of the given cell in
init_net and updates the value of their HPWL to the new
calculated value .
```

This function loops over the cell oriented net of the given cell. It updates the value in the corresponding init_net dictionary to the calculated value by HPWL_mod in case the change is accepted.

```
mapping() -
     Mapps every cell to an array of net indexes which include
this cell.
Mapping array
     Key = cell number
     Value = array of cell oriented net
```

This function loops over the netlist, and maps every cell to its cell oriented net to ease the process of calculating the new HPWL. So, instead of looping the whole net, the program loops over the nets that will be modified only.

```
HPWL_mod() -
     Loops over the cell oriented net of the given cell and
calculate the new HPWL of these nets only.
Save the updated HPWL of each net in an array
New_HPWL = HPWL_before - net_last_HPWL + net_new_HPWL
Vec array
     Key = net index
     Value = array of cell oriented net
```

This function loops over the cell oriented net of the given cell/ cells, and calculates their HPWL. Then, it subtracts the old value of the net and adds the new calculated value to the total HPWL. Also, it adds the new calculated HPWL for each updated net to an array inorder to replace the old values if the change is accepted. There are two versions of this function, if both cells are not empty and if one is empty and one is not.

```
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
          Check if not the same cell or two empty cells
          Swap value of X,Y in the cells array
          Get HPWL after swap (HPWL_mod)
          Calculate deltaL = HPWL(after swap)-HPWL(before swap)
          if deltaL > 0
               Update cells coordinates
               Swap in the core
               Update values in init_net dictionary
          else
               If rand > (1 - e^-deltaL/T)
                    update cells coordinates
                    Swap in the core
                    Update values in init_net dictionary
               Else
                    Re-swap cells coordinates
          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


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.

The performance

Testcase	**Run-time**
d0	**2 sec**
d1	**3 sec**
d2	**24 sec**
d3	**31 sec**
t1	**40 sec**
t2	**1:30 min**
t3	**6 min**

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 36 for the netlist in d0.txt file
The total wire length was decrimented from 185 to 64 for the netlist in d1.txt file
The total wire length was decrimented from 3888 to 1048 for the netlist in d2.txt file
The total wire length was decrimented from 3456 to 886 for the netlist in d3.txt file
The total wire length was decrimented from 6886 to 1927 for the netlist in t1.txt file
The total wire length was decrimented from 45990 to 5711 for the netlist in t2.txt file
The total wire length was decrimented from 43191 to 11199 for the netlist in t3.txt file

Bonus Feature:

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

We also, implemented the animated GIF of the resulted graph (TWL vs time)