EE 5301 - Fall 2021

Problem Assignment 2: (A Very Simple) Simulated Annealing Placement

Due dates:
Phase 1: Dec 1, 2021 by 11:59pm
Phase 2: Dec 10, 2021 by 11:59pm

This programming assignment requires you to use C/C++ to implement an annealing-based placement on the benchmarks you used in programming assignment 1 (not just C7552) using the standard cell style, i.e., rows of abutted logic gates. The main focus is to minimize wire length as the primary objective. The problem statement has been drastically simplified to allow you to finish each phase of the programming within a week or so: all gates and I/O are of equal height (1 unit), but with different widths. I/O pads can be placed anywhere (from this point on, "Gates" refers to both logic gates and I/O buffers and pads).

Gates are placed on rows of height 1 and there is no separation between rows, which is unrealistic but OK for our purposes. The chip area is initially set to be a square (or close to a square) with an initial chip width of:

$$\text{initChipWidth} = \text{ceil}\left(\sqrt{SumGatesArea}\right)$$

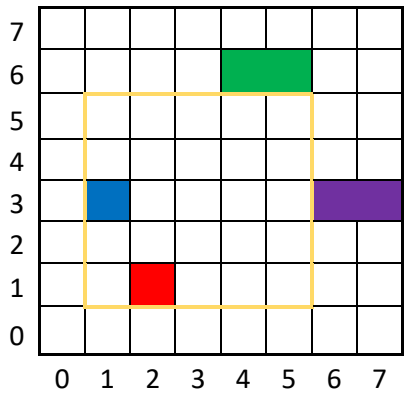
And an initial chip height of:

$$\text{chipHeight} = \text{ceil}\left(SumGatesArea \,/\, chipWidth\right)$$

The chipHeight value would be equal to the number of rows and will never change during the placement process. It is possible, however, that the algorithm might increase the width of a particular row.

Wire length is defined as the sum of all the half-perimeter wire lengths of all the nets. The half-perimeter wire length (HPWL) of a net is half the perimeter of the smallest rectangle that encloses all the bottom-left corners of the gates connected to that net.

For example, say we have a placement as shown below:



Here the red square and blue squares are a NOT gate, while the two green and purple squares are NAND gates. The numbers bottom and left of the placement above, refer to coordinates of the bottom-left square. Now, assuming there is a net *e* between the gates shown above, then the length of the net can be approximated using the half-perimeter width length (HPWL):

$$HPWL_e = \max_{i \in e}\{x_i\} - \min_{i \in e}\{x_i\} + \max_{i \in e}\{y_i\} - \min_{i \in e}\{y_i\}$$

Where (x_i, y_i) are coordinates of the bottom-left corner of each gate connected to net *e*. This formula can also be thought of as the half the perimeter of the smallest rectangle that touches the bottom-left corner of each gate placed. This will be the yellow rectangle above. So, the HPWL for the example above will be:

$$\max\{1,2,4,6\} - \min\{1,2,4,6\} + \max\{1,3,3,6\} - \min\{1,3,3,6\} = 6 - 1 + 6 - 1 = 10$$

The HPWL for the full circuit will the sum of HPWL for each net in the circuit:

$$\text{HPWL} = \sum_{e \in E} \text{HPWL}_e$$

Input File Format

The circuit description is in the same format as PA1, and it is expected that you'll reuse that code for reading in the circuit. The size of each gate is determined from the table below. Except for inverters, buffers and dff, the numbers below are for two-input gates. A k-input gate's area is k/2 * 2-inp-area.

Gate type	Width
not	1
buff	2
nand	2
nor	3
xor	5
and	3
or	4
xnor	6
I/O	1
dff	9

Note: *If there are other gate types that are missing, please let the TA know through Canvas.*

Phase 1

In Phase 1, you should build the backbone of the program: reading the input file, populating the data structures, generating a random initial placement, and calculating the area and the HPWL of the placement. No optimization is done at this phase.

You should output the chip width and height, the area, and the HPWL of the full circuit.

Phase 2

In Phase 2, you would implement the annealing engine, along with the move functions. It is up to you to decide which and how many move functions to implement. Similarly, you will have to decide on the strategy of choosing move functions. This strategy could be as simple as randomly choosing one, to a voting based scheme, etc. You are encouraged to experiment with this, and the annealing parameters to fine-tune your placement engine to get better solution convergence.

Your report should highlight your strategy and the thinking that went into tuning your code. You will report your area / wire length results for the given benchmarks. Part of your grade will be determined by the quality of your placements. I expect to see some innovation in this phase (e.g., optimizing the annealing engine, analyzing annealing behavior when some parameters change, etc.).

Your cost function would be HPWL. As an option, you could implement the cost as $\propto$ area + (1- $\propto$) HPWL, in which "area" is the area of the chip, and "HPWL" is the wirelength cost. In this optional setting, you can experimentally find the best $\propto$ value that works best for minimizing wire length, yet keeping the area from growing unreasonably.

It is highly recommended that you generate an auxiliary output file that dumps intermediate values such as number of accepted/rejected moves, wire length, area, etc. for each temperature step. Generating graphs using these values (e.g., in Excel) would give you insight into how the annealing engine is working. If you output the numbers using, e.g., fprintf (file, "%d\t%g\t%g\n", anIntVariable, aDoubleVar, aDoubleVar);, then you can easily read the file in Excel, where the tab characters ("\t") would show Excel the columns. The same can be achieved using fstream by using myOfs << anIntVar << "\t" << aDoubleVar << "\t" << aDoubleVar << endl;

Phase 2 Output Format

The output file format (ASCII, not binary) for each circuit should be as follows:

- The total HPWL of the placement.
- The width, height, and the total area of the placement.
- |V| lines, each stating the x y coordinate of a cell. The cells are numbered in the order that they are read (or the order they are referenced in the file). Use the same convention as your programming PA1.

You should also create one pdf file containing the results table. The table should look like this:

Ckt name	Chip Width	Chip Height	HPWL	Runtime
c17	...	...	...	...
c432	...			
...				

Below the table, show a bullet list of your strategy and any interesting techniques you used or observations you made during the assignment. If you implemented the optional cost function that combines area and wirelength costs, do discuss it in your report.

Reference for Benchmarks (the .bench format):
http://www.pld.ttu.ee/~maksim/benchmarks/

The benchmarks from ISCAS85 and ITC99 will be used for evaluation.

Submission Process

In each phase, please submit:

- Your code as one .zip containing all source and header files, Makefile to compile your program, and the output files for each of the circuits in ISCAS85 and ITC99.
- A readme to explain your implementation, approach, any issues you faced and the considerations to make when running your code. Upload this inside the .zip file with your code.
- Additionally, for Phase 2 include the pdf file with the results table and your strategy/observations. Upload the PDF separately on Canvas from the .zip file.

The name of the zip file should be <x500>.zip

Grading

- Submission and compilation of the program: 25%
- Legal placements (no overlaps, no out of chip boundary cells): 10%
- Wire length quality: 30% (the worst wire length quality in class gets 5%, and the best gets 30%, and the rest are linearly graded in between). Wire length quality is the arithmetic mean of the HPWL of all the circuits.
- Runtime: 20% (the slowest code will get 5%, the fastest will get 20%, and the rest linearly in between).
- Documentation, including the results table and your strategy/observations: 15%