CMU 18-760 VLSI CAD

Simple annealing placer code, and 4 benchmarks 25 Oct 2005 v1

This document describes how to use these files

760anneal.c (source code for a simpler placer)
760place.tiny (very small benchmark)
760place.medium (medium benchmark with around 800 gates)
760place.big (bigger benchmark with 2761 gates and 2478 nets)
760place.huge (even bigger benchmark with 6514 gates and 5742 nets)

This is the tiny benchmark:

```
7   7
1 p 0 6  1
2 p 0 0  2
3 p 6 6  3
4 p 6 0  4
5 g 3 1 5 9
6 g 3 5 6 10
7 g 3 6 7 11
8 g 3 7 8 12
9 g 3 3 8 13
10 g 3 9 14 18
11 g 4 10 14 15 19
12 g 4 11 15 16 20
13 g 4 12 16 17 21
14 g 3 13 17 22
15 g 3 18 23 27
16 g 4 19 23 24 28
17 g 4 20 24 25 29
18 g 4 21 25 26 30
19 g 3 22 26 31
20 g 3 27 32 36
21 g 4 28 32 33 37
22 g 4 29 33 34 38
23 g 4 30 34 35 39
24 g 3 31 35 40
25 g 3 2 36 41
26 g 3 37 41 42
27 g 3 38 42 43
28 g 3 39 43 44
29 g 3 4 40 44
-1
```

The placement grid is a 7 x 7 array of slots.
The numbering is 0..6 for rows, 0..6 for columns.
Rows 0 and 6 are for pads; ditto columns 0 and 6

There are 4 pads in this example, placed at fixed locations around the edges of the grid. Each object -- pad or gate -- has an integer ID. This line says object 3 is a pad, it's at x=6 y=6 in the grid, and it connects to electrical net 3.

This is a gate. It has ID 16. It has 4 pins on it. In order, the 4 pins connect to the nets numbered 19 23 24 and 28.

In this format, the objects get numbered sequentially starting with 1. Objects include pads (which don't move) and gates (which do). Nets are also numbered starting from 1. You have to read all objects to figure out how many nets there are and what they connect to. Nets can connect to several objects, i.e., it's not just 2-point connections here.

End of file sentinel, tells input reader we are done.

Compile it on your favorite machine with gcc, like this:

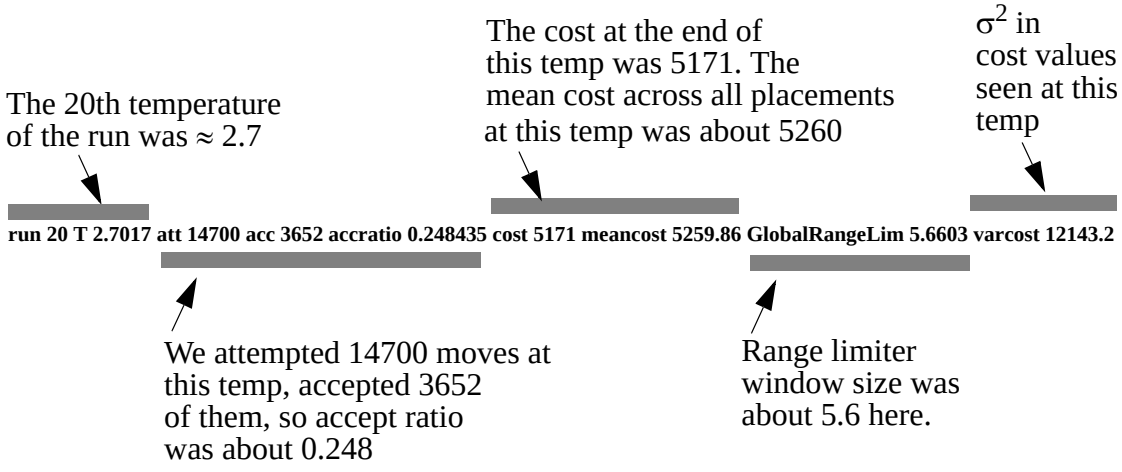
gcc 760anneal.c -lm -o anneal

You need the **-lm** for the math library for the exponentials and such. You run it like this:

anneal netlistfile SEED HOT COOL MOVESPER RANGE > outfile

where:

- **netlistfile** is the file name of the to-be-placed netlist.
- **SEED** is a big random integer, e.g., 1234567, which starts the random number generator in the code so you get a different answer with each seed.
- **HOT** is the starting temperature as a floating point number, e.g., 200.0.
- **COOL** is the cooling rate, a floating point fraction less than 1, e.g., 0.90. In the annealing we cool as Tnew = COOL • Told.
- **MOVESPER** is an integer that determines how many moves we do at each temperature. We multiply MOVESPER by the number of objects and we do this many moves at each temperature to get equilibrium. So, if you set MOVESPER to 20, and you have 100 objects to place, you do 2000 moves at each temperature.
- **RANGE** is a floating point fraction which determines how fast we shrink the range limiting window during move generation. Typical value is 0.98.
- **outfile** is where you want to see the results. The code just writes to the C stdout. The code writes out all the settings you picked, and also it writes a line of info for each temperature. And at the end it writes the final placement. A typical line of annealing temperature info looks like this:



To experiment with the code, you can try these options for the "**medium**" benchmark:

- **Typical run:** good values for the problem are HOT=50.0, COOL=0.95, MOVESPER=30, RANGE=0.98. Run the code a few times with different settings of the random SEED value; each run should take only a few minutes on a fast machine.

- **Quench:** rerun but with these new settings: HOT=50.0, COOL=0.75, MOVESPER=15, RANGE=0.98. This is going to cool **really** fast, and not explore very many moves at each temperature. Again, run the tool a few times with different seeds. **Look a**t the spread of final cost values. Pick the best one here and **plot** its cost vs. temperature progress on the same graph as the previous typical run. **Think about** on the difference you see.
- **No range limiting:** rerun but with these new settings: HOT=50.0, COOL=0.95, MOVESPER=30, RANGE=1.0. Again, run the tool a few times with different seeds. **Look at** the spread of the final cost values. **Think about** what happened here, i.e., try to figure out exactly what RANGE=1.0 means for the range limiting **in this code** (look at the source code).