University of Toronto, Faculty of Applied Science and Engineering
Department of Electrical and Computer Engineering

ECE 1387 - CAD for Digital Circuit Synthesis and Layout

Assignment #3 - Branch and Bound Partitioning with Parallelization

November 2015 J. Anderson

Assignment Date: November 1, 2015

Due Date: November 15, 2015, 11:59pm. Submit report, source, and executable on ECF (details to be announced on the course's Piazza page).

Late Penalty: **-2 marks per day late, with total marks available = 20**

You are to write an implementation of the branch-and-bound partitioning algorithm described in class. It is to take a netlist of (equal-sized) blocks and divide it **in half** with the objective being to minimize the number of **nets** that connect blocks in the two partitions; that is, you are to minimize the "cut set" (crossing count).

You are to implement **exact** branch and bound – that is, your lower bound function must represent a true lower bound. Try to design a clever bounding function to reduce the size of the explored search space.

In class, we discussed several potential decision trees for this problem. *You are welcome to try out one of the alternative decision trees we devised in the in-class brainstorming.*

As in the previous assignments, your program should display its progress and results using graphics. Illustrate, as best you can, the present state of the decision tree, and the pruning as it progresses.

The netlist format is similar to that used in A2 (analytical placement) and it specifies the blocks (cells) and the connectivity between them. Each line has the following form:

blocknum $netnum_1$ $netnum_2$ $netnum_3$ ... $netnum_n$ -1

where blocknum is a positive integer giving the number of the block, and the $netnum_i$ are the numbers of the nets that are attached to that block. Every block that has the same $netnum_i$ on its description line is attached. Note that each block may have a different number of nets attached to it. Each line is terminated by a –1. A –1 appearing by itself on a line terminates the netlist. Example input file:

```
1 2 3 4 -1
2 5 4 -1
3 5 6 2 -1
4 6 3 -1
-1
```

In this example, block 1 is connected to nets 2, 3 and 4. Note that a net may connect more than two blocks (i.e. there may be multi-fanout nets). Also note that net numbers are not related to block numbers.

Test your program on the **four** test circuits (available via link on the Piazza website).

You are to minimize the number of **nets** that connect blocks in the two partitions. This means that each net contributes either 0 or 1 to the total crossing count, and never contributes more than this. A net contributes 1 to the crossing count if it connects cells that lie in both partitions.

What to do:

1. Use your B&B partitioner to find the minimum crossing count for each of the 4 test circuits. For each test circuit, report three quantities in a table: 1) the minimum crossing count achieved, 2) your program's run-time (on remote.ecf.utoronto.ca), 3) the total number of decision tree nodes visited. Regarding the definition of "visited", if you call your bounding function for a node, that counts as visiting the node. Likewise, if a node is disqualified owing to balance constraints, the node still counts as visited. Also, in your report, provide a plot of your decision tree (graphics) for each of the 4 test circuits. Note: run-times should reported with graphics turned off.

2. Create a parallel version of your B&B partitioner using OpenMP or Pthreads (your choice). OpenMP is an API for creating multi-threaded programs that is gaining traction and is easier to use than POSIX threads. With OpenMP, parallelization directives are added as compiler pragmas (`#pragma omp`). See the Wikipedia page about OpenMP to help you get started and see some sample programs. For each test circuit, report three quantities in a table: 1) your parallel program's run-time (on remote.ecf.utoronto.ca), 2) the total number of decision tree nodes visited in the parallel version.

3. Write a 2-page description of the flow of your program. You should describe:

- The branching structure (that is, the design of your decision tree). What do the levels of the tree represent? How did you "order" the levels?

- How you determined the initial "best" solution.

- The bounding function that you used (and any others you investigated). Describe its effectiveness on reducing the exploration space.

- How you traverse the tree and prune it with the bounding function.

- An overview of your approach to parallelization. Please describe the number of threads you used and explain how you handled shared memory among the threads.

CONTEST: There will be a prize for the solution with the lowest number of decision-tree nodes visited for the 4th test circuit (non-parallel implementation).