Daniel Edmond

EECE 583 Assignment 1 : Routing

Algorithms

Lee Moore

To implement the Lee-moore algorithm, I expand in all directions from an input source net until a destination pin is reached.
When multiple nets are specified, routed nets become obstacles for subsequent nets. Handling of multiple sinks is discussed below.

Line-Probe

To implement the line-probe algorithm, I first move source net to the “old route” buffer, RO, sorted in descending order of the sum of manhattan distances from the source net to the destination pins. RN, the new wave buffer, starts empty. The following procedure runs in a loop until RO is empty:
One item is pulled from RO and added to RN to form the expansion set used to generate the next new wave. When expanding around each cell of the expansion set, if the cell is in the direction of its closest destination pin and hasn’t previously been explored by expansion, a probe is sent toward the target, adding the line-probe cells to the RO. If the expanded cells are not towards the target, they are added to RN.
When multiple nets are specified, routed nets become obstacles for subsequent nets. Handling of multiple sinks is discussed below.
Note that while expanding, a traceback code is set in the cell so that the path can simply be traced back to the source when the destination is reached. My algorithm aims to support quick expansion around obstacles hit by recent line-probes, while providing opportunities to escape to old routes for new opportunities if required. At this time, my algorithm is designed so that it can switch destination targets while expanding around obstacles if it finds that a different target is now closer. For this reason, it does not produce optimal routes and in some cases has difficulty tracing back to the source properly.

Multiple-sinks

Both the Lee-Moore and Line probe algorithms have been designed for an input source net and multiple destinations pins, stopping when a segment has been connected. The algorithms can easily be run iteratively by passing the output route into the input source net of the next iteration and removing the reached pins from the destination pins list of the next iteration.

Extensions

I was able to improve my results by re-ordering the nets intelligently. I first determined the bounding box for each set of pins. I then counted how many pins from other nets are contained within its bounding box. Nets were re-ordered in ascending order of the number not-net pins contained in the bounding box. While this strategy did not address all cases, it was effective in the the example shown in Figure 1.

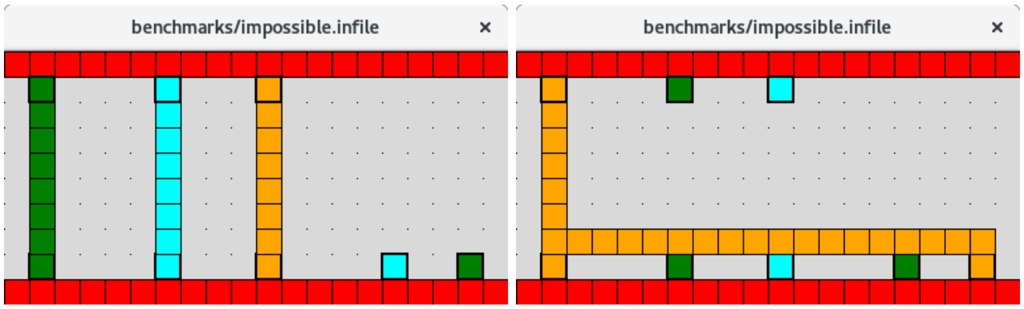

Figure 1 : Routing with bounding-box algorithm (left) and without (right)

Notice on the right routing diagram, run without net reordering, that the first routed net blocks three other segments. On the left routing diagram, run with net reordering, pins that bound fewer other pins are routed first, effectively increasing the number of segments routed by one.

Testing

No additional testing on top of the benchmarks circuits was performed for this assignment. I will make this a goal for future assignments and projects.

Results

Lee Moore

Final route diagrams using the lee-moore algorithm are shown in Figures 2 and 3 below.

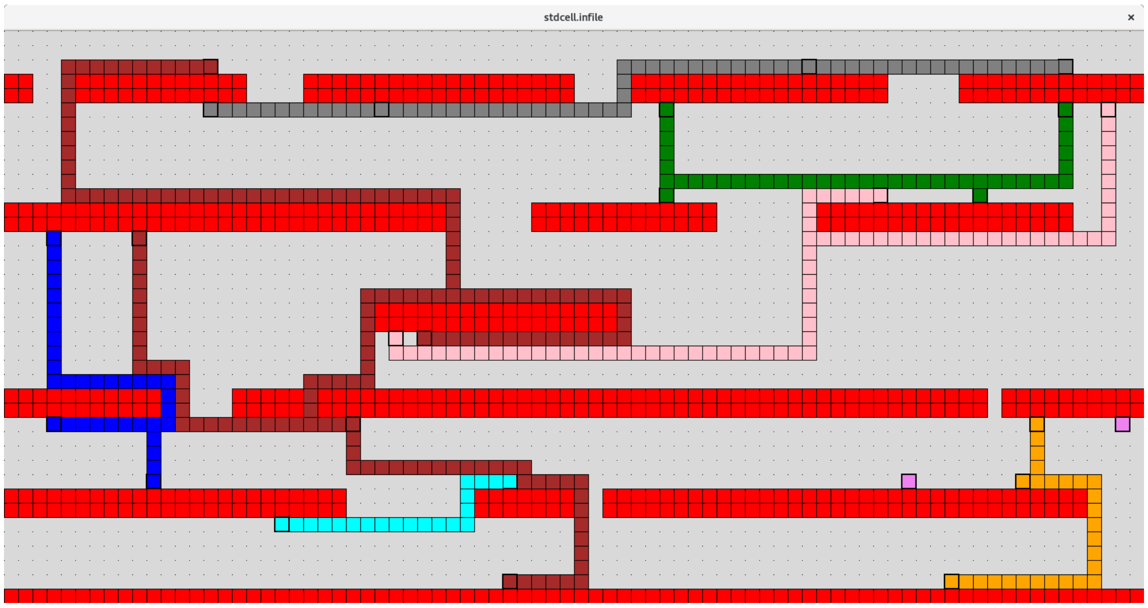

Figure 2 : Lee-moore routing solution for stdcell benchmark

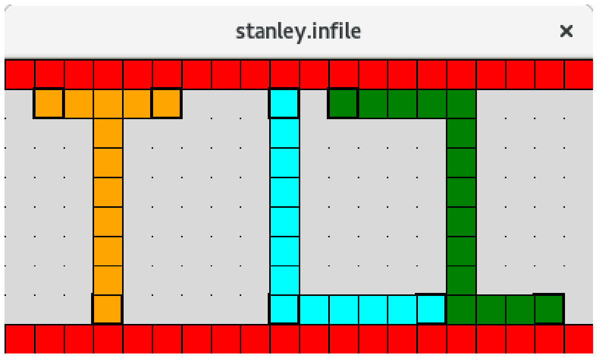

Figure 3 : Lee-moore routing solution for stanley benchmark

Line Probe

Final route diagrams using the line-probe algorithm are shown in Figures 3 and 4 below.

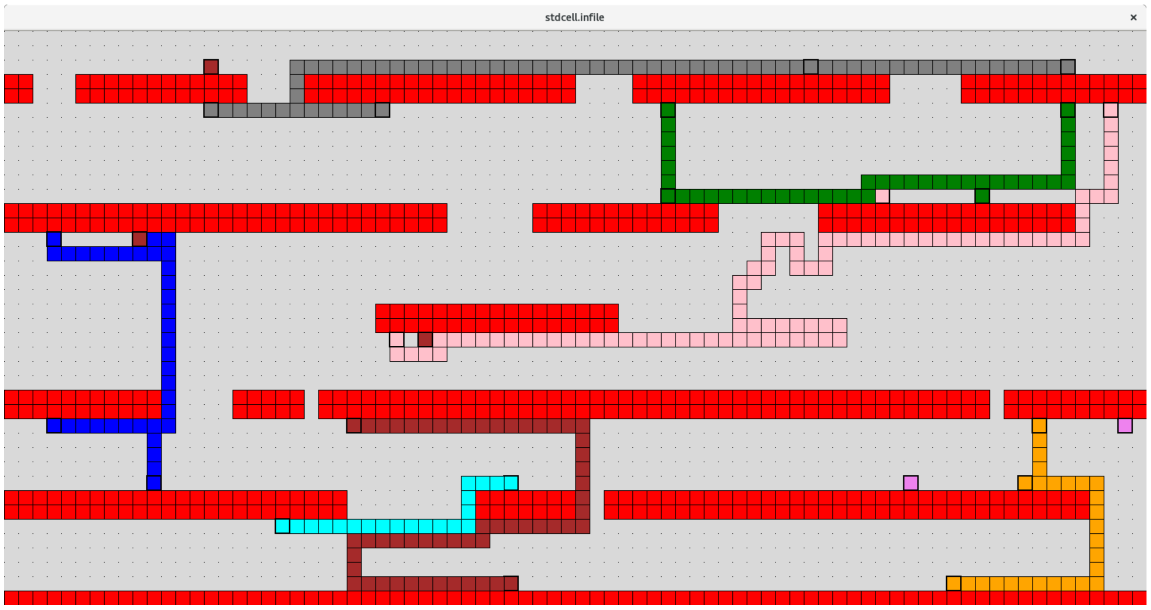

Figure 4 : Line-probe routing solution for stdcell benchmark

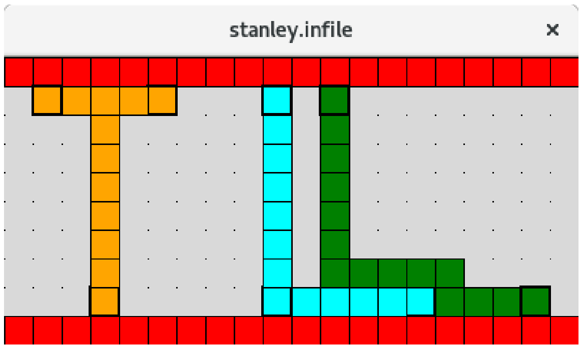

Figure 5 : Line-probe routing solution for stanley benchmark

The performance of the algorithms on provided benchmarks is detailed in Table 1 below.

Table 1 : Summary of routing results

Benchmark	**Lee-moore (routed / total)**	**Line probe (routed / total)**
example	3 / 3	3 / 3
impossible	3 / 5	3 / 5
misty	4 / 5	4 / 5
rusty	4 / 4	4 / 4
stdcell	17 / 18	14 / 18
temp	14 / 17	15 / 17
impossible2	3 / 4	3 / 4

kuma	5 / 6	4 / 6
oswald	1 / 2	1 / 2
stanley	5 / 5	5 / 5
sydney	3 / 3	3 / 3
wavy	7 / 7	7 / 7

User Guide

Clone the following repository:
https://github.com/edmondda/eece_583.git

Please ensure that python3-tk is installed.

```
> python3 assignment1.py --h
usage: assignment1.py [-h] [-f EXP_FILE] [-s] [-n] [-lp]

optional arguments:
  -h, --help            show this help message and exit
  -f EXP_FILE, --file EXP_FILE
                        process specified input file
  -s, --step            step through algorithm
  -n, --net_step        step through net by net
  -lp, --line_probe     Run line-probe algorithm (default is lee-moore)
```

To run Lee-moore, stepping on net segment route:

```
> python3 assignment1.py -n
```

To run Line-probe algorithm, stepping on net segment route:

```
> python3 assignment1.py -lp -n
```

The “-n” flag can be used to step through the algorithm segment wise. Flags “-n -s” together will enable stepping through the individual steps in the algorithm, particularly for the line-probe algorithm.

By default, the benchmark files in ./benchmarks/ are run. Users can optionally specify other input files using the -f flag, specifying the full directory path to the input file. For example, as follows:

```
> python3 assignment1.py -f "benchmarks/stdcell.infile" -n
```