Owen-Ethan_905452983_palatics_Lab3

Ethan Owen
SID: 905452983
EE201A, Winter 2026, MSOL

Part 1: HPWL

Algorithm Used

- Iterate over all pin figures for each terminal pin
- For all nets, use the center-point approximation and ensure that the HPWL includes all net endpoints
- For HPWL, find min/max x/y coordinates for all endpoint positions
- $HPWL = (max_x - min_x) + (max_y - min_y)$

Assumptions Made

1. Include all nets (power, ground, clock, signal, unconnected)
2. Calculate HPWL using the center point approximation and ensure all endpoints are included in each calculation

Values

- *Total nets processed*: 324
- *Total HPWL*: 5,599,970 DBU

Part 2: Placement

Scoring Metric: $HPWL^2 \times Time$

- Lower HPWL sees great returns
- Fast time is really important

Algorithm Approach

The algorithm is a greedy algorithm. It uses swaps of similarly oriented instances of the same type to improve the HPWL. Optimizations are made to the search scope to avoid looking at nets that we seen to have no returns.

I tried *really* hard to use complex approaches: Simulated Annealing, Centroid, etc, but these yielded massive execution times for no improvement on swaps. I think a greedy approach is naive, but it scored the best for this problem, hence its usage here.

Steps

1. Build a cache of all instances and center coordinates, group instances by cell type and orientation
    1. Swapping rules
    2. Same orientation

    3. Same instance type
    4. E.g. INVX at rotation 1 swapped with other INVX at rotation 1
    5. Avoid all instances that cannot be swapped with each other
2. Classify all nets and avoid considering power, clocking, reset, and unconnected nets, refine from 324 nets down to 309 nets
    1. Why do we do this?
    2. These span the entire chip
    3. Swapping never improved HPWL
    4. Locking would lock too much of the design
    5. Avoid filler cells that would have no contribution
    6. Total DBU goes from 5,599,970 to 5,254,450 without these nets considered
3. Remove instances with no net connections, call this pruning
4. Evaluate all candidates in one batch pass
    1. For each we do the following
    2. Compute HPWL delta of swap
    3. Use same HPWL calculation as in part 1
    4. Collect all pairs with negative deltas
5. Sort for best delta and perform swaps
6. Make a final sweet for any changes from swaps
7. Use `oatransforms` to handle swaps and preserve orientation

I made some optimizations after the fact to use batches, caching, and limit API and complex function usage to avoid timing overhead on repeated calculations.

Results

- *Original HPWL*: 5,599,970 DBU
- *Final HPWL*: 5,599,210 DBU
- *HPWL Reduction*: 760 DBU
- *Number of swaps*: 2
- *Execution time*: ~0.0011 sec
- *Score* ($HPWL^2 \times Time$): ~3.5×10^{10}