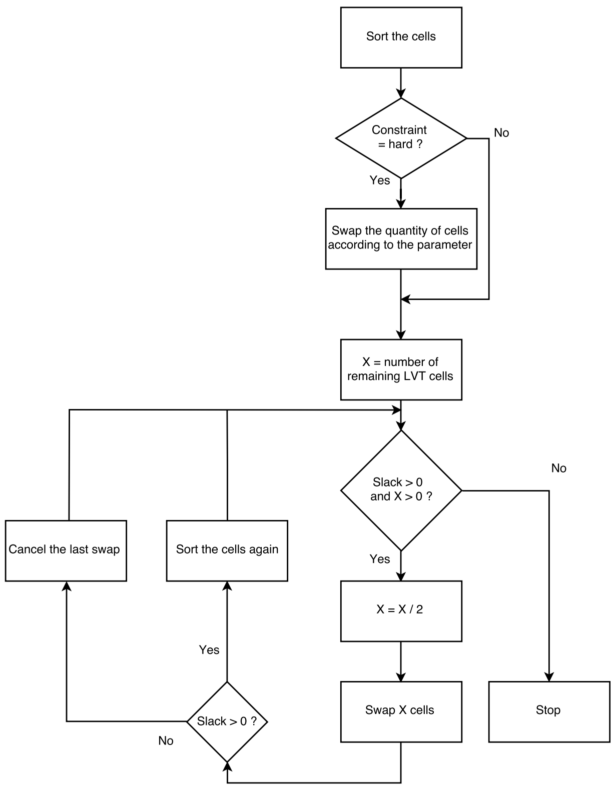
Synthesis and Optimization - Group 18

DualVth contest report

authors: Cesarano, Fogliato, Monti

The logical flow of the algorithm we implemented can be easily understood by looking at the flowchart in the Figure.

Our aim is to provide a very fast solution, while sacrificing a bit of efficiency.
The first step is organizing the cells in the circuit in such a way that they are sorted by decreasing slack. If a cell has a large slack, it probably means it can be swapped to HVT without incurring in a relevant timing penalty.
When executing the script under a hard constraint, since the most important condition is to satisfy a certain percentage of LVT cells, we apply the swap without looking at anything else, but still following the sorting we just implemented. On the other hand, if the constraint is soft, we skip this step.
After this phase we try to find, in the minimum possible amount of time, how many (more) cells can be swapped while keeping a total positive slack. Instead of trying all the cells one by one, which would imply a linear complexity, we follow a logarithmic approach by considering half of the available cells at each iteration.
A while loop is executed, where the number of swapped cells is halved each time. In case that, after the swap, the slack is still positive, we sort again the remaining cells and proceed to the next iteration. If it is negative, instead, we undo the swap and, at the next iteration, we will try swapping half of that amount.
The script terminates when the last iteration tried to swap just one cell, and it doesn't make sense to swap 0.


Sort the cells
Constraint = hard ?
No
Yes
Swap the quantity of cells according to the parameter
X = number of remaining LVT cells
Slack > 0 and X > 0 ?
No
Yes
X = X / 2
Swap X cells
Stop
Slack > 0 ?
Yes
No
Sort the cells again
Cancel the last swap