LOW POWER CONTEST 2017-2018

Group 15
Chiaberto Simone s251572
Ieva Antonia s253237
Tarantino Gabriele s253084

The objective of the contest is to implement a plug-in function able to perform a leakage power optimization, based on the usage of *dual-threshold libraries*. Starting from an initial combinational circuit synthesized entirely with LVT cells (faster but with higher leakage power), the optimization procedure swaps a certain number of LVT cells to equivalent HVT cells (slower but with lower leakage power), trying to obtain the required *power saving* (its value must be in the range 0 to 1) without changing their size and, simultaneously, attempts to minimize the overall slack penalty of the circuit and the script execution time.

The problem is substantially reduced in converting the minimum possible number of LVT cells, such as to achieve the required saving. Since no prediction or assumption can be made on the paths distribution, we chose to use a sort of *dichotomous algorithm* which, basing on a parameter called *swap_ direction*, "swaps" or "de-swaps" a certain amount of cells, according to the dichotomous research.

The written algorithm can be divided in these sections:

1. **PREPARATION STAGE**: it checks if the required saving is one of the critical situation (1 or 0) and acts as a consequence; otherwise, if the saving is an intermediate value, the algorithm creates a list of leakage powers and cell delays when all the cells are LVT; then swaps all the cells to HVT and creates other two similar lists with the new values. Moreover it sets the staring dichotomous variable to half of the total cells and finally restores the initial cells footprint (all LVT cells).
2. **MAIN LOOP**: the loop establishes at each operation how many cells need to be swapped and which are the cells with the higher priority (i.e. the ones to be swapped first). In particular, the operations performed are:
   - **section a:** First of all, for each LVT cell remaining in the circuit, a priority index is computed, creating the list *cell_ list_ priority*. The priority list takes into account the relative gain in terms of power, the relative loss in terms of delay and the maximum slack between the paths to which the cell belongs.
   - **section b:** At this point, basing on *swap_ direction* variable, we can have two different actions:
     - swap direction 0: an half of the total LVT cells remaining is switched to HVT cells, choosing them referring to the priority indexes list computed before.
     - swap direction 1: an half of the total last HVT cells swapped during the last iteration with swap direction equal to 0 is switched back to LVT cells, starting from the cell with the lowest priority index.
   - **section c:** Subsequently, the total current power saving is computed and compared to the required one: if it is greater, the value of the parameter *swap_ direction* is set to 1, otherwise, it is set to 0. The algorithm stops when no more division on the cell to swap number can be made. This means that the dichotomous algorithm has ended and power saving is reached with a resolution of a single cell. If the dichotomous research finishes with a "de-swapping" iteration, it could be that the power saving obtained is just smaller than the required one. In this case, the algorithm swaps the highest priority indexed cell to HVT to meet the wanted saving.

During the implementation phase and after some tests we noticed that the most critical section in terms of execution time was the index computation, in particular the cell delay extraction. This happens because in order to compute the indexes for both the HVT and LVT delays, a temporary switch is made in order to find HVT ones, with a consequent loss of time. To cope with this problem we performed an approximation that has guaranteed us a reduction of the CPU execution time, with a relative limited loss in terms of slack reduction: in particular it consists into use the initial data about cell delay, computed during preparation stage, for all the times we have to recompute priority indexes, except in the case we have just swapped more than 85% of the total circuit cells to HVT; the algorithm recomputes priority indexes using updated data on delay only for the remaining LVT cells belonging to the most critical paths (this operation is needed to guarantee reliable results for the priority index). Obviously, more the distribution is stressed in terms of slack constraint, more this approximation affects negatively the final result in terms of slack penalty. Anyway, we retained that this approximation can be accepted, since the execution time is drastically reduced.