Power optimizing Contest, Group 14

FLORIDIA Andrea *S224906*, POOLAD Pooya *S226588*

One technique for optimizing Power consumption of Electronic Integrated circuits is to use cells with two threshold voltage level. The goal of this approach is to use gates with higher threshold voltage (slower) in cells which belongs to non-critical paths of the design to reduce leakage power. In this approach we decided to use a different algorithm to implement dual V_th design. We first swap all the cells to HVT cells, then start optimizing and swapping back those for which is needed to remain LVT.

I. INITIAL COMPUTATIONS

GIVEN the inputs design with all LVT cells, first it checks for unfeasible conditions. These conditions are: user arrival time smaller than the actual design arrival time with all LVT cells and whether the number of paths already present in the slack window exceed the parameter set by the user. If at least one unfeasible condition is met, then the script terminates returning the list $\{0\ 0\ 0\ 0\}$. Before checking for unfeasible conditions, the script computes the $new\ zero$ for the slack. Since the arrival time inserted by the user becomes the new required time for the design, we have to compute a new value which corresponds to the $slack = 0$ with the new arrival time. This is done to avoid re-synthesizing the circuit with a new timing constraint. The Equation 1 is used for compute it.

$$Slack_{zero} = RequiredTime_{design} - ArrivalTime_{user} \quad (1)$$

II. OPTIMIZATION PHASE

SINCE the design has been synthesized with all LVT cells, the actual values for leakage power and arrival time are respectively the worst one (leakage) and the best one (arrival time). Therefore, we decided to implement an algorithm which first swap all the cells to HVT, then swap back again those cells that will violate the constraints. With this approach we will decrease a lot in term of computations and deciding. The first thing the script does is to swap all the cells to HVT. After doing this, the design is in the reverse situation. Now if the constraints are met, the script terminates gathering the requested percentages. If not, since a given cell may belong to different paths, it sorts the design cells by the arrival time of the worst timing path which the cell belongs to. At the same time it will store in the variable $swap\_value$ the number of cells that violate the user arrival time constraint. The cells will be sorted in decreasing order, thus assuming $swap\_value = n$, the first n cells are the one which are violating the timing constraint. So some of them absolutely need to swap back to LVT. Once that these cells have been individuated, the script swaps to LVT all the cells that are violating the timing constraint. It is very likely that at the end of this swap, the timing constraint have been met. If so, since a given cell may belong to different path, it very likely not all the n cells needs be swapped to LVT. Thus in order to obtain a better result, script starts swapping back to HVT cells, starting from the last one that have been swapped to LVT. The script swaps until the constraints are not met anymore, then it swaps to LVT the last cell that have been swapped to HVT. Doing so, the constraints are met again and the scripts terminates gathering the requested parameters. Nevertheless, it is also true that after swapping the cells that violate the timing constraint, there may still be a constraints violation (number of paths in the slack window). In this case, the script continues to swapping cells from HVT to LVT until the constraints are fully met.