# Low power contest report

Group 11: Matteo Isoldi s318980, Filippo Marostica s319962, Elena Roncolino s304719

## 1 Introduction

When analyzing a circuit there exist two different main contributions for what concerns the total power: the dynamic power, that is the power dissipated when the circuit is active, and the static power, that is the power dissipated when the circuit is not switching. The aim of the contest is to write a TCL program for PrimeTime that implements a post-synthesis leakage power minimization procedure. We decided to start from the situation in which all the gates are taken from the High-Vth (HVT) library so that the leakage power is minimized as much as possible. From this starting point we progressively convert the cells, starting from the most critical one, to Low-Vth to improve the timing of the circuit and to satisfy the slack. Our algorithm is divided into two main phases: score calculation and cell swapping.

## 2 Score calculation

An important part of our algorithm was to find an optimal way of ordering the cells of the circuit so as to speed up the process of converting them from one library to the other starting from the most critical one. For this purpose we designed a function that calculates a score based on the two constraints that we need to satisfy, i.e., slack and fanout endpoint cost. The formula used is:

$$Score = \frac{P_{LVT\_leak} - P_{HVT\_leak}}{SLACK_{LVT} - SLACK_{HVT}} + \frac{FOEC_{cell} - MaxFOEC}{\#contributingCellsEC}$$

## 3 Cell swapping

Finally, when all the cells are ordered, the algorithm starts a recursive procedure in which it converts from HVT to LVT the most critical cells and it checks if the constraints are met and, in case they are not, it continues to go through the other cells of the list.

At the end of this process, we decided to do some further steps to improve the optimization; in fact, when the previous for cycle has ended, we may be in a situation in which some cells were previously swapped to LVT even if this was not really necessary. To solve this problem we implemented an extra cycle in which, after reordering the cells in decreasing way, we are swapping back to HVT the less critical cell of the circuit. The cycle is executed by the script as long as some cells are swapped otherwise, when more than 5% of the total cells are not consequently swapped or when the constraints are no longer satisfied, it stops.

If, at the end of this final cycle, the constraints are not satisfied, we swap back the cells to LVT in this way we "cancel" the effect due to the execution of the last for.

## 4 Script functions

To perform all the operations, we implemented the following functions:

- *swap_to_hvt, swap_to_lvt*: used to swap all the cells to HVT or LVT library;
- *swap_cell_to_hvt, swap_cell_to_lvt*: used to swap a specific cell to HVT or LVT;
- *extractCellsInfo*: used to extract all the necessary information;
- *compute_fanout_cost*: used to compute the endpoint fanout cost of a specific cell;
- *rankCells*: used to organize all the cells depending on their score;
- *compute_score*: used to calculate the score as explained above;
- *check_contest_constraints*: every time the algorithm is executed this function is used to check the progress and, in the case of satisfied constraints or if all the cells are LVT, it ends the execution of the operations.