Low power contest report

Group 11: Isoldi Matteo, Marostica Filippo, Roncolino Elena

1 Introduction

When analyzing a circuit, there exist two main different 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 plugin 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. From this starting point, we first compute a score for each cell, based on its attributes. After this step, we progressively convert the cells, starting from the most critical one, to Low-Vth (LVT) to improve the timing of the circuit and to satisfy the slack. We will now analyze briefly this two main parts of the algorithm.

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:

$$Score = \frac{P_{LVT\_leak} - P_{HVT\_leak}}{SLACK_{LVT} - SLACK_{HVT}} + BP$$

Where the bonus priority points (BP) are computed as:

$$BP = \begin{cases} \dfrac{FOEC_{cell} - MaxFOEC}{\#contributingCellsEC} & \text{When } FOEC_{cell} > MaxFOEC \\ 0 & \text{When } FOEC_{cell} <= MaxFOEC \end{cases}$$

3 Cell swapping

Finally, when all the cells are ordered, the algorithm starts an iterative process in which it converts from HVT to LVT the most critical cells and, when it notices that the constraints are met, stops the switching procedure. At the end of this process, since some cells were previously swapped to LVT even if this was not really necessary, we enter another iterative phase where we try to do some more optimization.

After reordering the cells in decreasing way, we are swapping back to HVT the less critical cells 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, it stops. If, at the end of this final cycle, the constraints are not satisfied, we swap back the cells to LVT in order to "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.