Low-Power Contest 16/17

Piergiovanni Ferrara (ID: 243745), Danilo Moceri (ID: 243790)

June 20, 2017

Assignment

Write a new TCL command integrated into Prime Time that performs a post synthesis Dual-Vth cell assignment for leakage power optimization, the new command has two main arguments which are :

-lvt –> The percentage of LVT cells allowed after the command execution.
-constaint [soft | hard] –> Optimization effort.

Introduction

To solve the assignment we have developed a new tcl command to be integrated into Prime Time.
As it is easy to imagine, the most important thing to take into account is how to decide which cells should be swapped before others, to achieve a good power reduction without losing too much in performance due to cells inside critical paths.
After different attempts to find a good and quick solution we decided to use a "not so much expensive" cost function which allows us to have: good performances in terms of execution time and leakage power saving.

Implementation details

The first step of our algorithm is to compute a sorted list of cells such that we can start to swap the cells in head of the list. To do that we defined a cost function computed as:

$$Priority = LeakagePower_{cell} \cdot Slack_{cell} \tag{1}$$

as explained before, this cost function is not so complex and can be computed quickly and easily. We use the *Priority* to sort the list in decreasing order, such that the cells with a higher leakage power not belonging to the critical paths are swapped before.
In both the cases (soft and hard) of the algorithm we use the same previously defined cost function, the main difference between the two cases is given by the way in which we swap the cells.
In fact concerning the hard version, we just start to swap cells from the list head till the requested percentage of Low Vth cell is reached. This is possible keeping in mind that in this case we don't take care about possible negative slacks, however thanks to the ordered list by priority we are going to start swapping the cells with a higher leakage and not belonging to critical paths.
In the other case (soft constraint option), taking into account that negative slack is not allowed, we should check the report timing each time a cell is swapped, doing so, a huge overhead due to the timing analysis arises. To improve the algorithm in terms of execution time it is possible to find a smarter solution. In fact, in our algorithm we use a sort of "binary search algorithm" and in particular, given the value of cells percentage that the user would like to swap without timing violations, we as first point swap half the cells value and check for possible negative slacks. If any, the algorithm proceeds restoring the right part of half of the swapped cells checking again for possible timing violations, otherwise it continues to swap another half of the remaining upper half. Thanks to this approach it is possible to reduce the number of timing checking, avoiding to run useless timing analysis.

Conclusion

In our algorithm we attempted to find a good solution based on a trade-off between execution time and leakage reduction. Without any doubts other improvements are possible, the first one is to compute a more complex cost function such that a better swapping priority can be defined. For instance the priority can be based on the differences between the leakage and the delay of the LVth and those of the HVth, but certainly doing so the execution time needed to compute all those parameters will increase. Another possible choice is to maintain the current cost function doing some improvement on the product we defined in 1, for example computing a weighted mean with some coefficients that give more importance to the leakage power rather than to the slack or vice versa.