Project report – Group 16

Synthesis and Optimization of digital systems

Academic year: 2022- 2023

Jessica Marossero s309840, Fabrizio Cavallone s314703, Tartaglia Federico s319857

Algorithm description:

Dual-Vth optimization involves assigning different threshold voltage values to cells in a design to balance power consumption and timing performances. The algorithm begins by initializing the required parameters, including the slack threshold and maximum fanout endpoint cost. These parameters define the timing constraints and help evaluate the quality of the optimized design. To identify the cells eligible for swapping, the algorithm first filters the LVT cells, then, it computes the timing slack for each LVT cell and sorts them in increasing order of slack values. Next, the script calculates the number of cells that should remain unchanged based on a predetermined percentage (80%). This percentage has been experimentally determined based on the results of a more iterative algorithm, noting that the cells swapped back from HVT to LVT were predominantly found within the top 80% of the total cells ordered by slack. Cells beyond this threshold are marked for swapping and a list of them to be swapped is created. The algorithm performs an initial round of cell swapping by selecting the cells with the lowest leakage power from the HVT group. Then, it enters in an iterative loop to swap cells until the timing constraints are met where it swaps the first two cells to LVT for decreasing the CPU time making a trade-off between quality of results and speed. The iterative swapping process continues until the design satisfies all timing constraints. Once the timing constraints are met, the algorithm terminates, and the final design is obtained. The algorithm follows a systematic approach to swap cells between different voltage threshold groups, considering their power consumption and timing slack.

Main procedures description:

- **dualVth {slackThreshold maxFanoutEndpointCost}**: This is the main procedure for the dual Vth optimization by repeatedly swapping cells between HVT and LVT libraries until the timing constraints are met. It calls other procedures to retrieve sorted lists of cells and it continues this process until the timing constraints are satisfied, and then it returns 1.
- **fitered_and_sorted_by_slack {}**: It retrieves the LVT cells from the design based on a filter condition and iterates over each cell taking the cell_name and the slack value. The list is then sorted in increasing order of slack and the procedure returns the lvt_sorted list.
- **list_to_swap {lvt_sorted}**: It takes the sorted list lvt_sorted as input and determines the number of cells that should not be changed (80% of the total cells) and it appends the cell names to the lvt_to_swap list if they should be swapped.
- **swap_some_cells_to_hvt {lvt_to_swap}**: It takes the list as input and it iterates over each cell in lvt_to_swap and searches for corresponding cell references. If a match is found, it modifies the reference name and resizes the cell to use the HVT library.
- **sort_cells_by_power {list_cells}**: This procedure sorts the HVT list of cells by their power. It takes the list with the cell_name and the leakage_powers arguments as input and returns a new list with the cells sorted in increasing order of power.