Group 18

Cioffi Giulia 249504

Conte Fiorenza 251535

Melibeo Alessio 253297

Synthesis and Optimization of Digital Systems

Low-Power Contest 17/18

The presented script is intended to work on *PrimeTime* on a circuit previously synthesized by *Design Compiler* using only LVT (Low Threshold Voltage) cells. LVT cells are faster but produce more power consumption due to leakage. The function *dualVth* receives a parameter (ranging from 0 to 1) that represents the aimed power saving of the circuit. The goal of the algorithm implemented in the function is to replace a certain amount of LVT cells with HVT (High Threshold Voltage) cells, which are slower but are better in terms of leakage power consumption. In order to perform an efficient swap, the most important parameter to be taken into account is the slack: the idea is to swap cells with higher slack first.

As a first step, the aimed power saving is saved in the variable *savings*. Then the power analysis is performed to find the power consumption of the circuit synthetized using only LVT cells; this value is saved in the variable *start_power*.

At this point we create a list, *full_list*, in which each element has two fields (name of the cell and slack). Starting from this list, we generate other two lists sorted by decreasing slacks, *sorted_full_list* and *sorted_name_list* . The first one contains both name and slack, while the second one contains only the name of the cells. The total number of cells included in the circuit is saved in the variable *ncells*. Our algorithm exploits only the *sorted_name_list* to perform the cells swap: it keeps replacing sets of cells and saving the related final leakage power in the variable *leak_power* until the leakage power of the circuit synthetized with LVT and HVT cells matches the following condition:

$$leak\_power < start\_power * (1 - savings)$$

The variable *iterations*, that contains the number of cells which form a set entirely swapped at each iteration, is computed according to the aimed power saving:

$$iterations = k * savings * ncells$$

$$k = \begin{cases} 0.05 & if savings < 0.4 \\ 0.1 & otherwise \end{cases}$$

i.e. if a small power saving is required, then it's likely that less cells need to be swapped.

Each iteration swaps LVT cells with HVT cells of the same type, in order to keep the original behaviour of the circuit. It's necessary to acquire the *ref_name* of the cells, replace *_LL_* with *_LH_* and then look for this cell into the HVT library; if they are still available, the replacement is performed. This way, we are sure that the behaviour of the cell is not modified, while its threshold voltage is.

Some controls are performed during the swap: it is possible that after a certain number of replacements of cells of the same type, the corresponding HVT cells are not available anymore. We avoided that the related error messages were displayed.

When the algorithm jumps out of the swapping loop, the final leakage power of the circuit is available in *leak_power,* so the obtained *power_gain* can be computed with the following relation and displayed on the screen:

$$power\_gain = \frac{start\_power - leak\_power}{start\_power}$$

The power gain finally reached strictly depends on the *savings* chosen: if the aimed power saving is very low, the obtained *power_gain* could be slightly higher than the first one; if the aimed power saving in very high, the obtained *power_gain* could not be reached and saturate to its maximum reached by making all its available swaps.