Low Power Contest 2025 Submission

Group XX

Gianmarco Fortunelli
Francesca Di Giuseppe
Mattia Cesaraccio

7th July 2025

1. Introduction

This report summarizes our approach for the Low Power Contest 2025, focused on minimizing power leakage in various circuits and timing constraints, while exploiting multi-Vt cell replacement. It was required to consider a slack threshold and a limit on critical paths below that threshold.

2. Optimization algorithm

Our algorithm is composed of four steps. In the first two, we replace as many LVT cells as possible with the HVT ones, keeping the number of critical paths under the given threshold. The last two steps focus on replacing LVT cells with SVT ones.

To identify the most beneficial swap, we order the cell list using two different priority criteria and run the optimization using each priority list, both for HVT and SVT swaps. The first criterion prioritizes cells providing the highest leakage power reduction with minimal impact on slack, while the second focuses on cells with the highest absolute slack. In both cases, priorities are computed at the beginning of the optimization process and are recomputed only if more than half of the cells are successfully changed without errors. Both priority lists are computed on additional procedures for better code reading. In the deltaP/deltaS criterion, we first save the initial leakage power and slack, then apply temporary cell replacements, read the new leakage power and slack values, compute the differences and their ratio, sort the list accordingly, and finally revert all the changes. In the Slack-based criterion, the procedure is simpler: we sort the cell list directly based on timing information.

To optimize the algorithm's run time, we start by swapping 10% of the LVT cells simultaneously. If a rule violation occurs, we swap back the latest changed cells, halve the group size and try again. This process stops when either we fail to change even one cell or all cells have been changed. Rule violations are controlled with a dedicated procedure that returns '1' when a violation is present.

If the former exit case happens, we skip that cell and initialize a counter for consecutive skipped cells. It is reset when a cell (or group of cells) is successfully changed and it is incremented when a single cell needs to be skipped. At this point, a second phase of the algorithm begins, in which we try to swap cells until we get a series of 10 (deltaP/deltaS priority) or 15 (slack priority) errors. We start by trying to replace the 2% following cells simultaneously.

To avoid excessive and unproductive trials, a fine-tuning reduction strategy has been added for the number of cells swapped simultaneously. Indeed, after the second failure, the maximum group size is reduced to 0.67%. If four consecutive errors occur or more than 25% of cells have been swapped, it is reduced to 1. In case of failure, when changing multiple cells concurrently, the group size is halved. However, if the last cell has been skipped, it is directly reduced to two after the first trial with 0.67%.

In all the steps of the algorithm, time is taken into account as an enter and exit condition, allowing the program to return a result that could be under the target but without stalling or raising errors.

The table 1 summarizes the results we have obtained for each configuration. As can be noticed, we have been able to reach the desired power saving every time.

Block	Clock (ns)	SlackThr	ViolPaths	Runtime (s)	Obtained (%)	Wanted (%)
c1908	2.0	0.10	100	39.37	84.90	50
c5315	2.0	0.10	100	174.93	92.58	50
c1908	2.0	0.10	100	40.95	84.90	80
c1908	2.0	0.25	200	45.03	77.14	75
c1908	1.5	0.05	1500	71.93	61.88	55
c1908	1.5	0.07	5000	145.23	60.46	55
c1908	1.0	0.01	1000	58.78	16.95	15
c1908	1.0	0.02	3500	197.69	14.33	10
c5315	2.0	0.10	100	176.03	92.58	90
c5315	2.0	0.20	200	232.20	90.01	80
c5315	1.5	0.05	300	271.79	81.46	75
c5315	1.5	0.07	700	299.21	80.41	75
c5315	1.0	0.01	500	239.78	27.29	25
c5315	1.0	0.02	1500	299.14	25.25	25

Table 1: Results for each configuration (SlackThreshold, ViolatingPaths, Saving and Runtime).