Slack-Driven Dual-Vth Leakage Reduction

PARISI Emanuele *S231543*, ROSSI Simone *S228146*

Leakage power is emerging as a key challenge in IC design. Leakage is increasingly exponentially with each technology generation and is becoming the dominant part of total power. Device threshold voltage scaling, shrinking device dimensions, and larger circuit sizes are causing this dramatic increase in leakage. As leakage varies exponentially with process parameters, yield of the chip is often directly influenced by leakage. Increasing amount of leakage is also critical for power constraint ICs. This significant increase of leakage now warrants that it be considered as the key design variable in all IC designs.

To reduce this kind of static power consumption, we propose a solution based on Cell-Based Multi-Threshold Voltage Design and we developed an algorithm that first of all works at path-level, taking into account the slack distribution while choosing which set of cell can be swapped to HVT, and then moving to a cell-based approach, grouping the cells into clusters.

This algorithm has been tested using Synopsys Design Compiler on three different benchmarks (`AES Cipher`, `C3515`, `C1908`) and the results collected. Due to the small size of the circuits, the power saving using only the path-based approach was unfortunately near to 0, while the cell-based algorithm reached an high efficiency, up to 65% of leakage reduction.

***Index Terms*—Low Power Contest, Synthesis and Optimization of Digital System, Leakage Power, Dual-Vth design**

I. INTRODUCTION

LEAKAGE power minimization is an issue of significant concern in nanometer scale CMOS designs. One of the possible solution for this issue is the dual-Vth design.

In the dual-Vth a circuit is partitioned into critical and non-critical paths, and use low threshold voltage (LVT) transistors only for gates in the critical paths. This approach will reduce subthreshold leakage current. It may achieve large leakage power reduction if the circuit contains many gates in non-critical paths. Thus, dualVTth scheme is efficient in realizing high-speed and low-power systems.

II. ALGORITHM: CELL-BASED APPROACH

THE main idea of this approach is to divide the set of cells into clusters, with its dimension fixed to $1/250$ of the total cells, and iteratively cluster per cluster switch from LVT to HVT. Then, the algorithm will check for the requirements, in particular the arrival time for the most critical path will be evaluated as well as the number of paths that can fall in the slack window[1]. If the requirements are not met, all the cells in the current cluster will be swapped back to LVT.

```
for {set i 0} {$i < [expr $n_cells/$n_per_cluster]}
    {incr i} {
  set cluster [list]
  for {set j 0} {$j < $n_per_cluster} {incr j} {
    lappend cluster [lindex $cells [expr $i *
    $n_per_cluster + $j]]
  }
  set HVT_cells [expr $HVT_cells + [cell_swapping
    $cluster HVT]]
  set flag [check_requirements $arrival_time
    $critical_paths $slack_win]
  if {$flag == 0} {
    set HVT_cells [expr $HVT_cells - [cell_swapping
    $cluster LVT]]
  }
}
```

.

[1] `arrivalTime`, `criticalPaths` and `slackWin` are command-line arguments of the procedure

III. ALGORITHM: PATH-BASED APPROACH

THE idea of this approach for choosing which cells to be swapped is to divide the spectrum of the slacks into different windows and for each window to identify the lowest critical paths. Starting from these ones, the algorithm will start to swap to HVT continuously checking for the requirements. This approach will stop as soon as in one window no paths can be swapped to HVT. The algorithm can summarized with these steps:

1) Divide the histogram of slack into W bins from 0 ns to ATns (where AT is the given arrival time)
2) Select the K worst critical paths in window W_i

$$CP_i = \{p_j, j = 0, \ldots, K-1\}$$

3) For each $p_j \in CP_i$ switch to HVT all the cells
   - If $p_j|_{HVT}$ meets the requirements, we can move to the next path
   - If $p_j|_{HVT}$ does not meet the requirements, path is restored to the previous snapshot
4) If at the end of one windows no paths can be swapped to HVT, starting from this point the algorithm will continue with a cell-based approach.

IV. RESULTS

THESE approaches have been tested with three different benchmarks: `C5315` and `1908` (small design) and `AES Cipher` (medium design), all of them synthetized with Synopsys Design Compiler 2014 with an arrival time of 3 ns.

However, for such a small complex designs, the path-based algorithm cannot perform any leakage reduction; only for the `AES Cipher` we measured a power saving but with a leakage reduction not greater than $1.5\%/2.0\%$; this result cannot be justified by the time required to complete the script (couple of minutes).

On the contrary, the only cell-based algorithm shows an incredible leakage reduction[2]:

- `AES Cipher`: -23.39% in $115\,\mathrm{s}$ with 26% of the total number of cells swapped to HVT;
- `C5315`: -55.47% in $2.0\,\mathrm{s}$ with 52% of the total number of cells swapped to HVT;
- `C1908`: -61.93% in $2.0\,\mathrm{s}$ with 55% of the total number of cells swapped to HVT.

V. Conclusion

For all these reasons, in the final script we use only the cell-based approach. The path-based approach is more rigorous but shows its limit for low complex design. Further studies of this approach could include the testing with a modern high complex design to finally evaluate the actual benefits of this algorithm with respect to the more empirical cell-based swapping.

[2]For these tests, `arrivalTime = 3`, `criticalPaths = 1000` and `slackWin = 0.01`