Enhanced Clustered Voltage Scaling for Low Power

Monica Donno Luca Macchiarulo Alberto Macii Enrico Macii Massimo Poncino

Politecnico di Torino
Corso Duca degli Abruzzi 24
10129, Torino, Italy

ABSTRACT

This paper presents a voltage scaling approach that is based on an enhanced variant of *clustered* voltage scaling originally proposed by Usami and Horowitz ([1]) The results show that subtituting the original depth first strategy with a breadth first one results in improved speed and quality of results. Data are validated through power and timing analysis performed with a commercial tool.

Categories & Subjects Descriptors

B. Hardware, B.7 Integrated Circuits, B.7.0 General.

General Terms

Design.

1. INTRODUCTION

Although static power consumption is becoming increasingly significant in current technologies, dynamic power is still the predominant source of power consumption in CMOS designs. This fact, together with the well-known quadratic dependency of dynamic power with respect to the supply voltage, explains the popularity of *supply voltage scaling* techniques as power reduction approaches.

Recently, voltage scaling has been increasingly applied at higher abstraction levels, where the benefits of voltage scaling can be combined to those derived from addressing the problem in the early stages of the design flow, possibly allowing voltage levels to change over time (*dynamic* voltage scaling).

Clearly, voltage cannot be scaled arbitrarily, because the delay of a CMOS device is roughly inversely proportional to the supply voltage. Therefore, the possibility of running accurate timing analysis on a design is essential, in order to assess the validity of voltage scaling approaches. This issue represents somehow the weakness of high-level approaches: Although they potentially provide sizable energy savings, they tend to be excessively conservative because of the lack of a reliable validation of their effect on timing. For example, techniques that operate at the task level evaluate the effects of scaling on latency by using estimates of the worst-case execution time of the tasks, an extremely pessimistic figure that may cancel most of the potentially achievable energy savings.

For this reason, gate or circuit-level solutions are usually more well accepted by designers, because they can guarantee that timing constraints are not violated by introducing lower supply voltages.

Several authors have addressed the voltage scaling problem at the gate level. All the approaches share two main features:

- Voltage scaling is applied *locally*, in the sense that the supply voltage-delay tradeoff allows to scale voltage only in some portions of a design;
- Voltage is considered a discrete quantity, that can assume a limited set of voltage levels (typically two). This is mainly due to technological reasons, because technology library are designed for specific supply voltage values. The small number of allowable voltages is instead due to efficiency reasons, because the routing of too many supply voltages would results in an unacceptable wiring overhead.

The various approaches basically differ in the way they try to solve the power-delay tradeoff. The problem of voltage scaling shares many similarities with that of gate resizing, in the sense that both methods consist of the replacement of some library cells with functionally equivalent ones.

Usami and Horowitz [1] propose the idea of *clustered voltage scaling* (CVS), where gates are clustered in two sets (corresponding to the high and low voltage levels), in such a way that the number of level shifters between the two voltage levels is minimized; shifters only needed at the primary outputs to restore the nominal (high) voltage level. In [9], Sundararajan and Parhi approach the problem in a similar way, and solved it using linear programming.

Chen and Sarrafzadeh [8] propose a more effective realization of the phase of cluster construction, that is based on a formulation of the problem as a maximum-weighted independent set problem.

Some other approaches combine voltage scaling with gate resizing, typically adopting a two-phase approach. Yeh and Chang [6] build clusters using the same formulation as in [8], and allow scaling and sizing to be carried out independently. Energy reductions provided by the scaling alone are however marginal with respect to the basic CVS.

GLSVLSI'02, April 18-19, 2002, New York, New York, USA.

In this work, we propose a voltage scaling technique that provides some basic variants to the basic CVS approach of [1] to improve effectiveness and performance. The results show that average improvements of up to 9 % can be achieved, using an accurate, pre-characterized energy model for the level shifters.

The paper is organized as follows: Section 2 reviews the basics of CVS, that constitutes the basis of the proposed algorithm, which is described in Section 3. Section 4 provides some experimental data that compare the plain and our modified CVS. Finally, Section 5 draws a few conclusion about the proposed solution.

2. CLUSTERED VOLTAGE SCALING

In this section we will describe the details of the clustered voltage scaling originally presented Usami and Horowitz ([1]): Its basic idea was that of preserving a *clustered* structure as a means to both achieve simplicity of algorithm and good quality of results. As in all voltage scaling algorithms, this proposal tries to reduce voltage supply level for those gates that are not critical for the circuit performance (i.e. gates that have a sufficiently high slack), without modifying sizes of gates and circuit topology. The method was originally proposed for a dual-voltage scenario that seems to be the most viable from technological point of view as discussed in the previous section. In the following, we will indicate with V_{DDH} and V_{DDL} respectively, the high and low voltage levels. The algorithm tries to optimize power consumption without worsening performance figures, and therefore compares with the case of all cells supplied with V_{DDH}. This case provides the most performing solution, due to the monotonical behavior of the curves delay versus supply voltage. The algorithm's core is a strategy to successively supply cells with V_{DDL} while maintaining the maximum performance. The main problem of a multiple voltage scheme in a netlist consists in the possibility of connecting gates with the "wrong" relationships of voltage levels. As a matter of fact, while a connection between a cell supplied by V_{DDH} to a cell supplied by V_{DDL} does not introduce particular problems, the opposite is not acceptable. This is due to the fact that a logical '1' corresponds to an electrical level of V_{DDL}, that might be too small to drive correctly the following gate. The consequence of this connection is thus a significant degradation of noise margins, and, possibly, the introduction of a non-valid electrical level (the gate downstream may not interpret it as a '1'), as well as some static dissipation in the second gate (see Figure 1). Since this type of connections cannot be eliminated in a dual-voltage scheme, the use of *level shifters* is mandatory to compensate the uneveness of voltage levels [3, 4]. Level shifters take as inputs low-voltage swing signals and return full-swing signals as outputs. An example of a typical level shifter is shown in Figure 1. Even if the introduction of level shifters solves all the electrical problems of low-to-high voltage connections they do not come for free: They introduce a new source of power dissipation, take more silicon area, and can add to the delay of the circuit. Therefore, they should be added as sparingly as possible.

The previous considerations make it interesting the idea of a voltage scaling strategy that limits the number of level shifters by construction: Every path from primary inputs to primary outputs will meet only a sequence of V_{DDH} to V_{DDL} gates and not the opposite one. Therefore, at most one level

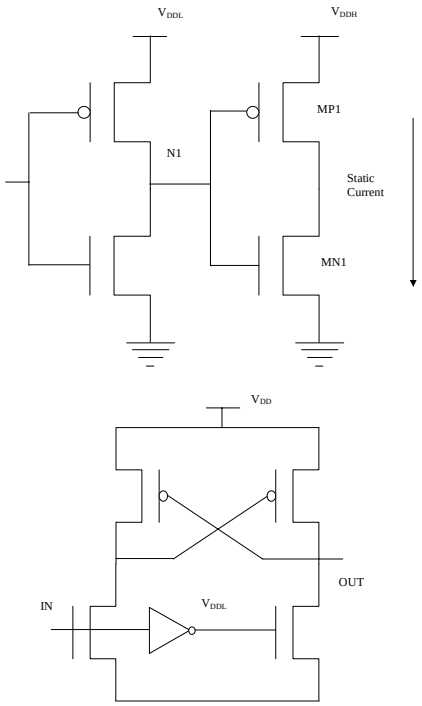

Figure 1: Level shifter

shifter will be needed, and only at the primary outputs, if they need to be supplied to V_{DDH}. Moreover, they can be embedded into the latches that are usually put at the POs of a cambinational circuit, thus reducing their impact on the overall design [1]. This kind of structure has been christened *Clustered Voltage Structure* because it results in the clustering of gates in two sets: A set of gates at V_{DDH}, connected to the primary inputs, followed by a set of gates at V_{DDL} feeding some primary outputs (see Figure 2).

CVS algorithm is a search algorithm on the netlist that tries to substitute as many cells as possible with V_{DDL} cells, *starting from the primary outputs*, with the constraint that the design performance is not changed. The details of the algorithm are reported in Figure 3.

In step 6, a cell is found viable if and only if its entire transitive fanout can be substituted without worsening the overall timing. This analysis is performed through a *forward* depth-first search (DFS), that stops when all the transitive fanout is analyzed or a substituition cannot be performed.

The order of visit of the primary outputs can be either obtained through decreasing slacks or decreasing loads. The experimental results show that the results are practically independent of this heuristics. An example of application of the algorithm is shown in Figure 4: the algorithm considers gate G8 first, substitutes it with a V_{DDL} cell, checks the new timing and concludes that substitution can be accepted. Then moves to cell G5. There is one child that is yet to be analyzed for substitution (G9), that is therefore substituted with a cell with V_{DDL}. Timing analysis reports an acceptable timing, therefore its substitution is retained. Then cell G5 is analyzed again: Its children are all V_{DDL} and therefore its own substitution is analyzed. It is accepted as well. then the analysis moves to cell G1, G3, G4, all of which give an unacceptable timing. In the end cell G10 is

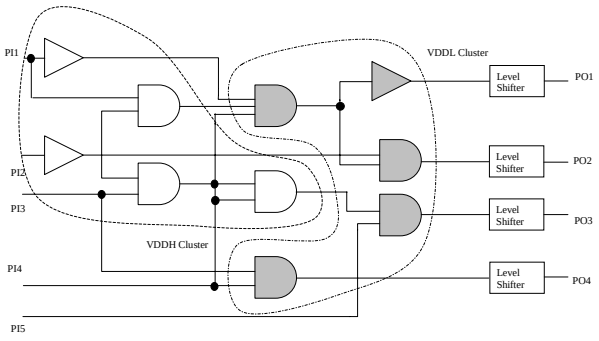

Figure 2: Clustered Voltage Structure

1 Pick a new cell C connected to a primary output
2 Substitute it with a V_{DDL} analogous cell
3 Perform a new static timing analysis
4 **If** the new timing worsen the original one go back to step 1
5 Pick a cell feeding the last substituted
6 Verify its viability for substitution through a DFS
7 **If** the new timing worsen the original one go back to step 5
8 **If** there are unanalyzed PO cells go back to step 1

Figure 3: Original CVS Algorithm.

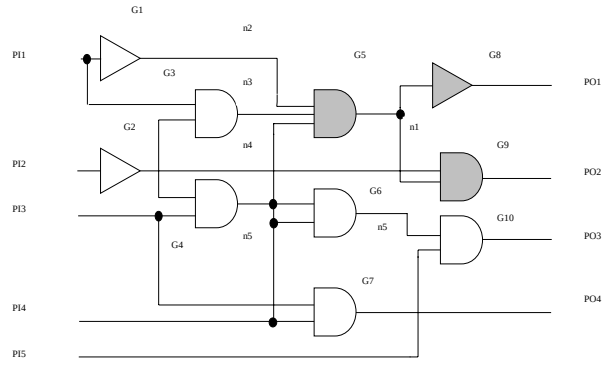

Figure 4: Original CVS algorithm: Example

considered, but it is not considered because it lies on the critical path.

The algorithm as it is suffers of some drawbacks, that are related mainly to the way search proceeds more than the rationale behind it (clustered voltage scheme). The next two sections show possible changes in the algorithm which will prove to be more efficient and/or effective.

3. ENHANCED CLUSTERED VOLTAGE SCALING

As explained in the previous section, search strategy of the original CVS algorithm doesn't strictly depend on the clustered structure. Therefore we looked for alternative implementations that improve results and/or execution time without changing the basic CVS.

3.1 Partial Depth First Search

The first observation on original CVS algorithm search is that the forward DFS might take a long time. In fact, whenever a node in the netlist is checked for its feasibility, if its fanout is greater than one, the algorithm runs forward DFS to check whether substitution is feasible for all its transitive fanout. The first modification we propose consists of stopping the search whenever a node is declared unfeasible. In practice, we perform a partial backward search following the flow of Figure 5.

1 Pick a new cell C connected to a primary output
2 Substitute C with a V_{DDL} analogous cell
3 Perform a new static timing analysis
4 **If** timing worsen
resubstitute C with the original one and go to step 9
5 Pick a cell C feeding the last substituted
6 Verify its viability for substitution through a DFS
7 Substitute C with a V_{DDL} analogous cell
8 **If** timing does not worsen go back to step 5
Else resubstitute C with the original one and go to step 9
9 **If** there are unanalyzed PO cells go back to step 1

Figure 5: Partial DFS Algorithm.

In this case, when a single DFS search on the transitive fanout is aborted, we skip to the following PO. The rationale behind this choice is that the search space will be pruned by cutting many subsitutions that are not likely to affect substantially the results, but saving computation time.

To show with an example the difference between the two approaches, let us consider Figure 6. All shaded nodes represent cells that could be substituted simultaneously without incurring in timing penalty. This is obviously possible only if the complete circuit has other - not shown - nodes, as the critical path is invariably made of V_{DDH} gates.

If we apply the Partial DFS algorithm, starting from node I10, we first substitute I10 with a V_{DDL} cell, then its parent node I7. In the following step the algorithm checks node I2 for feasibility and, as it has more than 1 children, its entire transitive fanout is checked. Node I3 turns out to be unfeasible, and therefore the entire search stops and no further nodes are analyzed. This concludes the search starting from PO I10. Analogously, starting from node I9, we substitute nodes I9 and I8 and then stop (I4 should feed I3 that has been already marked as unfeasible). The final result is shown in Figure 7: Only the four nodes I7, I8, I9 and I10

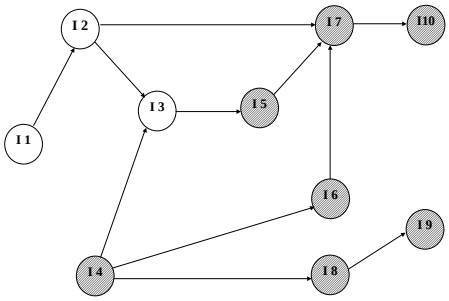

Figure 6: Example netlist

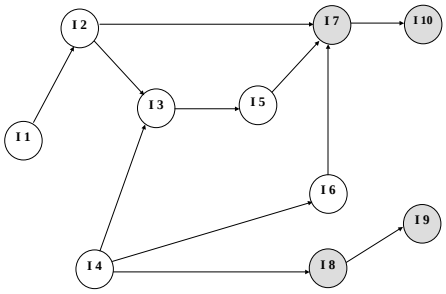

Figure 7: Application of Partial DFS algorithm

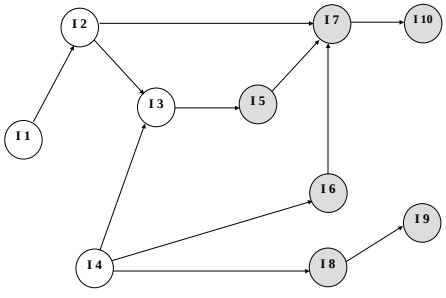

Figure 8: Application of original CVS algorithm

are implemented with V_{DDL} cells. If on the other hand we apply the original algorithm, after the visit of nodes I10 and I7, the recursive visit on node I2, before stopping for I3 infeasibility, analyzes node I5, which turns out to be feasible, and substitutes it. Analogously, node I6 is changed. The final result would be that of Figure 8, with two more V_{DDL} cells introduced. In the experimental section we will see that in fact the two algorithms differ slightly in their results.

3.2 Breadth First Search

An alternative implementation of the search algorithm consists of switching from a depth first to a breadth first search. This approach seems to be more suitable to the desired kind of optimization, as it visits the netlist level by level, starting from the POs and automatically building the clusters. The algorithm is detailed in Figure 9.

1 Put all POs but those on the critical path in list **NodeList**
2 Pick a new cell C from **NodeList**
3 Substitute C with a V_{DDL} analogous cell
4 Perform a new static timing analysis
5 **If** the new timing worsen the original one
  resubstitute C with the original V_{DDH} cell
  Else
  add all C's parents to NodeList
6 Delete C from NodeList
7 **If** Nodelist is not empty, go back to step 2

Figure 9: Full BFS Algorithm.

The algorithm implements a search strategy that subsequently analyzes gates that are increasingly farther from the POs; this solution represent a more "natural" ordering with respect to the desired clustering. This intuition is confirmed by the experimental results reported below.

4. EXPERIMENTAL RESULTS

In order to assess the advantages and drawbacks of the various algorithms, we have integrated them within Synopsys DesignCompiler [10], that allows to provide accurate power and timing figures to compare the various CVS implementations. The workflow is shown in Figure 10.

The given circuit is first analyzed, compiled and bound to a reference library with DesignCompiler. The results of the synthesis process are stored as structural VHDL files. These files are supplied to our parser which starts the process with an initial power estimation. Then the tool applies the chosen voltage assignment algorithm to the parsed netlist, and outputs an optimized VHDL netlist with some gates substituted with low-V_{DD} gates. The resulting netlist can therefore be fed back to DesignCompiler to provide the final power estimation together with timing verification (to guarantee that the flow does not increase the circuit delay).

The design flow has been validated through application to combinational ISCAS benchmarks [5], through evaluation of two parameters (percentage reduction of power, number of substituted instances) and their variance with respect to various input choices (technology library, PO visit heuristics, supply values).

The reference comparison has been made on the original CVS algorithm with respect to the case $V_{DDL} = 4V$, decreasing capacitance values for PO visit, and linear reduction of cell delay with respect to supply voltage increase.

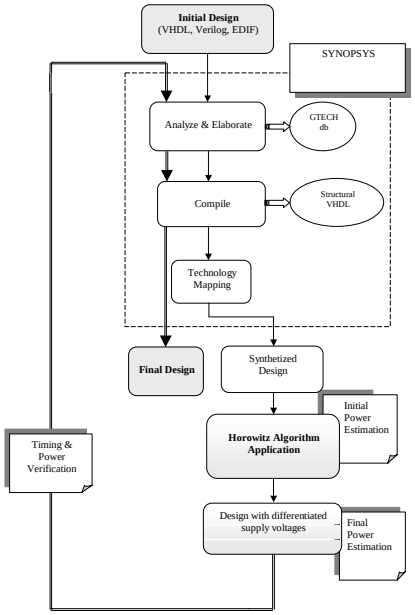

Figure 10: Work flow

Circuit	Initial Power μ W	Final Power μ W	Power Red.	Subst. Gates
c17	4.46	3.868	13.27%	50.00%
c432	168.783	154.427	8.51%	14.04%
c499	356.804	352.210	1.3%	26.8%
c880	215.563	181.566	15.77%	51.53%
c1355	405.087	400.84	1.05%	28.83%
c1908	273.217	260.892	4.51%	30.34%
c2670	569.65	486.108	14.66%	51.16%
c3540	781.1	749.862	4%	14.31%
c5315	1667.298	1464.582	12.15%	51.1%
c6288	5559.858	5540.156	0.35%	2.73%
c7552	2314.236	1999.324	13.6%	55.5%

Table 1: Results for original CVS algorithm, nominal conditions: $V_{DDL} = 4V$, C decreasing heuristics.

Under these conditions the results show an average reduction of power dissipation of 8.1% with the substitution of an avarage of 25.95% of gates. Table 1 reports the results for each benchmark circuit. Benchmark c6288 has proven to be particularly refractary to any power gain, due to the presence of a great number of gates in its critical path.

4.1 Algorithm comparison

Partial DFS algorithm shows a modest decrease in average power savings (8.1% to 7.7%). However, there is a significant gain (up to 2 times) in execution time (the number of visited instances is typically much smaller than the original CVS algorithm). For example for c6288, the biggest benchmark we have considered, the partial DFS algorithm requires 8 minutes of CPU time (linux workstation, PIII 256 Mb RAM), while the original CVS takes 20 minutes. Table 2 shows the details on the various benchmarks.

Circuit	Initial Power μ W	Final Power μ W	Power Red.	Subst. Gates
c17	4.46	3.868	13.27%	50.00%
c432	168.783	154.427	8.51%	14.04%
c499	356.804	353.227	1.00%	26.34%
c880	215.563	181.566	15.77%	51.53%
c1355	405.087	400.84	1.05%	28.83%
c1908	273.217	260.892	4.51%	30.34%
c2670	569.65	492.485	13.55%	47.37%
c3540	781.1	750.142	3.96%	14.11%
c5315	1667.298	1478.378	11.33%	47.59%
c6288	5559.858	5540.156	0.35%	2.73%
c7552	2314.236	2044.398	11.66%	47.80%

Table 2: Results for Partial DFS algorithm, nominal conditions: $V_{DDL} = 4V$, C decreasing heuristics.

As for BFS algorithm, results show this version of the algorithm is the best for both efficiency and performance (elaboration time 4 minutes, benchmark c6288): The power savings increase to an average of 9.34%, with the smallest values of CPU time. Table 3 shows the details on the various benchmarks.

4.2 Sensitivity Analysis

The first among the considered parameters is the intrinsic delays of gates, that influences overall timing and feasibility analysis. In order to test the sensitivity of the optimization results with respect to these parameters, experiments have been run with superlinear and sublinear variations of delays (20% variation in the nominal case, 15% and 25% in the sublinear and superlinear cases). The results show a negligible variation of substituted gates and power savings, thus illustrating the robustness of the approach with respect to the accuracy of timing models.

To investigate the influence of the heuristics on PO ordering, we ran the experiments by varying the optimization conditions: POs were ordered according to capacitance values, decreasing slacks and randomly. The results (not reported) show the limited influence of this heuristic to the quality of optimization.

V_{DDL} variations, on the other hand, turned to be important for the overall quality of optimization, as it might

Circuit	Initial Power μ W	Final Power μ W	Power Red.	Subst. Gates
c17	4.46	3.868	13.27%	50.00%
c432	168.783	154.427	8.51%	14.04%
c499	356.804	352.210	1.3%	26.8%
c880	215.563	179.732	16.62%	55.1%
c1355	405.087	400.84	1.05%	28.83%
c1908	273.217	260.606	4.6%	30.89%
c2670	569.65	457.539	19.68%	64.9%
c3540	781.1	747.315	4.3%	15.92%
c5315	1667.298	1430.295	14.2%	55.8%
c6288	5559.858	5540.156	0.35%	2.73%
c7552	2314.236	1878.539	18.82%	72.5%

Table 3: Results for BFS algorithm, nominal conditions: $V_{DDL} = 4V$, C decreasing heuristics.

have been guessed. We repeated the experiments with V_{DDL} equal to 3.3V and 2.5V. The results show a progressively decreasing number of substitution together with an increase of power savings (11.69 % and 15.45% respectively). This can be explained by the fact that the substituted instances, even if less in number, contribute with quadratically increasing power savings.

4.3 Timing verification

As outlined above, timing verification is performed with DesignCompiler. The resulting timing figures for the original and the modified circuits are compared in order to check potential discrepancies between their critical delays. The differences turned out to be negligible (less than 1 % of the total delay), well below the tolerance of timing models.

5. CONCLUSIONS

We proposed two alternative approaches of the clustered voltage scaling (CVS) algorithm, based on modified implementations of the underlying search strategy. The results show the possibility of meaningful power reduction with a limited impact on the design flow. The advantages are highly dependent on features of implemented circuits. However, even with delay-optimized designs, therefore having very small margins for intervention, savings are non-negligible.

6. REFERENCES

[1] K. Usami, M. Horowitz, " Clustered Voltage Scaling Technique for Low-Power Design" *ISLPD'95: International Symposium on Low-Power Design*, pp.3–8, Dana Point, CA, 1995.

[2] J.-M. Chang, M. Pedram, " Energy Minimization Using Multiple Supply Voltages" *ISLPD'96: International Symposium on Low-Power Design*, pp.157–162, Monterey, CA, 1996.

[3] N. Otsuka, M. Horowitz, " Circuit Techniques for 1.5-V Power Supply Flash Memory " *IEEE Journal of Solid State Circuits* vol. 32, No. 8, pp.1217–1230, August 1997.

[4] Y. Kanno, H. Mizuno, K. Tanaka, T. Watanabe, " Level Converters with High Immunity to Power-Supply Bouncing for High-Speed Sub-1-V LSIs " *2000 IEEE Symposium on VLSI Circuits*, pp. 202–203, 2000.

[5] F. Brglez, H. Fujiwara, " A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran" *ISCAS'85: IEEE International Symposium on Circuits and Systems*, pp.785-794, 1985.

[6] C. Yeh, M.-C. Chang "Gate-Level Voltage Scaling for Low-Power Design Using Multiple Supply Voltages", *IEE Proceedings – Circuits, Devices and Systems*, Vol. 146, No. 6, pp. 334–339, 1999.

[7] J.-M. Chang, M. Pedram " Energy Minimization Using Multiple Supply Voltages" *IEEE Transactions on VLSI Systems*, Vol 5., No. 4, pp. 1–8, 1997.

[8] C. Chen, M. Sarrafzadeh " Provably Good Algorithm for Low Power Consumption with Dual Supply Voltages" *ICCAD'99: ACM/IEEE International Conference on CAD*, pp. 76–79, San Jose, CA, 1999.

[9] V. Sundararajan, K. K. Parhi " Synthesis of Low Power CMOS VLSI Circuits using Dual Supply Voltages" *DAC-36: 36th ACM Design Automation Conference*, pp.72–75, New Orleans, LA, 1999.

[10] Synopsys DesignCompiler, *http://www.synopsys.com.*