Algorithms for Leakage Reduction with Dual Threshold Design Techniques

Konrad Engel Thomas Kalinowski Roger Labahn
University of Rostock
Institute for Mathematics
18051 Rostock, Germany
Email: {konrad.engel, thomas.kalinowski, roger.labahn}@uni-rostock.de

Frank Sill Dirk Timmermann
University of Rostock
Department of CSEE
18119 Rostock, Germany
Email: {frank.sill, dirk.timmermann}@uni-rostock.de

***Abstract*— The application of devices with different threshold voltages is a state-of-the-art VLSI design technique to reduce the power consumption based on leakage currents. As devices with reduced leakage dissipation have longer delay, the aim of such Dual Threshold CMOS (DTCMOS) approaches is the detection of non-critical gates regarding the circuit's performance to exchange these with slower but less leaky gates. In our talk, we consider the optimization of DTCMOS circuits, which are modeled as directed acyclic graphs. With each vertex v of a directed acyclic graph, we associate two delay values $d_0(v) \leq d_1(v)$ and two leakage values $c_0(v) \geq c_1(v)$. The objective is to choose one of the indices 0 or 1 for each vertex, such that the corresponding total delay along any directed path does not exceed the maximum circuit's delay and that the total leakage is minimized. It is well-known that this problem is NP-hard. We present heuristic approaches to the problem that are based on k–cutsets and k–Sperner families in partially ordered sets.**

I. Introduction

The reduction of power dissipation is one of the main tasks in current research. Users demand for high mobility, long runtimes, and high performance at the same time. These requirements resulted in aggressive downscaling of technology sizes, which leads to reduction of the capacitive load per logic gate. Unfortunately, the influence of short channel and tunneling effects has increased exponentially, which resulted in increasing leakage currents. Hence, in current predictions, the power dissipation due leakage currents will be up to 50 % of the whole power dissipation [1]. The consideration of idle modes is a very common approach to reduce leakage currents. This includes the technique of disconnecting the supply voltage from the design by sleep transistors if the design has no tasks [2]. Further, special input vectors can be applied in idle modes [3]. This Minimum Leakage Vector technique bases on the fact that the leakage of a gate depends on its input vector. Maken et. al. proposed in [4] the dynamic scaling of supply voltage to the required performance. Another widely used approach is the application of two device types, which vary in their threshold voltage V_{th}. This Dual Threshold CMOS (DTCMOS) design technique uses fast low threshold voltage (LVT) and slow high threshold voltage (HVT) devices. The advantage of HVT devices is their small leakage current. Thus, the aim of DTCMOS is to maximize the gain in leakage by the use of HVT devices without worsening the performance of the circuit. The problem of choosing adequate values for the two threshold voltages is addressed in [5, 6]. We assume these values are fixed, so our task is to assign one of two given threshold voltages to each gate. Li et al. [7] have shown that already very special cases of this problem are NP-complete. Several heuristics have been proposed in the literature. Sundararajan and Parhi [8] consider the set of gates that can potentially be implemented as HVT devices and determine an adequate subset by solving an integer linear programming problem (ILP). In the heuristic variant of their algorithm, solving of the ILP is replaced by a sequence of solvings of an LP-relaxation. The algorithm of Wei et al. [9] starts with all gates at low threshold voltage, visits the gates in a breadth first order, and switches a gate to high threshold voltage if this does not violate the timing constraint. Sill et al. [10] present a simple algorithm, which sequentially scans the circuit and transfers every device in critical paths into an LVT device. As usual, also in this paper the circuit is modeled as a directed acyclic graph (dag), where for each vertex we have to decide whether it operates on high or on low threshold voltage. The objective is to make this decision in such a way that the performance of the circuit is optimal and the leakage is minimized. In this formulation the problem is an instance of the discrete time–cost tradeoff problem, which arises also in other contexts and has been subject of intensive research (see [11] for a survey and [12] for results on approximation algorithms). The paper is organized as follows; Section II presents a basic introduction into the DTCMOS approach. In section III, we formally state the problem as an optimization problem on a dag. In sections IV-VI, we propose four heuristic algorithms for the selection of the HVT devices, and finally in section VII, we present some test results on benchmark circuits. For sake of brevity, proofs and implementation details are omitted.

II. Preliminaries

A. Transistor equations

In nanometer technologies with gate lengths smaller than 100 nm the delay of a gate can be approximated as follows:

$$delay \propto \frac{C_{load} V_{DD}}{(V_{DD} - V_{th})^{\alpha}}$$

C_{load} labels the load capacitance of the gate, V_{DD} is the supply voltage, V_{th} is the threshold voltage, and α is the velocity saturation index [13]. From this equation follows that the reduction of V_{th} results in longer gate delay.

Ideally, CMOS gates draw no current and dissipate no power when idle. Unfortunately, this is not true for real gates. The biggest influence originates from sub-threshold current I_{sub}, which is the current between source and drain of a transistor when the device should be cut off ($V_{gs} < Vth$). A common used approximation is [14]:

$$I_{sub} = I_0' \cdot e^{\frac{q}{nkT}(V_{gs}-V_{th})} \left(1 - e^{-\frac{kTV_{ds}}{q}}\right)$$

I_0' labels the zero-bias current, n is the subthreshold swing coefficient, k is the Boltzmann's constant, T is the operating temperature, q corresponds to charge on an electron, V_{gs} is the gate-source voltage, and V_{ds} is the drain-source voltage. Following from this equation the reduction of the threshold voltage results in a higher sub-threshold leakage current.

B. Dual Threshold CMOS (DTCMOS)

Data signals traverse integrated circuits through different paths of logic gates whereas the start- and endpoints of these paths are marked by sequential elements like registers. The maximum frequency to clock the registers is determined by the path with the longest propagation delay, called critical path. Thus, it is possible to trade off delay for leakage in all other, non-critical paths. Such an attempt is exploited by the Dual Threshold CMOS (DTCMOS) design technique. Therefore, this approach offer various gates for the same logical function that differ in evaluation delay and power dissipation due to leakage current. As shown, the transistor's threshold voltage V_{th} mainly determine delay and leakage. So, DTCMOS approaches apply fast gates with transistors that have low V_{th} (LVT gates). Further, gates are used that have the same logical functions but consist of transistors with high V_{th} (HVT gates). These gates offer slower evaluation but also decreased leakage currents. Finally, to reduce the design's overall leakage currents at constant performance, as many HVT gates as possible are applied to the non-critical paths so that the delays of the paths do not exceed the critical path delay which consists solely of LVT gates [8].

III. Mathematical formulation of the problem

Let $G = (V, E)$ be a dag and let $d_i, c_i : V \to \mathbb{R}_+$ ($i = 0, 1$) be weight functions on the vertex set such that for all $v \in V$, $d_0(v) \leq d_1(v)$ and $c_0(v) \geq c_1(v)$. $d_0(v)$ and $d_1(v)$ can be interpreted as the delays of vertex v, where v is of LVT and HVT type, respectively. Similarly, $c_0(v)$ and $c_1(v)$ are the corresponding leakage values of vertex v. In the following we use graph theoretic terminology and speak of length and cost instead of delay and leakage. Throughout we assume w.l.o.g. that G has a unique source q and a unique sink s with $d_0(v) = d_1(v) = c_0(v) = c_1(v) = 0$ for $v \in \{q, s\}$. Let $x : V \to \{0, 1\}$ be a decision function with the interpretation that $x(v) = 1$ if vertex v is HVT and $x(v) = 0$ otherwise. The pair (G, x) is called a *realization*. The *cost* of the realization is given by

$$c(G, x) := \sum_{v \in V} c_{x(v)}(v).$$

For a path $P = (v_0, \ldots, v_k)$ in G and a realization (G, x), the *length* of P is defined by

$$d(P, x) := \sum_{v \in P} d_{x(v)}(v).$$

The *length* of the realization (G, x) is given by

$$d(G, x) := \max\{d(P, x) \ : \ P \text{ is a } q\text{-}s\text{-path}\}.$$

Let x_0 be the zero function, i.e. $x_0(v) = 0$ for all $v \in V$. Clearly, (G, x_0) realizes the minimum length $d_{min}(G) := d(G, x_0)$. The DTCMOS-problem (equivalent to the deadline formulation of the discrete time-cost tradeoff problem [12]) is the following:

DTCMOS: Find a decision function $x : V \to \{0, 1\}$ such that $d(G, x) = d_{min}(G)$ and $c(G, x)$ is minimum.

Using a shifting we may assume w.l.o.g. that $c_1(v) = 0$ for all $v \in V$. For brevity we set $c(v) := c_0(v)$ and obtain

$$c(G, x) = \sum_{v \in V : x(v) = 0} c(v).$$

DTCMOS is known to be strongly NP-complete [15]. This gives reason to look for well-founded heuristics. Though the problem can be formulated as a 0-1-linear programming problem ILP-solvers are not the adequate tool since the graph can be very large. We present four such heuristics and compare their performance on practical examples. But first we need some more notation. For $x : V \to \{0, 1\}$ and $v \in V$, we put

$$\begin{aligned} e(v, x) &:= \max\{d(P, x) \ : \ P \text{ is a } q\text{-}v\text{-path}\} - d_{x(v)}(v), \\ h(v, x) &:= \max\{d(P, x) \ : \ P \text{ is a } v\text{-}s\text{-path}\}. \end{aligned}$$

The values $e(v, x)$ and $h(v, x)$ can be easily obtained by a recursive version of depth first search (DFS, cf. Sedgewick [16], 105 ff.). Interpreting the vertices as jobs, where an edge vw indicates that job v has to be completed before job w can be started, the value $e(v, x)$ can be considered as the earliest starting time for job v, while $d(G, x) - h(v, x)$ is the latest starting time under the timing constraint. The *slack* at vertex v is defined by $slack(v, x) := d(G, x) - (e(v, x) + h(v, x))$. For the sake of brevity we put $dif(v) := d_1(v) - d_0(v)$.

IV. The single and the multiple switch algorithm (SSA and MSA)

A vertex is called a *candidate vertex* if its decision can be switched from 0 to 1 without changing the length of the realization. The *candidate set* C is the set of all candidate vertices, i.e.

$$C := \{v \in V \ : \ x(v) = 0 \text{ and } slack(v, x) \geq dif(v)\}.$$

A more general approach is as follows:

Lemma 1. *Let $S \subseteq V$ be a subset of vertices such that $x(v) = 0$ and $slack(v,x) \geq \sum_{w \in S} dif(w)$ for all $v \in S$. Moreover, let*

$$x'(w) := \begin{cases} 1 & \text{if } w \in S, \\ x(w) & \text{otherwise.} \end{cases}$$

Then

$$d(G,x') = d(G,x) \quad \text{and} \quad c(G,x') \leq c(G,x).$$

In the first two algorithms, one starts with the zero function x_0 and then successively switches one or more vertices $v \in C$ from 0 to 1 without changing the length but decreasing the cost. The choice of such vertices can be done in the following way: from a heuristic point of view it is wise to take vertices with large slack (they do not have much influence on other vertices) and with large cost (they contribute more to decrease the cost). Let for some $\lambda > 0$

$$weight(v) = slack(v,x) - dif(v) + \lambda c(v).$$

In the Single Switch Algorithm we proceed by switching one single vertex of maximal weight in each step. The same method with other score functions is part of the algorithm proposed by Li et al. [7].

In practical examples, we observed that the number of switched vertices grows linearly with n. So for large n, it is better to switch more than one vertex in each iteration step. For this, we order the vertices of the candidate set C by decreasing weight and take a large initial segment of this ordering for the set S used in Lemma 1. This initial segment can be determined step-by-step as follows. Let $C = \{v_1, \ldots, v_k\}$ with $weight(v_1) \geq \cdots \geq weight(v_k)$. At stage $S = \{v_1, \ldots, v_l\}$ ($l < k$), we can insert v_{l+1} into S if $slack(v_i, x) \geq \sum_{j=1}^{l+1} dif(v_j)$ is true for all $i = 1, \ldots, l+1$. In an iteration of the Multiple Switch Algorithm we switch all vertices belonging to the current set S from 0 to 1.

V. THE k-FAMILY ALGORITHM (k-FA)

As in the previous section we start with the zero function x_0 and switch iteratively a set of vertices from the candidate set C to $x(v) = 1$ keeping the length $d(G,x)$ invariant. Let

$$maxdif(G) := \max\{dif(v) \ : \ v \in V\}.$$

We may assume $maxdif(G) > 0$, because otherwise an optimal solution is given by $x(v) = 1$ for all $v \in V$. For any real number $\kappa \geq 0$, let

$$k := \max\{\lfloor \kappa \rfloor, 1\}$$

and define a partially ordered set (poset) $(Q_x^\kappa, \leq)$ by

$$Q_x^\kappa := \{v \in V \ : \ x(v) = 0, \ slack(v,x) \geq \kappa \cdot maxdif(G)\}$$

if $\kappa > 1$ and $Q_x^\kappa = C$ if $\kappa \leq 1$, and $v \leq w \ :\Leftrightarrow$ There is a v-w-path in G.

A subset $S \subseteq Q_x^\kappa$ is called a *k-family* in $(Q_x^\kappa, \leq)$ if there is no chain in $(Q_x^\kappa, \leq)$ containing more than k elements from S.

Lemma 2. *Let S be a k-family in $(Q_x^\kappa, \leq)$ and let*

$$x'(w) := \begin{cases} 1 & \text{if } w \in S, \\ x(w) & \text{otherwise.} \end{cases}$$

Then

$$d(G,x') = d(G,x) \quad \text{and} \quad c(G,x') \leq c(G,x).$$

Switching the elements of a k-family S reduces the cost by $\sum_{v \in S} c(s)$, hence it is wise to choose a k-family of maximum cost. For the same reason as in section IV it is helpful to replace $c(v)$ by a weight $weight(v)$, where e.g.

$$weight(v) = 1 + maxdif(G) - dif(v) + \lambda c(v), \quad \lambda > 0.$$

So we have reduced the original problem to the task of finding a k-family of maximum weight. A min-cost-flow algorithm for this problem is described in Chapter 4 of [17].

VI. THE k-CUTSET ALGORITHM (k-CA)

Again, we start with the zero function x_0. Assume that after some steps, we have a realization (G,x) with $d(G,x) = d_{min}(G)$. As in the previous sections we work with the set C of candidate vertices. If C is nonempty we switch all its elements simultaneously. Clearly this improves the cost, but in general, this increases the length as well. Hence we have to switch back some vertices in order to obtain the minimum length $d_{min}(G)$. Now we change notation, and assume that (G,x) is a realization with $d(G,x) > d_{min}(G)$. We describe a step by which $d(G,x)$ is decreased and the cost increase is minimum. We restrict the presentation to one variant:

Let $G_x = (V_x, E_x)$ be the *critical graph*, i.e. the graph defined by

$$V_x := \{v \in V \ : \ slack(v,x) = 0\},$$
$$E_x := \{vw \in E \ : \ v, w \in V_x \text{ and } e(v,x) + d_{x(v)}(v) = e(w,x)\}.$$

We say that a set $S \subseteq V_x$ is a *k-cutset* in G_x iff every q-s-path in G_x has at least k vertices in S.

Let

$$k := \left\lceil \frac{d(G,x) - d_{min}(G)}{maxdif(G)} \right\rceil.$$

Lemma 3. *Let $d(G,x) > d_{min}(G)$ and assume that $S \subseteq V_x$ is a set of vertices with $x(v) = 1$ for all $v \in S$ and such that*

$$x'(v) := \begin{cases} 0 & \text{if } v \in S, \\ x(v) & \text{otherwise,} \end{cases}$$

defines a realization (G,x') with $d(G,x') = d_{min}(G)$. Then S is a k-cutset in G_x.

Our approach is to switch the vertices of a minimum weighted k-cutset in G_x to zero, and iterate this until $d_{min}(G)$ is reached. The following weight function is used:

$$weight(v) = c(v) - \mu \, dif(v),$$

where μ is some small positive constant. The determination of a minimum k-cutset can be accomplished by a min-cost-flow algorithm. This is completely described in Chapter 4 of [17] in terms of posets. When $d_{min}(G)$ is reached we determine the new candidate set C and start the whole procedure again.

VII. Experimental results

We implemented the four algorithms in C++. Further, we applied a gates library based on the modified predictive 65 nm BPTM technology [18, 19]. The ten ISCAS'85 [20] netlists form the test data. Clearly, the results strongly depend on the functions $d_i, c_i : V \to \mathbb{R}_+ (i = 0, 1)$. We applied average-leakage values (see table I), but it is also possible to use leakage values, which depends on gate's input states. Table II contains the percentage of improvement of the cost for leakage compared with the zero function x_0. For comparison we add results from a previous work [10]. The bold numbers indicate the best result for the respective circuit. One can see that MSA and k-CA are mostly the best ones. In contrast to the previous work the results of the new algorithm are better in almost every case, whereas the difference is in some cases greater than 20 %. Table III contains the running times (in seconds) of the algorithms on a 1.8 GHz PC for the largest ISCAS'85 [20] circuits c7552 and c6288. It can be observed that the MSA and k-CA algorithms need the shortest running times.

VIII. Conclusions

In this paper, we present four novel heuristic algorithms for the improvement of Dual Threshold CMOS designs to reduce the power consumption based on leakage currents. The results indicate a leakage reduction of up to 81 % compared to non-optimized designs with the same performance. In contrast to previous works the algorithms improve the leakage reduction by up to 25 %. As all new algorithms achieve almost equal leakage reduction results we recommend the MSA and k-CA algorithms for application in DTCMOS designs due to the shortest computation times.

	Delay $[ps]$		average-leakage $[nW]$	
Gate type	d_0 (LVT)	d_1 (HVT)	c_0 (LVT)	c_1 (HVT)
INV	37	46	92.8	12.6
NAND2	43	58	135.0	20.3
AND2	59	81	253.9	37.5
NOR2	66	90	86.0	10.6
OR2	71	98	151.9	20.9

TABLE I

The delay and average-leakage values of applied gates

design [20]	$\sum$ gates	SSA	MSA	k-FA	k-CA	[10]
c432	155	**44.76**	43.94	43.87	44.16	39.06
c499	474	56.10	56.43	57.08	**58.28**	35.69
c880	393	**72.27**	**72.27**	**72.27**	**72.27**	61.65
c1355	738	57.43	58.79	**62.39**	62.37	38.84
c1908	453	59.45	60.61	60.47	61.07	**63.94**
c2670	663	77.85	77.89	77.79	**77.90**	64.50
c3540	1093	**75.28**	**75.28**	75.05	75.06	61.26
c5315	1781	79.03	**79.12**	78.87	78.59	62.00
c6288	2701	53.13	52.59	53.08	**53.60**	36.89
c7552	1874	81.49	**81.54**	81.00	80.75	65.32

TABLE II

Leakage reduction by application of novel algorithms compared to non-optimized versions in %

References

[1] N.S. Kim, T. Austin, T. Blaauw, T. Mudge, K. Flautner, H.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *IEEE Computer*, 36(12):68–75, 2003.

[2] M. Anis and M. Elmasry. *Multi-Threshold CMOS Digital Circuits*. Kluwer Academic Publishers, 1st edition, 2003.

[3] Duarte D. E. Vijaykrishnan N. Tsai, Y. and M. J. Irwin. Characterization and modeling of run-time techniques for leakage power reduction. *IEEE Transactions on VLSI Systems*, 12(1):1221–1232, 2004.

[4] P. Maken, M. Degrauwe, M. Van Paemel, and H. Oguey. A voltage reduction technique for digital systems. In *1990 IEEE International Solid-State Circuits Conference*, pages 238–239, 1990.

[5] M. Hamada, Y. Ootaguro, and T. Kuroda. Utilizing surplus timing for power reduction. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 89–92, 2001.

[6] M. Liu, W.-S. Wang, and M. Orshansky. Leakage power reduction by dual-Vth designs under probabilistic analysis of Vth variation. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED'04)*, pages 2–7, 2004.

[7] W.-N. Li, A. Lim, P. Agrawal, and S. Sahni. On the circuit implementation problem. In *Proceedings of the 29th ACM/IEEE conference on Design Automation (DAC)*, pages 478–483, 1993.

[8] V. Sundararajan and K. Parhi. Low power synthesis of dual threshold voltage CMOS VLSI circuits. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 139–144, 1999.

[9] L. Wei, Z. Chen, K. Roy, M.C. Johnson, Y. Ye, and V.K. De. Design and optimization of dual-threshold circuits for low-voltage low-power applications. *IEEE Transactions on VLSI Systems*, 7(1):16–24, 1999.

[10] F. Sill, F. Grassert, and D. Timmermann. Low power gate-level design with mixed-Vth (MVT) techniques. In *17th Symposium on Integrated Circuits and Systems (SBCCI)*, pages 278–282, 2004.

[11] P. De, E.J. Dunne, J.B. Ghosh, and C.E. Wells. The discrete time–cost tradeoff problem revisited. *European Journal of Operational Research*, 81:225–238, 1995.

[12] M. Skutella. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, 23(4):909–929, 1998.

[13] T. Sakurai and R. Newton. Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2), 1990.

[14] Xuemei X. Mohan D. He J. Liu W. Cao K. M. Jin X. Ou J. J. Chan M. Hu, S. and A. M. Niknejad. Berkeley short channel igfet model version 4.5. Technical report, Dpt. of EECS, University of California, Berkeley, 2005.

[15] P. De, E.J. Dunne, J.B. Ghosh, and C.E. Wells. Complexity of the discrete time–cost tradeoff problem for project networks. *Operations Research*, 45:302–306, 1997.

[16] R. Sedgewick. *Algorithms in C++ Part 5: Graph Algorithms*. Addison-Wesley, 3rd edition, 2001.

[17] K. Engel. *Sperner Theory*. Cambridge University Press, 1997.

[18] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive mosfet and interconnect modeling for early circuit design. In *CICC*, pages 201–204, 2000.

[19] F. Sill, F. Grassert, and D. Timmermann. Total leakage power optimization with improved Mixed Gates. In *18th Symposium on Integrated Circuits and Systems (SBCCI)*, pages 154–159, Florianopolis, Brazil, September 2005.

[20] F. Brglez and H. Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits. In *IEEE International Symposium on Circuits and Systems*, pages 695–698, New Jersey, USA, 1985. IEEE Press.

	Design	$\sum$ gates	SSA	MSA	k-FA	k-CA
average-leakage	c7552	1874	6.64	0.45	1.32	0.39
	c6288	2701	10.27	1.97	7.00	7.78

TABLE III

Running times of the different algorithms in seconds