MINIMIZING LEAKAGE: WHAT IF EVERY GATE COULD HAVE ITS INDIVIDUAL THRESHOLD VOLTAGE?

Ralf Salomon, Frank Sill, and Dirk Timmermann
Faculty of Computer Science and Electrical Engineering
University of Rostock
18051 Rostock, Germany
email: {ralf.salomon,frank.sill,dirk.timmermann}@uni-rostock.de

ABSTRACT
Designers aim at fast but low-power consuming integrated circuits. Since high processing speed always comes with high energy demands, the literature provides several ways to reduce a circuit's power dissipation. Even though technologically not possible today, this paper hypothesizes that the design would allow for choosing a gate's from a continuous domain in order to explore options for future improvements. That is, the design goal consists in selecting a gate's delay such that both the circuit's delay and its power consumption assumes minimal values. Since the resulting optimization problem is multi-dimensional and might also contain local optima, this paper utilizes genetic algorithms for this task. On a selection of the well-known ISCACS test suite, the genetic algorithms reduced the circuits' leakage values by about 20-50 %; with respect to non-optimized circuits, the improvement is about 60-80 %.

KEY WORDS
Genetic Algorithms, Real-World Application, Low-Power VLSI, Optimization

1 Introduction

Off-the-shelf products offered virtually everywhere indicate that the processing speed of digital devices, such as personal computers, laptops, personal digital assistents, cellular phones, and the like, is of high importance to many end-users. In other words, end-users expect their devices to operate at a processing speed as *high* as possible. With respect to *mobile devices,* the markets today also suggest another trend: mobile devices are expected to yield times-of-operation as long as possible, probably in order to maximize the end-user's independence on electrical wires. A high processing speed paired with a long time-of-operation are one of *the* driving forces for research on low-power technologies. Section 2 briefly reviews the technological background as well as the relation between energy consumption and processing speed. It turns out, unfortunately, that these two parameters compete with each other.

The requirements of both high processing speed and long times-of-operation has led to an aggressive downscaling of technological sizes, as it has occurred in the last years [2]. Downscaling has greatly reduced the capacitive load per logic gate, which in turn is yielding higher performance at a reduced dynamic power dissipation per logic gate. Downscaling, however, has also its drawbacks: the influences of short channels and tunneling effects have an exponentially increasing effect on leakage currents [8]. Current predictions forecast that the power dissipation due to leakage currents will be of up to 50 % of the circuit's entire power dissipation [8].

The literature proposes several approaches to reduce the circuit's leakage currents without reducing its processing speed. One option is to consider idle modes of gates or larger parts of the circuit. Some, employ *sleep transistors* to disconnect gates from the power supply [3]. Others apply particular input vectors in order to keep those gates in a state in which they consume as less energy as possible [17]. Furthermore, the insertion of additional gates can further improve the energy savings [19]. In addition, the literature has proposed to dynamically scale the supply voltage to meet the user's performance requirements [10].

Despite its improvements, the insertion of additional devices is not without drawbacks. Sleep transistors, for example, increase the circuit's delay[3], and can only be used, if the *entire* part of interest is in idle mode. Moreover, design optimizations that are based on single transistors require huge computational resources [18]. In addition, most of the existing design tools support optimizations on rather a gate level [18]. Another option consists in the application of two device types, which vary in their threshold voltages V_{th} [16], or the thicknesses of the gate oxide (T_{ox}) [15]. These dual-V_{th} CMOS (DVTCMOS) and dual-T_{ox} CMOS (DTOCMOS) design techniques use fast gates in critical paths whereas they use slower gates with lower leakage power consumption in non-critical ones.

The limitation to two values is due to obvious technological reasons. In order to explore the possibilities for future achievements, this paper hypothesizes that the design would allow for choosing *any* value for V_{th} (T_{ox}), i.e., choosing from a continuous domain rather than just two or three values. That is, the design goal is to select n different values for the delay of n gates, such that both the circuit's delay and its power consumption is minimal. Since the resulting optimization problem is multi-dimensional and might also contain local optima, Section 3 describes how to utilize evolutionary algorithms for this task.

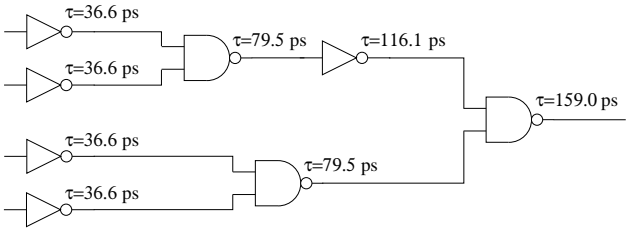

Figure 1. A simple CMOS-circuit with different path delays, caused by different numbers of gates in each path. The lower path is non-critical, and thus may be subject to the implementation in slow high-voltage technology. Here, τ refers to the cumulated delays from the circuit's input to the gate's output, rather than the gate's individual delays.

For an evaluation of the proposed exploration, this paper resorts to some test designs drawn from the well-known ISCAS test suite [5]. Section 4 provides a short description of these tasks, and also summarizes all relevant parameter settings. The results presented in Section 5 suggest that the selected algorithms evolve designs better than previously reported; On a selection of the well-known ISCACS test suite, the genetic algorithms reduced the circuits' leakage values by about 30-80 % as compared to non-optimized circuits. Finally, Section 6 concludes with a brief discussion.

2 Background and Previous Research

A digital VLSI circuit generally consists of very many gates of different types, such as `NAND`, `NOR`, inverters, etc., and varying numbers of inputs, which together realize the circuit's functionality, e.g., a network adapter, a serial port, etc. Each gate requires some time to process its input data. This time is generally called the gate's delay τ [1]. Figure 1 presents a simple example, which allows for the following two observations: First, the delay τ is not equivalent for all gates but depends on both the gate's functionality[2] and the number of inputs. Second, not all gates are equally important for the circuit's overall delay. In this example, the upper path has more gates in sequence and thus constitutes the circuit's critical path, since it determines its overall delay, whereas the lower path processes its signals faster anyhow. In other words, the gates residing in the non-critical path could be processing their signals at a lower speed without affecting the circuit's overall delay to some extent and thus could be saving valuable energy, if that was technologically possible.

It might be well known to many readers that every gate is realized by a certain number of transistors. A transistor's electrical characteristics depend, among other things, on a specific design parameter, called the gate threshold-voltage V_{TH}. Given a specific technology, this parameter determines both the transistor's processing delay as well as its power consumption. On the gate level, the (static) power consumption is described by a technical parameter called *leakage*; this paper uses the terms power consumption and leakage synonymously. Conversely, if either the transistor's delay or its power consumption is given, the other parameter is also determined. Unfortunately, these two parameters are inversely correlated by their very nature. That is, a transistor has either a short delay and a high power consumption or vice versa.

Table 1. This table shows three different realizations of a `NOR-2` with two inputs and an inverter.

`NOR-2`		`INV`	
Delay	Leakage	Delay	Leakage
66.3 ps	86.0 nA	36.6 ps	92.8 nA
78.0 ps	36.6 nA	37.6 ps	62.5 nA
90.0 ps	10.6 nA	45.8 ps	12.6 nA

It used to be that the very same threshold voltage V_{TH} had to be used for *all* transistors throughout the entire VLSI circuit. State-of-the-art design technologies [6, 7, 16], however, allow for using varying threshold voltages for the transistors. The designer can thus fine tune both the delay and the power consumption of every gate. Table 1 provides two examples of three different realizations with their resulting delays and leakage currents, which are the main contribution to the gate's power consumption.

By having the option of choosing from different gates, the designer may select fast gates (consuming high energy) for the critical path and slower gates (consuming low energy) for the non-critical paths. By offering p specific implementations per gate, i.e., a specific combination of delay and energy consumption, the design task consists of selecting a particular implementation for every gate such that the circuit's overall delay is as short as possible and that simultaneously the circuit's overall power consumption assumes a minimum. Once a particular gate implementations have been selected, state-of-the-art design tools[3] automatically determine both the circuit's entire delay and its entire power consumption.

With a total of g gates, the very same circuit can be realized in potentially $n{=}g^p$ alternatives. Previous research [9] has suggested that $p{=}3$ different implementations per gate are optimal[4]. Due to the possible interconnections between different paths, this optimization problem *can* be *NP*-hard in the general case.

For the task of finding optimal designs, previous research [7, 13, 16] has employed various algorithms, from which two serve as a baseline for comparison purposes. The first algorithm, denoted as SFA-I (straight-forward al-

[1] On the transistor as well as gate level, the term *delay* rather than processing speed is more commonly used.

[2] Different functionalities require a different number of internal elements, i.e., transistors, which influences a gate's overall delay τ.

[3] The HSpice simulator `www.synopsys.com`, for example.

[4] Unfortunately, the literature [9] does not provide any substantial indication, why $p{=}3$ is supposed to be optimal.

gorithm, variant I) for short, starts off by using the slowest implementation for all gates. It then accelerates the critical path by substituting some of them with their fastest counterparts until either this path has turned into a non-critical one or no further gates can be accelerated. This step is repeated as long as it can change a critical path into a non-critical one. Finally, all fast gates are substituted by the medium ones as long as this does not affect the circuit's overall delay. The second algorithm, denoted as SFA-II for short, starts off by selecting the slowest alternatives for all gates. It then consecutively substitutes them with medium or fast alternatives, until no further acceleration can be achieved, i.e., the minimal delay has been reached. Finally, the algorithm tests whether fast gates can be substituted by medium (or even slow) ones without increasing the circuit's overall delay. For further details, the interested reader is referred to the literature [13].

3 The Evolutionary Approach

The term *evolutionary algorithms* refers to a class of heuristic population-based search procedures that incorporate random variation and selection, and provide a framework that mainly consists of genetic algorithms, evolutionary programming, and evolution strategies [1]. Despite some peculiarities, all evolutionary algorithms maintain a population of μ individuals, also called parents. In each generation, an evolutionary algorithm generates λ offspring by copying randomly selected parents and applying variation operators, such as mutation and recombination. It then assigns a fitness value (defined by a fitness or objective function) to each offspring. Depending on their fitness, each offspring is given a specific survival probability. A canonical evolutionary algorithm works as follows:

Step 0: Initialization of the population's individuals and evaluation of the individuals' fitness

Step 1: Selection of the parents according to a preselected selection scheme (e.g., roulette wheel, linear ranking, truncation selection)

Step 2: Recombination of selected parents by exchanging parts of their genes

Step 3: Mutation of some genes by a prespecified probability

Step 4: If not termination criterion met, go to Step 1

By selecting certain individuals as parents, an evolutionary algorithm advances from one generations to another. The two most-commonly used selection schemes are denoted as either (μ,λ) or (μ+λ). The first selection scheme indicates that the algorithm chooses the parents for the next generation *only* from the offspring, whereas the latter selection scheme selects from the union of the previous parents *and* the current offspring; the latter form is also known as μ-fold elitism. For a good overview on the various variations as well as further details, the interested reader is referred to [1].

The application of evolutionary algorithm to the minimization of the circuit's leakage is straight forward. Since a particular circuit consists of n gates, it is represented by a genome consisting of n positions each of which codes for the chosen threshold voltage V_{th} (or any other design parameter, such as T_{ox}). Since this paper considers continuous values for the threshold voltage, the genome thus consists of n floating-point numbers. In other words, a circuit is represented by an array of n floating-point variables $x_{1\leq i\leq n}$. Because of their superior behavior in the presence of local optima [1], this paper focuses on genetic algorithms, which are denoted as (μ,λ)-GA or (μ+λ)-GA for short.

The mutation operator changes each position of the genome with the mutation probability p_m, which is usually set to $p_m = 1/n$ in genetic algorithms. Since the genome consists of floating-point numbers, a mutation is implemented by adding a Gaussian-distributed random number $N(0,1)$ with expectation value 0 and standard deviation 1. In summary, every variable x_i of the genome is mutation with probability $p_m = 1/n$ as follows: $x_i \leftarrow x_i + N(0,1)$.

Many genetic algorithms also apply recombination. The uniform recombination operator would then exchange the corresponding parameters x_i^k and x_i^l of two different individuals k and l with recombination probability p_r=0.5.

As a second option, this paper also considers a very similar algorithm known as simulated annealing [11] denoted as SA for short. Like genetic algorithms, simulated annealing applies a mutation to a randomly chosen gene position. In case that mutation is beneficial, it is taken. But in case that mutation is detrimental, it is taken with probability $p_{SA} = 1/(1 + \exp(\Delta P_{\text{leakage}}/T))$, with T and $t_{\max}$ denoting the current and maximal temperature, respectively. With having a temperature scheduling such as $T = T_{\max}(t_{\max} - t)/t_{\max}$, with t and $t_{\max}$ denoting the current and maximal number of generations, respectively, simulated annealing models the physics that can be observed in annealed glasses.

4 Methods

As has already been mentioned above (see, also, [9]), this paper adopts a direct encoding in which the n gates are represented by a genome consisting of n floating-point values. Each of these values codes for the gate's particularly chosen delay from which the leakage current I_{ds} can be easily determined. In accordance with the literature [4, 12] a mutation probability $p_m = 1/n$ was chosen in all experiments. For both genetic algorithms and simulated annealing, the maximal optimization time was set to $t_{\max} = 3000$.

Table 2. This table shows three examples of how a genetic algorithm improves the circuit design as compared to deterministic algorithms when using only two gate threshold voltages.

C432	Delay	Leakage
SFA-I	1045	43,699
SFA-II	1045	38,482
(1+6)-GA	1045	38,779

C1335	Delay	Leakage
SFA-I	925	79,037
SFA-II	925	73,443
(1+6)-GA	925	70,781

C3540	Delay	Leakage
SFA-I	1618	145,504
SFA-II	1618	137,857
(1+6)-GA	1618	132,012

The literature [1, 4] offers a large selection of various recombination operators for evolutionary algorithms. But since recombination could not yield any performance advantage in the present task, non of these operators is been used in this paper. Furthermore, the literature [1] suggests that μ=1 parent and λ=6 offspring yield the highest sequential efficiency.

As has been outlined above, the fitness function should incorporate both the network's delay and its energy consumption. Since the network's delay is of primary interest (by definition), the following strategy has been used: First of all, both values delay and leakage is calculated for the circuit. The, an offspring is worse if its delay is larger than that of the parent. In case the delay is unaffected or even better (recall that the circuit's delay depends on the longest path only), the leakage values decide.

For the goal of doing a comparative study, this paper has selected eight standard designs from the well-known ISCAS test suite [5]:

Name	Gates	Ins	Outs	Function
C432	160	36	7	27-channel interrupt controller
C499	202	41	32	32-Bit single-error correction
C880	383	60	26	8-bit ALU
C1355	546	32	8	32-Bit single-error correction
C2670	1193	233	140	12-bit ALU and controller
C3540	1669	50	22	8-bit ALU
C5315	2406	178	123	9-bit ALU
C7552	3512	207	108	32-bit adder/comparator

In the experiments, all transistors of a single gate had varying but identical values. Upon the specifically chosen value, the `HSPICE` simulator has calculated the gates overall delay and power consumption. The transistor models are based on on the 65 nm Berkeley predictive technology models (BPTM).

With this approach, the electrical characteristics of each employed gate has been simulated for 21 different parameter sets, i.e., T_{ox} and NDP. These parameters have been selected, since previous research (reference omitted because of blind reviewing) has indicated that they have the most prominent influence.

5 Results

Table 2 summarizes the results on three selected examples. It can be clearly seen how a genetic algorithm reduces the circuit's power comsumption over the deterministic algorithms SFA-I and SFA-II. In order to have a fair comparison, the genetic algorithm was selecting only grom two different threshold voltages. By contrast, Figure 2 shows possible further energy savings, when selecting from continuous threshold voltages. The figure Figure 2 summarizes the results when optimizing the eight test designs C432, C499, C880, C1355, C2670, C3540, C5315, and C7552 [5] by means of a (1+6)-GAs. The figure presents the achieved energy savings due to leakage in comparison to optimized DVTCMOS/DTOCMOS circuits.

It can be clearly seen that for every problem, the improvements depend on the number of clusters. The general figures are that in comparison to optimized DVTCMOS/DTOCMOS circuits, the energy savings are between about 20-45 %. It does not come to a surprise that without any clustering, i.e., columns labeled with '∞', the improvements are maximal. But it may be noted that even for a small number of clusters of about 3-5, the achievable energy savings are significant.

It was interesting to note that in all experiments, the genetic algorithms have shown a very similar results in terms of leakage like simulated annealing. The differences are marginal and thus not explicitly reported in the figures.

6 Conclusions

This paper has argued that processing speed and energy consumptions are properties, which end-users consider important, especially for mobile devices. This paper has reviewed that these two parameters depend on each other due to technological reasons. This paper has applied genetic algorithms to find circuits with minimal energy consumptions with still providing minimal delays.

It has been mentioned that both the approach and results presented in this paper are hypothetical in that current technologies do not offer to fine tune every gate. Currently, the designer can choose from two or at most three different values for the threshold voltage V_{th} or the thickness of the gate oxide T_{ox}. However, the experiments reported here have been achieved with state-of-the-art simulation tools and thus represent plausible configurations. The results suggest that for energy-critical applications, the offerings

of more than two or three options might be worth it.

Currently, the possible clusters are pre-defined. Consequently, future research will be dedicated to the development of an adaptive clustering algorithm that determines an energy-optimal clustering.

A second avenue for future research concerns the development of a problem-specific optimization algorithm. Yet, this paper resorts to a generic form of an evolutionary algorithm. It is well known and rather obvious that a general-purpose algorithm is rather universal by its very nature, and thus, yields sub-optimal performance with respect to the algorithm's runtime. It can be expected that a problem-specific algorithm will be yielding at least the same energy savings but in a much shorter time.

Acknowledgements

The authors gratefully thank Ralf Joost for fruitful discussions and many valuable comments on draft versions of this paper. Part of this research has been supported by the German Research Foundation (DFG), grant number 466.

References

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, Evolutionary Computation: Comments on the History and Current State, *IEEE Transactions on Evolutionary Computation,* **1**(1), 1997, 3-17.

[2] Y.S. Borkar, VLSI Design Challenges for Gigascale Integration, keynote address at the 18th Conference on VLSI Design, Kolkata, India, 2005.

[3] M.I. Elmasry and M. Anis, *Multi-Threshold Cmos Digital Circuits,* (Kluwer Academic Publishers, 2003).

[4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* (Reading, MA: Addison-Wesley, 1989).

[5] M. Hansen, H. Yalcin, and J. P. Hayes, Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering, *IEEE Design and Test,* **16**(16), 1999, 72-80. `http://www.eecs.umich.edu/~jhayes/iscas/`

[6] J.K. Kao and A. Chandrakasan, Dual-Threshold Voltage Techniques for Low-Power Digital Circuits, *IEEE Journal of Solid State Circuits,* **35**(7), 2000, 1009-1018.

[7] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar, Total Power Optimization by Simultaneous Dual-V_{th} Allocation and Device Sizing in High Performance Microprocessors, *Proceedings of the 39th Conference on Design Automation,* New Orleans, USA, 2002, 486-491.

[8] N.S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan, Leakage Current: Moore's Law Meets Static Power, *IEEE Computer,* **36**(12), 2003, :68-75.

[9] T. Kuroda, Low-Power, High-Speed CMOS VLSI Design, *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computer and Processors (ICCD 2002),* San Jose, USA, 2002, 310-315.

[10] P. Maken, M. Degrauwe, M. Van Paemel, and H. Oguey, A Voltage Reduction Technique for Digital Systems, *Proc. of the 37th IEEE International Solid-State Circuits Conference,* 1990, 238-239.

[11] R. Rojas, *Neural Networks: A systematic Introduction,* (Berlin: Springer-Verlag, 1996).

[12] R. Salomon, Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems,* **39**(3), 1996, 263-278.

[13] F. Sill, F. Grassert, and D. Timmermann, Reducing Leakage with Mixed-Vth (MVT), *Proceedings of 18th Conference on VLSI Design,* Kolkata, India, 2005, 874-877.

[14] A. Srivastava, D. Sylvester, and D. Blaauw, Statistical Optimization of Leakage Power Considering Process Variations using Dual-V_{th} and Sizing, *Proceedings of the 41st Design Automation Conference (DAC 2004),* San Diego, USA, 2004, 773-778.

[15] A.K. Sultania, D. Sylvester, S.S. Sapatnekar, Transistor and Pin Reordering for Gate Oxide Leakage Reduction in Dual Tox Circuits, *Proceedings of 22nd IEEE International Conference on Computer Design (ICCD 2004),* San Jose, USA, 2004, 228-233.

[16] V. Sundararajan and K. Parhi, Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits, *Proceedings of the IEEE International Symposium on Low Power Electronics and Design (ISLPED 1999),* 1999, 139-144.

[17] Y.-F. Tsai, D. Duarte, N. Vijaykrishnan, and M.J. Irwin, Characterization and Modeling of Run-Time Techniques for Leakage Power Reduction, *IEEE Trans. on VLSI Systems,* **12**(11), 2004, 1221-1233.

[18] N.H.E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective,* 3rd edition, (Reading, MA: Addison-Wesley, 2004).

[19] L. Yuan, and G. Qu, Enhanced Leakage Reduction Technique by Gate Replacement, *Proceedings of the 42nd Design Automation Conference (DAC 2005),* Anaheim, USA, 2005, 47-50.

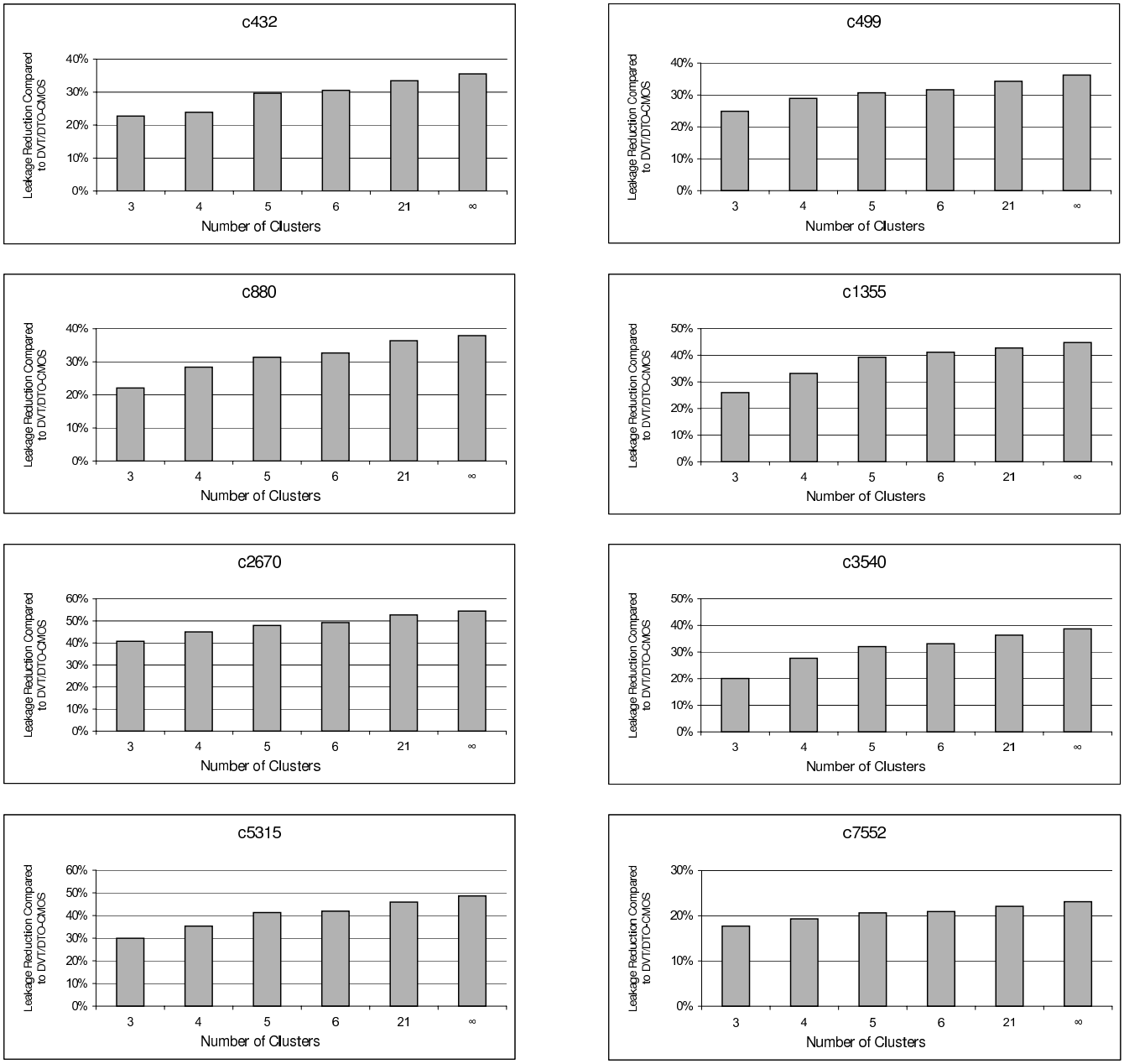

Figure 2. This figure shows the improvements in comparison to optimized DVTCMOS/DTOCMOS circuits when using (1+6)-GAs to the eight test problems C432, C499, C880, C1355, C2670, C3540, C5315, and C7552 taken from the ISCAS test suite [5].