Biologically-Inspired Optimization of Circuit Performance and Leakage: A Comparative Study

Ralf Salomon and Frank Sill

Faculty of Computer Science and Electrical Engineering,
University of Rostock, 18051 Rostock, Germany
{ralf.salomon, frank.sill}@uni-rostock.de

Abstract. State-of-the-art technologies in very large scale integration (VLSI) allow for the realization of gates with varying energy consumptions and hence delays (i.e., processing speeds) in the very same circuit. By considering this technological advent as an option, the design process can pursue two different goals: (1) making the circuit as fast as possible and (2) making non-time-critical gates slower in order minimize the circuit's overall energy consumption. This paper utilizes evolutionary algorithms, a population-based heuristic optimization technique, in order to find optimal solutions. From a technological point of view, this goal can be accomplished by varying the individual threshold voltages, which determine both the device's processing speed and its *leakage currents*. The experimental results indicate that evolutionary algorithms yield significantly better solutions than rather traditional optimization algorithms. By maintaining populations of candidate solutions, evolutionary algorithms are able to escape from sub-optimal designs, which contrasts traditional single-point optimization approaches.

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.

The issue of a suitable power supply, e.g., by means of rechargeable batteries, becomes even more important in *small* mobile devices, such as cellular phones and personal digital assistants. For example, it would probably be unacceptable for most end-users, if the battery was larger and/or heavier than the actual cellular phone. High processing speed and long time-of-operation are probably *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 by their very nature.

W. Grass et al. (Eds.): ARCS 2006, LNCS 3894, pp. 352–366, 2006.

Among other aspects, current research on low-power [3, 9, 10, 18, 19] tries to minimize a circuit's energy consumption without tampering its processing speed. Normally, a circuit consists of very many interconnected gates. But as Section 2 argues, not all of these gates are equally responsible for the circuit's processing speed. Based on this observation, previous research has proposed to *simultaneously* use both fast high-energy-consuming and slow low-energy-consuming gates in the very same circuit. For this approach, the term dual-threshold CMOS (DTCMOS) has been coined.

It is obvious that all gates within the critical path must be implemented in fast high-energy-consuming technology, in order to obtain maximal processing speed. For all other gates, however, it remains to be determined, which technology is to be used in order to reach both *optimization goals*: fastest processing speed by paying minimal energy consumption.

Previous research [18, 19, 22, 24] has already reported on some encouraging results when applying special-purpose algorithms to the present optimization problem. This is also discussed in Section 2. However, a comparison with human-optimized designs indicates that these algorithms yield good but only sub-optimal solutions. Apparently, these algorithms got stuck at sub-optimal solutions, also known as diverting local optima in the pertinent literature on optimization.

Since the optimization procedures mentioned above do not reliably yield optimal solutions, this paper applies evolutionary algorithms to the problem at hand. Evolutionary algorithms are heuristic population-based search procedures that incorporate random variation and selection. This paper focuses on the application of evolutionary algorithms to the optimization problem at hand, since both numerous experiments and theoretical analyses [1, 6, 17] stress their superior global optimization performance, especially in the presence of unwanted local optima. Therefore, Section 3 presents a short description of this class of algorithms.

In order to allow for an evaluation, this paper applies selected evolutionary algorithms to some rather standard design problems, which are drawn from the ISCAS benchmark suite [7]. Section 4 provides a short description of these tasks, and also summarizes all the relevant parameter settings. The results presented in Section 5 suggest that the selected algorithms evolve designs better than previously reported. Finally, Section 6 concludes with a brief discussion.

2 The Circuit Model and Previous Research

The introduction has already indicated that a device's processing speed f as well as its energy consumption P_{total} are tightly coupled. From a technological point of view, this relation can be approximated by the sum of a static and a dynamic term P_{static} and P_{dynamic}, respectively:

$$P_{\text{total}} = P_{\text{static}} + P_{\text{dynamic}}$$

with

$$P_{\text{static}} \approx P_{leakage} \approx V_{DD} I_{DS} \quad \text{and} \quad P_{\text{dynamic}} \sim \alpha C_{\text{load}} f V_{DD}^2 \ , \tag{1}$$

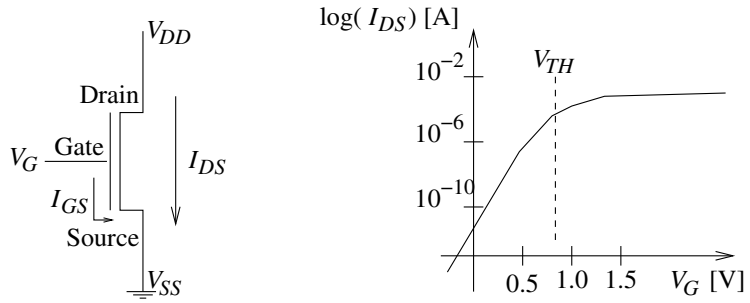

Fig. 1. A CMOS transistor with the main voltages and currents as well as the drain-to-source current $I_{DS} = f(V_G)$ as a function of the gate voltage V_G for a constant power supply V_{DD}. The voltage V_{TH} is called the threshold voltage.

with C_{load} denoting the sum of the device's dynamic capacities (e.g., the gate capacities), V_{DD} denoting the device's power supply, I_{DS} denoting the drain-to-source (static) current (see below), and P_{leakage} denoting the static leakage energy consumption. The dynamic term, as equation (1) indicates, is proportional to the square of the power supply V_{DD}. Research on low-power has consequently reduced V_{DD} from $5V$ to approximately $1.4V$ in recent years. In order to make the CMOS devices still function properly, the gate threshold-voltage V_{TH} has similarly been reduced from approximately $0.7V$ to $0.4V$ [9, 25].

Figure 1 illustrates a CMOS transistor with some of its major voltages and currents. The figure also illustrates the drain-to-source current $I_{DS} = f(V_G)$ as a function of the gate voltage V_G for a constant power supply V_{DD}. It should be noted that the y-axis is in logarithmic scale, thus expressing that the drain-to-source current $I_{DS} \sim \exp(V_G)$ grows exponentialy with the gate voltage V_G. The term V_{TH} is called threshold (gate) voltage, and the regime $V_G < V_{TH}$ is called *sub-threshold*. In current VLSI technologies, the drain-to-source sub-threshold currents $I_{DS} \gg I_{GS}$ are much larger than the gate-to-source sub-threshold currents I_{GS} and thus dominate the static energy consumption P_{leakage}.

Unfortunately, the reduction of the gate threshold voltage to $V_{TH} \approx 0.4$ has led to a left-shift of the $I_{DS}(V_G)$ curve, and thus to a significant increase of the sub-threshold drain-to-source current. By contrast, technological advances in the development of new insulators, prevent other leakage currents, such as the gate-to-drain currents I_{GD}, from further increases. Currently available low-power technologies indicate that the static energy consumption P_{leakage} is and will be dominated by the sub-threshold drain-to-source leakage current I_{DS}.

The particularly chosen value of the threshold voltage V_{TH} not only influences the sub-threshold drain-to-source leakage current I_{DS} but also the transistor's delay[1] and thus the device's processing speed: a higher threshold voltage V_{TH}, with a lower sub-threshold drain-to-source leakage current I_{DS} associated to it, requires more time to charge and discharge the transistor's capacitor. Hence,

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

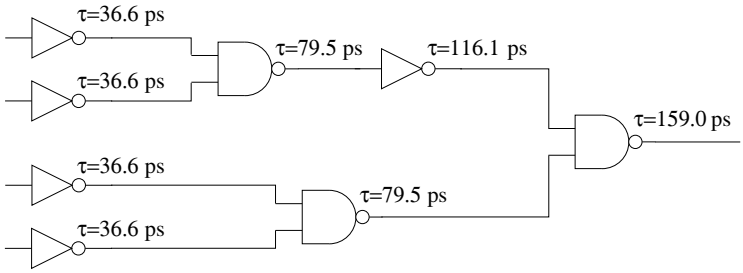

Fig. 2. A simple CMOS-circuit with different path delays, caused by different numbers of gates in each path. The lower path is non-critical, and thus may be subject to the implementation in slow high-voltage technology.

saving energy consumption by increasing the threshold voltage also increases a transistor's delay.

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 or a microprocessor. It used to be that throughout the entire VLSI circuit, the very same threshold voltage V_{TH} was used for all transistors. But the simple example presented in Figure 2 allows for the observation that not all gates are equally important for the circuit's overall delay: 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 can be processing their signals slower without affecting the circuit's overall delay to some extent.

Based on the observation discussed above, dual-threshold CMOS (DTCMOS) design techniques [9, 10, 22, 24] have proposed to employ both slow high-threshold and fast low-threshold gate types in the very same circuit. These two gate types are obviously realized by high-threshold and low-threshold transistors, respectively. In addition, previous research [3, 18, 19, 23] has also generalized this idea by allowing a variable number of fast and slow transistors, thus providing a rather fine-grained differentiation of the gate's delay and its energy consumption. Table 1 provides three examples of three different realizations with their resulting delays and leakage currents. For further details, the interested reader is referred to [18, 19].

Table 1. This table shows three different realizations with their resulting delays and leakage currents for `NOR-2, NAND-2,` and an inverter

NOR-2		NAND-2		INV	
66.3 ps	86.0 nA	42.9 ps	135.0 nA	36.6 ps	92.8 nA
78.0 ps	36.6 nA	51.3 ps	46.5 nA	37.6 ps	62.5 nA
90.0 ps	10.6 nA	58.3 ps	20.3 nA	45.8 ps	12.6 nA

By offering p specific implementations, i.e., delay and energy consumption, per gate and a total of g gates, the very same circuit can be realized in potentially $n = g^p$ alternatives. Previous research [11] has suggested that p=3 different implementations per gate are optimal; unfortunately, no substantial indication was provided, why this choice is supposed to be optimal. One possible reason might be that the number of possible realizations grows exponentially in the number p implementations per gate.

For the task of finding optimal designs, previous research [10, 18, 19, 22] has employed various algorithms, from which two serve as a baseline for comparison purposes. The first algorithm [18], denoted as SFA-I (straight-forward algorithm, 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 [19], denoted as SFA-II for short, is a modification of a previous development [12, 21]. It starts off by selecting the fastest alternatives for all gates. It then consecutively substitutes them with medium or slow alternatives by preferring gates with a high fan-out. The step is repeated until no further gate can be slowed down without affecting the circuit's overall delay. For further details, the interested reader is referred to the literature [18, 19].

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 [6], evolutionary programming [4, 5], and evolution strategies [15, 17].

Even though all evolutionary algorithm have their own peculiarities, they share many common features. 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. The canonical form of an evolutionary algorithm can be "formally" described 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 pre-specified probability
Step 4: If not termination criterion met, go to Step 1

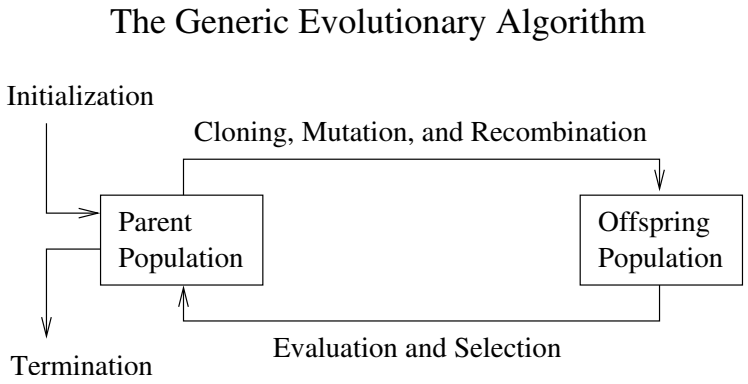

Fig. 3. A graphical visualization of an evolutionary algorithm in its canonical form

Figure 3 presents a graphical visualization of an evolutionary algorithm, and for a good overview as well as further details, the interested reader is referred to [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 $(\mu+\lambda)$. The first selection scheme, i.e., (μ,λ) 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.

As has been discussed above and exemplified in Table 1, all gates can be configured with three different leakage currents[2]. Therefore, the optimization problem at hand is *discrete* by its very nature for which the traditional form of genetic algorithms is particularly suited. In the experimental comparisons, these algorithms are denoted as $(\mu+\lambda)$-GA or (μ,λ)-GA for short. The other evolutionary algorithm variants, particularly evolutionary programming and evolution strategies, are *rather* tailored to continuous parameter optimization and are thus not further considered in this paper.

4 Methods

The first task of almost any optimization procedure is to find a proper machine coding, also called genome or genotype, for the problem at hand. Here, this paper adopts a direct coding in which every gate is represented by a particular allele, which codes for the particularly chosen leakage current I_{DS}. Thus, a device that consists of n gates is represented by a genome of n positions with each being able to assume three different values (see above and also [11]).

Since each gate can choose its leakage current only from three different values, the implementation of an appropriate mutation operator is straight forward: it

[2] It should be noted that the actual values of the leakage currents are not equivalent for all gates, but depend on their number of inputs, functionality, and other parameters.

chooses the next lower or higher value. In accordance with the literature [6, 16] a mutation probability $p_m = 1/n$ with n denoting the number of gates was chosen in all experiments.

The literature [1, 6, 15] 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 [15] 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 fitness function has been used:

$$f = \text{delay} - \frac{1}{\text{leakage}} \ . \tag{2}$$

For the goal of doing a comparative study, this paper has selected the following five standard designs (for further details, see [7]):

C432 is a 27-channel interrupt controller [26] with a total of 36 inputs and 7 outputs. The controller has 27 interrupt request inputs, which are grouped into 3 buses with 9 lines each. It has further 9 control inputs, which activates/ de-activates the associated interrupt lines. The implementation of such an interrupt controller, requires 160 gates.

C1355 is a 32-Bit single-error correcting circuit [27]. By utilizing a (40,32) Hamming code matrix, it generates a 8-bit long syndrom by reading the 32 input lines. The 41 input lines are forwarded along with the 8-bit syndrom to a correction unit. The implementation of this device requires 546 gates.

C3540 is a 8-bit arithmetic-logical-unit (ALU) [28] with 50 inputs and 22 outputs. It realizes various arithmetic, logical, BCD, shift, and other operations on 8 input lines, and its implementation requires 1669 gates.

C5315 is an extension of the `C3540`-circuit [29], in that it realizes a 9-bit ALU with 178 inputs and 123 outputs, which requires 2406 gates for its implementation.

C7552 is a device [30] that contains a 34-bit adder, a 34-bit comparator, which requires an additional 34-bit adder, and an 34-bit parity checker. The circuit requires 3512 gates to map the 207 inputs onto 108 outputs.

For the realization, this paper used a previously developed gate library [20], which is based on the 65 nm Berkeley predictive technology models (BPTM).

5 Results

Direct performance comparisons are not straight forward for the following two reasons: first, two quality measures are simultaneously subject to the optimization process, and second, the optimization procedures considered in this paper operate on different time scales. Therefore, this section starts off with a detailed

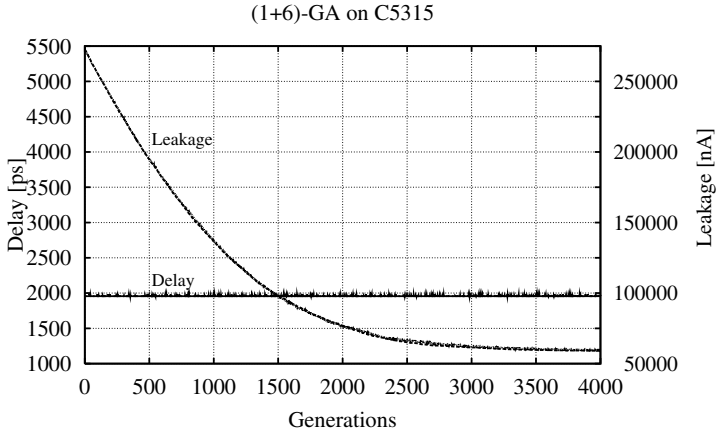

Fig. 4. The evolution of both the circuit's delay (y-axis on the left-hand-side) and leakage (y-axis on the right-hand-side) when using (1†6)-genetic algorithms for the C5315 problem

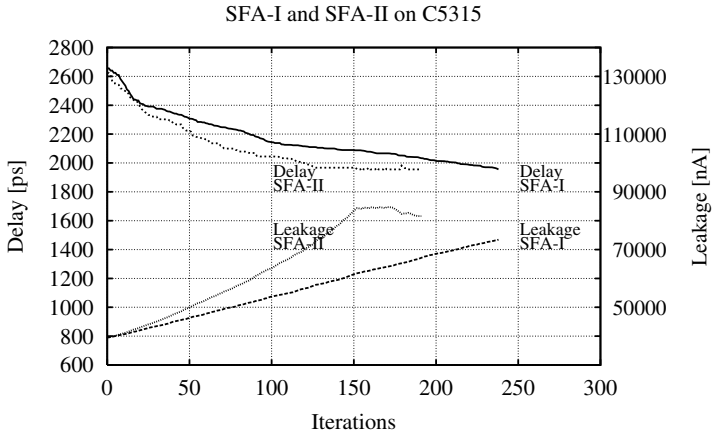

Fig. 5. The evolution of both the circuit's delay (y-axis on the left-hand-side) and leakage (y-axis on the right-hand-side) when applying a straight-forward optimization algorithm on the C5315 problem

discussion of Figures 4-7, which show various performance figures obtained on the ALU-design problem C5315.

Figure 4 shows the evolution of the both the delay and leakage when using both a (1+6)-GA and a (1,6)-GA. Since the genetic algorithms initialize all gates with the fastest realizations, the delay starts at 1955 ps and a (total) leakage of 272,600 nA. During the corse of evolution, then, the leakage drops to almost a fifth of that value, i.e., about 58,597 nA, without increasing the circuit's delay as requested. Since the performance graphs of both procedure are virtually

identical, the remainder of this section focuses on the (1+6)-GA and does not consider the (1,6)-GA any further.

For comparison purposes, Figure 5 presents the development of both delay and leakage when using the procedures SFA-I and SFA-II previously developed [18, 19]. It can be seen that both procedures start off with a relatively large delay of about 2656 ps, but arrive at the same final value of 1955 ps after about 150 to 250 iterations. In order to attain this improved processing, both procedures have increased the leakage to about 73,443 nA and 81,580 nA, respectively.

For a better comparison of the two parameters under optimization, Figures 6 and 7 combine those graphs into two figures. To this end, the time scale, i.e., x-axis, has been rescaled to a 100 time units. It can be clearly seen that the circuit's final delay arrive at the same values (Figure 6), whereas the genetic

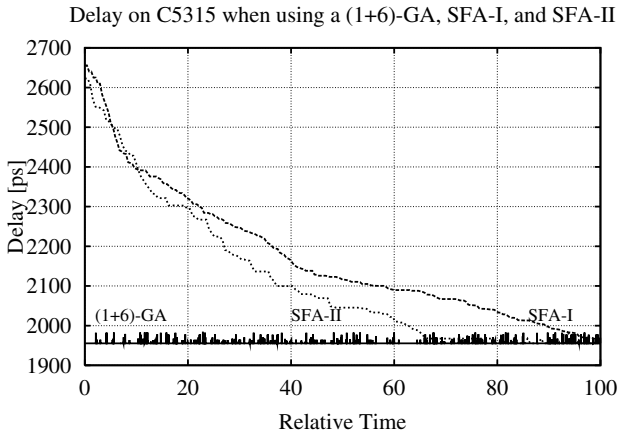

Fig. 6. The evolution of the delay for all algorithms

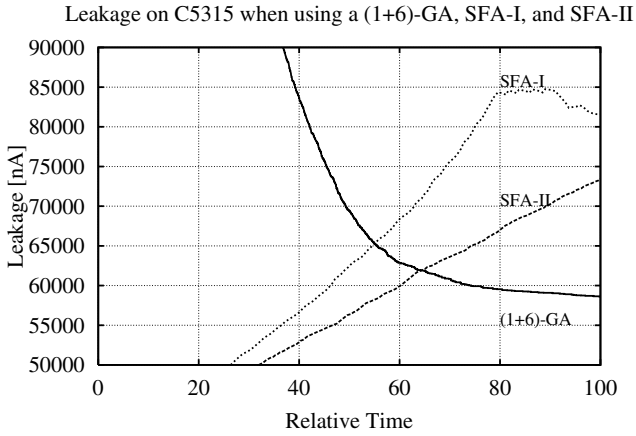

Fig. 7. The evolution of the leakage for all algorithms

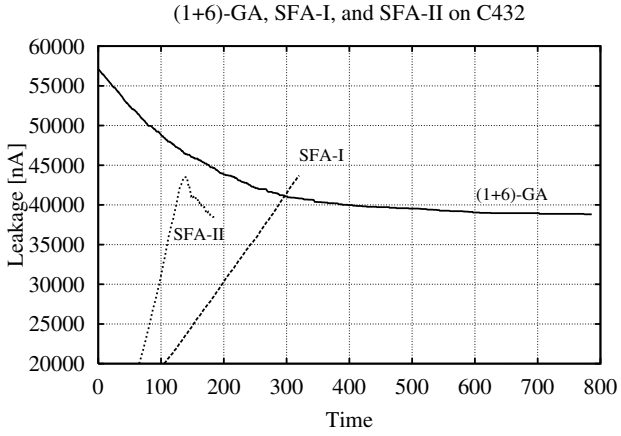

Fig. 8. The evolution of the leakage when applying a (1+6)-GA, SFA-I, and SFA-II to the C432 problem

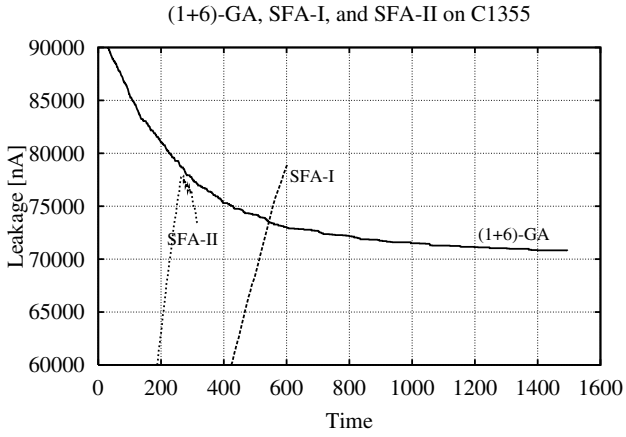

Fig. 9. The evolution of the leakage when applying a (1+6)-GA, SFA-I, and SFA-II to the C1355 problem

algorithms were able to improve the leakage by about 20-30% (Figure 7). This, however, came at the cost of a significant increase in the computational requirements. With respect to the end-users expectations on the time-of-operation, this additional optimization effort might by worth it, especially since this has to be done only once during the circuit's design process.

Figures 8 to 11 show how the optimization procedures under consideration evolve the leakage over time for the other four design problems C432, C1355, C3540, and C7552, respectively. As for the C5315 problem discussed first, all procedures exhibit a similar behavior. Furthermore, all procedures arrive at the same final delay (not shown in any figure) for each problem.

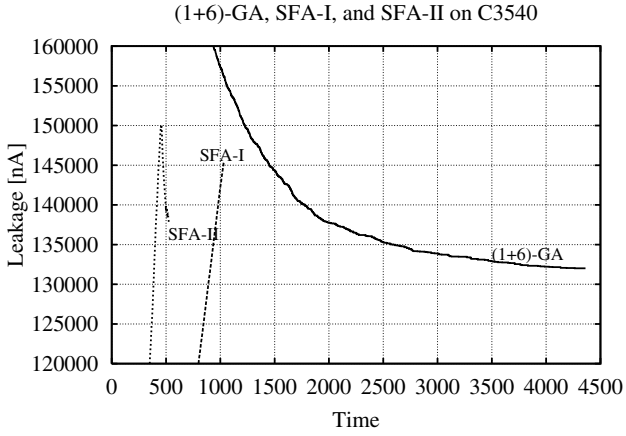

Fig. 10. The evolution of the leakage when applying a (1+6)-GA, SFA-I, and SFA-II to the C3540 problem

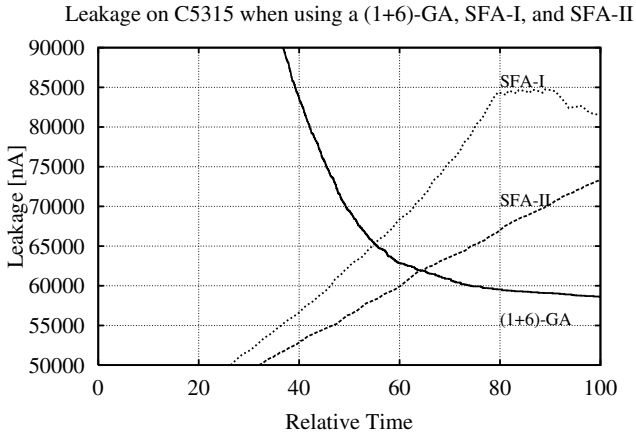

Fig. 11. The evolution of the leakage when applying a (1+6)-GA, SFA-I, and SFA-II to the C5315 problem

The performance graphs may be summarized as follows: In comparison to SFA-I, SFA-II constitutes a significant improvement in that it requires shorter optimization time and often yields lower leakage values. SFA-II increases the leakage by substituting slow high-voltage gates by their fastest counter parts, until the circuit has the shortest delay possible. It then reduces the resulting leakage by also considering medium-voltage gates.

The genetic algorithms by contrast, yielded the lowest overall leakage and thus energy consumption values, but required substantially more time. This observation goes in-line with the pertinent literature [15]: evolutionary algorithms are a general framework, which might be slower than special-purpose procedures

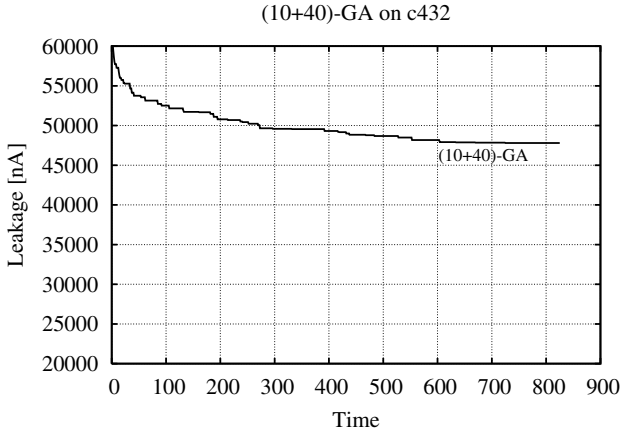

Fig. 12. The evolution of the leakage when applying a (10+40)-GA to the C432 problem

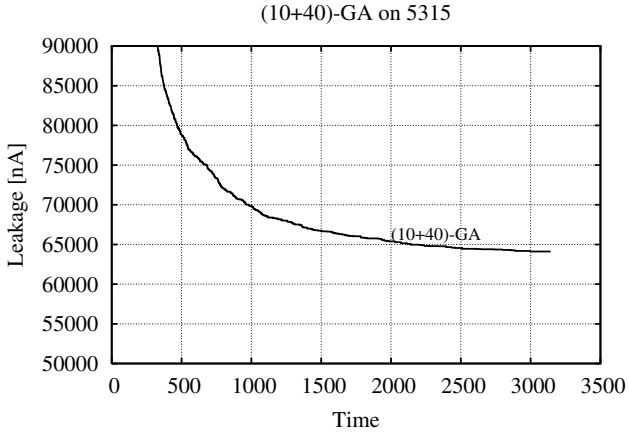

Fig. 13. The evolution of the leakage when applying a (10+40)-GA to the C5315 problem

in many cases but have the ability to escape from local optima, and are thus often able to yield superior results. For comparison purposes, Table 2 presents the final values for delay and leakage for all algorithms over all problems considered in this paper.

In order to assess the utility of large population sizes, Figures 12 and 13 illustrate the application of a (10+40)-GA to the C432 and C5315 problems, respectively. A comparison with Figures 8 and 11, respectively, indicates that a (10+40)-GA might be faster in terms of the number of generations but significantly slower in terms of the number of functions evaluations, which is the product of the number of generations and the number of offspring λ.

Table 2. Comparison of the final delay and leakage values over all procedures and all problems

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

C5315	Delay	Leakage
SFA-I	1955	73,443
SFA-II	1955	81,580
(1+6)-GA	1955	58,597

C7552	Delay	Leakage
SFA-I	1198	180,714
SFA-II	1198	195,898
(1+6)-GA	1198	169,948

6 Conclusions

This paper has argued that processing speed and energy consumptions are properties, which end-users consider important for mobile devices. It has been discussed that these two parameters depend on each other due to technological reasons. This paper has furthermore reviewed two optimization procedures, which have been investigated in previous research. Since previous research has led to optimized designs, which are inferior to devices designed by humans, this paper has applied genetic algorithms to this optimization problem. The experimental results indicate that genetic algorithms were able to reduce the leakage by about 10-40% as compared to previously optimized designs. The results also indicate, however, that genetic algorithms require substantially more computation time.

Future research will be dedicated to the investigate of further optimization approaches, such as simulated annealing and other evolutionary algorithm variants, Furthermore, future research will be investigating to what extent an increase of the number p of different gate implementations can benificial for the overall energy consumption.

Acknowledgements

The authors gratefully thank Ralf Joost for fruitful discussions and many valuable comments on draft versions of this paper. The authors also thank Prof. Timmermann for his continuous support. Part of this research was 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):3-17, 1997.
2. Y.S. Borkar. VLSI Design Challenges for Gigascale Integration, keynote address at the 18th Conference on VLSI Design, Kolkata, India, 2005.
3. W. Chen, W. Hang, P. Kudva, G.D. Gristede, S. Kosonocky, and R.V. Joshi. Mixed Multi-Threshold Differential Cascode Voltage Switch (MT-DCVS) Circuit Styles and Strategies for Low Power VLSI Design, in E. Macii, V. De, and M.J. Irwin (Eds.) *Proceedings of the 2001 International Symposium on Low Power Electronics and Design (ISLPED'01),* pp. 263-266, 2001.
4. Fogel, L.J., 1962. Autonomous Automata. *Industrial Research,* **4**:14-19.
5. D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence.*. IEEE Press, NJ, 1995.
6. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.
7. M. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering, in *IEEE Design and Test,* **16**(16):72-80, 1999.
8. J.K. Kao, A. Chandrakasan, and D. Antoniadis. Transistor sizing issues and tool for multi-threshold CMOS technology, in *Proceedings of the 34th Conference on Design Automation (DAC),* pp. 409-414, 1997.
9. J.K. Kao and A. Chandrakasan. Dual-Threshold Voltage Techniques for Low-Power Digital Circuits, *IEEE Journal of Solid State Circuits,* **35**(7):1009-1018, 2000.
10. T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar, Total Power Optimization by Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessors, in *Proceedings of the 39th Conference on Design Automation,* pp. 486-491, 2002.
11. T. Kuroda. Low-Power, High-Speed CMOS VLSI Design. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computer and Processors (ICCD 2002),* pp. 310-315, 2002.
12. M. Liu, W.S. Wang, and M. Orshansky. Leakage Power Reduction by Dual-Vth Designs under Probabilistic Analysis of V_{th} Variation, in R.V. Joshi, K. Choi, V. Tiwari, and K. Roy (Eds.), *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED 2004),* pp. 2-7, 2004.
13. D.G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Menlo Park, CA, 1984.
14. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1987.
15. I. Rechenberg, *Evolutionsstrategie* (Frommann-Holzboog, Stuttgart, 1994).
16. 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):263-278, 1996.
17. H.-P. Schwefel. *Evolution and Optimum Seeking.* John Wiley and Sons, NY. 1995.
18. F. Sill, F. Grassert, and D. Timmermann. Low Power Gate-level Design with Mixed-Vth (MVT) Techniques, in *Proceedings of the 17th Symposium on Integrated Circuits and Systems (SBCCI),* 2004.
19. F. Sill, F. Grassert, and D. Timmermann. Reducing Leakage with Mixed-Vth (MVT), in *Proceedings of 18th Conference on VLSI Design,* pp. 874-877, 2005.

20. F. Sill, F. Grassert, and D. Timmermann. Total Leakage Power Optimization with Improved Mixed Gates, in *Proceedings of the 18th Symposium on Integrated Circuits and Systems Design (SBCCI 2005),* 2005.
21. A. Srivastava, D. Sylvester, and D. Blaauw. Statistical Optimization of Leakage Power Considering Process Variations using Dual-V_{th} and Sizing, in S. Malik, L. Fix, and A.B. Kahng (Eds.), *Proceedings of the 41st Design Automation Conference (DAC 2004),* pp. 773-778, 2004.
22. V. Sundararajan and K. Parhi. Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits, in *Proceedings of the IEEE International Symposium on Low Power Electronics and Design,* pp. 139-144, 1999.
23. L. Wei, Z. Chen, and K. Roy. Mixed-vth (MVT) CMOS Circuit Design Methodology for Low Power Applications. in *Proceedings of the 36th Design Automation Conference,* pp. 430-435, 1999.
24. L. Wei, K. Roy, and C. Koh. Power Minimization by Simultaneous Dual-Vth Assignment and Gate-sizing, in *Proceedings of the IEEE Custom Integrated Circuits Conference,* pp. 413-416, 2000.
25. N.H.E. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective,* 3rd edition. Addison-Wesley. 2004.
26. `http://www.eecs.umich.edu/~jhayes/iscas/c432.html`
27. `http://www.eecs.umich.edu/~jhayes/iscas/c499.html`
28. `http://www.eecs.umich.edu/~jhayes/iscas/c3540/c3540.html`
29. `http://www.eecs.umich.edu/~jhayes/iscas/c5315/c5315.html`
30. `http://www.eecs.umich.edu/~jhayes/iscas/c7552/c7552.html`