A Nature-Inspired Firefly Algorithm Based Approach for Nanoscale Leakage Optimal RTL Structure

Elias Kougianos[a,b,1], Saraju P. Mohanty[a,c,1,*]

[a] *NanoSystem Design Laboratory (NSDL), University of North Texas, Denton, TX 76207, USA.*
[b] *Department of Electrical Engineering Technology, University of North Texas, Denton, TX 76207, USA.*
[c] *Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, USA.*

Abstract

Optimization of leakage power is essential for nanoscale CMOS (nano-CMOS) technology based integrated circuits for numerous reasons, including improving battery life of the system in which they are used as well as enhancing reliability. Leakage optimization at an early stage of the design cycle such as the register-transfer level (RTL) or architectural level provides more degrees of freedom to design engineers and ensures that the design is optimized at higher levels before proceeding to the next and more detailed phases of the design cycle. In this paper, an RTL optimization approach is presented that targets leakage-power optimization while performing simultaneous scheduling, allocation and binding. The optimization approach uses a nature-inspired firefly algorithm so that large digital integrated circuits can be effectively handled without convergence issues. The firefly algorithm optimizes the cost of leakage delay product (LDP) under various resource constraints. As a specific example, gate-oxide leakage is optimized using a 45nm CMOS dual-oxide based pre-characterized datapath library. Experimental results over various architectural level benchmark integrated circuits show that average leakage optimization of 90% can be obtained. For a comparative perspective, an integer linear programming (ILP) based algorithm is also presented and it is observed that the firefly algorithm is as accurate as ILP while converging much faster. To the best of the authors' knowledge, this is the first ever paper that applies firefly based algorithms for RTL optimization.

Keywords: Firefly Algorithm, Nano-CMOS, Leakage Power, Gate Leakage, Low Power Synthesis, Scheduling, Binding, RTL Optimization, Integer-Linear Programming

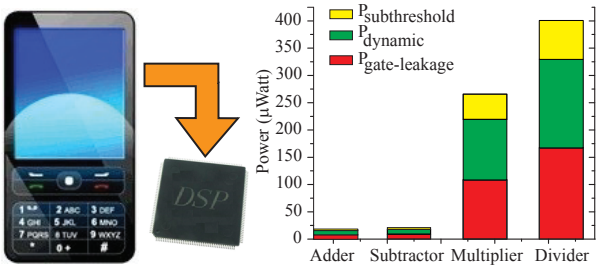
1. Introduction

Scaling of CMOS technology has been taking place at a very fast pace in the last several years. Digital, analog as well as mixed-signal integrated circuits have been built using 65nm, 45nm, and even 22nm technologies [1, 2]. Specifically, digital integrated circuits are made of aggressively scaled CMOS technology nodes with very high transistor count. For example, microprocessors for desktop PCs from Intel® are made of billions of transistors at the 22nm technology node. At the same time the graphic processor or graphics processing unit (GPU) from NVIDIA® has 7 billion transistors. A specific field-programmable gate-array (FPGA) chip from Xilinx® contains more than 20 billion transistors. The ARM® based analog/mixed-signal system-on-chip (AMS-SoC) which powers smart mobile phones as well as tablets has its own transistor count trend as well as a heterogeneous architecture containing hardware and

*Corresponding author
Email addresses: elias.kougianos@unt.edu (Elias Kougianos), saraju.mohanty@unt.edu (Saraju P. Mohanty)
[1] http://nsdl.cse.unt.edu

software on the same die. Power dissipation of the circuits and systems is an important axis in the design space exploration for nanoscale CMOS design. In short channel CMOS technologies the leakage power as well as the dynamic power are significant [3, 4].

The heavy demand for portable communication and computing services has promoted both industrial and academic research. Due to device scaling and the consequent increase in leakage components in both active and sleep states, the power dissipation of CMOS circuits remains a major concern as depicted in Fig. 1. In a typical nano-CMOS digital IC, power has diverse forms such as subthreshold leakage, gate-oxide leakage, reverse biased diode leakage, gate induced drain leakage (GIDL), and reverse-biased drain-substrate and source-substrate junction band-to-band tunneling [5, 6, 3, 7]. Each leakage component has diverse origins and flows between different terminals and in various operating conditions of a nanoscale MOS transistor. In this paper, RTL optimization of gate-oxide leakage is proposed to reduce the leakage of nano-CMOS chips consisting of short channel devices with ultra-thin oxide thickness in both their ON and OFF states. However, subthreshold leakage optimization can be also considered.



(a) Digital Signal Processor in a SoC (b) Power Components in Digital Circuits

Figure 1: Digital ICs and the power components for a typical nano-CMOS technology [5, 6, 3, 7].

In digital integrated circuits and systems, leakage optimization can be performed at various levels of abstraction ranging from system level to physical level. However, the current paper proposes to optimize leakage at the architectural level as the possibility of reduction is higher and circuit complexity is lower which provides faster convergence with more degrees of freedom [5, 8]. High-level synthesis (HLS), architecture-level synthesis, or RTL synthesis is an automatic design process which allows exploration of design alternatives prior to layout. In order to provide designers with an efficient automated path to silicon (i.e., from system to silicon), leakage reduction RTL frameworks are required. At the architecture level, there are balanced degrees of freedom to tune the design parameters for fast and correct design decisions at an early phase of the design cycle without propagating design errors to lower levels of circuit abstractions, where they are costly to correct. Moreover, designing at higher levels of abstraction is an efficient way to handle complexity, facilitate design verification, and increase design reuse. Hence, the current paper focuses on leakage optimization during RTL synthesis.

The rest of the current paper is organized in the following manner. The notation used in describing the algorithm is presented in Table 1. The contributions to the state-of-the-art along with discussion of prior related research is presented in Section 2. An overview of the proposed leakage-optimal RTL design flow is depicted in Section 3. A technology-based RTL optimization technique called dual oxide approach is discussed in Section 4. The detailed algorithm that uses dual-oxide technique during RTL for leakage optimization is presented in Section 5. A nano-CMOS based datapath component library is presented in Section 6 along with discussion on its characterization. The experiments performed on various benchmark integrated circuits for different constraints are discussed in Section 7. The paper is concluded in Section 8 with a summary and directions for future research.

2. Contributions of this Paper to Advancing the State-of-the-Art

Research on optimization during HLS or RTL optimization is of major interest for the academic and industrial communities [5, 6]. Genetic algorithms have been used in various evolutionary approaches for power, delay, or area optimization during high-level synthesis [9, 10, 11]. Particle swarm optimization (PSO) has been explored for power and performance trade-offs during architecture-level synthesis [12]. The widely used multiple supply voltage based RTL optimization techniques use higher supply voltage for critical path components and lower supply voltage for non-critical path components [13, 14, 15]. Many mature solutions are available in the current literature for their use for dynamic power reduction. The use of multiple threshold voltage technique for subthreshold leakage reduction during HLS is also well-researched [8, 16, 17, 18, 19]. The use of high-threshold voltage transistors in non-critical paths and low-threshold transistors in the critical paths serves as subthreshold leakage and delay trade-off in these approaches. Selective shutdown of different portions of the integrated circuit or system using power gating mechanisms has been explored for aggressive reduction of subthreshold leakage [20, 21]. Gate-oxide leakage,

Table 1: Notation used in this paper.

FF_i	Denotes a firefly
N_F	Total number of fireflies in a population
S_i	Denotes location of firefly FF_i
N_S	Denotes dimension of the location S_i
I_{ϕ_i}	Luminous (or light) intensity of firefly FF_i
d_{ij}	Distance between firefly FF_i and firefly FF_j
β_{attract}	Attractiveness between fireflies
γ_{light}	Fixed light absorption coefficient
η	Coefficient of randomization
Θ_i	Vector of random numbers
$G(V, E)$	Sequencing data flow graph (DFG)
$V\{v_i\}$	Set of nodes or vertices in a DFG
E	Set of edges in the DFG
U	Unscheduled DFG
S	Scheduled DFG with resource binding
N_c	Number of clock cycles or control steps
t	Technology corner in dual-T_{ox} or dual-κ
$R_{k,t}$	Resource of type k made using corner t
$M_{k,t}$	Maximum number of $R_{k,t}$ resources
N_{R_c}	Number of resources active in a control step c
T_{ox_H}	Higher gate oxide thickness
T_{ox_L}	Lower gate oxide thickness
κ_H	Higher dielectric
κ_L	Lower dielectric
R_{Avl}	Availability matrix for each c of k corner t
AS_i	As soon as possible time stamp of vertex v_i
AL_i	As late as possible time stamp of vertex v_i
$C[v_i]$	Final time stamp of vertex v_i
$X_{i,t,c}$	Decision variable; 1 if v_i uses $R_{k,t}$ and in a c
$L_{i,t}$	Latency in number of cycles for v_i using $R_{k,t}$
$P_{ox_{ST}}$	Leakage power for single oxide thickness
$P_{ox_{DT}}$	Leakage power for dual oxide thickness
$T_{pd_{ST}}$	Critical path delay for single oxide thickness
$T_{pd_{DT}}$	Critical path delay for dual oxide thickness
$T_{pd_{CON}}$	Target delay or delay constraint of the DFG
$Delay_{CON}$	Performance or delay trade-off factor
LDP	Leakage Delay Product
A_{ST}	Total Area for single oxide thickness
A_{DT}	Total Area for dual oxide thickness

which is prominent for technology nodes below 65nm, has been addressed by several researchers [22, 23, 24].

In the current paper, to advance the state-of-the-art in RTL leakage optimization, an RTL design optimization flow for gate leakage is presented that uses a specific nature-inspired algorithm called "firefly algorithm" as the core. The proposed approach performs simultaneous scheduling, allocation and binding and optimizes gate leakage of datapath circuits for specified time and resource constraints. Dual-oxide (using low-T_{ox} and high-T_{ox}) and dual dielectric (or dual-κ using low-κ and high-κ) technologies are explored for total gate leakage, propagation delay, and circuit area trade-offs. In sub-65 nm CMOS technology nodes the oxide thickness is quite thin, approximately 1.2 nm. However, a single layer of SiO_2 is approximately 0.2 nm. Thus, the misplacement of one SiO_2 layer can lead to a significant variations in the effective T_{ox}, and corresponding gate leakage and delay. Thus process variation effects need to be considered during estimation of the leakage and delay. The proposed RTL flow takes process variations into consideration while optimizing gate leakage. This paper presents analytical models for the estimation of gate leakage, propagation delay, and area of nano-CMOS based architectural units which are used in the RTL flow. Extensive experiments are performed over real-life digital signal processing (DSP) circuits whose applications are widespread, from media players to mobile phones. The current archival journal paper is based on preliminary research presented as a conference publication in [25]. In the conference publication a simulated annealing algorithm based RTL optimization is presented. The firefly algorithm based RTL optimization presented in the current journal paper further advances the state-of-art with 51.4 % convergence time improvement as evident from the experimental results in Section 7.

3. Proposed Methodology for RTL Leakage Minimization

This section presents the overall methodology for the gate-leakage optimization at register-transfer level that uses a firefly based algorithm for design exploration. The section is divided into 3 subsections to improve readability.

3.1. Proposed RTL Overall Methodology

Typical steps of high-level synthesis include compilation, transformation, datapath scheduling, functional unit allocation, operation binding, connection allocation and architecture generation. A broad overview of the proposed RTL leakage methodology that shows selected steps of HLS and uses the firefly algorithm as the core is depicted in Fig. 2. In this methodology, during the compilation phase the behavioral Hardware Description Language (HDL) representation of the circuit is compiled to obtain a data flow graph (DFG). The transformation phase may perform various operations over the DFG to obtain an efficient DFG representation. The resource-constrained as soon as possible (ASAP) scheduling of the DFG gives the lower bound on the time stamps. The resource-constrained as late as possible (ALAP) scheduling of the DFG gives the upper

bound on the time stamps. The mobility graph obtained from the ASAP and ALAP time stamps provides the degree of freedom for iteration of the algorithm to select a specific time stamp while meeting the time constraints. The above steps of ASAP and ALAP scheduling and mobility calculations use standard approaches and have not be detailed here for brevity [26, 27]. The firefly based algorithm performs the resource and time constrained scheduling of the operations in the DFG so that the operations which are part of a specific group can be executed concurrently. The firefly algorithm uses a multiple-dielectric RTL library for leakage, delay, and area trade-offs. The allocation step of the flow determines the number and types of functional units to be used in the target datapath circuit. The binding phase of the flow deals with attaching various operations to functional units and variables to memory units. The connection allocation phase of the flow fixes types or number of buses, buffers, and multiplexors for the communication among the resources of the target architecture. Finally, leakage-optimal RTL descriptions for datapath and control circuits are generated. Scheduling and binding steps are the two major phases of the RTL optimization flow. In the overall digital integrated circuit design flow, high-level synthesis is followed by logic synthesis and physical synthesis to generate the layout of the target circuit. However, optimization at the logic or physical synthesis steps is not within the scope of the current paper.

The RTL flow will generate integrated circuits that dissipate minimal leakage power for the specified constraints when the proposed scheduling algorithm is used along with leakage, propagation delay, and area estimators. In the RTL design flow, it is assumed that the target architecture datapath is specified as a sequencing data flow graph (DFG), which is essentially a directed acyclic DFG. In the sequencing DFG, each vertex represents an operation and each edge represents a dependency. The sequencing DFG does not support hierarchical entities and the conditional statements are handled using comparison operations. The advantages of the sequencing DFG includes simplicity in modeling during high-level synthesis (HLS). It may be noted that the assumption of sequencing DFG doesn't limit the applicability of the proposed RTL optimization flow. It can be used for any type of DFG or control data flow graph (CDFG). However, as a specific example, the current paper considers sequencing DFG for easy implementation as the research focus of this paper is not on the data structures or compilation of HLS to represent digital integrated circuit behavior. Each vertex of the sequencing DFG contains attributes that specify the operation type associated with that vertex. The propagation delay of a control step is determined by a combination of the delays of the functional unit, the multiplexer, and registers. The proposed flow assumes that each vertex which is connected to the primary input is assigned two registers and one multiplexer. On the other hand, the inner vertices of the sequencing DFG have one register and one multiplexer. The estimator uses the analytical models developed for the components of the datapath or RTL library. It also estimates the total leakage and critical path delay for a scheduled DFG and corresponding binding. The simultaneous reduction of leakage power dissipation and critical path delay can be combined to the single metric of leakage delay product (LDP). Thus, the objective of the scheduler is to minimize the LDP of the whole DFG while assigning time stamps for the nodes of the DFG for a specific allocation and binding. This implicitly facilitates minimization of the leakage power along with critical path delay while considering resource constraints.

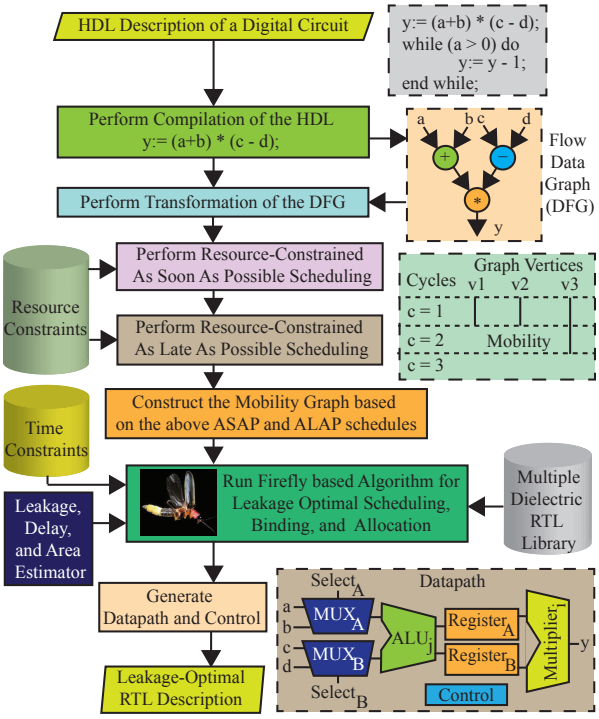


Figure 2: The proposed firefly algorithm based RTL nanoscale leakage optimization flow.

3.2. *Proposed Technology based RTL Optimization*

The leakage delay product of the DFG for a specific scheduling, allocation, and binding during the design it-

eration can be estimated as follows:

$$LDP_{DFG} = \sum_{c=1}^{N_c} \sum_{r=1}^{N_{Rc}} P_{ox\,R}(c,r) T_{pd\,R}(c,r), \qquad (1)$$

where N_c is the number of control steps in a DFG, N_{Rc} is the number of resources active in any control step c. $P_{ox\,R}(c,r)$ is the total leakage power of the r-th functional unit active in the control step c and $T_{pd\,R}(c,r)$ is its propagation delay. The proposed flow generates various forms of the target architecture, such as scheduled DFG with appropriate functional unit assignment to a datapath operation and estimates of leakage power and propagation delay. The design flow assumes that different resources are characterized for leakage power, propagation delay, and area for various oxide thicknesses and dielectrics and are available in the RTL library. In the RTL library construction, it is assumed that all transistors inside a specific RTL component have the same oxide thickness, whereas oxide thickness may differ from one RTL component to another. In such a situation, for nanoscale process variation consideration, a specific oxide thickness T_{ox} is assumed to have a value in the range $(T_{ox} - \Delta T_{ox}, T_{ox} + \Delta T_{ox})$. The leakage and delay estimator assumes such process variations to be Gaussian and hence generates a specific T_{ox} accordingly [28]. It may be noted that a constant L/T_{ox} is maintained and L is scaled along with T_{ox}. Furthermore a constant W/L ratio is maintained to reduce short-channel effects. Thus, all three process parameters T_{oxp}, L, and W are assumed to have Gaussian variations.

3.3. Firefly Algorithm for RTL Design Exploration

It may be noted that the firefly algorithm was introduced in 2008 and since then it has been applied to various areas involving combinatorial and numerical optimization [29, 30, 31]. The firefly algorithm is a biologically-inspired metaheuristic algorithm which mimics the behavior and motion of fireflies. Other similar biologically-inspired algorithms include the bat algorithm, cuckoo search, and ant colony optimization. In general, metaheuristic algorithms are simpler to implement and converge due to low memory requirements and built-in mechanisms to overcome local minima. However, metaheuristic algorithms may not necessarily provide true optimal solutions and can also produce different results when executed at different times. It is possible to use other algorithms such as simulated annealing and Monte Carlo simulations in the context of RTL optimization. Simulated annealing and Monte Carlo have been used previously by the authors and the results are presented in [5]. The current paper is a paradigm shift in which Monte Carlo, which is usually computationally intensive, is not used. The current paper uses the firefly algorithm for its advantages and is for the first time used for RTL optimization in the current paper. The firefly algorithm is a metaheuristic algorithm which has been proven in the existing literature [32, 29]. It is observed that the firefly algorithm has the minimum run time as compared to similar algorithms. In addition, the firefly algorithm convergence is superior to that of the bat and cuckoo search algorithms. Thus, the firefly algorithm has the potential to handle the optimization of large and complex digital designs in the nanoscale era and hence has been adopted in the current paper.

4. Dual Dielectric Approach for RTL Optimization

As a specific example of leakage optimization, this paper deals with gate leakage optimization. However, in a similar manner subthreshold leakage can also be handled [5]. For ultra-thin gate-oxides, leakage is mainly due to direct tunneling. The probability of gate-oxide tunneling is affected by the barrier height which is the voltage drop across the gate oxide as well as the barrier thickness. The gate-oxide tunneling current in a CMOS, for supply voltage V_{dd} and effective gate-oxide thickness T_{ox}, has the following form [33]:

$$I_{ox} \propto W \left(\frac{V_{dd}}{T_{ox}}\right)^2 \exp\left(-\alpha \frac{T_{ox}}{V_{dd}}\right), \qquad (2)$$

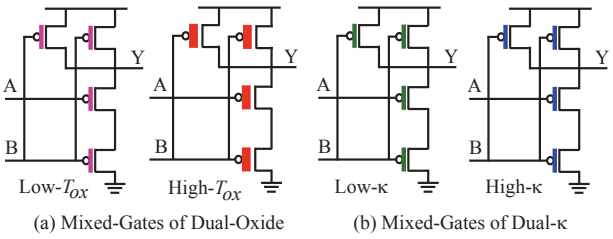
where α is an experimentally derived factor. Based on the above relation, the possible options for gate-oxide leakage are the following: (1) supply voltage reduction, (2) gate-oxide thickness increase, and (3) gate width decrease. Supply voltage reduction has been widely used for dynamic power reduction and hence may work for leakage reduction, however may not be able to be fully effective to counter the exponential growth of gate-oxide leakage. Gate-oxide thickness increase will reduce gate-oxide leakage significantly, but the propagation delay is affected. Gate width decrease is not as effective as gate-oxide leakage changes with it only linearly. Thus, the use of multiple gate-oxide thicknesses can be used as a gate-oxide leakage power and propagation delay trade-off.

In the current paper, the dual-oxide approach is explored for reduction of gate-oxide leakage during RTL synthesis. In addition, the "dual dielectric" approach is being introduced for logic level gate leakage reduction in which SiO_2 is selectively replaced with high-κ materials such as SiON and Si_3N_4 and hence is ex-

plored for RTL synthesis in the current paper as a natural progression of our previous research [33]. The proposed dual-T_{ox}/κ technique can be used with other techniques, such as multi-V_{dd}, multi-V_{Th}, or clock gating for a more comprehensive low power solution for sub-65 nm CMOS integrated circuits. It may be noted that low power high level synthesis has been an area of active research for the last two decades and still remains an active topic due to the ever increasing use of portable devices [26, 34]. At the early stages of research in low-power high-level synthesis capacitive switching power dissipation was of primary attention. Then the research focus changed to subthreshold leakage and later on to gate-oxide leakage. At the RTL, the dual-oxide datapath component library is used for gate-leakage and delay tradeoffs during various HLS tasks. This is analogous to subthreshold leakage optimization using a multiple threshold CMOS (MTCMOS) datapath library. Power (including leakage) optimization at the RTL (higher abstraction with coarse granularity), versus circuit level (lower abstraction with finer granularity), has always been a topic of debate. RTL level is faster and feasible, but the disadvantage is accuracy loss. Circuit level is slower and accurate, but may be computationally intensive and even infeasible to run optimization for large circuits. Optimization at higher levels of abstraction such as RTL ensures that the design is optimized before moving to lower or detailed levels of abstractions, such as logic or circuit.

In the current paper, the three dimensional design space of gate leakage, performance, and area, is explored for reduction of gate leakage during RTL synthesis using dual-T_{ox} or dual-κ. The idea behind the use of dual oxide or dielectric for a small circuit is presented in Fig. 3. The dual thickness approach where functional units containing transistors with either higher T_{ox} or lower T_{ox} is shown in Fig. 3(a). The dual dielectric approach where functional units containing transistors with either higher κ or lower κ is shown in Fig. 3(b). A mix of the different types of functional units during high-level synthesis or RTL optimization can be tradeoffs of leakage and performance.

The gate-oxide leakage optimization problem during HLS phases can be formalized as follows: *Given an unscheduled sequencing data flow graph (UDFG) $G_U(V, E)$, determine the scheduled sequencing data flow graph (SDFG) $G_S(V, E)$ with appropriate resource binding such that the total leakage and delay product (LDP) is minimal for specified resource constraints.* In the sequencing DFG, V denotes the set of all vertices or nodes whereas E denotes the set of all edges. V_{cp} is used to denote the set of vertices in the critical path from the source vertex to the sink vertex of the sequencing DFG. The RTL optimization problem can be formulated as follows:



(a) Mixed-Gates of Dual-Oxide (b) Mixed-Gates of Dual-κ

Figure 3: Concept of dual dielectric approach for gate leakage tradeoff.

$$\text{Objective:} \quad \text{Minimize}\left(\text{LDP(DFG)}\right), \tag{3}$$

$$\text{Constraints:} \quad \text{Allocated}\left(R_{k,t}\right) \leq \text{Available}\left(R_{k,t}\right), \quad \forall c \in N_c. \tag{4}$$

In the above expression, $R_{k,t}$ denotes a resource (i.e. functional unit or datapath component) of type k consisting of transistors of a specific T_{ox} or κ. t is a technology index which represents high-T_{ox}/low-T_{ox} for dual-oxide or high-κ/low-κ for dual-dielectric. For N_c number of clock cycles in a DFG, c is a specific clock cycle.

The use of dual-T_{ox} or dual-κ can increase the fabrication cost of the integrated circuits [33, 35]. These technologies may need additional processing steps and more masks during the photolithographic processes. However, the fabrication costs would be compensated by the reduction of the energy or power costs and hence a trade-off cab be achieved between the energy efficiency and fabrication cost which is very important for mobile electronics. In addition, research on the manufacturing aspects is very active and new process technologies in the future addressing these issues may become available. It is possible to use dual-T_{ox} and dual-κ together, but in that case the fabrication process will need additional steps as compared to dual-T_{ox} only or dual-κ only.

5. The Proposed Firefly based RTL Optimization Algorithm

In this Section a detailed discussion of the firefly based optimization algorithm is presented. For the purpose of comparison with more established algorithms, integer linear programming (ILP) based optimization is also presented.

5.1. Firefly based RTL Optimization Algorithm

The firefly optimization algorithm is based on the following 3 assumptions [29, 30, 36, 31]:

(1) It is assumed that all fireflies are of the same sex. Fireflies are attracted towards each other irrespective of their sex. However, in nature both male fireflies and female fireflies exist, and the male fireflies take flight and emit flashes which are different from species to species in terms of color, rate, length, and intensity. Female fireflies are wingless or have short wings and fly little [36].

(2) The level of attraction or attractiveness between fireflies is proportional to their brightness. For any two flashing fireflies, the firefly with less bright flashing will move towards the firefly with more bright flashing. The brightness (and hence attractiveness) increases as the distance between the two fireflies decreases whereas it decreases as the distance between the two fireflies increases. A particular firefly moves randomly if there are no brighter fireflies available within its visible vicinity. The attractiveness of a firefly can be modeled as follows [29, 31]:

$$\beta_{\text{attract}}\left(d_{\text{distance}}\right) = \beta_{\text{attract},0}\exp\left(-\gamma_{\text{light}} d^2_{\text{distance}}\right), \quad (5)$$

where β_{attract} is the attractiveness between the fireflies. d_{distance} is the distance between two fireflies. $\beta_{\text{attract},0}$ is the attractiveness when the distance between the fireflies is zero, i.e. $r_{\text{distance}} = 0$. As an example, $\beta_{\text{attract},0} = 1$; it may be noted that $\beta_{\text{attract},0} = 0$ makes the algorithm a simple random walk. γ_{light} is the light absorption coefficient for a specific medium. While in theory $\gamma_{\text{light}} \in [0, \infty)$, it is typically a value in the range $(0.1, 10)$ for the firefly based algorithm.

(3) The brightness of a firefly is determined by the nature of the objective function. For maximization problems, the brightness is proportional to the value of their objective functions. On the other hand, in the case of minimization problems, the problem can be translated to the maximization problem by negating the objective function.

The basis of the firefly optimization algorithm is depicted in Fig. 4. In a population of N_F fireflies each firefly is denoted as FF_i. In other words, there are N_F solutions. The location of each firefly represents a solution. For example, for firefly FF_i the location is S_i. Each location S_i is of dimension N_S, each dimension representing a tuning variable for optimization. For example, a clock cycle for scheduling a vertex, a resource of specific type, and a resource of a specific technology. The luminous intensity of the flash produced by a firefly FF_i is I_{ϕ_i} which is proportional to the cost/objective function when the optimization is formulated as a maximization problem. In the current case of LDP_{DFG} minimization, it is proportional to the negative of the cost function, i.e. $I_{\phi_i} \propto -LDP_{DFG_i}$. LDP_{DFG_i} is the cost function at location S_i. The movement of a firefly FF_i which is attracted to another more attractive or brighter firefly FF_j, is determined by the following expression [29, 31]:

$$\begin{aligned} S_i = & \; S_i + \left(S_j - S_i\right)\beta_{\text{attract}}\left(d_{\text{ij}}\right) + \eta\Theta_i & (6)\\ = & \; S_i + \left(S_j - S_i\right)\beta_{\text{attract},0}\exp\left(-\gamma_{\text{light}} r^2_{\text{ij}}\right) + \eta\Theta_i, & (7)\end{aligned}$$

where S_i is the location of firefly FF_i and there are N_F fireflies in a population each representing a possible solution. S_i is a vector at a location of N_S dimensions. The coefficient of randomness η has a value in the range 0 to 1, i.e. $\eta \in [0, 1]$. The 2nd term is due to the attraction of the fireflies. The 3rd term introduces randomization to the algorithm. Θ_i is a vector of random numbers which assumes a Gaussian or uniform distribution in the range $[0, 1]$. The distance between two fireflies FF_i and FF_j is their Cartesian distance [31]:

$$d_{ij} = ||S_i - S_j|| = \sqrt{\sum_{m=1}^{N_S}\left(s_{i,m} - s_{j,m}\right)^2}. \quad (8)$$

In the current algorithm d_{ij} is estimated as the error between two solutions S_i and S_j, i.e. the difference of the negative of the LDP values.

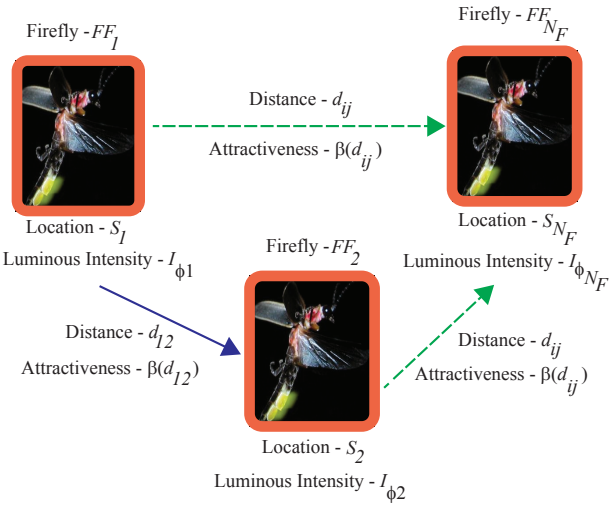


Figure 4: The concept of the firefly based algorithm.

The pseudo-code of the proposed firefly based approach that performs simultaneous scheduling, alloca-

tion, and binding during RTL synthesis while minimizing gate leakage is presented in Algorithm 1. The algorithm searches for an RTL realization that has minimum leakage power for specified time and resource constraints for either dual-T_{ox} or dual-κ techniques. For either technique, it is important that every vertex is scheduled in a way such that less leaky resources can be assigned to non critical resources so that the total delay is not effected while maximizing the leakage reduction. The algorithm assumes ASAP and ALAP schedules, mobility graph, multiple-dielectric RTL library, and resource/time constraints as inputs. Various parameters such as, γ_{light}, $\beta_{\text{attract},0}$, and η are initialized. The target delay $T_{pd_{CON}}$ is calculated for a specific delay trade-off with the number of clock cycles determined from the resource-constrained ASAP and ALAP time stamps. The mobility of the vertices in the sequencing DFG, which is an indicator of the chances of assigning a higher thickness or dielectric resource, is dependent on the total number of available lesser leaky resources in the RTL library. The propagation delay of the slowest resource in the baseline corner/technology is typically the divider with low oxide thickness or lower dielectric.

The main iteration of the algorithm starts with the generation of a random population of N_F fireflies, each denoted as FF_i. Each location of the firefly S_i is essentially a configuration in terms of time stamp of the DFG and binding of the operations to specific resources. The critical path delay for each of the solutions can be estimated as $T_{pd_{ST,i}}$. For the cases of multicycling target architecture, the critical path delay of the circuit is calculated as the product of the number of clock cycles and the maximum propagation delay of any resource in the specific target architecture. The assignment of the higher T_{ox}/κ resources for leakage reduction will increase the critical path delay of the target architecture which can be compensated using chaining and multicycling. The multicycling operation increases the number of clock cycles in the architecture and chaining can be incorporated when there are only few operations. The use of both multicycling and chaining in the architecture ensures execution of the ready operations when a resource is available for execution.

In the algorithm, locations which do not meet the timing constraints are discarded and need not be considered further. This speeds up the proposed algorithm as compared to the original firefly algorithm. In generating a solution in the proposed flow, a vertex is selected by the algorithm at random and checked if a less leaky resource can be assigned in all possible clock cycles and that it satisfies a time constraint. In the estimation of leakage power, critical path delay, or LDP, a random thickness is assumed in the range of $(T_{ox} - \Delta T_{ox}, T_{ox} + \Delta T_{ox})$ to take nanoscale process variations into account. ΔT_{ox} is approximately 15% for a monolayer misplacement of

Algorithm 1 The Nature-Inspired Firefly based Architecture-Level Leakage Optimization.

Input: ASAP schedule, ALAP schedule, Mobility Graph, Multiple-dielectric RTL library, Resource constraints, Time constraints, Maximum number of iterations N_{Count}.
Output: Leakage optimal schedule of DFG, and Leakage, delay, and area estimates.
Initialize light absorption coefficient γ_{light}.
Initialize attractiveness when the distance is between the fireflies is zero, $\beta_{\text{attract},0}$.
Initialize the coefficient of randomness η.
Initialize the iteration counter as $Count \leftarrow 0$;
Fix the number of clock cycles N_c as the maximum of the resource-constrained ASAP and resource-constrained ALAP schedules.
Assume target delay as $T_{pd_{CON}}$ as $Delay_{CON} \times N_c \times$ Delay of the slowest resource in the baseline corner.
Generate N_F fireflies FF_i, $i = 1, 2, \cdots, N_F$.
Generate N_F solutions S_i. Each solution is a time stamp in the mobility range and with an allocation and binding from the available resource $R_{Avl}[c][k][t]$.
Estimate the critical path delay or latency of the target architecture for these solutions S_i.
if $\left(T_{pd_{ST,i}} > T_{pd_{CON}}\right)$ **then**
    Discard S_i for these i's as time constraint is violated.
end if
Estimate the LDP_i of the available locations S_i.
Assume luminous intensity I_{ϕ_i} of firefly FF_i as $-LDP_i$.
while $(Count < N_{Count})$ **do**
    for $(i = 1 \rightarrow N_F$, other than the discarded $S_i)$ **do**
        for $(j = 1 \rightarrow N_F$, other than the discarded $S_j)$ **do**
            if $\left(I_{\phi_i} < I_{\phi_j}\right)$ **then**
                Calculate distance d_{ij} between S_i and S_j.
                Calculate the attractiveness between fireflies FF_i and FF_j as $\beta_{\text{attract}}\,(d_{\text{ij}})$.
                Update location S_i using the movement expression.
                Evaluate the new solution LDP_i and update the luminous intensity I_{ϕ_i}.
            end if
        end for
    end for
    Rank the fireflies according to the associated LDP values and update the best intensity value $I_{\phi_{\text{best}}}$.
    Update the best location S_{best}.
    Update the counter, $Count \leftarrow Count + 1$;
end while
return Schedule data flow graph DFG_S, estimate of leakage, delay, and area.

SiO_2. As LDP minimization is the objective for specified resource and time constraints, the luminous intensity I_{ϕ_i} is assumed as the negative of the LDP. The algorithm iterates for the specified maximum number of iterations for the set of fireflies FF_is/FF_js with locations S_is/S_js which meet the timing constraints in the previous step. The locations of the fireflies are updated based on the calculations of the distance and attractiveness. The best solution is the one which provided maximum I_ϕ for minimal LDP.

5.2. ILP Based Optimization Algorithm

As an alternative and for comparison purposes, this subsection briefly presents an integer linear programming (ILP) based RTL algorithm to perform LDP minimization. It is a fact that ILP has scalability issues due to its exponential complexity. However, the novel contribution of this paper is not ILP; rather it is the firefly algorithm based RTL optimization. ILP is briefly presented for a direct comparative perspective. Further broad comparison with selected existing low-power RTL approaches is presented in Section 7.

(a) *Objective Function*: The objective of the ILP based RTL algorithm is to minimize the LDP of the whole DFG over all clock cycles. The objective function can be expressed in the following manner:

$$\text{Minimize :} \quad LDP_{DFG} \tag{9}$$

$$\text{Minimize :} \quad \sum_c \sum_i \sum_t X_{i,t,c} LDP(i,t). \tag{10}$$

where $LDP(i,t)$ is the leakage delay product of $R_{k,t}$ used by vertex v_i of the DFG and $X_{i,t,c}$ is a binary decision variable.

(b) *Uniqueness Constraints*: The uniqueness constraints in the ILP formulations at the RTL ensure that each vertex v_i of the sequencing DFG is scheduled in the appropriate clock cycle(s) within the mobility range (AS_i, AL_i) with the resource $R_{k,t}$ assignment. Sometimes a vertex in the DFG can be scheduled for more than one clock cycle based on the delay of a resource for multicycle operations. The uniqueness constraints of the RTL ILP formulations are represented as, $\forall i$, $1 \leq i \leq V$:

$$\sum_c \sum_t X_{i,t,c} = 1. \tag{11}$$

(c) *Precedence Constraints*: The precedence constraints in the ILP formulations at the RTL ensure that for a vertex v_i of the sequencing DFG, all its predecessor vertices are scheduled in earlier clock cycles and its successor vertices are scheduled in later clock cycles to maintain data dependency. The precedence constraints of the RTL ILP formulations are modeled as, $\forall i, j, v_i \in Pred_{v_j}$:

$$\sum_t \sum_{d=AS_i}^{AL_i} dX_{i,t,d} - \sum_t \sum_{e=AS_j}^{AL_j} eX_{j,t,e} \leq -1. \tag{12}$$

(d) *Resource Constraints*: The resource constraints in the ILP formulations at the RTL guarantee that each of the clock cycles in the DFG needs resources not exceeding the available number of resources, which are specified by allocation as a part of resource constraints. The resource constraints of the RTL ILP formulations can be enforced as, $\forall t$ and $\forall c$, $1 \leq c \leq N$:

$$\sum_{i \in R_{k,t}} X_{i,t,c} \leq M_{k,t}. \tag{13}$$

The flow of the proposed ILP based RTL optimization approach is presented in Algorithm 2. The inputs to the algorithm are the same as Algorithm 1 in terms of representation and constraints. For given resource constraints, the ILP-based algorithm determines a target architecture RTL description that has minimum LDP. The resource constraints of the RTL optimization algorithm are expressed as the number of different types of resources made of transistors of each technology for dual-T_{ox} or dual-κ.

Algorithm 2 ILP Based Architecture-Level Leakage Optimization.

1: **Input:** ASAP schedule, ALAP schedule, Mobility Graph, Multiple-dielectric RTL library, Resource constraints, Time constraints, Maximum number of iterations N_{Count}.
2: **Output:** Leakage optimal schedule of DFG, and Leakage, delay, and area estimates.
3: Construct the resource allocation table and available resource table based on the specified resource constraints R_{con} of the optimization.
4: Find the number of different resources for each technology index t for both dual-T_{ox} or dual-κ technology using the resource allocation table of the above step.
5: Model the RTL ILP formulations of the DFG for the specified constraints using AMPL.
6: Solve the ILP formulations which are modeled in AMPL using an ILP-solver to obtain the optimal solution.
7: **return** Schedule data flow graph DFG_S, estimate of leakage, delay, and area.

6. Nano-CMOS based RTL Component Library

The design of the datapath component library, also know as RTL library or architecture-level library, uses a

3-level hierarchical approach in the current paper. At the highest level of hierarchy, RTL, functional units such as adders, subtractors, multipliers, divider, comparator, multiplexer, and register are available. These RTL units utilize logic level components (logic gates) which comprise the second level of the hierarchy (logic level). At the bottom is the CMOS realizations of these gates which constitute the transistor level or circuit level. For much higher accuracy of the library the physical design level (layout) can also used. This section describes a characterization methodology that is appropriate for technology in development, such as sub-65nm CMOS, and which generates estimates for gate leakage current and propagation delay that are used by the optimization algorithms. The current datapath component is based on the datapath component library presented by the authors in [37], but has been extended to different dielectrics (κ) as well as to analytical model format. It may be noted that the datapath component library creation along with the HLS phases of the previous sections provides a complete perspective of generation of optimal RTL. At the same time this library information can be used by other researchers in their research works. In cases in which the datapath or RTL library is purchased from a 3rd party vendor, this library creation is not necessary. The following subsections discuss the approaches used at the different levels of abstraction in a bottom-up fashion.

6.1. CMOS Transistor Level Characterization

At the characterization stage of the digital integrated circuit design cycle, it is assumed that initial silicon is available and compact/SPICE models are derived. The different components of gate-oxide leakage current manifested during ON and OFF conditions of a NMOS transistor are depicted in Fig. 5. In the gate-oxide leakage current components, I_{gd} and I_{gs} are originating from the diffusions layers, drain and source, respectively, directly to gate region. On the ON state, I_{gcd} and I_{gcs} are the components from the drain and source diffusions to gate oxide through the active channel. The quantitative significance of each of the components is different based on the type as well as the state of the transistor. The gate to body leakage current component (I_{gb}) is observed to be minimal as compared to other components and hence not shown. In a similar manner the various components can be shown for a PMOS transistor. In addition, the leakage components values may change depending on the oxide thickness and gate dielectric. The advantage of using SPICE or an analog simulator for leakage characterization is that very accurate analysis can be performed within the complete dynamic range of the transistor operation. The lack of real-life manufacturing processes with published data is a challenge for performing such nanoscale leakage characterization. The current paper uses the predictive technology model (PTM) since it is well accepted by the research community [38].

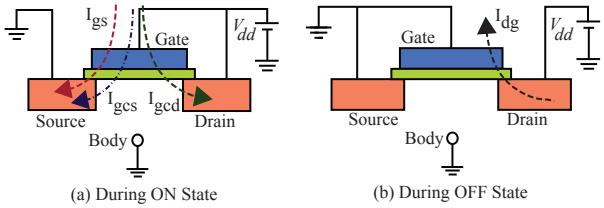


Figure 5: Gate leakage current flow in the two states of an NMOS transistor.

6.2. Logic Gate Level Characterization

The leakage analysis for a nano-CMOS logic gate is more involved than the case of a single transistor: leakage is affected by the type of transistors, location of the transistors, as well as input conditions. As a specific example, the gate-oxide leakage components are shown in Fig. 6 for a nano-CMOS based 2-input NAND gate. The figure depicts the gate-oxide leakage current components for the 4 possible input combinations for the 2-input NAND logic. In this figure, the individual leakage components such as I_{gd}, I_{gs}, I_{gcd}, and I_{gcs} are not shown as was the case in Fig. 5, for brevity.

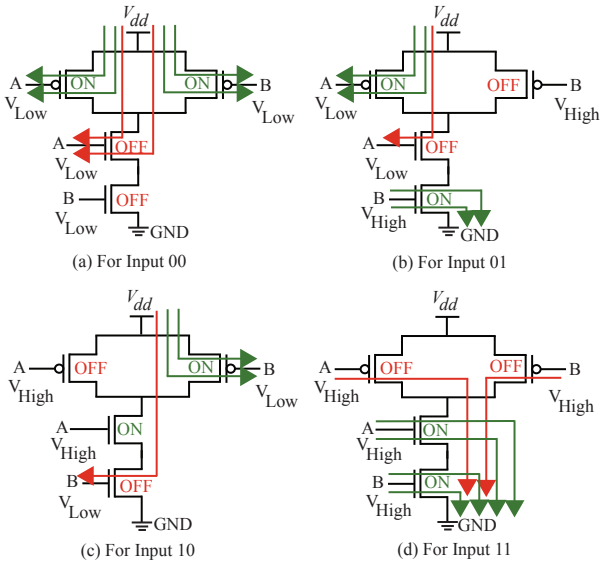


Figure 6: Leakage current flow the four states of a 2-input NAND nano-CMOS circuit.

The total gate-oxide leakage current for each of the MOSFETs is calculated by summing all individual components as follows [37, 33]:

$$I_{ox}\text{MOS}[i] = I_{gs}[i] + I_{gd}[i] + I_{gcs}[i] + I_{gcd}[i] + I_{gb}[i], \quad (14)$$

where the index i denotes the i-th MOSFET within a logic gate. For a specific state, the gate-oxide leakage for the nano-CMOS based NAND logic gate $I_{ox}\text{NAND, state}$ is calculated by summing the absolute gate-oxide leakage currents over all MOSFETs present in the logic gate as both positive as well negative gate-oxide leakage currents contribute towards overall gate-oxide leakage. For the nano-CMOS based NAND, the state-specific gate-oxide leakage is the following:

$$I_{ox}\text{NAND, state} = \sum_{\forall \text{MOS}_i} |I_{ox}\text{MOS}[i]|, \quad (15)$$

where "state" denotes the operating state of a nano-CMOS NAND gate based on input conditions.

For each of the four possible states of a 2-input NAND logic gate (00, 01, 10 and 11), the overall gate leakage current ($I_{ox}\text{NAND}_{00}$, $I_{ox}\text{NAND}_{01}$, $I_{ox}\text{NAND}_{10}$, and $I_{ox}\text{NAND}_{11}$, respectively) is calculated from the above equations. Assuming that all four states are equiprobable, an average gate-oxide leakage for the nano-CMOS NAND logic gate is calculated as follows:

$$\begin{aligned} I_{ox}\text{NAND} &= \left(\tfrac{1}{4}\right)\left(I_{ox}\text{NAND}_{00} + I_{ox}\text{NAND}_{01}\right. \\ &\quad \left. + I_{ox}\text{NAND}_{10} + I_{ox}\text{NAND}_{11}\right). \end{aligned} \quad (16)$$

Following the above approach, statewise gate-oxide leakage and average gate-oxide leakage of other logic gates can be estimated. The characterization of propagation delay for the nano-CMOS logic gates can be performed in a similar manner which is not presented due to lack of space [37]. The characterization results for AND, NOT, OR and NOR logic gates are presented in Table 2 and Fig. 7. It is clear that the NAND gate outperforms all other 2-input gates with respect to total gate leakage and propagation delay, and hence the NAND realization is beneficial for low gate leakage ultrathin nano-CMOS circuit design.

The accurate calculation of silicon area of a circuit can be done from its physical design. However, for RTL optimization it is sufficient for a quick estimate. In the current paper, the area of a NAND logic gate is calculated using the following analytical formula [39]:

$$\begin{aligned} A_{NAND} &= k_{inv}\left(1 + 4\left(n_{in} - 1\right)\sqrt{\tfrac{AR_{NAND}}{k_{inv}}}\right) \\ &\quad \left(1 + \frac{\left(\frac{W_{NMOS}}{f} - 1\right)(1+\beta_{NAND})}{\sqrt{k_{inv} AR_{NAND}}}\right), \end{aligned} \quad (17)$$

Table 2: Gate-Oxide Leakage and Propagation Delay for Various Nano-CMOS 2-Input Logic Gates.

	I_{ox} in $\left(\frac{nA}{\mu m}\right)$ for logic states				T_{pd}
	I_{ox00}	I_{ox01}	I_{ox10}	I_{ox11}	in ps
NAND2	55.8	172.0	35.8	247.6	256.9
NOR2	102.1	128.5	121.3	246.6	378.2
AND2	179.6	295.7	160.0	298.5	350.0
OR2	225.4	179.6	171.8	297.7	340.3

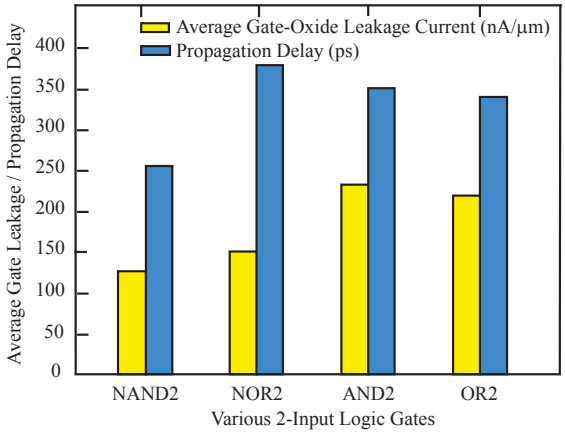


Figure 7: Gate-oxide leakage and delay for nano-CMOS logic gates.

where W_{NMOS} is the width of the NMOS, f is the minimum feature size, k_{inv} is the area of minimum size inverter, AR_{NAND} is the aspect ratio of the NAND gate, n_{in} is the number of inputs, and β_{NAND} is the ratio of PMOS width to NMOS width.

6.3. Register Transfer Level Characterization

In hierarchical modeling it is assumed that datapath components are constructed using universal logic gates. In the remainder of the current paper NAND logic gates will be used as the universal element to realize the RTL components and therefore the rest of this subsection will concentrate on the gate level characterization of the NAND. To address the problem of unknown *a priori* probabilities for the internal nodes in the logic netlist of RTL components, one approach is to simulate the RTL component designs at the transistor level using the compact models and SPICE. In order to obtain accurate probabilities, the simulations must use random input vectors of very large length. Considering that each NAND logic gate comprises of 4 transistors and each RTL component may contain hundreds or thousands of gates, it becomes apparent that such simulations are not expected to complete in a reasonable amount of time

and the design cycle may be lengthened. Therefore the current paper proposes to estimate the required probabilities via the use of the powerful hardware description and verification language (HDVL) SystemVerilog.

As a specific simple example, consider the NAND implementation of a half-adder as shown in Fig. 8. Structural SystemVerilog simulations are performed for combinational design assuming random input vectors a and b with 10^6 bits each. From the digital simulations the probabilities are calculated as shown in the figure. The total simulation time has been 4.7 sec. However, a similar SPICE simulation would have taken much longer time. If equal probabilities for all nets are assumed then the gate-oxide leakage current estimate for this half adder is the following:

$$I_{ox,\text{Res}} = 5 \times I_{ox}\text{NAND} \quad (18)$$

$$= 5 \times \left(\frac{1}{4}\right)\left(I_{ox}\text{NAND}_{00} + I_{ox}\text{NAND}_{01} + I_{ox}\text{NAND}_{10} + I_{ox}\text{NAND}_{11}\right), \quad (19)$$

where $I_{ox,\text{Res}}$ is the average gate-oxide leakage of a datapath component, functional unit, or RTL resource. For the specific 45nm CMOS technology used in this paper, $I_{ox,\text{Res}} = 829\frac{nA}{\mu m}$, for the above half adder.

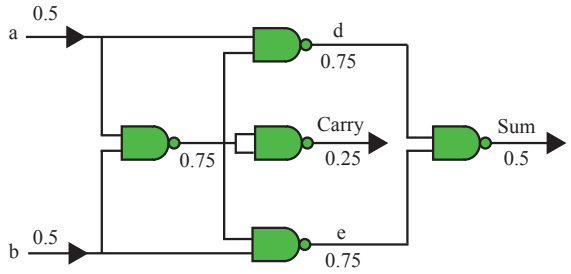


Figure 8: A half-adder as NAND netlist with probabilities for a specific net to be at logic level "1".

In general, it can be assumed that in a n-bit RTL component there are N_{total} NAND logic gates out of which N_{cp} are in its critical path. The gate-oxide leakage current ($I_{ox,\text{Res}}$) of an RTL component is the following:

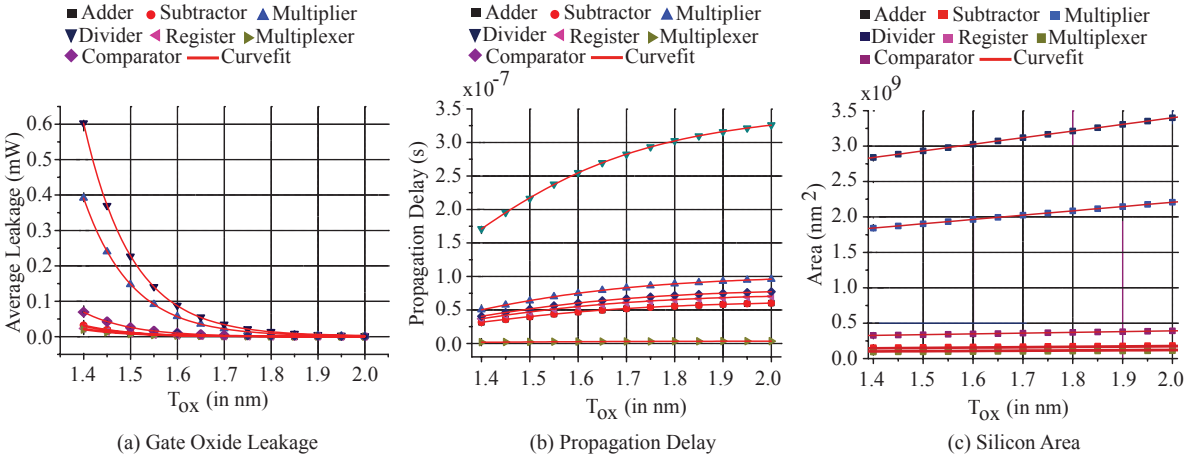
$$I_{ox,\text{Res}} = \sum_{i=1}^{N_{total}} Pr_i I_{ox}\text{NAND}_i, \quad (20)$$

where Pr_i is the signal probability for i^{th} 2-input NAND is logic "1". $I_{ox}\text{NAND}_i$ is the average gate leakage current dissipation of the i^{th} 2-input NAND, assuming all states to be equiprobable. Similarly, the propagation delay of the n-bit RTL component is the following: $T_{pd}\text{Res} = \sum_{i=1}^{N_{cp}} T_{pd}\text{NAND}_i$. The silicon area of a n-bit RTL component is the following: $A_{\text{Res}} = \sum_{i=1}^{N_{total}} A_{\text{NAND}_i}$. In this model the effect of interconnect wires is not considered as the gate-oxide leakage current is restricted to active devices and does not contribute on power dissipation in the interconnect.

Using the steps presented above, data are generated to thoroughly characterize each of the datapath components. However, for optimization during RTL synthesis one option is to describe the characterization data obtained as analytical functions of oxide thickness or dielectric constant for the use of dual-T_{ox} and dual-κ. The simulated results are plotted and different analytical functions are fit for gate-oxide leakage, propagation delay, and area in terms of T_{ox} and κ. The gate oxide leakage, propagation delay, and silicon area for different functional units are shown in Fig. 9 with respect to the oxide thickness. The corresponding coefficients of the curve fitting are provided in Table 3. Using these models leakage, delay, or area of any resource can be calculated for specific T_{ox}. Similarly, gate-oxide leakage and propagation delay of different RTL components in terms of dielectric constants is presented in Fig. 10. The coefficients of corresponding different analytical functions are presented in Table 4. Using these models leakage, delay, or area of any resource can be calculated for specific κ. Separate analytical models are not presented for area as the area of resources can be calculated from the same area model which is presented in terms of T_{ox}. It may be noted that the analytical models obtained have a correlation coefficient of approximately 0.99. In other words, the analytical models faithfully represent the simulation data.

7. Experimental Results

The algorithm is implemented in C and integrated into the in-house high-level synthesis framework [5, 27]. AMPL is used to model the RTL ILP formulations and are then solved using LP-Solve. The computational platform for the experiments is an Intel® quad Xeon® server with 24GB RAM. The algorithm has been exhaustively tested with several RTL benchmark circuits for several constraints. However, the experimental results presented in this section are for a selected set of circuits and resource-time constraints, for brevity. Specifically experimental results for the following digital signal processing (DSP) benchmark circuits are presented [5, 27]: (1) Auto-Regressive filter (ARF) (total 28 nodes, 16*, 12+, 40 edges), (2) Band-Pass filter (BPF) (total 29 nodes, 10*, 10+, 9-, 40 edges), (3) Discrete cosine transformation filter (DCT) (total 42 nodes,



(a) Gate Oxide Leakage (b) Propagation Delay (c) Silicon Area

Figure 9: RTL unit characterization for different oxide thicknesses.

Table 3: Gate-Oxide Leakage, Propagation Delay, and Area of RTL Components as Analytical Functions of T_{ox} (nm).

Datapath Components	$I_{ox}(\mu A) = a\, e^{(-T_{ox}/\alpha)} + b$			$T_{pd}(ns) = \frac{(a_1 - a_2)}{(1+(T_{ox}/\beta)^{\gamma})} + a_2$				$A(nm^2) = \alpha T_{ox} + \beta$	
	α	a	b	a_1	a_2	β	γ	α	β
Adder	0.10	24.82	0.08	-7.37	64.47	1.36	7.24	0.45 $\times 10^8$	0.74 $\times 10^8$
Subtractor	0.10	27.76	0.09	-7.37	64.47	1.36	7.24	0.50 $\times 10^8$	0.83 $\times 10^8$
Multiplier	0.10	331.62	1.09	-11.75	102.74	1.36	7.24	6.07 $\times 10^8$	9.92 $\times 10^8$
Divider	0.10	510.89	1.68	-39.93	349.12	1.36	7.24	9.35 $\times 10^8$	15.28 $\times 10^8$
Register	0.10	19.70	0.06	-8.63	75.48	1.36	7.24	0.36 $\times 10^8$	0.58 $\times 10^8$
Multiplexer	0.10	16.77	0.05	-0.41	3.66	1.36	7.24	0.30 $\times 10^8$	0.50 $\times 10^8$
Comparator	0.10	58.83	0.19	-9.47	82.82	1.36	7.24	1.07 $\times 10^8$	1.76 $\times 10^8$

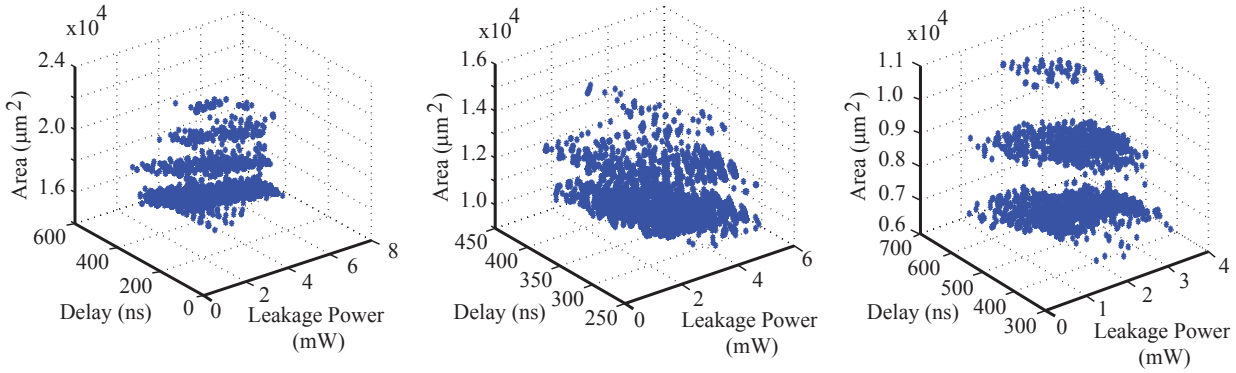
13*, 29+, 68 edges), (4) Elliptic-Wave filter (EWF) (total 34 nodes, 8*, 26+, 53 edges), (5) Finite impulse response filter (total 23 nodes, 8*, 15+, 32 edges), and (6) MPEG motion mector (MMV) (total 32 nodes, 14*, 14+, 2 Load, 2 Store, 29 edges).

The algorithm was applied on various RTL benchmark circuits for both dual-T_{ox} and dual-κ approaches. For the dual-T_{ox} approach the gate-oxide leakage is calculated for single thickness of 1.4 nm as the base case value from the BSIM4 model with supply voltage V_{dd} = 1 V. For dual-T_{ox} approach, thicknesses 1.4 nm and 1.7 nm are considered. Similarly, for the dual-κ approach the base case gate-oxide leakage is calculated using SiO_2 as the dielectric. For dual-κ, a dual dielectric pair $SiO_2(\kappa = 3.9)$ - $Si_3N_4(\kappa = 7)$ is considered with the gate oxide thickness for both low-κ and high-κ resources a constant 1.4 nm. The experiments are performed for single cycle and multicycling operations of the datapaths. The 3-dimensional design exploration by the algorithm before converging to an optimal solution is shown in Fig. 11 for selected DSP benchmark circuits. As evident, the algorithm widely iterates in the design space and does not get stuck in local minima. Based on the resource and time constraints any point in the design space is a solution. The points in the 3-D space represent a number of design alternatives for the datapath during each iteration of the algorithm. The time for the convergence of the algorithm is in the range of 3 min to 5 min in the specific computational platform. A selected set of resource constraints is shown in Table 5.

A selected specific set of results for specific resource and time constraints for the RTL benchmark circuits is presented in Table 6. The experimental results take into account the gate-oxide leakage, propagation delay, and area of functional units, interconnect units, and storage units present in the target architecture. For a dual-T_{ox} approach the percentage reduction in leakage is calculated as follows: $\Delta P_{ox} = \left(\frac{P_{ox\,ST} - P_{ox\,DT}}{P_{ox\,ST}}\right) 100\%$. The percentage area overhead is calculated as follows: $\Delta A = \left(\frac{(A_{DT} - A_{ST})}{A_{ST}}\right) 100\%$. For a dual-$\kappa$ approach reduction in gate-oxide leakage is calculated as follows:

Table 4: Gate-Oxide Leakage and Propagation Delay of RTL Components as Analytical Functions of κ.

Datapath Components	$I_{ox}(\mu A) = ae^{\left(\frac{-K}{\alpha}\right)} + I_{ox0}$			$T_{pd}(ns) = \begin{cases} a_2 + \frac{a_1 - a_2}{\left(1+e^{\left(\frac{\kappa-\kappa_0}{d\kappa}\right)}\right)}, & 2.5 \leq \kappa < 6 \\ \beta e^{\left(\frac{-\kappa}{\alpha}\right)} + T_{pd_0}, & 6 \leq \kappa < 30 \end{cases}$					
	I_{ox0}	a	α	a_1	a_2	κ_0	β	α	T_{pd_0}
Adder	-1.20	2.53	0.36	5.06	63.22	4.02	0.47	-10.63	0.37
Subtractor	-1.34	2.83	0.36	5.06	63.22	4.02	0.47	-10.63	0.37
Multiplier	-16.10	33.81	0.36	8.07	100.74	4.02	0.47	-10.63	0.37
Divider	-24.80	52.08	0.36	27.43	342.32	4.02	0.47	-10.63	0.37
Register	-0.95	2.00	0.36	5.93	74.01	4.02	0.47	-10.63	0.37
Multiplexer	-0.81	1.70	0.36	0.28	3.59	4.02	0.47	-10.63	0.37
Comparator	-2.85	5.99	0.36	6.50	81.21	4.02	0.47	-10.63	0.37



(a) For Auto-Regressive Filter (b) For Band-Pass Filter (c) For Elliptic-Wave Filter

Figure 11: Plots of leakage, propagation delay and area for different benchmark integrated circuits for RTL design space exploration.

$\Delta P_{ox} = \left(\frac{P_{ox\,SK} - P_{ox\,DK}}{P_{ox\,SK}}\right) 100\%$. The critical path delay (T_{pd}) of the target architecture is the sum of the delays of the vertices in the longest path of the sequencing DFG for single cycle case and number of clock cycle times the slowest delay resource for multicycling-chaining case. The delay trade-off factor (DTF) is used to represent various time constraints for the experiments. For brevity for each time constraint, results averaged over the resource constraints are presented as power and performance trade-offs are more important in the design exploration space than the area in the current scope of this research.

As evident from the results, for dual-T_{ox}, the gate leakage reduction for all ranges is between 58 % - 89 %. The area penalty is ranging from 7 % to 35 % for different timing constraints specified in the experiments. For dual-κ, the gate leakage reduction for all ranges is between 62 % - 94 %. The area penalty for dual-κ is in the range of 5 % to 37 %. It is observed from the results that the area penalty increases with reduction of the gate-oxide leakage for a dual-T_{ox} approach. The variation in area with the dual-κ approach is due to the allocation of more resources. Overall average area penalty is 14.13 % for dual-T_{ox} technique and 15.14 % for dual-κ technique. The delay penalty for both dual-T_{ox} and dual-κ has been in the range of 0 % to 30 %. It is observed that the extent to which gate-oxide leakage reduction happens increases as the number of available high-T_{ox} or high-κ resources increases. The results averaged over the different resource and time constraints are presented in Fig. 12 for both techniques used in the firefly algorithm. Different benchmark circuits based on different types and number of operations and accordingly resources needed to implement them yielded diverse results. The reduction for the FIR circuit is highest whereas for the EWF circuit lowest. The reduction obtained from the dual-κ approach is more than that from the dual-T_{ox}, in the range of 1% to 4 %. It may be noted that the delay and area overheads have been presented in the range of experiments between various constraints and benchmarks. A design engineer can make design decisions based on the power, delay, and area trade-offs

Table 5: Selected Resource Constraints used in the Experiments.

Number of Resources for Various Oxides						Resource Constraint No.
Multiplier		Adder		Subtractor		
Low-T_{ox}/κ	High-T_{ox}/κ	Low-T_{ox}/κ	High-T_{ox}/κ	Low-T_{ox}/κ	High-T_{ox}/κ	
1	1	2	0	2	0	RC1
2	1	1	1	1	1	RC2
2	0	0	2	0	2	RC3
3	0	1	1	1	1	RC4

Table 6: Gate-Oxide Leakage Reduction for Various Benchmarks using Dual-T_{ox} or Dual-κ Approaches.

Benchmark Digital Circuits	Delay Trade-off Factor	Dual-T_{ox} (For $\kappa = 3.9$)				Dual-κ (For $T_{ox} = 1.4nm$)			
		$P_{ox_{DT}}$ (μW)	$T_{pd_{DT}}$ (ns)	ΔP_{ox} $(\%)$	ΔA $(\%)$	$P_{ox_{DK}}$ (μW)	$T_{pd_{DK}}$ (ns)	ΔP_{ox} $(\%)$	ΔA $(\%)$
ARF	Base Case P_{ox} = 6618.2 μW, T_{pd} = 308.9 ns, A =15293.7 μm^2								
	1.0	1628.1	308.9	75.3	7.6	1321.7	308.9	80.0	8.2
	1.1	1600.5	329.4	75.8	7.8	1321.7	308.9	80.0	8.6
	1.2	1231.5	362.2	81.3	7.6	930.1	360.4	85.9	8.4
	1.3	890.2	374.4	86.5	7.8	930.1	360.4	85.9	8.6
BPF	Base Case P_{ox} = 5222.4 μW, T_{pd} = 290.1 ns, A = 9798.2 μm^2								
	1.0	1215.8	290.1	76.7	31.6	969.8	290.1	81.4	32.4
	1.1	1184.9	310.7	77.3	31.6	969.8	290.1	81.4	32.4
	1.2	815.9	343.5	84.3	30.1	578.1	341.7	88.9	30.6
	1.3	788.3	364.0	84.9	30.1	574.7	373.9	88.9	31.2
DCT	Base Case P_{ox} = 5941.6 μW, T_{pd} = 308.9 ns, A = 8474.54 μm^2								
	1.0	1644.2	308.9	72.3	9.2	1380.3	308.9	76.7	10.5
	1.1	1589.0	308.9	73.2	33.1	1351.0	308.9	77.2	37.0
	1.2	1330.5	341.7	77.6	7.6	1351.0	308.9	77.2	8.2
	1.3	1330.5	341.7	77.6	9.2	1047.3	360.4	82.3	10.5
EWF	Base Case P_{ox} = 3895.4 μW, T_{pd} = 498.4 ns, A = 6080.0 μm^2								
	1.0	1636.2	498.4	57.9	7.0	1497.5	498.4	61.5	5.0
	1.1	1267.2	531.2	67.4	35.0	1468.2	530.6	62.3	33.2
	1.2	870.6	584.5	77.6	7.0	1076.6	582.2	72.3	8.7
	1.3	815.4	646.2	79.0	7.6	684.9	633.7	82.4	9.4
FIR	Base Case P_{ox} = 3572.9 μW, T_{pd} = 303.0 ns, A = 15845.5 μm^2								
	1.0	796.7	282.4	77.6	8.6	626.3	303.0	82.4	8.8
	1.1	769.1	323.5	78.4	8.6	626.3	303.0	82.4	8.9
	1.2	741.5	344.1	79.2	9.8	234.7	354.5	93.4	10.2
	1.3	400.1	356.3	88.7	9.7	205.3	386.8	94.2	9.3
MMV	Base Case P_{ox} = 3468.5 μW, T_{pd} = 486.0 ns, A = 7582.4 μm^2								
	1.0	957.3	486.0	72.4	7.5	804.6	486.0	76.8	10.2
	1.1	929.5	532.6	73.2	8.8	780.3	530.4	77.5	10.4
	1.2	884.3	578.2	74.5	10.2	717.9	576.2	79.3	12.6
	1.3	742.1	627.8	78.6	6.7	624.2	625.3	82.0	9.5

and has the choice of not picking the high overhead solutions.

For a comparative perspective of the firefly algorithm with the widely used integer linear programming (ILP) approach, the average results are presented for ILP in Fig. 13. The average reduction in gate leakage for the benchmark circuits is similar to the results obtained from the firefly algorithm. The ILP results are only 2 % to 5 % better than that of the firefly algorithm for both dual-T_{ox} and dual-κ approaches. However, the formu-

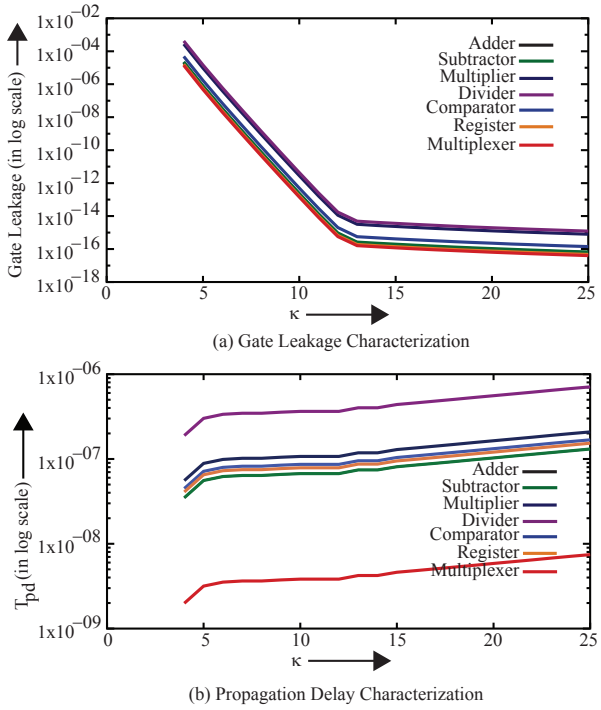


Figure 10: RTL unit characterization for different dielectrics.

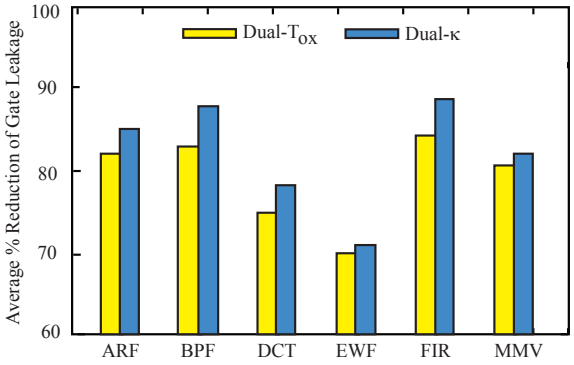


Figure 12: Average results for firefly based algorithm showing comparison of Dual-T_{ox} and Dual-κ for various RTL benchmark DSP circuits.

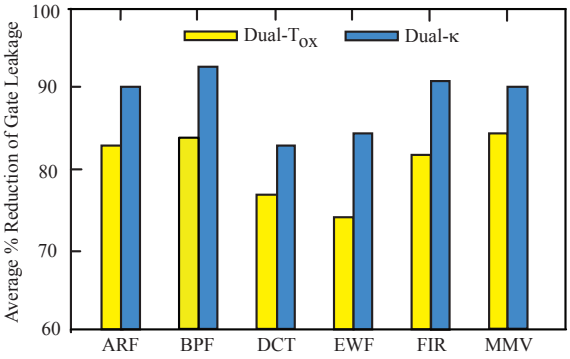


Figure 13: Average results for integer linear programming (ILP) based algorithm showing comparison of Dual-T_{ox} and Dual-κ for various RTL benchmark DSP circuits.

lation of the ILP for larger circuits is more complex than the firefly metaheuristic as ILP complexity increases exponentially with the number of vertices of the DFGs. Convergence of the ILP algorithm was an issue in many instances of resource and time constraints. On the other hand, the firefly based algorithm converges to solutions much faster, in the range of 55% to 70%, even for large size circuits with large DFGs as evident from Fig. 14. The plot represents the time of convergence in the computational platform averaged over various resource and time constraint cases used in the experiments.

The firefly based algorithm is further experimented with for different types of datapath operations as well as different datapath components. For multicycling and chaining operations of the datapath with a dual-T_{ox} approach it is observed that the gate-oxide leakage reduction ranges from 30.5% to 91.1%. For these cases the area penalty of the target architecture ranges from 4.80% to 28.67% for different time constraints. One important observation is that there is a drastic reduction in delay for multicycling and chaining operations as compared to single cycle operation. The trend of the dual-κ approach also is observed to be similar. In another set of experiments, the datapath components were characterized for a supply voltage of $V_{dd} = 0.7$ V. In this set of experiments the gate leakage reduction is lower than that of the 1 V supply voltage case. For both dual-T_{ox} and dual-κ approaches the reduction in the gate leakage decreased by 4 % to 17 %. This can be attributed to the fact that the gate-oxide leakage reduces with V_{dd}.

For a direct comparison, the results of the firefly based RTL optimization of the current paper are considered along with the results of the simulated annealing based RTL optimization [25]. The firefly based optimizations converged much faster, in the order of 40% to 60% for most of the benchmark circuits as compared to the simulated annealing based RTL optimization as observed from Fig. 14. Direct comparison results for the various benchmark circuits averaged over various resource and time constraints are presented in Table 7 for the dual-T_{ox} technique. From the results it is observed that for the same delay trade-off factor the percentage leakage reductions from the firefly based optimization as well as the simulated annealing based optimization are the same for the different benchmark circuits. However, the area penalty in the case of firefly based algo-

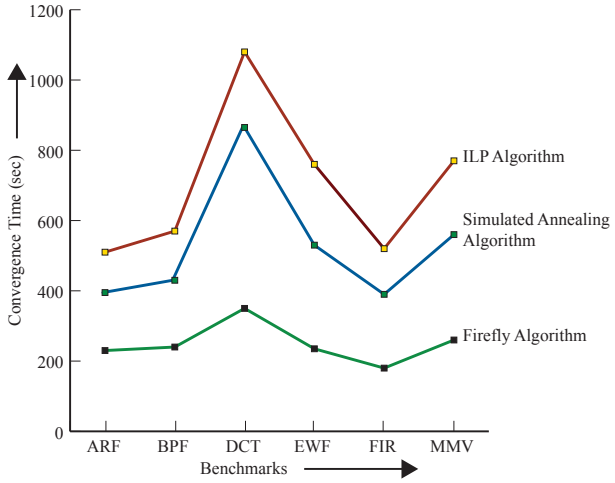


Figure 14: Average convergence time for 3 algorithms for different benchmark DSP circuits.

rithm was lower compared to the simulation annealing based optimization results: 12.7 % lower when averaged across all benchmarks. Moreover, the firefly optimization converged much faster as compared to the simulated annealing optimization for all benchmark circuits and it is 51.4 % faster when averaged across all benchmarks.

The firefly algorithm of the current paper converges faster and had leakage reduction exceeding 80 % on an average. A direct and fair comparison with prior research is difficult due to different scopes, different benchmarks, different constraints, different technology nodes, etc. Hence, a broad comparison with other related research is now presented to provide a comparative perspective. Dynamic power reduction using dual-V_{dd} techniques results in power reduction of 34 % to 46 % with 50 % delay penalty [40]. Subthreshold leakage reduction in the range of 8.4 % to 21.6 % is possible using the technique in [17]. The particle swarm optimization (PSO) in [12] does not explicitly target leakage as the current paper does. The leakage reduction in the current paper is much more significant as compared to the dual-V_{th} based leakage reduction of [8], however due to use of different benchmarks these results can not be used for a direct comparison. The current paper presents a new algorithm as compared to the heuristic algorithm of [37] and also considers area as an explicit dimension in the design exploration process. The gate-oxide leakage technique in [22] is presented at logic level as compared to RTL of the current paper. However, in terms of the percentage reduction the results are comparable. A comparison of the percentage reduction of a selected existing low-power RTL techniques is presented in Table 8. It is evident that the leakage reduction from the firefly approach far exceeds other similar research.

8. Conclusions and Future Research

The current paper presented a firefly based optimization algorithm for leakage, delay, and area trade-offs at the architectural level during high-level synthesis. The proposed approach performed scheduling, allocation, and binding to generate low leakage RTL. Both dual-T_{ox} and dual-κ approaches were explored for leakage optimization under resource and time constraints. The experimental results proved that both techniques are quite effective. The use of dual-κ is proven to be more effective than the dual-T_{ox} approach for gate-oxide reduction and trade-offs. The results obtained with the proposed RTL optimization flow outperformed other architecture level leakage reduction works available in the literature when compared in terms of percentage reduction. The leakage reduction due to dual-T_{ox} or dual-κ far exceeds the widely known dual-V_{th} technique. However, both dual-T_{ox} and dual-κ may need additional steps during the fabrication of the integrated circuit. The firefly algorithm converges faster and yields results as good as ILP even for larger circuits. It is possible to use dual-T_{ox} and dual-κ together [33]. However, the fabrication cost of such integrated circuits will be much higher as compared to dual-T_{ox} only or dual-κ only technology. At the RTL, the use of dual-T_{ox} and dual-κ together leads to a power savings increase by 10% to 20%, as compared to dual-T_{ox} only or dual-κ only technology for similar time constraints. It may be noted that the use of the dual-T_{ox} or dual-κ technology even though is considered for gate leakage optimization in the current paper may have impact on other forms of nanoscale leakage such as subthreshold leakage. Future extensions of the research of the current paper will involve total leakage power including the subthreshold leakage as well as junction leakage in addition to the gate leakage. Future optimization will account for process variations in the performance metrics of the integrated circuits at the RTL through statistical modeling. Moreover, future research of the firefly algorithm will be extended to analog circuits as well as emerging technology based digital circuits.

Acknowledgements

This research was supported in part by NSF awards CNS-0854182 and DUE-0942629. Preliminary ver-

Table 7: Direct Comparison of Firefly and Simulated Annealing Optimizations.

Bench-mark Circuits	Delay Trade-off Factor	Firefly Optimization (Current Paper)			Simulated Annealing Optimization ([25])		
		Leakage Reduction (%)	Area Penalty (%)	Running Time (Sec)	Leakage Reduction (%)	Area Penalty (%)	Running Time (Sec)
ARF	1.15	79.7	7.7	230	79.7	8.6	395
BPF	1.15	80.8	30.8	240	80.8	31.7	430
DCT	1.15	75.2	14.7	350	75.2	20.7	865
EWF	1.15	70.5	14.1	235	70.5	16.2	530
FIR	1.15	81.0	9.2	180	81.0	9.4	390
MMV	1.15	74.7	8.3	260	74.7	8.6	560

Table 8: A Broad Comparative Perspective with Existing Low-Power RTL Optimization Techniques.

Research →	Dynamic [40]		Subthreshold [17]		Gate-Oxide [37]		Gate-Oxide (Current Paper)	
Technique →	(Multi-V_{dd})		(Multi-V_{th})		(Multi-T_{ox})		(Multi-T_{ox})	
Benchmark Circuit	ΔP (%)	ΔT_{pd} (%)	ΔP (%)	ΔT_{pd} (%)	ΔP (%)	ΔT_{pd} (%)	ΔP (%)	ΔT_{pd} (%)
ARF	46.1	50.0	8.4	NA	61.7	23.0	79.7	15
EWF	35.7	50.0	19.7	NA	66.8	5.8	70.5	15
FIR	41.3	50.0	21.6	NA	63.0	12.4	81.0	15

sions of this archival journal paper have been presented in the following conferences: [25, 41].

References

[1] P. Dautriche, Analog Design Trends and Challenges in 28 and 20nm CMOS Technology, in: Proceedings of the 37th European Solid-State Circuits Conference, 2011, pp. 1–4.

[2] C. H. Diaz, K. H. Fung, Y. K. Leung, C. C. Wu, C. P. Chao, G. J. Chern, W. Lin, C. Lee, F. S. Lai, M. C. Chang, Y. C. Sun, Device Trends and Implications On Circuit Design in Advanced CMOS Technologies, in: Proceedings of the IEEE Custom Integrated Circuits Conference, 2005, pp. 675–679.

[3] T. Li, W. Zhang, Z. Yu, Full-chip leakage analysis in nano-scale technologies: Mechanisms, variation sources, and verification, in: Proceedings of the 45th ACM/IEEE Design Automation Conference, 2008, pp. 594–599.

[4] A. Agarwal, S. Mukhopadyaya, A. Roychowdhury, K. Roy, C. H. Kim, Leakage Power Analysis and Reduction for Nanoscale Circuits, IEEE Micro 26 (2) (2006) 68–80.

[5] S. P. Mohanty, M. Gomathisankaran, E. Kougianos, Variability-Aware Architecture Level Optimization Techniques for Robust Nanoscale Chip Design, Computers & Electrical Engineering 40 (1) (2014) 168–193.

[6] S. P. Mohanty, Unified Challenges in Nano-CMOS High-Level Synthesis, in: Proceedings of the 22nd International Conference on VLSI Design, 2009, pp. 531–531.

[7] K. Roy, S. Mukhopadhyay, H. M. Meimand, Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits, Proceedings of the IEEE 91 (2) (2003) 305–327.

[8] Y. Chen, Y. Xie, Y. Wang, A. Takach, Minimizing Leakage Power in Aging-bounded High-level Synthesis with Design Time Multi-V_{th} Assignment, in: Proceedings of the Asia and South Pacific Design Automation Conference, 2010, pp. 689–694.

[9] D. S. H. Ram, M. C. Bhuvaneswari, S. M. Logesh, A Novel Evolutionary Technique for Multi-objective Power, Area and Delay Optimization in High Level Synthesis of Datapaths, in: Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2011, pp. 290–295.

[10] F. C. C. Mei, S. Phon-Amnuaisuk, M. Y. Alias, P. W. Leong, Adaptive GA: An Essential Ingredient in High-Level Synthesis, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2008, pp. 3837–3844.

[11] V. Krishnan, S. Katkoori, A Genetic Algorithm For The Design Space Exploration of Datapaths During High-Level Synthesis, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 213–229.

[12] A. Sengupta, V. K. Mishra, Integrated Particle Swarm Optimization (i-PSO): An Adaptive Design Space Exploration Framework for Power-Performance Tradeoff in Architectural Synthesis, in: Proceedings of the 15th International Symposium on Quality Electronic Design, 2014, pp. 60–67.

[13] C. Hao, S. Chen, T. Yoshimura, Network Simplex Method Based Multiple Voltage Scheduling in Power-Efficient High-Level Synthesis, in: Proceedings 18th Asia and South Pacific Design Automation Conference, 2013, pp. 237–242.

[14] C. Ping-Yuan, Y. Chien-Cheng, A Voltage Level Converter Circuit Design with Low Power Consumption, in: Proceedings of the 6th International Conference on ASIC, 2005, pp. 358–359.

[15] S. H. Kulkarni, D. Sylvester, High Performance level Conversion for Dual VDD Design, IEEE Transactions on VLSI Systems 12 (9) (2004) 926–936.

[16] X. Tang, H. Zhou, P. Banerjee, Leakage power optimization with dual-V_{th} library in high-level synthesis, in: Proceedings of the 42nd annual conference on Design automation, 2005, pp. 202–207.

[17] C. Gopalakrishnan, S. Katkoori, Knapbind: An Area-Efficient Binding Algorithm for Low-Leakage Datapaths, in: Proceedings of 21st International Conference on Computer Design, 2003, pp. 430–435.

[18] R. M. Rao, J. L. Burns, R. B. Brown, Circuit Techniques for Gate and Sub-Threshold Leakage Minimization in Future CMOS Technologies, in: European Solid-State Circuits Conference, 2003, pp. 313–316.

[19] K. S. Khouri, N. K. Jha, Leakage power analysis and reduction during behavioral synthesis, IEEE Transactions on VLSI Systems 10 (6) (2002) 876–885.

[20] D. Dal, N. Mansouri, Power Optimization With Power Islands Synthesis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28 (7) (2009) 1025–1037.

[21] D. Helms, O. Meyer, M. Hoyer, W. Nebel, Voltage- and ABB-Island Optimization In High Level Synthesis, in: Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2007, pp. 153–158.

[22] S. Yang, H. Wang, Z. jia Yang, Low Leakage Dynamic Circuits With Dual Threshold Voltages and Dual Gate Oxide Thickness, in: Proceedings of the 7th International Conference on ASIC, 2007, pp. 70–73.

[23] N. Sirisantana, K. Roy, Low-power Design using Multiple Channel Lengths and Oxide Thicknesses, IEEE Design and Test of Computers 21 (1) (2004) 56–63.

[24] D. Lee, D. Blaauw, D. Sylvester, Gate Oxide Leakage Current Analysis and Reduction for VLSI Circuits, IEEE Transactions on VLSI Systems 12 (2) (2004) 155–166.

[25] S. P. Mohanty, R. Velagapudi, E. Kougianos, Dual-K Versus Dual-T Technique for Gate Leakage Reduction: A Comparative Perspective, in: Proceedings of the 7th International Symposium on Quality of Electronic Design, 2006, pp. 564–569.

[26] S. Mohanty, N. Ranganathan, E. Kougianos, P. Patra, Low-Power High-Level Synthesis for Nanoscale CMOS Circuits, Springer, 2008.

[27] S. P. Mohanty, N. Ranganathan, A Framework for Energy and Transient Power Reduction during Behavioral Synthesis, IEEE Transactions on VLSI Systems 12 (6) (2004) 562–572.

[28] R. Rao, A. Srivastava, D. Blaauw, D. Sylvester, Statistical Analysis of Subthreshold Leakage Current for VLSI Circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 12 (2) (2004) 131–139.

[29] S. Arora, S. Singh, A Conceptual Comparison of Firefly Algorithm, Bat Algorithm and Cuckoo Search, in: Proceedings of the International Conference on Control Computing Communication Materials (ICCCCM), 2013, pp. 1–4.

[30] D. M. Munoz, C. H. Llanos, L. Dos Santos Coelho, M. Ayala-Rincon, Hardware-Based Parallel Firefly Algorithm For Embedded Applications, in: Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2013, pp. 39–46.

[31] X. S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.

[32] P. Kaur, T. Kaur, A Comparative Study of Various Metaheuristic Algorithms, International Journal of Computer Science and Information Technologies 5 (5) (2014) 6701–6704.

[33] V. Mukherjee, S. P. Mohanty, E. Kougianos, A Dual Dielectric Approach for Performance Aware Gate Tunneling Reduction in Combinational Circuits, in: Proceedings of the 23rd IEEE International Conference of Computer Design (ICCD), 2005, pp. 431–436.

[34] A. Raghunathan, N. K. Jha, S. Dey, High-Level Power Analysis and Optimization, Springer US, 1997.

[35] A. K. Sultania, D. Sylvester, S. S. Sapatnekar, Tradeoffs Between Gate Oxide Leakage and Delay for Dual T_{ox} Circuits, in: Proceedings of Design Automation Conference, 2004, pp. 761–766.

[36] A. Smith, Pennsylvania Firefly, `https://bioweb.uwlax.edu/BIO203/2011/smith_ash2/index.htm`, last Accessed on 30 Aug 2014 (2011).

[37] S. P. Mohanty, E. Kougianos, D. K. Pradhan, Simultaneous Scheduling and Binding for Low Gate Leakage Nano-Complementary Metal-Oxide-Semiconductor Datapath Circuit Behavioural Synthesis, IET Computers Digital Techniques 2 (2) (2008) 118–131.

[38] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, C. Hu, New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design, in: Proceedings of the IEEE Custom Integrated Circuits Conference, 2000, pp. 201–204.

[39] K. A. Bowman, L. Wang, X. Tang, J. D. Meindl, A Circuit-Level Perspective of the Optimum Gate Oxide Thickness, IEEE Transactions on Electron Devices 48 (8) (2001) 1800–1810.

[40] A. Manzak, C. Chakrabarti, A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages, IEEE Transactions on VLSI Systems 10 (1) (2002) 6–14.

[41] S. P. Mohanty, ILP Based Gate Leakage Optimization Using DKCMOS Library during RTL Synthesis, in: Proceedings of the 9th International Symposium on Quality of Electronic Design, 2008, pp. 174–177.