RELIABILITY AND ENERGY OPTIMIZATION IN HIGH LEVEL SYNTHESIS OF INTEGRATED CIRCUITS

RAWAN SMRI

Prof. Dr. SÜLEYMAN TOSUN

Supervisor

Submitted to Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering

January 2020

”For my mother and father

Ahlam and Adel SMRI”

ABSTRACT

RELIABILITY AND ENERGY OPTIMIZATION IN HIGH LEVEL SYNTHESIS OF INTEGRATED CIRCUITS

Rawan SMRI

Master of Science, Computer Engineering Department

Supervisor: Prof. Dr. Süleyman TOSUN

January 2020, 60 pages

Ever-increasing performance demand for the computer applications has resulted in shrinking the technology sizes of the CMOS circuits over the past 50 years, which made it possible to increase the number of transistors on a single chip. On the other hand, the increase in circuit densities makes the design process more challenging. For example, circuits become more vulnerable to radiation effects due to lower supply and threshold voltage levels; thus, the number of transient faults in circuits increases. While a reduced technology size makes circuits more susceptible to transient faults, some energy reduction techniques also negatively affect their reliability. Traditional high level synthesis (HLS) methods usually consider only area and latency along with either energy or reliability. To the best of our knowledge, there is no prior work that takes area and latency as constraints and energy and reliability as optimization parameters. Especially, the effect of DVS on reliability is completely ignored by the previous studies. In this work, we aim to develop new HLS methods for application specific integrated circuit (ASIC) design under area and timing constraints with the objectives of low energy consumption and high reliability. For the mapping and scheduling steps of HLS, we propose genetic algorithm (GA)-based optimization method, and also use a selective duplication method. And for comparison purposes we introduced integer linear programming

(ILP) method. While the ILP-based method determines the optimum results, the CPU time exponentially increases with the number of the application nodes. Therefore, we propose a GA-based metaheuristic that is faster and determines optimum or near-optimum results in shorter times than ILP. In addition, we characterize a resource library consisting of three adders and two multipliers with varying area, delay, energy, and reliability parameters under two voltage levels

Keywords: High-Level Synthesis (HLS), Dynamic Voltage Scaling (DVS), reliability, soft errors, energy.

ÖZET

ENTEGRE DEVRELERİN YÜKSEK SEVİYESİNDE GÜVENİLİRLİK VE ENERJİ OPTİMİZASYONU

Rawan SMRI

Yüksek Lisans, Bilgisayar Mühendisliği

Danışman: Prof. Dr.Süleyman TOSUN

Ocak 2020, 60 sayfa

Bilgisayar uygulamalarına yönelik artan performans talebi, CMOS devrelerinin teknoloji boyutlarının son 50 yılda azalmasına neden oldu ve bu da tek bir yonga üzerindeki transistörlerin sayısını artırmayı mümkün kıldı. Öte yandan, devre boyutlarındaki artış tasarım sürecini daha da zorlaştırmaktadır. Örneğin, devreler, daha düşük besleme ve eşik voltaj seviyeleri nedeniyle radyasyon etkilerine neden olur; böylece devrelerdeki geçici hataların sayısı artar. Azalan teknoloji boyutu devreleri geçici arızalara karşı daha hassas hale getirirken, bazı enerji azaltma teknikleri de güvenilirliklerini olumsuz yönde etkilemektedir. Geleneksel yüksek seviyeli sentez (HLS) yöntemleri genellikle enerji ve güvenilirliğin yanı sıra onearea ve gecikmeyi de dikkate alır. Bildiğimiz kadarıyla, alan ve gecikmeyi kısıtlayıcı, enerji ve güvenilirliği op-enimizasyon parametreleri olarak alan önceki bir çalışma yoktur. Özellikle DVS'nin güvenilirlik üzerindeki etkisi önceki çalışmalarda tamamen göz ardı edilmektedir. Bu çalışmada, düşük enerji tüketimi ve yüksek güvenilirlik hedefleri ile alan ve zamanlama kısıtlamaları altında özel entegre devre (ASIC) tasarımı uygulaması için yeni HLS yöntemleri geliştirmeyi hedefliyoruz. HLS'nin haritalama ve çizelgeleme adımları için genetik algoritma (GA) tabanlı optimizasyon yöntemi öneriyoruz ve ayrıca seçici bir du-plication yöntemi kullanıyoruz. Karşılaştırma amacıyla tamsayı doğrusal programlama

(ILP) yöntemini tanıttık. ILP tabanlı yöntem optimum sonuçları belirlerken, CPU uygulama düğümlerinin sayısıyla birlikte katlanarak artar. Bu nedenle, ILP'den daha hızlı ve optimum veya optimumya yakın sonuçları daha kısa sürede belirleyen aGA tabanlı bir metaheuristic öneriyoruz. Buna ek olarak, üç kademeli ve değişken alan, gecikme, enerji ve güvenilirlik parametrelerine sahip iki çarpandan oluşan gerilim seviyelerinin altındaki kaynak kütüphanesini karakterize ediyoruz.

Anahtar Kelimeler: Yüksek Seviyeli Sentez (HLS), Dinamik Gerilim Ölçekleme (DVS), güvenilirlik, yumuşak başlıklar, enerji

ACKNOWLEDGEMENTS

First, I would like to sincerely thank my supervisor **Prof. Dr. Süleyman TOSUN** for his time, patience and for his valuable guidance in every stage of my research all along this long way. Additionally, I would like to thank my colleague **Ms. Selma DILEK** for writing and implementing the ILP method part and **Assist. Prof. Dr. Deniz DAL** for his helping in library characterization of arithmetic circuits.

Finally, I would express the most gratitude to my husband **Tariq ASI**, the one who did not hesitate to show his support and compassion to me all the time, to my beloved baby girl **Meryem** who had to live an atypical life with a student mom, to **Omar** for his unlimited companionship , to my parents **Ahlam** and **Adel**, to my brothers **Mohammad** and **Yazeed** and my friend and sister **Aseel** last but not least ,can not forget my Turkish mother **Gülsüm NALINCI** for her help through my staying in Turkey, and lastly for my and mother in low **Jamal** and **Ibtisam**. Because they all truly encouraged , believed in me and taught me what unconditional love means.

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under project number 116E095.

Contents

Page

FIGURES

Page

TABLES

1. INTRODUCTION

Ever increasing performance demand for the computer applications has resulted in shrinking the technology sizes of the complementary metal-oxide-semiconductor (CMOS) circuits every 18 months over the past 50 years driven by the Moore's Law. Shrinking the technology sizes made it possible to increase the number of transistors on chips, thus allowing the designers to embed more components in their designs than before. While the smaller transistor sizes reduce the cost of the integrated circuits as a result of having a smaller chip area, the increase in the circuit densities makes the design process more challenging. Furthermore, each technology generation also introduces new design problems in digital systems. For example, when the technology sizes are reduced, circuits become more vulnerable to radiation effects due to lower supply and threshold voltage levels; therefore, the number of the transient faults in circuits increases [3]. Figure 1.1, adopted from [1], shows how the soft error rates (SERs) of sequential circuits (SRAM and latches) and combinational logic with different sizes change with each technology generation. While sequential elements still has high SER rates by keeping almost the same values, the SER of combinational logic increases dramatically. Therefore, tackling the soft error (SE) problem of combinational circuits has also become a major concern. Although combinational circuits can mask the transient errors to some extend, they cannot eliminate them completely without some extra precautions. Thus, new design methods for mitigating them before they are latched to memory elements are crucial.

While a reduced technology size makes the circuits more susceptible to transient faults, some energy reduction techniques also negatively affect their reliability. For example, when dynamic voltage scaling (DVS) is applied as an energy reduction method, a circuit consumes less energy under lower voltage levels; however, lowering the supply voltage also reduces the reliability of the circuit [4, 5]. Furthermore, when we consider the design of an application with large number of components, tackling all system requirements such as area, performance, energy consumption, and reliability becomes cumbersome. Therefore, a design automation tool is a must to ease the design process and to determine the best design in terms of the given objective function and the constraints. Generally, it is much more practical and efficient to tackle several constraints and optimization parameters at higher levels of abstraction for designing Application Specific Integrated Circuits (ASICs). High-level synthesis (HLS) process aims to integrate all system requirements on higher level of abstraction

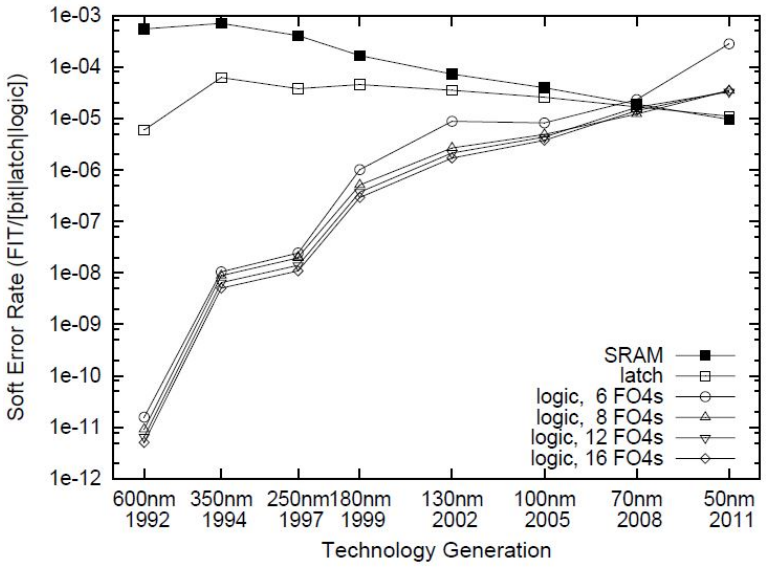


FIGURE 1.1: Predicted Soft Error rates of sequential and combinational logic under different technology sizes [1].

and shields the designer from lower level design burdens [6].

Traditional HLS methods usually consider only area and latency together with either energy [7] or reliability [8]. To the best of our knowledge, there is no prior work that takes area and latency as constraints and energy and reliability as optimization parameters. Especially, the effect of DVS on reliability is completely ignored by the previous studies. In this work, we aim to develop new HLS methods for ASIC design under area and timing constraints with objectives of low energy consumption and high reliability. In our work, we use different versions of the same resources in terms of area, performance, energy, and reliability. For this purpose, we implemented several adders and multiplier circuits to utilize in the design of the given application. For our optimization function with two parameters, we blend the energy and reliability values by assigning weights to each of them in order to be able to handle our multi-optimization problem. For the mapping and scheduling steps of the HLS, we use Genetic Algorithm (GA) based optimization methods, and compare it to Integer Linear Programming (ILP) method. While the ILP-based method determines the optimum results, it takes too much time for some problems consisting of large number of variables. Therefore, we propose a GA-based metaheuristic that determines optimum or near-optimum results in a reasonable amount of time.

We can summarize the contributions of this work as follows:

- We characterize a resource library with three adders and two multipliers under varying area, delay, energy, and reliability parameters. We list the same resource parameters under two voltage levels. We believe that our resource library will also be useful for future HLS studies.

- We present a GA-based metaheuristic method for mapping and scheduling steps of our HLS design flow. Our GA-based method obtains optimal or near-optimal results for most of the test instances in very short run-times, even for very large-sized applications. The strength of our GA-based method comes from the intelligent mutation and cross-over operators that diversify the solution population.

- We show that there is still a room for the reliability improvement after the mapping and scheduling steps are completed, and use a selective duplication method in this respect.

- We illustrate the effectiveness of GA-based methods on several benchmarks in terms of energy and reliability by conducting rigorous experimental analysis, and compare it to integer linear programming (ILP) based method.

The rest of this work are organized as follows. Related work is presented in the next Chapter. We explain SEs, DVS, and effects of DVS on SEs in Chapter 3. In Chapter 4.0, we introduce our library characterization and the problem definition. We present our GA-based method in Chapter 5. In section 5.5.0, we explain the duplication method for further maximization of the reliability. In Chapter 6.0 we introduced the ILP Formulations for comparing purpose. We illustrate the effectiveness of the proposed methods by discussing the experimental results in Chapter 7. We finally conclude this paper in Chapter 8.

2. RELATED WORK

There have been several HLS-related studies in the literature [9]. Earlier publications usually focus on latency and area as constraints and/or objective functions [6]. In this study, we incorporate energy and reliability metrics into the HLS process unlike the previous studies that only focus on one of these metrics along with area and latency. In the following subsections, the related studies are reviewed according to their field of concern.

2.1. Reliability-Aware HLS

Reliability was treated as a first-class citizen in a very old work under the fault-tolerance criteria for HLS designs in [10]. This work aimed to design circuits under area and performance constraints to maximize the fault-tolerance by adding extra duplicated resources. Some other studies took advantage of the fact that the reliability of different implementations of the same function may be different due to their internal logic and masking capabilities. In addition, their area and latency values are also different. Optimization of a circuit by using these different resources is known as an NP-hard problem, therefore a heuristic method was proposed in [2] . There have also been metaheuristic attempts for optimizing the reliability using different versions of a particular resource [11]. Some prior studies also presented HLS methods to design fault-tolerant data-paths in case of multi-cycle transient faults [12]. Authors of [13] presented a simulation-based method for combinational circuit synthesis considering soft errors. Reliability-aware resource allocation and binding in HLS is an NP-hard optimization problem. There have been several HLS-related studies in the literature [9]. Earlier publications usually focus on latency and area as constraints and/or objective functions [6]. In this study, we incorporate energy and reliability metrics into the HLS process unlike the previous studies that only focus on one of these metrics along with area and latency. In the following subsections, the related studies are reviewed according to their field of concern.

2.2. Energy-Aware HLS

Dynamic voltage scaling (DVS) has been the most commonly used energy consumption minimization method since it was introduced by [14] and [15]. Since the dynamic energy consumption decreases proportionally with the square of the voltage level, many commercial

CPUs are implemented with this in mind, and new scheduling methods for varying voltage level assignments have been proposed for these new architectures [16–19]. However, the target platform for these studies are either homogeneous or heterogeneous multiprocessor systems unlike our ASIC design platform. There are also prior studies in the context of HLS focusing on energy or power consumption [20–23]. [20] presents a novel scheduling algorithm for minimizing energy, while [21] and [22] propose new methods to reduce the power consumption of circuits. There are even game theoretical scheduling algorithms using DVS for HLS [23]. The interested readers can find several methods about low power HLS design for nanoscale CMOS circuits in [24].

2.3. Energy- and Reliability-Aware Design

There has been some prior research that focused on both energy and reliability simultaneously [25–28]. However, none of the existing studies incorporate these metrics in the HLS steps. Additionally, their target platforms are multiprocessor systems rather than ASICs. To the best of our knowledge, there has not been any previous research in the HLS field that considered area, performance, reliability, and energy all together in a single study. Most of the prior research discussed the methods of increasing the system reliability under the area and latency constraints without considering the energy consumption of a system, or they suggested approaches to minimize the energy consumption of a system ignoring the effect of those approaches on the reliability.

3. BACKGROUND

In the following subsections, we first discuss soft errors and reliability in digital systems. We then explain DVS, which is used to minimize the energy consumption. We finally present the effects of DVS on reliability of the circuits.

3.1. Soft Errors and Reliability

Function values in digital systems are generated as a result of switching in transistors. These physical elements can be affected by a number of external factors, which can cause undesirable switching situations that may lead to wrong results. If these non-persistent errors cause the data stored in memory to be erroneous even for a short period of time, all operations using that data will yield erroneous results until the data is updated. These errors in digital systems are called transient errors or soft errors (SEs).

A soft error is a signal fluctuation or an unexpected bit flip in semiconductor fabrics, which may occur due to radiation, alpha particles, or high-energy cosmic rays in the environment of the device containing the digital system. These errors generally do not corrupt the device; however, they may result in malfunctions. They usually occur when the energy accumulated in a transistor (Q) exceeds the critical energy ($Q_{critical}$). Figure 3.1 shows the occurrence of transient errors in the silicon view and the transistor view.

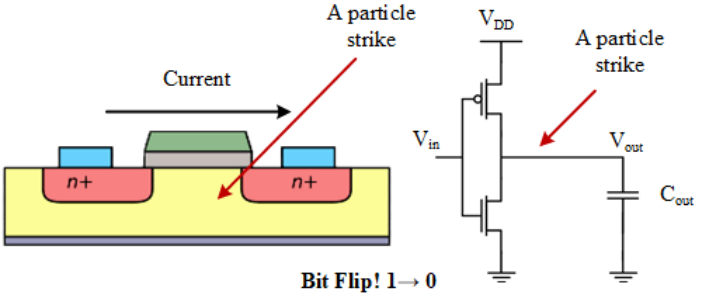


FIGURE 3.1: Occurrence of SEs: silicon view (left) and transistor view (right) (adapted from [2]).

As the technology size decreases and the chip circuit densities increase, SERs increase significantly, particularly in combinational circuits as shown in Figure 1.1. This increase in

SERs negatively affects the reliability of a running system during its operation. Hence, it has become inevitable to consider the effects of transient errors during the design process.

The reliability of a system can be calculated with the formula given in Equation (1), where λ is SER, while t is the running time of the system.

$$R(t) = e^{-\lambda t} \tag{1}$$

From the equation it is evident that the higher the SER value the lower the reliability. One way to improve the reliability of a system is to back up its components (i.e., to create a replica.) If two different versions of the system produce two different outputs, the result is incorrect. In such case, the system can be restarted or, alternatively, checkpoints can be added to the system to avoid the necessity for a complete restart. When a replica of a thread is created, the increased reliability value is calculated with the formula given in Equation (2), where R_s represents the total reliability after duplication, while R_i and $R_{i'}$ represent the reliability values of a system component and its replica, respectively.

$$R_s = R_i + R_{i'} - R_i R_{i'} \tag{2}$$

There are a lot of studies that focus on using replicas or multiple spare circuit elements to increase the reliability of the systems with multiple circuit elements. These backups are usually selected to be the same as the original ones. However, circuit elements implemented in a different manner exhibit different behavior against transient errors. For instance, an adder circuit with a larger but faster operating area may have lower reliability values than a smaller but slower one. A previous study showed that the SER values of circuit elements implemented in different ways can also be different, and that by taking this into consideration during the design process the reliability of a system can be affected [2]. Nonetheless, in the design of integrated circuits, the energy consumption of a circuit is also an important criteria along with reliability, area, and performance constraints. We also incorporate duplication in our final design to further increase its reliability without increasing its area, which will be explained in Section 5.5..

3.2. Dynamic Voltage Scaling

Performance (runtime of an application) is the most important requirement that needs to be achieved for ASICs. Furthermore, while the energy consumption plays a significant role, especially in battery-powered systems, the reliability comes to the fore in critical applications such as rocket control circuits, satellites, and nuclear power plant control circuits. Therefore, when designing such systems, it should be ensured that they meet the given time and energy constraints, while maximizing the reliability.

DVS was introduced in 1996 by [14], and since then it has become the most popular method for reducing the energy consumption in digital circuits. The reason behind the widely adoption of this method relies on the fact that when digital circuits operate under a low voltage, their energy consumption decreases in proportion to the square of the voltage, while the worst case execution time (WCET) only increases proportionally to the decrease in voltage. If the operation time requirement for a digital circuit is sufficient for the application of DVS, the circuit may be operated at a lower voltage to reduce the overall energy consumption. Modern digital circuits can be designed to operate at multiple voltage levels, allowing for the implementation of the DVS method. The effect of DVS on a circuit's energy and performance can be explained by the power consumption of a CMOS circuit. The dynamic power consumption of CMOS circuits is expressed by the Equation (3), where P is the power consumption, C_L is the load capacitance, N_s is the number of switching cycles per hour, v is the source voltage, and f is the operating frequency of the circuit.

$$P = C_L N_s v^2 f \tag{3}$$

If the source voltage of the circuit is reduced, the execution time of the circuit will also change proportionally according to the Equation (4), where k and α are constants varying based on the applied technology dimensions, and v_t is the threshold voltage value.

$$t = C_L v / k(v - v_t)^{\alpha} \tag{4}$$

If the WCET of a digital circuit under high voltage (v_h) is known, the WCET value under low voltage (v_l) can be calculated from the Equation (5) derived from the Equation (4).

$$t_{v_l} = t_{v_h}(\frac{v_l}{v_h})(\frac{v_h - v_t}{v_l - v_t})^2 \tag{5}$$

Similarly, if the high-voltage energy consumption (E_{v_h}) of the circuit is known, the low-voltage energy consumption (E_{v_l}) can be calculated using the Equation (6).

$$E_{v_l} = E_{v_h}(\frac{v_l}{v_h})^2 \tag{6}$$

Although most of the previous studies have adopted DVS as main energy minimization method, they have only considered the negative effects of DVS on performance, while neglecting its negative impact on the reliability. One of the most unique contributions of this study is the analysis of the effect of different voltage levels on the reliability of a system when employing the DVS.

3.3. Effects of DVS on Reliability

DVS is a very efficient technique when it comes to reducing the energy consumption. On the other hand, when a digital circuit operates at a low voltage, it becomes more vulnerable to soft errors since $Q_{critical}$ values of the transistors can be more easily exceeded, and as a result, the total reliability of the system decreases. In other words, when we use the DVS technique to reduce the energy consumption, lowering the operating voltage of a digital circuit (and therefore its frequency as well) will lead to both an increase in its execution time and a decrease in its reliability.

The fault rate of a system at frequency f (voltage v) is expressed by means of the Equation 7, where λ refers to the SER, and λ_0 refers to the average error rate at frequency f.

$$\lambda(f) = \lambda_0 g(f) \tag{7}$$

Let the operating frequency at the highest operating voltage (v_{max}) be $f_{max} = 1$. Transient errors generally occur when the critical voltage of the circuit is reached. This critical voltage is proportional to the system voltage. That is, when the system voltage is reduced, the critical threshold voltage will also decrease. Thus, at low voltages, the circuit will be more sensitive

to soft errors. Error rates according to the voltage changes can be calculated by the Equation (8).

$$\lambda(f) = \lambda_0 10^{\frac{d(1-f)}{(1-f_{min})}} \tag{8}$$

Here, the maximum error rate is expressed as $\lambda(f_{max}) = \lambda_0 10^d$, which is the minimum operating frequency. $d > 0$ is a constant. The higher the value of d the higher the error rate in the circuit (i.e., the lower the value of d the more resistant the circuit to faults.)

Using the Equation (8), the new reliability values can be calculated according to the changing energy levels and the execution time of a digital circuit. In this study, we consider the effect of DVS on reliability, the energy consumption, and the latency of digital circuits at different voltage levels.

4. LIBRARY CHARACTERIZATION AND PROBLEM DEFINITION

4.1. Library Characterization

A function can be implemented in multiple ways in the hardware using different design methods, which produce several different versions of the same function. For example, an adder can be implemented as a ripple-carry adder, carry-lookahead adder, prefix adder etc. [29]. Different implementations of the same function may have different area, latency, and energy consumption values. Additionally, they can exhibit diverse behavior in terms of the error resilience when a soft error hits a part of the circuit. Some circuits can tolerate faults better than others since their transistor layouts and logic functions are different. This is due to the fact that each combinational circuit has fault masking capabilities to some extent. Therefore, different versions of the same function may have different reliability values in addition to the area, latency, and energy consumption. The HLS methods proposed in this thesis utilize different versions of the same resource in an attempt to find the optimum energy and reliability values under given latency and area constraints for a given application. In this respect, a resource library is characterized to be employed in the proposed methods.

We implemented three adders and two multipliers in Verilog and synthesized them using Cadence Genus synthesis tool [30]. We obtained the area, latency, and energy values for each resource under 1.2V supply and 0.5V threshold voltage levels. We then scaled each parameter. We used the reliability values estimated in [8] for these resources. Finally, we obtained the new latency, energy, and reliability values for 1.0V supply and 0.5V threshold voltage levels using the Equations (6), (5), and (8), respectively. The details of the resource library after the characterization are given in Table 4.1. In this table, A is the area of the resource measured in mm^2. L_h and L_l represent the latency values of the corresponding resources under high and low voltage respectively, and they are measured in time steps. Similarly, R_h and R_l represent the reliability values, whereas E_h and E_l represent the energy consumption under high and low voltage measured in nanojoules, respectively (nJ).

TABLE 4.1: Resource Library.

Type	Resource Name	A	$\mathbf{L}_h$ / $\mathbf{L}_l$	$\mathbf{R}_h$ / $\mathbf{R}_l$	$\mathbf{E}_h$ / $\mathbf{E}_l$
Adder (A1)	Ripple Carry	2	5 / 8	0.999 / 0.998	12.00 / 8.33
Adder (A2)	Brent Kung	3	3 / 5	0.969 / 0.938	5.00 / 3.47
Adder (A3)	Kogge Stone	5	2 / 3	0.987 / 0.976	6.00 / 4.17
Multiplier (M1)	Carry Save	8	10 / 16	0.999 / 0.998	80.00 / 55.56
Multiplier (M2)	Carry Lookahead	12	15 / 25	0.969 / 0.938	160.00 / 111.11

4.2. Problem Definition

The aim of this study is to propose HLS methods for the resource allocation and scheduling steps to maximize the reliability and minimize the energy consumption of the final design under the given latency and area constraints. HLS is an automated design process that converts a given behavioral description of an application into a synthesized hardware. A behavioral description can be written in a high-level language and it is converted to a data flow representation in the form of directed acyclic graph (DAG) before the HLS process starts. In Figure 4.1, we show the behavioral model for the differential equation solver, its data flow representation with the precedence constraints, and the final DAG adopted from [6]. The first and last dummy nodes (*source* and *sink* nodes) are added to the DAG as reference points to ease the implementation of the scheduling algorithms.

The goal of the resource allocation is to assign a resource from the resource library to each node of the DAG while taking the area constraint into consideration whereas a scheduling algorithm assigns the start times for the each node of the DAG under latency constraints. The objective here is to minimize the total energy consumption and maximize the reliability of the final design.

There are several challenges to this problem, which make its optimization a very cumbersome task. First of all, we have a variety of possible resources (i.e. functional units) with different reliability, area, latency, and energy values that need to be taken into consideration simultaneously in the process of resource allocation and scheduling. Additionally, scheduling must take the task dependencies into account, so that dependent tasks will execute in

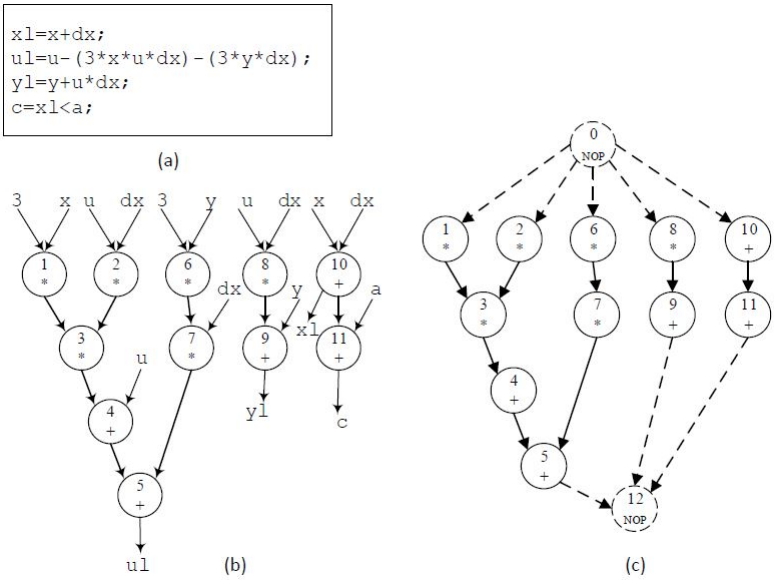


FIGURE 4.1: (a) An example design specification, (b) Data flow representation with precedence constraints, and (c) Directed Acyclic Graph (DAG) of the design specification.

the required order, and no precedence constraint is violated. Finally, different voltage levels assigned to different resources (DVS) introduces further complication into the model. We propose GA-based method to solve such a problem and compare it with ILP method in order to show the accuracy and execution times of different optimization methods.

5. GA-BASED METHOD

In the past decade, genetic algorithms have evolved as an optimization technique that is more practical, time saving and efficient compared to other traditional optimization techniques. GAs are categorized as metaheuristic methods that obtain optimum or near-optimum solutions in a reasonable amoun of time. They search the whole solution space randomly via the genetic operators; hence, they reduce the chance of trapping of local minima. In our work, we propose a GA-based method for finding the most reliable and concurrently the least energy consuming solution for our HLS problem. The proposed method follows the three main stages of GAs: population generation, applying genetic operators, and selection based on the fitness function. These stages are explained in more detail in the following subsections.

5.1. Population Generation

The very first step in a genetic algorithm is to generate an initial population of solutions to the problem. Individuals of the population could be initialized either totally at random or heuristically to a certain extent. In our study, the population is created totally at random with a size of 100 individual. We adopt the chromosome representation to symbolize a solution. Figure 5.1 shows the chromosome representation of the DES graph given in Figure 4.1, with the chosen voltage level under each resource (1 represents high while 0 represents the low voltage level).

Node	1	2	3	4	5	6	7	8	9	10	11
Resources	M1	M1	M2	A2	A2	M1	M2	M1	A2	A3	A2
Voltage Level	1	1	1	1	1	1	1	0	1	1	1

A	L	R	E
?	?	?	?

FIGURE 5.1: A chromosome symbolization of a solution for the DES benchmark after resource assignment, with its Area (A), Latency (L), Reliability (R) and Energy (E) values not calculated yet.

A chromosome is an array of genes, where each gene's value represents its randomly assigned resource from our resource library in Table 4.1. The Area (A), Latency (L), Reliability (R), and Energy (E) of this solution are not calculated before the scheduling and resource assignment stages. This is denoted by question marks in their related cells.

5.2. Scheduling and Binding

The second stage of our GA is the calculation of the fitness value of each solution, with the goal of keeping the best individuals for the next generation. Consequently, the proposed GA first applies the HLS step to all solutions, and then calculates their fitness values (i.e. the reliability of each solution). The well-known List Scheduling (LS) [6] is performed in two steps: first, the mobility of each node/gene in the chromosome is determined, and then the resource binding is carried out. The mobility (m) of a node (i) is the time difference between the earliest and latest time steps that a node can be assigned. The earliest time step ($t(i)^{ASAP}$) and the latest time step ($t(i)^{ALAP}$) are determined using the As Soon As Possible (ASAP) and As Late As Possible (ALAP) scheduling algorithms based on the resource assignment of the chromosome. We then use Equation 9 to calculate the mobility of each node.

$$m(i) = t(i)^{ALAP} - t(i)^{ASAP} \tag{9}$$

The LS algorithm does not change the nodes with zero mobility (critical path nodes), even if more than one node on different critical paths need to use the same resource at the same time, which results in higher area values. We adjusted the LS results to allow the critical path nodes to share resources with other nodes; hence, the total area is decreased. The change was made by adding only one extra time step to the critical path. This approach slightly increases the latency, but at the same time it significantly reduces the area.

ASAP scheduling identifies the minimum latency that can be obtained with the assigned resources. In ASAP, every node is scheduled at the earliest time step possible. The overall minimum latency is the time step at which the last node of the chromosome is scheduled. ALAP scheduling does the opposite of ASAP as it returns the starting time steps for every node in the solution with the maximum possible latency. This is done by comparing the latency constraint with the latency returned from ASAP, and then using the higher of both.

The ASAP and ALAP scheduling for the chromosome in Figure 5.1 are shown in Figure 5.2. In this figure, each dashed horizontal line shows the starting steps of some nodes. We do not draw each step in our scheduling figures to prevent overcrowding the illustration; instead, we only show the time steps if there is a node starting its execution in these time steps. To explain the GA operators on our running example, we assume the area ($\bar{A}$) and latency ($\bar{L}$)

constraints as 23 units and 30 time steps, respectively. The minimum latency returned from ASAP is 31, which is calculated by subtracting 1 from the starting step of the sink node (i.e., $L_{min} = t(n)^{ASAP} - 1$). Then, after applying the ALAP scheduling under $L = 31$ and determining the ALAP starting time step of each node, the algorithm calculates the mobility of each node as shown in Table 5.1.

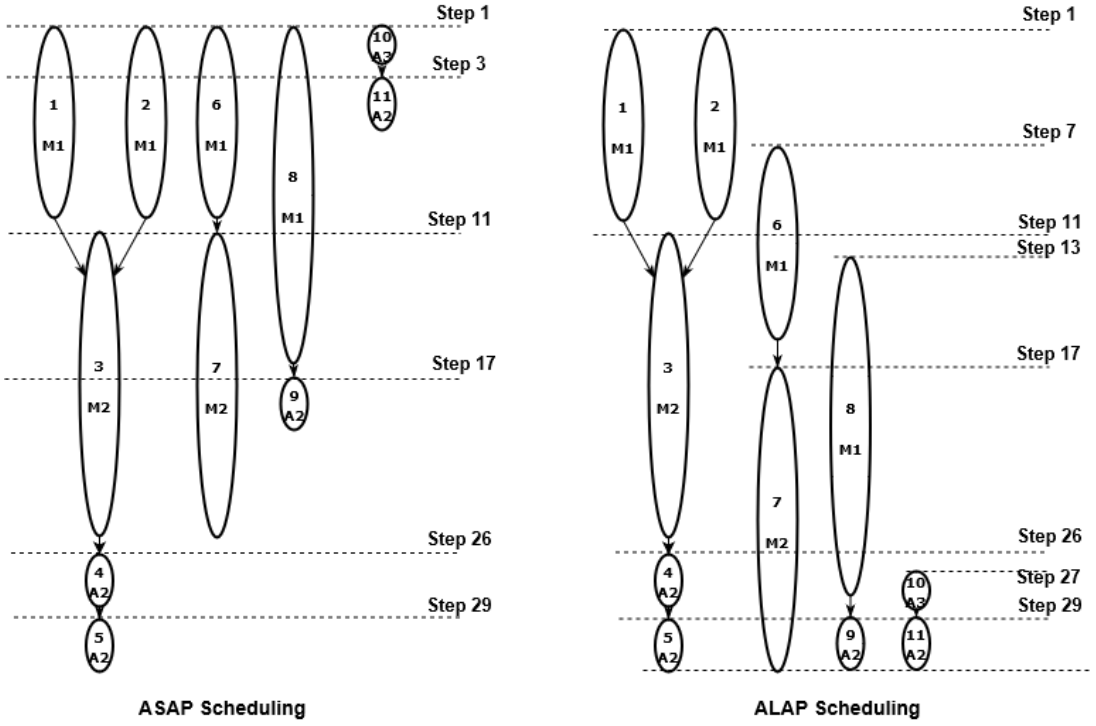


FIGURE 5.2: ASAP and ALAP scheduling for the chromosome representation given in Figure 5.1.

After calculating the mobility of the nodes, our algorithm applies the modified list scheduling, and subsequently returns A, L, R, and E values of the solution as shown in Figure 5.3.

TABLE 5.1: Node mobilities for the schedules in Figure 5.2.

Node(i)	1	2	3	4	5	6	7	8	9	10	11
$t(i)^{ASAP}$	1	1	11	26	29	1	11	1	17	1	3
$t(i)^{ALAP}$	1	1	11	26	29	7	17	13	29	27	29
m_i	0	0	0	0	0	6	6	12	12	26	26

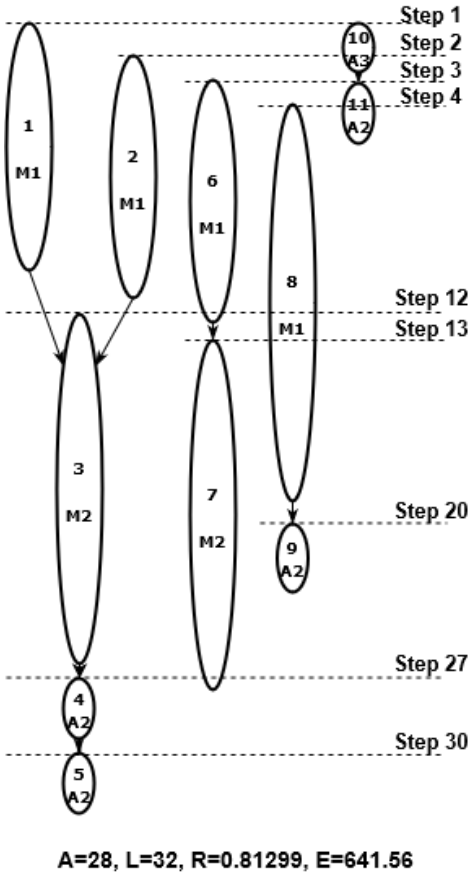


FIGURE 5.3: Final scheduling of the chromosome in Figure 5.1.

5.3. Genetic Operators

The third stage of a genetic algorithm is to apply the genetic operators. We use crossover, mutation, and selection operators, which are the three most common ones.

Crossover

In crossover operation, we select two chromosomes randomly from the population to be the parent . We then swap part of each parent chromosomes to create two offspring in an attempt to inherit better parts of each parent to their children. The most commonly used crossover types are one-point, two-point, and uniform crossover. In order to identify which one is the most suitable for our problem, we applied all three of them on a study similar to ours mentioned in [8], and found that uniform crossover gives the best results for such

problems. Uniform crossover swaps genes between both parents and makes two new children chromosomes. It assigns a random number u to every gene, where $0 \leq u \leq 1$, and compares that number with the swapping probability p_s. We chose p_s to be 0.5 in order to give the equivalent swapping chance for all genes as described in Algorithm 1.

In Figure 5.4, we demonstrate an example of applying the crossover operator on the solution in Figure 5.1. This chromosome is the first parent for the crossover and the second parent is randomly chosen by the algorithm. After applying crossover on parents, two new children chromosomes are produced as demonstrated. It is evident that although the first child meets both latency and area constraints with better reliability value than both parents, the second child does not.

Algorithm 1 Uniform Crossover

Data: X_p: Crossover population; n_i, i= 0,1,..., n: number of nodes; p_s: Probability of swapping;

Result: Two new chromosomes: $C^{(t+1)}$, $D^{(t+1)}$

begin
  Randomly select two chromosomes $A^{(t)}$ and $B^{(t)}$ from X_p. Create two empty chromosomes $C^{(t+1)}$ and $D^{(t+1)}$ wirh the size n.
  for $0 \leq i \leq n$ **do**
  Choose a random real number $u \in [0, 1]$ **if** $u \leq p_s$ **then** /* Swap genes */
    $C_i^{(t+1)} = B_i^{(t)}$ $D_i^{(t+1)} = A_i^{(t)}$
    else /* Don't Swap genes */
      $C_i^{(t+1)} = A_i^{(t)}$
      $D_i^{(t+1)} = B_i^{(t)}$
    end
  end
end

Mutation

Mutation is an important and effective genetic operator to converge to optimum solution. It modifies the randomly or heuristically selected genes to obtain a new chromosome. The goal of mutation is to diversify the population so that the chance of escaping from the local

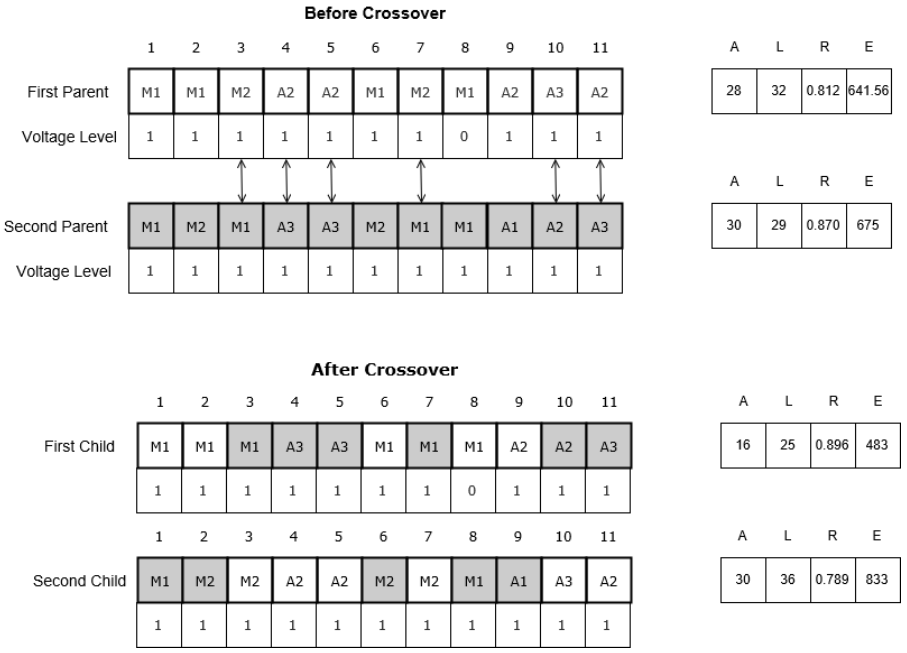


FIGURE 5.4: An example of uniform crossover.

minima increases. However, if the mutation ratio becomes very high, then our GA-based search algorithm behaves like a random search. Thus, we set the mutation ratio to 10% of the total population in our method.

We followed the **random resetting mutation** operator. At first it randomly select a chromosome from the population. Then, to increase the diversity of the solution space, we directly pick genes randomly from the chromosome at hand. The number of mutated genes is also determined randomly so that the number of modified genes is less than N nodes of the circuits.

In Figure 5.6, we illustrate the aforementioned mutation operation. After choosing the chromosome to be mutated, it picked the number of genes to be mutated to be 3. In reference to Figure 5.3, the node number 3 has the resource M_2 assigned to it, it was randomly chosen and changed to M_1 at high voltage level with the latency of 10 time steps instead of 15, e.g. reducing the critical path latency by five time steps. Also node 5 was randomly chosen and assigned the resource A_3 (L_h= 2) instead of A_2 (L_h= 3) which also reduces the latency by one step, since it is a critical path node. The last randomly modified node is the seventh from M_2 to M_1, which in turn reduced the used resources in the design, and hence the total area is reduced by 12 units (area of M_2). Ultimately, after the mutation process, both design constraints are met in this example.

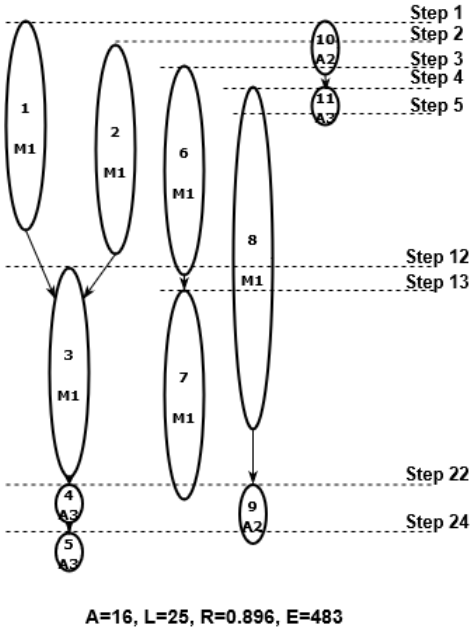


FIGURE 5.5: The scheduling of first child of the crossover given in Figure 5.4.

While we are applying both crossover and mutation, we make sure the shared resources use the same voltage level to construct power islands [31]. Although it is possible to switch from one voltage level to another, this process also takes extra latency and power consumption. Instead, we simply place the resources running on low voltage level to one power island and others to high voltage level.

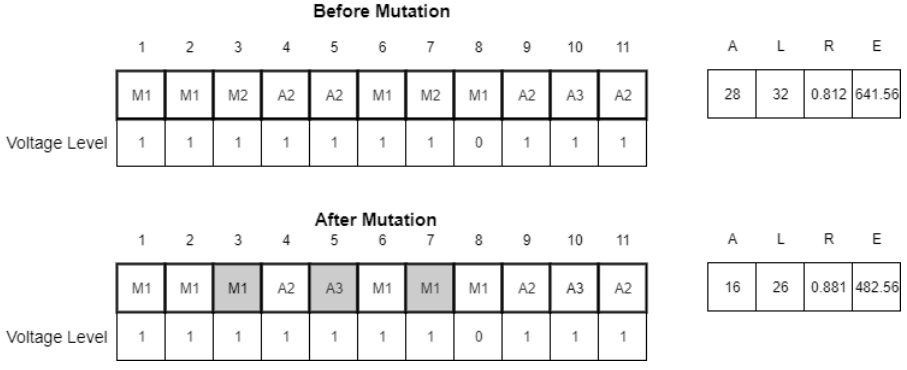


FIGURE 5.6: The mutation operator applied to the chromosome in Figure 5.1.

Selection

Our algorithm selects the parent chromosomes randomly to apply crossover and mutation. After applying crossover and mutation, the total population doubles because of the newly

added individuals. While 90% of the new population comes from the crossover, the remaining 10% comes from the mutation. Since the total population size must be kept fixed, the algorithm applies a **fitness based selection**. It adds the new generated chromosomes to the ones from previous iteration, then order them according to the objective function presented in section 5.4., the objective function given in Equation (12) attempts to maximize the reliability and minimize the energy consumption for a single chromosome. We use this objective function in our fitness calculation. We then select the best 100 chromosomes from the ordered chromosomes set.

Our three main steps is iterated for fixed number of times. We selected our iteration count experimentally. We finally return the the chromosome with the best fitness value as our solution.

5.4. Energy and objective function

The goal is to maximize the overall reliability of the circuit while minimizing its total energy consumption (formulated as objective functions (10) and (11) respectively).

$$\textbf{Maximize } R_{total} = \sum_{i \in Tasks} \rho_i \tag{10}$$

$$\textbf{Minimize } E_{total} = \sum_{i \in Tasks} \epsilon_i \tag{11}$$

This bi-objective problem is formulated as a single objective function given in Equation (12) by employing the scalarization technique in which we combine the weighted sum of energy and reliability values. The parameter α serves for the purpose of assigning weight to both reliability maximization and energy minimization. That is, through choosing different α values we can prioritize either objective function to a certain degree, or assign equal weight to both (by taking $\alpha = 0.5$).

$$\textbf{Minimize } obj = \alpha \cdot (1 - R_{norm}) + (1 - \alpha) \cdot (E_{norm}) \tag{12}$$

R_{norm} and E_{norm} are the values of the total reliability and the total energy consumption normalized to the range [0,1], calculated as given in (13) and (14) respectively.

$$R_{norm} = \frac{R_{total} - R_{min}}{R_{max} - R_{min}} \tag{13}$$

$$E_{norm} = \frac{E_{total} - E_{min}}{E_{max} - E_{min}} \tag{14}$$

R_{min} and R_{max} are the minimum and maximum values the reliability of a given circuit can have. The minimum (or maximum) achievable reliability of a circuit can be calculated by assigning the least (or most) reliable resources in the resource library to every task. Similarly, we can calculate E_{min} and E_{max}, which are the minimum and maximum values of the energy amount a circuit can consume.

5.5. Duplication-Based Post-processing

After the algorithm described in Section 5. returns the final scheduled solution, we take that solution with its latency, area, reliability and energy as an input to a subsequent process to enhance its reliability without violating the constraints. We employ a method similar to the duplication algorithm proposed in [8], which duplicates the nodes as much as it can using simple heuristic rules: it tests the potential of each node to have a duplicate resource from other resources that are previously allocated in the design. The nominated nodes are not on the critical path, so the latency of the solution is maintained. It also checks that the nominated resource is not scheduled for use at the same time step, so the area will not be increased either. For cases of multi-duplicable candidates, the precedence is given to the nodes with lower reliability values. The difference in our approach is that we also incorporate DVS in the duplication process. That is, when we are selecting a resource for duplication we give priority to the one with lower voltage level. To calculate the reliability of the duplicated version, we use Equation (2).

The pseudocode of the duplication process is given in Algorithm 2. The duplication process applied to the solution given in Figure 5.5 is illustrated in Figure 5.7. The shaded nodes are the added duplicate nodes. Furthermore, checkers are added at the end of each node and

its duplicate to ascertain the similarity of the results. From figure 5.7, it is evident that the duplication process increased the total system reliability from 0.896 to 0.938 (4.2% higher). The algorithm simultaneously ensures that the duplicates do not add to the overall area or latency of the solution. Nevertheless, the overall energy consumption is increased. However, we try to decrease this energy increase as much as possible. For example, in Figure 5.7, we have two voltage level options for resource M1 when we duplicate node 8. Although the version that uses low voltage has longer latency than its high voltage counterpart, we select M1 with low voltage since it does not violate the latency constraint and results in smaller energy increase.

Algorithm 2 Selective Duplication

Data: B= b_r, r= 1,2,...,k: resources library; T= t_i, i= 0,1,...,n: nodes' start times; β: Resources allocation; $\bar{A}$, A, $\bar{L}$, M= m_i, i = 0,1,...,n: nodes' mobilities.

Result: β', T'.

```
begin
    l= 1;  T'= T;  β'= β;
    while l ≤ L̄ do
        foreach n_i where t_i= l do
            foreach b_r ∈ β where n_i^type = r^type do
                foreach Time Step t_l in m_i do
                    Determine number of b_r operations in step t_l (i.e., | b_rl |)
                    if (| b_rl | + b_rl ≤ | b_r | ∨ b_r + A ≤ Ā ) ∧ (t_i + d_i ≤ t_j ; ∀ e_ij ∈ E) then
                        Duplicate n_i by binding to b_r in step t_l:  T' = T' + t'_l;  β' = β' + b_ir;
                        Add checker to time step max(t_i + d_i , t_i' + d_i' );
                    end
                end
            end
        end
    end
end
```

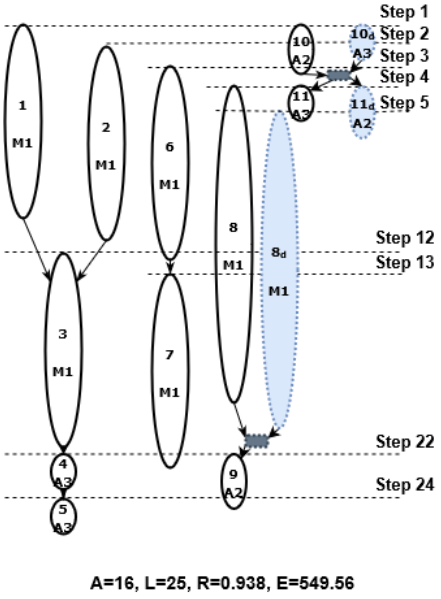


FIGURE 5.7: The scheduling after the duplication of the solution from Figure 5.5.

6. ILP FORMULATION

In this section, we present the ILP formulation of the problem, which maximizes the total reliability while minimizing the total energy consumption, to compare its results with our GA method. The notations used in the ILP formulation of the problem are defined in Table 6.1.

$\zeta_{i,j}$ refers to the compatibility of T_i with R_j (e.g. an addition operation can only be assigned an adder resource) and is formulated in Equation (15).

$$\zeta_{i,j} = \begin{cases} 1 & \text{If } TType_i = RType_j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$Assigned_{i,j}$ is a Boolean variable which specifies if R_j is assigned to T_i (see Equation (16)).

$$Assigned_{i,j,v} = \begin{cases} 1 & \text{if } T_i \text{ is assigned to } R_j \\ & \text{under } V_v \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Only one resource should be assigned to each task under a single voltage level, while taking the compatibility into consideration. This is formulated in Equation (17).

$$\begin{aligned} &\text{While } \zeta_{i,j} = 1 \\ &\forall i \in T : \sum_{j \in R, v \in V} Assigned_{i,j,v} = 1 \end{aligned} \tag{17}$$

$Start_{i,s}$ is a Boolean variable which specifies if T_i started at $Cstep_s$ (see Equation (18)).

$$Start_{i,s} = \begin{cases} 1 & \text{if } T_i \text{ started at } Cstep_s \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

A task may start at only one control step (see Equation (19)).

TABLE 6.1: ILP Notations.

$T = \{T_i : i = 1, ..., N\}$	A set of N tasks (additions, multiplications, NOPs) where T_i is the i^{th} task in T
$TType_i$	The type of T_i (addition, multiplication, NOP)
$R = \{R_i : i = 1, ..., M\}$	A library of M available hardware resources with different area, latency, reliability, and energy consumption values (adders, multipliers), where R_i is the i^{th} resource in R
$V = \{V_l, V_h\}$	A set of available voltage levels (high voltage $V_h = 1.2V$, low voltage $V_l = 1.0V$)
V_i	The voltage at the voltage level i
$RType_j$	The type of R_j
$\zeta_{i,j}$	The compatibility of the task T_i with the resource R_j
$Assigned_{i,j,v}$	Denotes whether R_j is assigned to T_i under V_v
$Csteps$	A set of control steps
$Start_{i,s}$	Denotes whether $Cstep_s$ is the start time of the task T_i
$Start_N$	The start time of the last $sink$ task
$G = (\tau, PREC)$	Precedence graph G where $PREC(i, j)$ means T_i precedes T_j
$Rel_{j,v}$	The reliability of R_j under V_v
A_j	The area occupied by R_j
$L_{j,v}$	The latency of R_j under V_v
$E_{j,v}$	The energy consumption of R_j under V_v
ρ_i	The reliability of T_i
δi	The delay of T_i
ϵ_i	The energy consumed by T_i
$\kappa_{i,s,r,v}$	$\begin{cases} 1 & \text{If } T_i \text{ started at } Cstep_s \text{ and } R_r \text{ is assigned to it under } V_v \\ 0 & \text{otherwise} \end{cases}$
$NumR_{j,s,v}$	The total number of instances of R_j used at $Cstep_s$ under V_v
$\Upsilon_{r,v}$	The total number of instances of R_j used within the circuit under V_v
R_{min}	The minimum reliability value of a given circuit
R_{max}	The maximum reliability value of a given circuit
E_{min}	The minimum energy consumption of a given circuit
E_{max}	The maximum energy consumption of a given circuit
R_{total}	The final total reliability of a given circuit
R_{norm}	The normalized value of the total reliability to the range [0,1]
E_{total}	The final total energy consumption of a given circuit
E_{norm}	The normalized value of the total energy consumption to the range [0,1]
obj	The objective function
Λ	Area constraint
λ	Latency constraint

$$\forall i \in T : \sum_{s \in Csteps} Start_{i,s} = 1 \tag{19}$$

The delay of a task depends on the latency of the resource assigned to it and the voltage level (see Equation (20)).

$$\begin{aligned} &\forall i \ \in T : \\ &\delta i \ = \sum_{r \in R, v \in V} L_{r,v} \cdot Assigned_{i,r,v} \end{aligned} \tag{20}$$

For dependent tasks, precedence constraints must be considered. The start time of a task that depends on a completion of another task must be greater than the end time of the precedent task. This is formulated in Equation (21).

$$\begin{aligned} &\forall (i,j) \ \in T : \text{ If } PREC(i,j) = 1 \\ &\sum_{s \ \in Csteps} Start_{j,s} \cdot s \geq \sum_{s \in Csteps} Start_{i,s} \cdot s + \delta i \end{aligned} \tag{21}$$

The reliability of a task depends on the reliability of its assigned resource under the applied voltage level. This is formulated in Equation (22).

$$\begin{aligned} &\forall i \ \in T : \\ &\rho_i \ = \sum_{r \in R, v \in V} Rel_{r,v} \cdot Assigned_{i,r,v} \end{aligned} \tag{22}$$

$\kappa_{i,s,r,v}$ is a Boolean variable which specifies if T_i started at $Cstep_s$ and if R_r has been assigned to it under V_v (see Equation (23)).

$$\kappa_{i,s,r,v} = \begin{cases} 1 & \text{If } T_i \text{ started at } Cstep_s \text{ and } R_r \text{ is assigned} \\ & \text{to it under } V_v \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Each task can only start at one control step and only one resource can be assigned to it under a single voltage level. We ensure this using Equations (24) and (25)).

$$\begin{aligned} &\forall i \ \in T : \\ &\sum_{r \in R, s \in Csteps, v \in V} \kappa_{i,s,r,v} \ = 1 \end{aligned} \tag{24}$$

$$\begin{aligned} &\forall (i \in T, r \in R, s \in Csteps, v \in V) : \\ &\qquad \kappa_{i,s,r,v} \geq Assigned_{i,r,v} + Start_{i,s} - 1 \end{aligned} \tag{25}$$

The total amount of energy consumed by a task depends on the energy consumption of its assigned resource under the applied voltage. This is formulated in Equation (26).

$$\begin{aligned} &\forall i \ \in T : \\ &\epsilon_i \ = \sum_{r \in R, v \in V} E_{r,v} \cdot Assigned_{i,r,v} \end{aligned} \tag{26}$$

To calculate the number of instances of each available resource used in the overall design, we have to determine the resources that are assigned to tasks starting at that control step. We only check the start times for each task at each control step since we assume that pipelined resources will be used in the design. $NumR_{r,s,v}$ represents the total number of instances of R_j at $Cstep_s$ under V_v and is formulated in Equation (27).

$$\begin{aligned} &\forall (r \in R, s \in Csteps, v \in V) \ : \\ &NumR_{r,s,v} \ = \sum_{i \in Tasks} \kappa_{i,s,r,v} \end{aligned} \tag{27}$$

$\Upsilon_{r,v}$ represents the total number of instances of each available resource under each voltage level that needs to be used in the overall circuit design, and it is the maximum of all $NumR_{r,s,v}$ (see Equation (28)).

$$\begin{aligned} &\forall (r \in R, v \in V) \ : \\ &\Upsilon_{r,v} \ = \max_{s \in Csteps} NumR_{r,s,v} \end{aligned} \tag{28}$$

6.1. Constraints

The total area should not exceed the given area constraint. This is formulated in Inequality (29).

$$\sum_{r \in R, v \in V} \Upsilon_{r,v} \cdot A_r \leq \Lambda \tag{29}$$

Latency constraint will be met if the start time of the last $sink$ task (denoted as $Start_N$ and defined in Equation (30) is less than or equal to the given maximum allowed latency. This is formulated in Inequality (31).

$$Start_N = \sum_{s \in Csteps} Start_{N,s} \cdot s \tag{30}$$

$$Start_N \leq \lambda \tag{31}$$

7. RESULTS AND ANALYSIS

In this chapter, the effectiveness of our proposed methods is investigated through several sets of experiments. We selected four most commonly used benchmarks in literature: Differential Equation Solver (DES), Finite Impulse Response (FIR) filter, Auto-Regressive (AR) filter, and Elliptic Wave Filter (EWF). The benchmark features (the number of nodes and edges in their respective data flow graphs, as well as addition and multiplication operations) are briefly summarized in Table 7.1. More detailed specifications and data flow graphs for the benchmarks we used can be found in [2] and [32].

TABLE 7.1: Summary of Benchmarks.

Benchmark	Nodes	Edges	Additions	Multiplications
DES	11	8	5	6
FIR	23	22	15	8
EWF	26	40	26	0
AR	28	30	12	16

We give the resource library we used in our experiments in Table 4.1, where we list area, latency, reliability, and energy consumption values of each resource under low and high voltage levels. We measure latency in time steps (e.g., clock cycles), area in mm^2, and energy consumption in nanojoules (nJ). It is worth noting that our proposed methods can be used with any resource library that has clearly defined area, latency, reliability, and energy consumption values for each resource under different voltage levels.

The experiments of the proposed methods which attempt to solve the bi-objective problem formulated in Equation (12) were performed using four benchmarks from Table 7.1 for varying area and latency constraints, as well as different α values.

The minimum latency constraints for each benchmark were obtained from ASAP scheduling algorithm by using the fastest resource for each type of the operations. Once the minimum circuit delay is obtained, the latency constraint can be increased gradually to test for less delay-sensitive cases, which may allow utilization of slower but more reliable resources in the design, and/or allow operation of certain resources at the low voltage level, as to reduce the overall energy consumption of the circuit. The minimum area constraints, on the other hand, were obtained by assigning a single resource with 38 the smallest area for each different type of operation within a benchmark. Testing for different area constraints allows the algorithms to find solutions with lower latency and higher reliability values.

Since the objective function combines the weighted sum of energy and reliability values, by choosing different values of the parameter α (which assigns weight to reliability maximization and energy minimization) we can prioritize either objective function to a certain degree, or assign an equal weight to both. The experiments with varying area and latency constraints were performed for α values of 0.0, 0.5, and 1.0. The α value of 0.0 means that we give the maximum priority to the minimization of energy consumption, while disregarding the reliability of the circuit altogether. Similarly, the α value of 1.0 means that we give the maximum priority to maximizing reliability, without taking energy consumption into consideration at all. Finally, by taking $\alpha = 0.5$, we assign an equal weight to both objective functions, making the problem bi-objective in nature. Furthermore, in Subsection 7.3., we present more detailed results that demonstrate how reliability and energy consumption values change when the parameter α varies in steps of 0.1 (assigning varying weights to either reliability maximization or energy minimization).

The experiments were performed on a computer with the following configurations: Intel Core(TM)2 Duo CPU E8500, at 3.16 GHz, with 2 cores, 2 logical processors, and a total physical memory of 5,823 MB.

7.1. Comparison of GA and ILP Methods

Tables 7.2, 7.3, 7.4, and 7.5 show the reliability and energy consumption results of the proposed GA method compared to ILP for DES, FIR, AR, and EWF benchmarks, respectively.

The first column specifies the value of the α. The second column indicates the latency (L) and area (A) constraints used in that particular test instance. The third and fourth columns give the reliability values from the solutions obtained by ILP and GA-based methods, respectively. Similarly, the sixth and seventh columns give the energy consumption values. Delta (Δ) represents the percentage change of the GA result relative to the ILP result, and the percentage changes in reliability and energy consumption results are given in the fifth and eighth columns respectively. The reliability Δ is calculated according to the percentage change increase formula since a higher reliability value means a better solution, whereas the energy consumption Δ is calculated according to the percentage change decrease formula as lower energy consumption is a more desirable outcome.

TABLE 7.2: Results of DES Benchmark.

		Reliability			**Energy**		
Alpha	(L , A)	ILP	GA	$\Delta(\%)$	ILP	GA	$\Delta(\%)$
1.0	(31,10)	0.99	0.99	0	540	540	0
	(31,20)	0.99	0.98	-1.2	540	534	1.1
	(31,30)	0.99	0.98	-1.2	540	534	1.1
	(28,20)	0.98	0.98	0	534	534	0
	(28,30)	0.98	0.98	0	534	534	0
	(28,40)	0.98	0.97	-1.2	534	528	1.12
	(25,20)	0.97	0.97	0	528	528	0
	(25,30)	0.97	0.97	0	528	528	0
	(25,40)	0.97	0.97	0	528	528	0
	Average Δ (%)			**-0.4**	**Average Δ (%)**		**0.37**
0.5	(31,10)	0.99	0.99	0	540	540	0
	(31,20)	0.98	0.99	0.4	480.11	490.4	-2.14
	(31,30)	0.96	0.94	-1.81	419.23	430	-2.57
	(28,20)	0.97	0.98	0.3	522.99	538	-2.87
	(28,30)	0.97	0.95	-1.83	474.11	472.16	0.41
	(28,40)	0.97	0.94	-2.95	474.11	450.68	4.94
	(25,20)	0.96	0.97	0.3	516.99	528	-2.13
	(25,30)	0.96	0.94	-2	492.55	516	-4.76
	(25,40)	0.96	0.94	-2	492.55	516	-4.76
	Average Δ (%)			**-1.06**	**Average Δ (%)**		**-1.54**
0.0	(31,10)	0.99	0.99	0	540	540	0
	(31,20)	0.72	0.74	2.6	448.47	450.44	-0.44
	(31,30)	0.80	0.80	0.9	404.65	405.62	-0.24
	(28,20)	0.85	0.86	1.9	480.56	481.56	-0.21
	(28,30)	0.76	0.75	-1.3	451	452.1	-0.24
	(28,40)	0.76	0.77	1.4	451	452.1	-0.24
	(25,20)	0.80	0.81	1	502.41	508	-1.11
	(25,30)	0.80	0.81	1.1	477.97	481.56	-0.75
	(25,40)	0.79	0.81	2.3	476.14	481.56	-1.14
	Average Δ (%)			**1.1**	**Average Δ (%)**		**-0.41**

For α values of 1.0 and 0.0 the GA-based method obtains optimum or near-optimum results in most of the cases. In these two cases, the solver is trying to either maximize the total reliability (for $\alpha = 1.0$) or minimize the overall energy consumption (for $\alpha = 0.0$). For ($\alpha = 0.5$) the deference from optimum solutions is a little bit higher than other values of alpha, which means that when assigning equal weight to the bi-objective problem of maximizing reliability and minimizing energy GA is not doing as well as ILP,this could be explained by the randomness of GAs in general. Also, while the number of benchmark nodes increases the disparity in the obtained values grows as well.

TABLE 7.3: Results of FIR Benchmark.

		Reliability			**Energy**		
Alpha	(L , A)	ILP	GA	Δ(%)	ILP	GA	Δ(%)
1.0	(51,10)	0.98	0.98	0	820	820	0
	(35,15)	0.91	0.91	0	784	784	0
	(40,15)	0.93	0.93	0	796	796	0
	(40,20)	0.93	0.93	0	796	796	0
	(50,20)	0.98	0.97	-1.2	820	814	0.7
	(35,30)	0.92	0.90	-2.4	790	778	1.5
	(40,30)	0.93	0.93	0	796	796	0
	(45,30)	0.95	0.95	0	808	808	0
	(50,30)	0.98	0.97	-1.2	820	814	0.7
	Average Δ (%)			**-0.53**	**Average Δ (%)**		**0.33**
0.5	(51,10)	0.98	0.98	0	820	820	0
	(35,15)	0.85	0.84	-2.14	546.13	549	-0.53
	(40,15)	0.88	0.86	-2.24	553.12	556.11	-0.54
	(40,20)	0.88	0.85	-3.64	553.12	580.64	-4.98
	(50,20)	0.90	0.87	-3.45	561.45	562.46	-0.18
	(35,30)	0.89	0.87	-2.16	564.13	572	-1.4
	(40,30)	0.88	0.84	-4.75	553.12	560.48	-1.33
	(45,30)	0.89	0.90	0.91	555.45	568.48	-2.35
	(50,30)	0.90	0.90	-0.1	557.78	564.48	-1.2
	Average Δ (%)			**-1.95**	**Average Δ (%)**		**-1.39**
0.0	(51,10)	0.98	0.98	0	820	820	0
	(35,15)	0.81	0.79	-2.89	534.48	548	-2.53
	(40,15)	0.81	0.79	-2.89	534.48	548	-2.53
	(40,20)	0.69	0.67	-2.77	508.86	510.83	-0.39
	(50,20)	0.44	0.46	3.74	499.33	500.73	-0.28
	(35,30)	0.58	0.59	2.06	513.81	528.48	-2.86
	(40,30)	0.54	0.52	-3.9	502.83	510.7	-1.57
	(45,30)	0.48	0.50	5.21	500.73	517.15	-3.28
	(50,30)	0.42	0.43	1.2	498.63	500.03	-0.28
	Average Δ (%)			**-0.03**	**Average Δ (%)**		**-1.52**

7.2. Execution Time Analysis

While the ILP-based method determines the optimum results, it takes too much time for problems with a large number of variables. Resource scheduling and binding under given constraints are NP-hard problems. Thus, the execution time of an ILP-based solution grows exponentially as the number of nodes increases, and becomes computationally impractical for more complex circuits. Therefore, the proposed GA-based method provides a faster solution with near-optimum results. Figure 7.1 demonstrates the comparison between the average execution times of ILP and GA methods for varying number of benchmark nodes.

TABLE 7.4: Results of AR Benchmark.

		Reliability			**Energy**		
Alpha	(L , A)	ILP	GA	Δ	ILP	GA	Δ
1.0	(65,15)	0.97	0.96	-1.36	1,424	1,412	0.84
	(55,20)	0.93	0.93	0.00	1,400	1,400	0.00
	(60,20)	0.97	0.96	-1.36	1,424	1,412	0.84
	(65,20)	0.97	0.96	-1.36	1,424	1,412	0.84
	(50,30)	0.92	0.90	-1.20	1,394	1,420	-1.87
	(55,30)	0.95	0.94	-1.20	1,412	1,406	0.42
	(60,30)	0.97	0.97	0.00	1,424	1,424	0.00
	(50,40)	0.93	0.93	0.00	1,400	1,400	0.00
	(55,40)	0.97	0.96	-1.20	1,424	1,418	0.42
	Average Δ (%)			**-0.85**	**Average Δ (%)**		**0.17**
0.5	(65,15)	0.86	0.88	2.43	978.96	1,012	-3.38
	(55,20)	0.92	0.92	-0.43	1,392.66	1,405	-0.89
	(60,20)	0.97	0.93	-3.78	1,409.32	1,411	-0.12
	(65,20)	0.86	0.90	4.81	971.62	1,106	-13.83
	(50,30)	0.85	0.83	-2.51	1,161.14	1,211	-4.29
	(55,30)	0.83	0.83	-0.55	1,058.72	1,108	-4.65
	(60,30)	0.85	0.89	5.45	965.62	1,024	-6.05
	(50,40)	0.88	0.87	-1.09	1,180.48	1,211	-2.59
	(55,40)	0.85	0.85	0.11	1,063.38	1,125	-5.79
	Average Δ (%)			**0.49**	**Average Δ (%)**		**-4.62**
0.0	(65,15)	0.83	0.79	-5.10	960.96	1,005	-4.58
	(55,20)	0.67	0.71	6.53	1,144.48	1,176	-2.75
	(60,20)	0.67	0.64	-4.75	1,071.16	1,140.68	-6.49
	(65,20)	0.77	0.76	-1.26	949.98	961.96	-1.26
	(50,30)	0.74	0.73	-1.49	1,147.76	1,188	-3.51
	(55,30)	0.83	0.75	-9.37	1,058.72	1,075	-1.54
	(60,30)	0.75	0.75	0.11	955.9	971.88	-1.67
	(50,40)	0.71	0.75	4.62	1,142.27	1,165	-1.99
	(55,40)	0.73	0.79	8.91	1,048.17	1,131	-7.90
	Average Δ (%)			**-0.2**	**Average Δ (%)**		**-3.52**

As it is evident from the figure, while the execution time of the ILP method starts growing exponentially for benchmarks that have more than 20 nodes, the average execution time of the GA-based method remains to be around one second for any benchmark size, making it a practical method of choice for complex circuits with a large number of nodes.

7.3. Effects of DVS on Reliability

Figures 7.2, 7.3, 7.4, and 7.5 demonstrate the changes in reliability and energy consumption values for different values of α, for DES, FIR, AR, and EWF benchmarks, respectively.

TABLE 7.5: Results of EWF Benchmark.

		Reliability			**Energy**		
Alpha	(L , A)	ILP	GA	Δ	ILP	GA	Δ
1.0	(30,10)	0.82	0.8	-2.34	228	220	3.51
	(30,20)	0.82	0.8	-2.34	228	220	3.51
	(40,10)	0.91	0.91	0	276	276	0
	(40,20)	0.91	0.9	-1.2	276	270	2.17
	(40,30)	0.91	0.91	0	276	276	0
	(50,5)	0.89	0.86	-3	291	284	2.41
	(50,10)	0.95	0.95	0	300	300	0
	(50,20)	0.95	0.95	0	300	300	0
	(50,30)	0.95	0.95	0	300	300	0
	Average Δ (%)			**-0.99**	**Average Δ (%)**		**1.29**
0.5	(30,10)	0.57	0.59	4.03	119.4	116	2.85
	(30,20)	0.55	0.57	3.95	113.91	119	-4.47
	(40,10)	0.53	0.57	7.50	108.42	119	-9.76
	(40,20)	0.53	0.56	5.62	108.42	116	-6.99
	(40,30)	0.53	0.57	6.47	108.42	106	2.23
	(50,5)	0.53	0.56	5.60	108.42	105.55	2.65
	(50,10)	0.53	0.56	5.60	108.42	116	-6.99
	(50,20)	0.53	0.56	5.60	108.42	116	-6.99
	(50,30)	0.53	0.56	5.60	108.42	116	-6.99
	Average Δ (%)			**5.55**	**Average Δ (%)**		**-3.83**
0.0	(30,10)	0.57	0.55	-3.01	119.4	122	-2.18
	(30,20)	0.43	0.44	2.34	109.71	115	-4.82
	(40,10)	0.30	0.33	8.27	98.62	100.02	-1.42
	(40,20)	0.30	0.32	4.05	98.62	99.32	-0.71
	(40,30)	0.30	0.32	4.1	98.62	101.42	-2.84
	(50,5)	0.53	0.49	-6.98	108.42	111.7	-3.03
	(50,10)	0.21	0.22	3.58	92.32	93.72	-1.52
	(50,20)	0.21	0.22	3.28	92.32	94.42	-2.27
	(50,30)	0.21	0.21	0	92.32	92.32	0
	Average Δ (%)			**1.74**	**Average Δ (%)**		**-2.09**

The figures present more detailed results that demonstrate how reliability and energy consumption values change when the parameter α varies in steps of 0.1 (assigning varying weights to either reliability maximization or energy minimization) for specific area (A) and latency (L) constraints. As α increases, we give more and more weight to maximizing the circuit reliability at the expense of higher energy consumption, and vice versa. Hence, with growing α, both reliability and energy consumption values should increase.

From the (a) charts in Figures 7.2-7.5 that demonstrate the change in reliability values, we observe that the proposed GA-based method obtains the optimum or near-optimum results

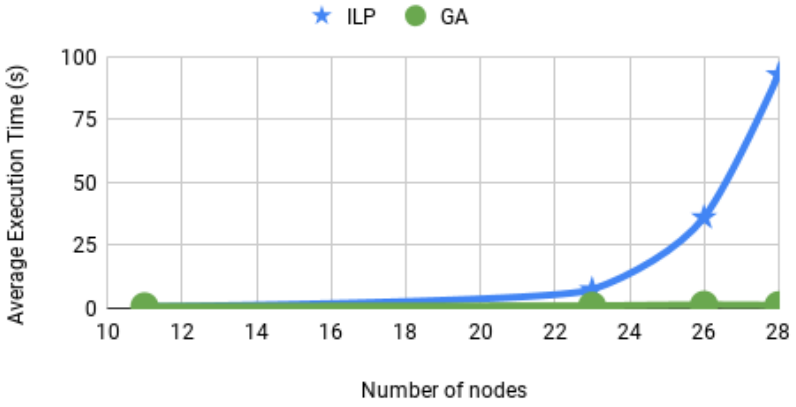
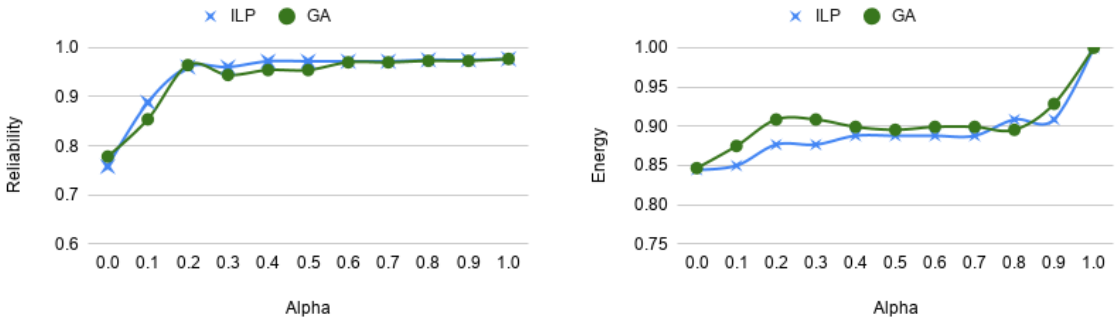


FIGURE 7.1: Average execution times of ILP and GA methods for varying number of benchmark nodes.

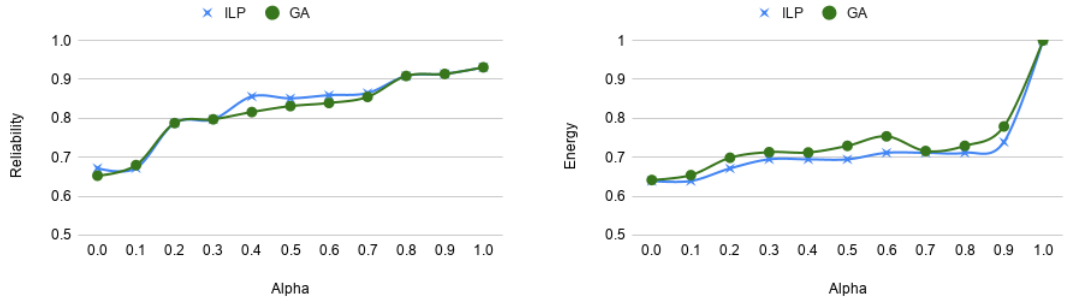


(a) (b)

FIGURE 7.2: Changes over different α values for DES benchmark ($A = 30, L = 28$): (a) Change in reliability; (b)Change in energy consumption

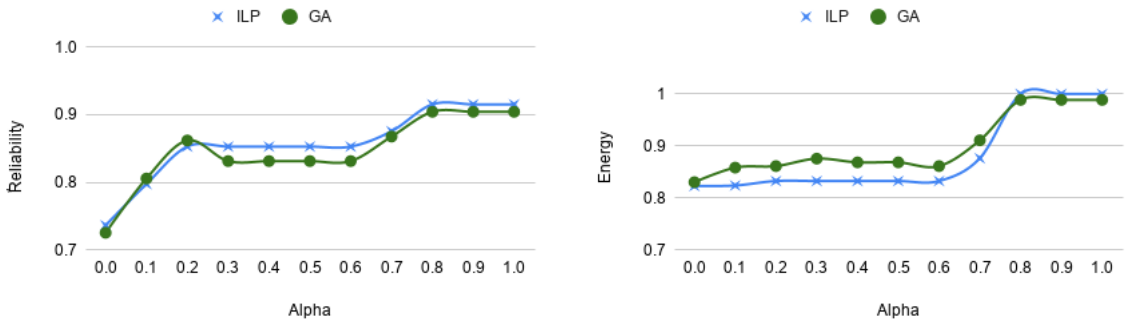
for all test cases. However, the GA solutions appear to result in relatively higher overall energy consumption.

Overall, we observe that DVS negatively affects the final reliability of a circuit. Giving more than 50% weight to minimizing energy consumption generally results in unacceptably low reliability values of the final circuit design. Therefore, one should consider balancing the gain and loss in terms of the energy and reliability. For example, when we aim to optimize reliability we can define energy as a constraint or vice versa. Both ILP and GA-based methods can easily be modified for assigning either reliability or energy as constraints since they offer very flexible infrastructure to the user to play with the area, latency, reliability, and energy of the design at the same time.



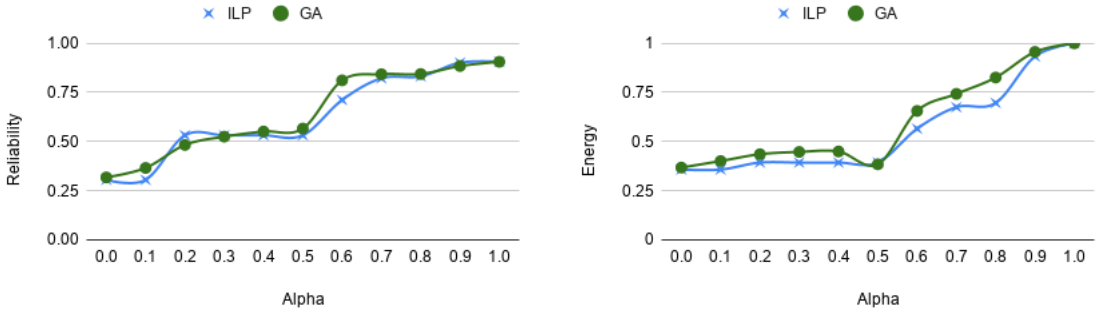
(a) (b)

FIGURE 7.3: Changes over different α values for FIR benchmark ($A = 20, L = 40$): (a) Change in reliability; (b)Change in energy consumption



(a) (b)

FIGURE 7.4: Changes over different α values for AR benchmark ($A = 30, L = 50$): (a) Change in reliability; (b)Change in energy consumption



(a) (b)

FIGURE 7.5: Changes over different α values for EWF benchmark ($A = 30, L = 40$): (a) Change in reliability; (b)Change in energy consumption

7.4. Effects of Duplication

Tables 7.6, 7.7, 7.8, and 7.9 show the reliability and energy consumption results of the duplicated-GA for DES, FIR, AR, and EWF benchmarks, respectively, for α value of 1.0. The symbols RΔ (reliability change) and EΔ (energy change) represent the percentage change of the GA, ILP, and Fully Duplicated (FD) results. While we determine FD version of the design, we consider that all the resources of the design obtained by GA-based method are duplicated by the same resource. Therefore, the area and energy consumption of FD are increased 100%. We compare our duplication method with FD to see if we can achieve similar reliability improvement with less area and less energy increase.

In the given tables, the first column defines the latency and area constraints for which the tests were performed. The second and third columns represent the reliability and energy consumption results of the GA-duplication. The fourth and fifth columns indicate the reliability and energy percentage change of the GA results relative to the results after duplication. The sixth and seventh columns show the reliability and energy percentage change of the ILP results relative to the results of the duplicated GA method. In the last two columns, the results of the duplicated GA method were compared relative to the results of the fully duplicated solutions. In the fully duplicated solutions, the reliability is calculated according to the Equation (2) using the resources obtained by GA, while the energy consumption and area are doubled.

From the results we observe that the GA duplication method improves the reliability results by around 5% on average over the original GA method, while increasing the energy with an average of 15%. Furthermore, the GA duplication results are better compared to the ILP results in terms of reliability; however, the energy consumption is increased.

When we compare our GA duplication method against the FD counterpart, we see that our reliability difference on four benchmark is around 4% on average. On the other hand, FD method increases the energy consumption by an average of 70% and area 100%. These results show that our duplication-based post processing is very effective when compared to FD version since it improves the reliability similar to fully duplication with half area and much less energy consumption.

TABLE 7.6: Duplicated GA results of DES compared to the results of GA, ILP and Fully Duplicated methods

	GA. Duplication		**Compared to GA**		**Compared to ILP**		**Compared to FD**	
(L , A)	Reliability	Energy	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)
(31,10)	0.9948	635	-0.58	14.96	-0.58	14.96	0.51	-70.08
(31,20)	0.9948	635	-1.77	15.91	-0.58	14.96	0.47	-68.19
(31,30)	0.9948	635	-1.77	15.91	-0.58	14.96	0.47	-68.19
(28,20)	0.9829	629	-0.58	15.10	-0.58	15.10	1.69	-69.79
(28,30)	0.9829	629	-0.58	15.10	-0.58	15.10	1.69	-69.79
(28,40)	0.9829	629	-0.76	16.06	0.44	15.10	2.67	-67.89
(25,20)	0.9711	623	-0.58	15.25	-0.58	15.25	2.86	-69.5
(25,30)	0.9711	623	-0.58	15.25	-0.58	15.25	2.86	-69.5
(25,40)	0.9711	623	-0.58	15.25	-0.58	15.25	2.86	-69.5

TABLE 7.7: Duplicated GA results of FIR compared to the results of GA, ILP and Fully Duplicated methods

	GA. Duplication		**Compared to GA**		**Compared to ILP**		**Compared to FD**	
(L , A)	Reliability	Energy	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)
(51,10)	0.99	931	-1.16	11.9	-1.16	11.92	1.09	-76.15
(35,15)	0.94	929	-3.35	15.6	-3.35	15.61	5.45	-68.78
(40,15)	0.96	908	-3.21	12.3	-3.21	12.29	3.45	-75.42
(40,20)	0.97	908	-3.71	12.3	-3.71	12.33	2.92	-75.33
(50,20)	0.98	998	-1.25	18.4	-0.05	17.84	2.15	-63.13
(35,30)	0.94	989	-4.23	21.3	-1.88	20.12	5.54	-57.33
(40,30)	0.96	908	-3.21	12.3	-3.21	12.33	3.45	-75.33
(45,30)	0.98	998	-2.17	19	-2.17	19.04	2.34	-61.92
(50,30)	0.98	998	-1.25	18.4	-0.05	17.84	2.15	-63.13

TABLE 7.8: Duplicated GA results of AR compared to the results of GA, ILP and Fully Duplicated methods

	GA. Duplication		**Compared to GA**		**Compared to ILP**		**Compared to FD**	
(L , A)	Reliability	Energy	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)	RΔ (%)	EΔ (%)
(65,15)	0.985	1611	-2.61	12.35	-1.27	11.61	1.37	-75.29
(55,20)	0.964	1610	-3.90	13.04	-3.90	13.04	3.16	-73.91
(60,20)	0.997	1724	-3.83	18.10	-2.51	17.40	0.09	-63.81
(65,20)	0.985	1611	-2.61	12.35	-1.27	11.61	1.37	-75.29
(50,30)	0.944	1534	-4.21	0.00	-3.05	9.13	4.95	-85.14
(55,30)	0.994	1776	-5.67	20.83	-4.53	20.50	0.20	-58.33
(60,30)	0.989	1544	-1.73	7.77	-1.73	7.77	0.98	-84.46
(50,40)	0.977	1730	-5.12	19.08	-5.12	19.08	1.85	-61.85
(55,40)	0.989	1638	-2.87	13.43	-1.69	13.06	0.94	-73.14

TABLE 7.9: Duplicated GA results of EWF compared to the results of GA, ILP and Fully Duplicated methods

	GA. Duplication		**Compared to GA**		**Compared to ILP**		**Compared to FD**	
(L , A)	Reliability	Energy	R$\triangle$ (%)	E$\triangle$ (%)	R$\triangle$ (%)	E$\triangle$ (%)	R$\triangle$ (%)	E$\triangle$ (%)
(30,10)	0.83	256	-2.79	14.06	-0.46	10.94	16.32	-71.88
(30,20)	0.83	256	-2.79	14.06	-0.46	10.94	16.32	-71.88
(40,10)	0.97	316	-6.89	12.66	-6.89	12.66	1.85	-74.68
(40,20)	0.95	310	-5.98	12.90	-4.84	10.97	3.86	-74.19
(40,30)	0.97	316	-6.89	12.66	-6.89	12.66	1.85	-74.68
(50,5)	0.88	294	-2.27	3.40	-0.37	1.02	9.93	-93.20
(50,10)	0.98	350	-2.80	14.29	-2.80	14.29	1.96	-71.43
(50,20)	0.98	350	-2.80	14.29	-2.80	14.29	1.96	-71.43
(50,30)	0.98	350	-2.80	14.29	-2.80	14.29	1.96	-71.43

8. CONCLUSION

Constantly shrinking technology sizes have allowed for a significant increase in the number of transistors on chips, resulting in smaller integrated circuit areas and increased circuit densities. Higher circuit densities, in turn, have introduced new design problems in digital systems, such as more vulnerability of circuits to radiation effects due to lower supply and threshold voltage levels. Higher SERs of combinational logic in particular makes reliability-oriented HLS a priority. Nonetheless, achieving a high reliability of a circuit should not come at the expense of unlimited energy consumption.

This study incorporates both energy and reliability metrics into the optimization of the HLS process of ASIC design, unlike the previous studies that only focus on one of these metrics, while taking area and latency constraints into consideration. Especially, the effect of DVS on reliability is examined. Our goal was to develop new HLS method for ASIC design under area and timing constraints with objectives of both low energy consumption and high reliability. We propose GA-based optimization method, as well as a selective duplication method which further maximizes the reliability and compare it to ILP method. While the ILP-based method determines the optimum results, it takes too much time for some problems; the execution time grows exponentially as the number of nodes increases, and it becomes computationally impractical for problems with a large number of variables. Therefore, we propose a GA-based method that is faster and yields optimum or near-optimum results.

REFERENCES

[1] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 389–398. **2002**.

[2] Suleyman Tosun, Nazanin Mansouri, Ercument Arvas, Mahmut Kandemir, and Yuan Xie. Reliability-centric high-level synthesis. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pages 1258–1263. IEEE Computer Society, **2005**.

[3] A. Dixit and A. Wood. The impact of new technology on soft error rates. In *2011 International Reliability Physics Symposium*, pages 5B.4.1–5B.4.7. **2011**. ISSN 1541-7026. doi:10.1109/IRPS.2011.5784522.

[4] Vikas Chandra and Robert Aitken. Impact of voltage scaling on nanoscale sram reliability. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 387–392. European Design and Automation Association, **2009**.

[5] F. Dabiri, N. Amini, M. Rofouei, and M. Sarrafzadeh. Reliability-aware optimization for dvs-enabled real-time embedded systems. In *9th International Symposium on Quality Electronic Design (isqed 2008)*, pages 780–783. **2008**. ISSN 1948-3295. doi:10.1109/ISQED.2008.4479837.

[6] Giovanni De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, **1994**.

[7] Zhiru Zhang, Deming Chen, Steve Dai, and Keith Campbell. High-level synthesis for low-power design. *IPSJ Transactions on System LSI Design Methodology*, 8:12–25, **2015**.

[8] Suleyman Tosun and Tohid Taghizad Gogjeh Yaran. Genetic algorithm-based reliability optimization for high-level synthesis. *Journal of Circuits, Systems and Computers*, 28(03):1950039, **2019**.

[9] Robert A Walker and Raul Camposano. *A survey of high-level synthesis systems*, volume 135. Springer Science & Business Media, **2012**.

[10] Alex Orailoğlu and Ramesh Karri. A design methodology for the high-level synthesis of fault-tolerant asics. In *6th IEEE Workshop on VLSI Signal Processing*, pages 417–426. Institute of Electrical and Electronics Engineers Inc., **1992**.

[11] Michael Glaß, Martin Lukasiewycz, Thilo Streichert, Christian Haubelt, and Jürgen Teich. Interactive presentation: Reliability-aware system synthesis. In *Proceedings of the conference on Design, automation and test in Europe*, pages 409–414. EDA Consortium, **2007**.

[12] Tomoo Inoue, Hayato Henmi, Yuki Yoshikawa, and Hideyuki Ichihara. High-level synthesis for multi-cycle transient fault tolerant datapaths. In *2011 IEEE 17th International On-Line Testing Symposium*, pages 13–18. IEEE, **2011**.

[13] Aiman H El-Maleh and Khaled AK Daud. Simulation-based method for synthesizing soft error tolerant combinational circuits. *IEEE Transactions on Reliability*, 64(3):935–948, **2015**.

[14] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced cpu energy. In *Mobile Computing*, pages 449–471. Springer, **1994**.

[15] Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing algorithms for dynamic speed-setting of a low-power cpu. In *MobiCom*, volume 95, pages 13–25. Citeseer, **1995**.

[16] Yumin Zhang, Xiaobo Sharon Hu, and Danny Z Chen. Task scheduling and voltage selection for energy minimization. In *Proceedings of the 39th annual Design Automation Conference*, pages 183–188. ACM, **2002**.

[17] Tohru Ishihara and Hiroto Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202. ACM, **1998**.

[18] Yung-Chia Lin, Yi-Ping You, Chung-Wen Huang, Jenq Kuen Lee, Wei-Kuan Shih, and Ting-Ting Hwang. Energy-aware scheduling and simulation methodologies for parallel security processors with multiple voltage domains. *The Journal of Supercomputing*, 42(2):201–223, **2007**.

[19] Ruibin Xu, Daniel Mossé, and Rami Melhem. Minimizing expected energy consumption in real-time systems through dynamic voltage scaling. *ACM Transactions on Computer Systems (TOCS)*, 25(4):9, **2007**.

[20] Jinfeng Liu, Pai H Chou, Nader Bagherzadeh, and Fadi Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. In *Proceedings of the 38th annual Design Automation Conference*, pages 840–845. ACM, **2001**.

[21] Sumit Ahuja. *High level power estimation and reduction techniques for power aware hardware design*. Ph.D. thesis, Virginia Tech, **2010**.

[22] John Hansen and Montek Singh. An energy and power-aware approach to high-level synthesis of asynchronous systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 269–276. IEEE Press, **2010**.

[23] Ashok K Murugavel and Nagarajan Ranganathan. A game theoretic approach for power optimization during behavioral synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(6):1031–1043, **2003**.

[24] Saraju P Mohanty, Nagarajan Ranganathan, Elias Kougianos, and Priyardarsan Patra. *Low-power high-level synthesis for nanoscale CMOS circuits*. Springer Science & Business Media, **2008**.

[25] Shengqi Yang, Wenping Wang, Tiehan Lu, Wayne Wolf, Narayanan Vijaykrishnan, and Yuan Xie. Case study of reliability-aware and low-power design. *IEEE transactions on very large scale integration (VLSI) systems*, 16(7):861–873, **2008**.

[26] Farshad Firouzi, Mostafa E Salehi, Fan Wang, Sied Mehdi Fakhraie, and Saeed Safari. Reliability-aware dynamic voltage and frequency scaling. In *2010 IEEE Computer Society Annual Symposium on VLSI*, pages 304–309. IEEE, **2010**.

[27] Farshad Firouzi, Mostafa E Salehi, Fan Wang, and Sied Mehdi Fakhraie. An accurate model for soft error rate estimation considering dynamic voltage and frequency scaling effects. *Microelectronics Reliability*, 51(2):460–467, **2011**.

[28] Edin Kadric, Kunal Mahajan, and André DeHon. Energy reduction through differential reliability and lightweight checking. In *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 243–250. IEEE, **2014**.

[29] David Harris and Sarah Harris. *Digital design and computer architecture*. Morgan Kaufmann, **2010**.

[30] Cadence genus synthesis solution. `https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html`. Accessed: 23.12.2019.

[31] Deniz Dal and Nazanin Mansouri. Power optimization with power islands synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7):1025–1037, **2009**.

[32] Suleyman Tosun, Ozcan Ozturk, Nazanin Mansouri, Ercument Arvas, Mahmut Kandemir, Yuan Xie, and W-L Hung. An ilp formulation for reliability-oriented high-level synthesis. In *Sixth international symposium on quality electronic design (isqed'05)*, pages 364–369. IEEE, **2005**.