ISSN: 0067-2904

Multi-Objective Genetic Algorithm-Based Technique for Achieving Low-Power VLSI Circuit Partition

Sharadindu Roy[1], Siddhartha Banerjee[2,*]
[1]*Department of Computer Science, Sonarpur Mahavidyalaya, Sonarpur, Kolkata, India*
[2]*Department of Computer Science, Ramakrishna Mission Residential College (Autonomous), Narendrapur, Kolkata, India*

Received: 23/6/2022 Accepted: 29/11/2022 Published: 30/11/2023

Abstract
Minimizing the power consumption of electronic systems is one of the most critical concerns in the design of integrated circuits for very large-scale integration (VLSI). Despite the reality that VLSI design is known for its compact size, low power, low price, excellent dependability, and high functionality, the design stage remains difficult to improve in terms of time and power. Several optimization algorithms have been designed to tackle the present issues in VLSI design. This study discusses a bi-objective optimization technique for circuit partitioning based on a genetic algorithm. The motivation for the proposed research is derived from the basic concept that, if some portions of a circuit's system are deactivated during the processor's idle time, the circuit's power consumption is automatically reduced. To reduce the overall system's power consumption, maximization of sleep time and minimization of net cuts are required. To achieve these, an effective fitness function has been constructed in such a way that the balance criteria are also maintained. The approach has been tested on a set of net lists from the ISPD'98 benchmark suite, each containing 10 to 30 nodes. The experimental results are compared with two existing methods that clearly indicate the acceptability of the suggested method. The suggested strategy achieves an average reduction of 24.69% and 31.46% for net cut, whereas average extensions of 15.20% and 12.31% are observed in sleep time when compared with two existing methods. The proposed method also achieves an average power efficiency of 14.98% and 12.09% with respect to these two state-of-the-art methods.

Keywords: VLSI Design; Partitioning; Multi-Objective Optimization; Genetic Algorithm; Net list; Sleep time; Net Cut.

1. Introduction

VLSI chips are widely employed in today's world, possibly in almost every sector of engineering, including computers, electronics, automobiles, voice and data communication networks, and many more. Thus, VLSI physical design and automation have become rapidly increasing industries. Reduced average power consumption in integrated circuits is a significant challenge for VLSI physical design. Researchers and industries are always making their best efforts to develop improved algorithms and methodologies for improving VLSI circuit performance. If the power consumption of the circuit is low, modern electronic gadgets such as smart phones, tablets, and laptops will have a longer battery life. This will minimize the chip's heat dissipation and energy consumption automatically. In CMOS devices, dynamic and sub-threshold leakage power should be decreased.

*Email: sidd_01_02@yahoo.com

Equation (1) can be used to measure the average dynamic power [1] of a circuit. Here, ***f*** denotes the operation's frequency, and ***C*** denotes the load capacitance, which comprises gate as well as wire capacitances**.** $\boldsymbol{V_{DD}}$ is the applied supply voltage, and ***N*** denotes the number of switches. The second term of the equation shows the power dissipation owing to leakage current, abbreviated as $\boldsymbol{I_{leak}}$.

$$A_{verage}=1/2\times C\times V_{DD}^{2}\times N\times f+I_{leak}\times V_{DD}^{2} \qquad (1)$$

The load capacitance (***C***) and switching activity (***N***) should be lowered to minimize dynamic and sub-threshold power. To achieve these objectives, a significant amount of effort and time was invested in developing a high-quality circuit partitioning algorithm. Partitioning is a method for dividing a complex component of the circuit into smaller components for easier and more efficient handling. Circuit partitioning is a difficult non-polynomial (NP) issue. It is more suitable to adopt some heuristic approaches or bio-inspired algorithms to improve the quality of circuit partitioning in a shorter amount of time. The first and most important purpose of these strategies should be to reduce the number of net cut sizes so that the number of partition-to-partition linkages is kept to a bare minimum. By minimizing net cuts among the circuit's partitions, the load capacitance gets reduced. Another goal of the partitioning strategies should be to maximize sleep durations while reducing switching activity. The components (or portions) are said to be in sleep mode when they exhibit no activity during a time interval. By using some control signals, power can be saved during this idle time. As a result, the whole system's power usage will be lowered. It is possible to maximize sleep time while minimizing transmission power loss [1].

Various optimization algorithms have been developed to address the existing VLSI design problems, notably in terms of optimizing the total wire length and area as traditional design factors. VLSI improvements in recent years have focused on extending design objectives and restrictions to include minimization of net cut and switching activity, too.

A genetic algorithm has higher parallelism, which means it has more parallel points. Chen et al. [2] suggest various strategies for achieving a better solution by altering the crossover point selection. Individual chromosomes were compared with the crossover operator in this procedure, and chromosomes were created based on the difference between the individual chromosomes. Yuen and Chow [3] developed an approach in which mutation operators were emphasized and the chromosomes were kept track of to minimize revisiting and lower the program's total run time. Another essential method developed by Jigang and Srikanthan [4] is efficient for hardware and software partitioning, which is concerned with improving the system's power estimation and overall running time. Gill et al. [5] suggested a genetic algorithm-based strategy for circuit partitioning, in which they discovered the average minimum and average net cut of circuits. They do, however, reveal that most previous methodologies are insufficient in terms of intelligent chromosomal selection for enhanced time. Arato et al. [6] offered a fascinating study in which partitioning is done using both integer linear programming (ILP) and genetic algorithms (GA). In this investigation, GA was found to be superior to ILP in terms of reaching system runtime. Prakash et al. [7] show a combination of Ant colony optimization and particle swarm optimization (PS-ACO) algorithms that deal with VLSI partitioning for multi-objective optimization with parameter cut net, delay, and sleep time. Another noteworthy work recently proposed by Prakash and Lal [8] was a Particle Swarm Optimization (PSO) based multi-objective VLSI circuit partitioning approach. But they employed bi-partitioning of circuits in both cases [7, 8].

Chellamani and Chandramani [9] suggested an effective optimization method. To optimize energy management, they applied a Satin Bowerbird Optimization (SBO) approach and a machine learning-based algorithm. An effective Satin Bowerbird Optimization (SBO) based VLSI circuit partitioning algorithm was devised by R. P. Guru and V. Vaithianathan [10]. They tested their method on the ISCAS'85 benchmark circuit, taking into account parameters like delay, area, and power. Finally, they established that their circuit partitioning method was efficient after comparing it with other methods like simulated annealing, PSO, ACO, etc.

Vinay Kumar et al. [11] reviewed 52 papers that were associated with optimization and mentioned their outcomes, including bio-inspired methods. They examined current trends in VLSI design improvements and future developments. A review of the many contributions of particle swarm optimization (PSO) and simulated annealing (SA), as well as VLSI design floor planning concerns, has been investigated.

The concept of multi-objective optimization was examined by Martins et al. [12] in the analog integrated circuit layout automation of placement procedures. They presented analog floor plan automation using simulated annealing and limited archive-based multi-objective optimization algorithms with better exploration of unworkable portions of the solution space. Funkej et al. [13] have studied optimal floor planning methods. On MCNC, there are block packing instances as well as an industrial example with hundreds of nets and 27 rectangles; they used their CONTAINMENT and SPARK algorithms for efficient floor layout. In addition, they demonstrate how to apply this approach to larger examples that cannot be addressed in a fair period of time in an optimal manner.

Several well-known VLSI circuit partitioning algorithms have been widely employed to solve a variety of VLSI layout optimization issues. There are still certain power and area issues that need to be addressed to keep up with current VLSI developments. These observations motivate us for this work. The main contribution of the proposed work can be listed as follows:

1. A hyper graph is used to represent the circuit partitioning problem, allowing complicated VLSI circuit relationships (connections) to be simply described over pair-wise relationships in a conventional graph.
2. Formulation of a new multi-objective fitness function that optimizes net cut and sleep time at the same time, considerably improving circuit performance, lowering system layout costs, and reducing power consumption.

In this work, a genetic algorithm-based multi-objective algorithm is introduced for k-way circuit partitioning that simultaneously optimizes sleep time and net cut. Because it separates the circuit into k-sub circuits, the suggested method is known as “k-way” (multi-way) circuit partitioning. The suggested method's performance is assessed using conventional benchmark circuits and compared to other current approaches. The proposed multi-objective algorithm's usefulness is demonstrated by the experimental findings.

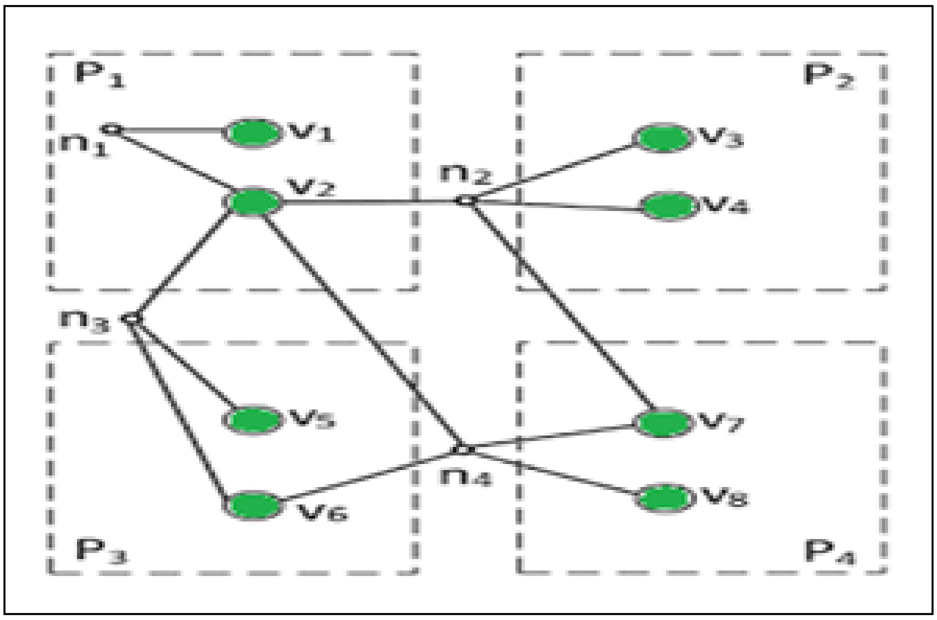

Figure 1: Hyper graph with 8 modules and 4 nets which is partitioned into 4 blocks.

2. Preliminaries

Definition 1: A hypergraph G = (M, Z) has a collection of vertices M which is called modules, and a collection of nets Z which is called hyper edges, among the modules. If a net ***n* ∈ *Z*** connects a set of modules ***v* ∈ *M*,** then net ***n*** is defined by the set of vertices ***v***. The set of vertices of a net ***n*** are referred to as its pins. The size of a net is determined by the number of pins in it.

Definition 2*:* A net ***n*** is considered to be cut in a hypergraph-based circuit partitioning if it connects more than one segment of a circuit block, i.e., the net is outside of a partition or block. If all connected modules with the net are within a partition, the net is considered to be uncut.

In a hyper graph, a hyper edge connects one or more vertices [14]. A hypergraph with 8 vertices $\{v_1, v_2, \ldots, v_8\}$ and 4 hyper edges $\{n_1, n_2, n_3, n_4\}$ is shown in Figure 1. The hyper edges n_1, n_2, n_3 and n_4 are defined by the set of vertices $\{v_1, v_2\}$, $\{v_2, v_3, v_4, v_7\}$, $\{v_2, v_5, v_6\}$ and $\{v_2, v_6, v_7, v_8\}$ respectively. There are four blocks in the circuit: P1, P2, P3, and P4. The nets n_2, n_3, and n_4 are severed in this example, leaving just n1 uncut. As a result, there are three (3) net cuts to be seen.

3. Problem Formulation

In this work, a GA-based multi-objective algorithm is introduced for k-way circuit partitioning that simultaneously reduces net cut and maximizes sleep time. Since maximizing sleep time also decreases the circuit's power consumption, the power efficiency is also calculated for the proposed method.

3.1 Net cut minimization

When a circuit is represented as a hypergraph, it is made up of a set of interconnected modules and nets. Every net is connected with multiple modules. Assume that the hyper graph H has ***m*** number of modules $T = \{t_1, t_2, \ldots, t_m\}$ and ***n*** number of nets $S = \{s_1, s_2, \ldots, s_n\}$. The circuit is divided into K blocks using the K-way partitioning algorithm. The purpose of this partitioning technique is to reduce the total number of nets cut. x_{ip} is set to 1 when the i-th module t_i is located in the p-th partition, otherwise, x_{ip} is set to 0. Similarly, if j-th net s_j connects the modules, all of which are within p-th partition, y_{jp} is set to 1; otherwise, y_{jp} is 0. Thus if net s_j does not provides cut, $\sum_{p=1}^{K} y_{jp} = 1$. Hence to minimize the net cut, this summation, overall net should be maximized. This can be mathematically expressed as Equation (2).

$$F_1 = \text{Maximize} \left(\sum_{j=1}^{n} \sum_{p=1}^{K} y_{jp}\right) \tag{2}$$

Equation (2) can also be expressed as equation (3).

$$F_1 = \text{Minimize} \left(n - \sum_{j=1}^{n} \sum_{p=1}^{K} y_{jp}\right) \tag{3}$$

Due to the fact that a module can be assigned to a single partition, module placement constraints can be specified by Eqs. 4 and 5.

$$\sum_{p=1}^{K} x_{ip} = 1 \tag{4}$$

and

$$B_{avg}*(1-\alpha/100) <= B_i <= B_{avg}*(1+\alpha/100) \tag{5}$$

where B_i represents the area of partition i. Since in the partitioning algorithm, area of each partition should be nearly equal with slight deviation, B_i should be between $B_{avg}*(1-\beta/100)$ and $B_{avg}*(1+\beta/100)$, where B_{avg} = (sum of areas of all modules / number of partitions) and β is the imbalance factor. This is an NP-hard problem that can be addressed using 0-1 linear integer programming [3].

3.2 Sleep Time Maximization

Recent studies [8] have demonstrated that sleep time can be maximized. As stated earlier, the hyper graph H has ***m*** number of modules T = {$t_1, t_2, ..., t_m$}. If ***i-th*** module t_i is in an idle state within the specified time frame ***(b, e)***, it is possible to put the module to sleep, where ***b*** and ***e*** denote the start and end of the time span, respectively. Two time periods (b_1, e_1) and (b_2, e_2) are said to be non-overlapping if $b_1 >= e_2$ or $b_2 >= e_1$. Assume that R_i indicates a set of non-overlapping intervals for t_i . During this time, it's possible to put the i-th module to sleep. R represents the entire idle sets of all modules within the circuit, i.e. R = {$R_1, R_2, ..., R_m$}. The time interval for a module may be empty, which indicates that the module has no idle time. If $b_1 \leq b_2 \leq e_2 \leq e_1$, it is assumed that the first interval covers the second. If I_1 and I_2 indicate two time intervals, then $I_1 \cap I_2$ is the collection of time units that both I_1 and I_2 have in common. As an example, if I_1 = (3, 7) and I_2 = (5, 9) then $I_1 \cap I_2$ = {5, 6, 7}. Again if R_j= {$I_{j1}, I_{j2}, ..., I_{jY}$} and R_i= {$I_{i1}, I_{i2}, .., I_{iX}$}, then $R_i \cap R_j = \{I_1 \cap I_2 \mid I_1 \in R_i, I_2 \in R_j\}$. ***D(t***$D(t_i, t_j)$ is the cardinality of $R_i \cap R_j$ and represents the total sleep duration between modules t_i and t_j.

Here, m denotes the total number of modules in the partitioning technique. Modules {$t_1, t_2, ..., t_m$} are partitioned into K blocks {$S_1, S_2, ... S_k$}. The sleep duration of p-th partition S_p i.e. ***SD*** (S_p) can be calculated as D ($t_i, t_j, \ldots, t_z$) where modules $t_i, t_j, \ldots, t_z$ belongs to S_p. Equation (6) can be used to maximize sleep duration in total for the circuit, where SW_p denotes the number of switches in the partition S_p. The amount of times a partition has to be turned off due to sleep is shown by the number of times it must be switched off over the overall time length. β is a parameter that is determined by the available technology and circuitry in modules, and controls the proportionate relevance of overhead terms (SW_p) and power savings ($SD(S_p)$) . In this experiment, β is taken into account as 1.

$$F2 = \text{Maximize} \left(\sum_{p=1}^{K} SD(S_p) - \beta \sum_{p=1}^{K} SW_p\right) \tag{6}$$

3.3 Composite function

To achieve dual objectives, i.e., minimizing net cut and maximizing sleep time duration, the composite objective function is defined in Equation (7).

$$F = \text{Minimize } (\mu_c \times F_1 + \mu_s \times 1/(1+ F_2)) \tag{7}$$

$\boldsymbol{\mu_c}$ and $\boldsymbol{\mu_s}$ denote the weights of the first and second objectives, respectively, such that $\mu_c + \mu_s = 1$. In this proposed methodology, equal weights have been assigned to these two objectives with $\mu_c = \mu_s = 0.5$.

3.4 Power efficiency

Let $\boldsymbol{PS_1}$ represent power consumption in sleep mode, and let $\boldsymbol{PS_2}$ denote power consumption when not in sleep mode in the suggested partitioning technique. Since there are K partitions (S1, S2,..., SK), PS_1 can be estimated as

$$PS_1 = P_o \times (T - S(S_1)) + S(S_1) \times P_s + P_o \times (T - S(S_2)) + S(S_2) \times P_s + \ldots + P_o \times (T - S(S_K)) + S(S_K) \times P_s$$

$$= K \times P_o \times T - (P_s - P_o) \times \{S(S_1) + S(S_2) + S(S_K)\}$$

Where ***S(Si)*** denotes the sleep time of the $\boldsymbol{S_i}$-th partition. $\mathbf{P_o}$ and $\mathbf{P_s}$ represent the power requirements of the partitions in active and sleep modes, respectively. T stands for the whole operation time. Because all modules are active during the whole operation time T when sleep mode is not considered, PS_2 may be calculated as

$$PS_2 = K \times P_o \times T$$

We typically have $P_o/P_s > 25$ [1] for a given memory chip. Thus, power efficiency can be estimated as

$$\begin{aligned}
P_{\text{efficiency}} &= 100 \times (PS_2 - PS_1) / PS_2 \\
&= 100 \times (1 - PS_1 / PS_2) \\
&= 100 \times (1 - [K \times P_o \times T - (P_s - P_o) \times \{S(S_1) + S(S_2) + .. + S(S_K)\}] / (K \times P_o \times T)) \\
&= 100 \times (1 - 1 - (P_s - P_o) \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times P_o \times T)) \\
&= 100 \times ((P_s - P_o) \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times P_o \times T)) \\
&= 100 \times (1 - P_o/P_s) \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times T) \\
&= 100 \times (1 - 1/25) \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times T) \\
&= 100 \times (24/25) \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times T)) \\
&= 96 \times \{ S(S_1) + S(S_2) + .. + S(S_K)\} / (K \times T))
\end{aligned} \tag{8}$$

For bi-partition K=2 power efficiency becomes

$$P_{\text{efficiency}} = \frac{48 \times (S(S1) + S(S2))}{T}. \tag{9}$$

4. Methodology

To produce the optimum bi-objective function given in Equation (7), a genetic algorithmic approach is applied. From the circuit description, two matrices are created. Net matrix N (n×m) and connectivity matrix C (m×m) are used to describe the circuit having m modules and n nets. Cij=1 in the connection matrix when the i-th module is linked to the j-th module via a net, else Cij=0. Similarly, in the Net matrix, Nij=1 if the i-th net is incident on the j-th module, else Nij=0. At first, an initial population is produced by dividing the circuit into K parts randomly. Chromosomes are used to symbolize the K-way partitions when the population is created. Then, to make new chromosomes, genetic operators such as crossover and mutation are applied to these chromosomes. To optimize the bi-objective function given in Equation (7), the fitness values of these additional chromosomes are determined. For the next generation, the best-fitting chromosomes are selected. These stages are repeated until the desired result is attained. The complete algorithm is listed below.

Proposed algorithm:
INPUTS: A circuit having n nets and m modules where m×m Connectivity matrix and n×m Net matrix, P_{mt}← Type of mutation, P_m← Probability of mutation, P_{ct} ← type of cross over, P_c ←Probability of crossover, G ←Total generations, P_{size} ←Size of the Population, $P_{\mathbf{activity}}$ ← Activity profiles of module, α← Imbalance factor .

OUTPUTS: Ideal solutions that achieve the objectives.
Step 1: Initial population: The starting population is created by slicing the circuits into K parts at random in order to satisfy the balance restrictions as described in Equation (5).

Step 2: Chromosome Encoding: In the beginning, the m-length chromosome $E_1E_2….E_m$ encodes each solution, and E_i=p when the i-th module is assigned to the p-th partition. Because K-way partitions are produced, a chromosome is a set of partition numbers ranging from 1 to K.

Step 3: Set generation number t=0.

Step 4: The rank of each **c**hromosome in the initial population is calculated by evaluating the fitness function $\mu_c \times F_1 + \mu_s \times 1/(1 + F_2))$ where F1 and F2 are given by Eqs. 3 and 6, respectively, assuming $\mu_c = \mu_s = 0.5$.

Step 5: Use the genetic operators mutation and crossover, depending on P_m, P_c, P_{mt}, and P ct, to generate a new chromosome set CS.

Step 6: Evaluation: The fitness function of each chromosome in CS is calculated using $\mu_c \times F_1 + \mu_s \times 1/(1 + F_2))$ where F1 and F2 are given by Eqs. 3 and 6, respectively, assuming $\mu_c = \mu_s = 0.5$.

Step 7: The population is updated with the chromosomes in CS whose fitness values meet the requirement.

Step 8: Increment the generation number, i.e., t=t+1.

Step 9: Repeat Step 6 to Step 9 while $t \neq G$.

Step 10: Output the optimal solutions.

Step 11: Record the optimized net cut and optimized sleep time.

Step 12: Calculate the power P_2 of the optimal solution.

Step 13: Calculate the power efficiency, $P_{efficiency}$ of optimized partition using Equation (8)

Step 14: Output power efficiency, $P_{efficiency}$.

Step 15: End.

Table 1: Representative circuits with number of nodes and number of nets.

SL. NO.	Two – Way Partitioning			Four –Way Partitioning		
	Circuit Description	**# Nodes**	**# Nets**	**Circuit Description**	**# Nodes**	**# Nets**
1	Spp_N10_E7_R1_1025	12	7	spp_N20_E20_R1_1344	22	20
2	Spp_N10_E37_R1_3228	12	37	spp_N20_E23_R2_2162	22	23
3	Spp_N11_E12_R1_3386	13	12	spp_N20_E23_R2_1909	22	23
4	Spp_N20_E20_R1_1344	22	20	spp_N20_E23_R2_1878	22	23
5	Spp_N20_E20_R2_942	22	20	spp_N20_E23_R2_1415	22	23
6	Spp_N21_E18_R2_1659	23	18	spp_N20_E22_R3_1063	22	22
7	Spp_N22_E22_R2_1232	24	22	spp_N20_E22_R2_3036	22	22
8	Spp_N23_E27_R2_1796	25	27	spp_N20_E22_R2_1529	22	22
9	Spp_N24_E25_R3_823	26	25	spp_N20_E20_R2_942	22	20
10	Spp_N25_E87_R3_812	27	87	spp_N20_E22_R2_659	22	22

5. Results and Discussions

On the ISPD 98 benchmark [16], the suggested method is evaluated. MATLAB 21a is used for implementation on an Intel Core i3 (10th Gen) system that has a RAM size of 8 GB. ISPD 98 contains 997 benchmark circuits with 10 to 30 nodes and 7 to 25 nets. The experiment has been carried out for two and four-way partitioning separately. Table 1 shows ten circuits for 2-way partitioning and ten circuits for 4-way partitioning, along with the number of nodes and nets in each of these benchmarks. Initially, the benchmark circuit descriptions are represented as a hypergraph in terms of the matrix of connectivity and the matrix of nets. Following that, the hypergraph is initially partitioned into two and four blocks randomly, maintaining the balanced condition, to create the initial population. On these initial partitions, multi-objective GA is used. The proposed fitness function optimizes net cut and sleep time simultaneously. The optimal net cut, sleep duration, and power efficiency for the representative circuits are illustrated after applying the proposed methodology, which is shown in Tables 2 and 3 for two-way and four-way partitioning, respectively. The power efficiency of the optimized partition is calculated using Equation (8).

To assess the acceptability of the proposed method, the experimental result is compared to that of other existing methods. Because the majority of works in the literature use the bi-partitioning method, the result of two-way partitioning is compared with two different methods proposed by Prakash and Lal [7, 8]. In [7], they proposed a circuit bi-partitioning method using PS-ACO, whereas in [8], only a PSO-based method is used. Comparisons are made concerning the number of net cuts, sleep time, and power efficiency and are given in Table 4. It can be observed from Table 4 that the numbers of net cuts are decreasing and sleep times, as well as power efficiencies, are increasing for most of the representative circuits. The average number of net cuts, sleep time, and power efficiency are also calculated for these ten representative benchmarks. The average net cuts are decreased by 24.69% and 31.46%, average sleep times are increased by 15.20% and 12.31%, and power efficiency is achieved at 14.98% and 12.09% with respect to [7] and [8].

Table 2: Optimized net cut, optimized sleep time and power efficiency for 2-way partitioning.

SL. NO	Circuit Description	Net cut	Sleep time	Power Efficiency
1	Spp_N10_E7_R1_1025	2	42	20.16
2	Spp_N10_E37_R1_3228	10	71	34.08
3	Spp_N11_E12_R1_3386	3	42	20.16
4	Spp_N20_E20_R1_1344	3	41	19.68
5	Spp_N20_E20_R2_942	4	43	20.64
6	Spp_N21_E18_R2_1659	7	42	20.16
7	Spp_N22_E22_R2_1232	7	41	19.68
8	Spp_N23_E27_R2_1796	10	42	20.16
9	Spp_N24_E25_R3_823	6	40	19.2
10	Spp_N25_E87_R3_812	9	43	20.64
Average		6.1	44.7	21.45

Table 3: Optimized net cut, optimized sleep time and power efficiency for 4-way partitioning.

SL. NO	Circuit Description	Net cut	Sleep time	Power Efficiency
1	Spp_N20_E20_R1_1344	5	7	7.64
2	Spp_N20_E23_R2_2162	12	7	7.64
3	Spp_N20_E23_R2_1909	11	10	10.91
4	Spp_N20_E23_R2_1878	10	9	9.82
5	Spp_N20_E23_R2_1415	9	9	9.82
6	Spp_N20_E22_R3_1063	10	18	19.64
7	Spp_N20_E22_R2_3036	11	10	10.91
8	Spp_N20_E22_R2_1529	9	9	9.82
9	Spp_N20_E20_R2_942	9	22	24.0
10	Spp_N20_E22_R2_659	10	8	8.73
Average		9.6	10.9	11.90

Table 4: Comparisons of proposed method with two existing methods proposed by Prakash and Lal [7, 8]

Circuit Description	Proposed method			PS-ACO based method [7]			PSO based method [8]		
	net cut	Sleep time	Power efficiency	net cut	Sleep time	Power efficiency	net cut	Sleep time	Power efficiency
Spp_N10_E7_R1_1025	2	42	20.16	4	38	18.24	4	40	19.2
Spp_N10_E37_R1_3228	10	71	34.08	8	36	17.28	9	34	16.32
Spp_N11_E12_R1_3386	3	42	20.16	4	36	17.28	4	32	15.36
Spp_N20_E20_R1_1344	3	41	19.68	6	39	18.72	7	42	20.16
Spp_N20_E20_R2_942	4	43	20.64	6	46	22.08	9	49	23.52
Spp_N21_E18_R2_1659	7	42	20.16	6	42	20.16	7	39	18.72
Spp_N22_E22_R2_1232	7	41	19.68	9	42	20.16	10	41	19.68
Spp_N23_E27_R2_1796	10	42	20.16	12	38	18.24	12	41	19.68
Spp_N24_E25_R3_823	6	40	19.2	9	32	15.36	9	17	8.16
Spp_N25_E87_R3_812	9	43	20.64	17	39	18.72	18	63	30.24
AVERAGE	6.1	44.7	21.41	8.1	38.8	18.62	8.9	39.8	19.1

5. Conclusions

A dual-objective evolutionary algorithm has been suggested for K-way circuit partitioning that maximizes sleep time while minimizing net cut size. The circuit partitioning problem is NP-hard. The initial population is created by arbitrarily partitioning a circuit in K-ways such that the balanced condition is met. Each chromosome is coded appropriately, and crossover and mutation operators are used to produce new solutions. To create a high-quality solution, the likelihood of crossing and mutation has been kept low. The GA can easily encode the design variable into bits because it is discrete by nature. For example, in 2-way partitioning, it is easy to encode the design variable from 1 to 2. The suggested method is used to improve circuit partitioning using the ISPD'98 benchmark suite. From the experimental results, it can be observed that the suggested technique improves the quality of the results significantly. The results are compared with two existing methods. The average improvement of net cut and sleep time is 24.69% and 15.20%, respectively, when compared with [7], whereas it is 31.46% and 12.31% when compared with [8]. Since the proposed method maximizes sleep time, the system's power usage has been reduced. The proposed method also achieves power efficiencies of 14.98% and 12.09% with respect to [7] and [8]. The proposed strategy will aid in achieving faster convergence without sacrificing solution quality.

References

[1] P. Ghafari, E. Mirhard, M. Anis, S. Areibi and M. Elmary, “A low power partitioning methodology by maximizing sleep time and minimizing cut nets,” *in proceedings of Fifth International Workshop on System-on-Chip for Real-Time Applications*, BaufAlberta Canada, pp. 368-371, 2005.

[2] Z. Q. Chen and Y. F. Yin, “A new crossover operator for real-coded genetic algorithm with selecting breeding based on difference between individuals,” *in proceedings of Eighth International Conference on Natural Computation (ICNC)*, 2012.

[3] S. Y. Yuen and C. K. Chow, “A genetic algorithm that adaptively mutates and never revisits,” *Evolutionary Computation,* vol.13**,** pp. 454-472, 2009.

[4] W. Jigang and T. Srikanthan, “Efficient algorithms for hardware/software partitioning to minimize hardware area,” *in proceedings of IEEE Asia Pacific Conference on Circuits and System*, pp. 1875-1878, 2006.

[5] S. S. Gill, R. Chandel and A. Chandel, “Genetic algorithm-based approach to circuit partitioning,” *International Journal of Computer and Electrical Engineering,* vol. 2**,** pp. 1793-1863, 2010.

[6] P. Arato, S. Juhasz, Z. A. Mann and D. Papp, “Hardware-Software partitioning in embedded system design,” *in proceedings of International Conference on Complex, Intelligent and Software Intensive Systems,* pp. 197-202, 2003.

[7] A. Prakash and R. K. Lal, “Hybrid PS-ACO Algorithm in Achieving Multiobjective Optimization for VLSI Partitioning,” *International Journal of Control Theory and Applications,* vol. 8, pp. 227-242, 2015.

[8] A. Prakkash and R. K. Lal, “PSO: An approach to multi-objective VLSI Partitioning,” *in proceedings of the 2nd International Conference on Innovations in Information, Embedded and Communication systems, IEEE,* 2015.

[9] G. K. Chellamani and P. V. Chandramani, “An Optimized Methodical Energy Management System for Residential Consumers Considering Price-Driven Demand Response Using Satin Bowerbird Optimization,” *Journal of Electrical Engineering & Technology*, vol. 15, pp. 955–967, 2020.

[10] R. P. Guru and V. Vaithianathan, “An efficient VLSI circuit partitioning algorithm based on satin bowerbird optimization (SBO*)*,” *Journal of Computational Electronics*, vol. 19, pp. 1232–1248, 2020.

[11] S. B. Vinay Kumar, P. V. Rao, H. A. Sharath, B. M. Sachin, U. S. Ravi and B. V. Monica, “Review on VLSI design using optimization and self-adaptive particle swarm optimization,” *Journal of King Saud University - Computer and Information Sciences*, vol. 32, pp. 1095-1107, 2020.

[12] R. Martins, N. Lourenço and N. Horta, “Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates,” *Expert Systems with Applications*, vol. 42, pp. 9137-9151, 2015.

[13] J. Funke, S. Hougardy and J. Schneider, “An exact algorithm for wire length optimal placements in VLSI design,” *Integration*, vol. 52, pp. 355-366, 2016.

[14] X. Ouvrard, “Hypergraphs an introduction and review,” 2020.

[15] S. Banerjee, S. Majumder and B. B. Bhattacharya, “A Graph-Based 3D IC Partitioning Technique,” *in proceedings of IEEE Computer Society Annual Symposium on VLSI*, pp. 613-618, 2014.

[16] Data Set: https://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/TESTCASES/netD20-29.tar.gz