Electronics and Electrical Engineering

https://www.aimspress.com/journal/ElectrEng

AIMS Electronics and Electrical Engineering, 9(3): 390–404.
DOI: 10.3934/electreng.2025018
Received: 06 March 2025
Revised: 22 June 2025
Accepted: 30 June 2025
Published: 08 July 2025

Research article

A balanced computational approach for multilevel VLSI circuit partitioning strategy with bat algorithm

Velamala Pavan Kumar and Aravindhan Alagarsamy*

Centre for Multi-Core Architecture Computation (C-MAC), Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, 522501, India

* **Correspondence:** Email: aravindhan.alagar@gmail.com.

Abstract: The selection of an appropriate partitioning approach significantly impacts integrated circuit (IC) size and performance. Inaccurate initial partitioning may lead to cascading issues throughout the physical design flow, affecting the performance of subsequent stages in the physical design process. Clustering the chip design into functional units and placing heavily communicating units in proximity can optimize wiring efficiency and improve overall chip performance. In this paper, we introduce a unique application of the bat algorithm (BA) to address the minimum cut cost in partitioning problems. Further, the BA is a nature-inspired metaheuristic algorithm capable of handling intricate and restricted optimization scenarios. The performance of the proposed partitioning with the BA was verified with several experiments on the MCNC and ISPD-98 benchmark datasets.

Keywords: VLSI physical design; circuit partitioning; cut-cost; partitioning algorithms; nature inspired algorithms; bat algorithm

1. Introduction

Partitioning is crucial for VLSI physical design, especially for large-scale circuits, as it identifies performance-critical areas. This enables optimized resource allocation, enhanced performance, and more efficient placement and routing. Further, it enables the designers to focus on the complexities of specific sections, reducing the overall design difficulty and improving the efficiency of later stages in the physical design process. Partitioning/clustering the design into well-defined blocks enables more efficient routing, minimizing the distance signals travel and improving chip performance [1]. This also leads to minimizing the distance between components, and we can reduce wire length, leading to lower power consumption and improved signal integrity, which is essential to ensure that high-performance circuits meet their timing requirements. Over the past decades, substantial research endeavors have been undertaken within the domain of VLSI partitioning. Due to the conflictive objectives, partitioning problems are considered NP-hard [2]. The selection of optimal solutions for NP-Hard problems requires careful consideration of individual objectives, goals, and the appropriateness of chosen algorithms concerning specific criteria. On the other hand, researchers have documented numerous heuristics that effectively solve optimization problems across NP-Hard domains. As a result, natural-inspired algorithms play a vital role in producing the optimized solution for most of the VLSI partitioning problems. Consequently, various heuristic and metaheuristic algorithms, such as genetic algorithms (GA) [3], simulated annealing (SA), and particle swarm optimization (PSO) [4]. have been developed to address challenging optimization problems. Recent trends show that the bat algorithm (BA) has emerged as a prominent metaheuristic algorithm, leveraging the echolocation behavior of bats to provide promising solutions. In this work, we propose a unique dimension of BA for effective partitioning with a minimum cut degree. Our primary objective of this work is to synthesize the strengths of existing algorithms to address the specific partitioning challenges encountered in BA [5–7]. Furthermore, the performance of the proposed partitioning with the BA is analyzed through various experiments run on the MCNC and ISPD-98 benchmark datasets. Similarly, the proposed work is compared with other algorithms like discrete particle swarm optimization (DPSO), satin bowerbird optimization (SBO), The whale optimization algorithm (WOA), and adaptive bird swarm optimization algorithm (ABSO). The experimental results indicate that the proposed work outperforms existing algorithms. The major highlights of the proposed work are outlined below:

- The proposed BA-based circuit partitioning aims to divide benchmark circuits into balanced partitions as like the property of the Kernighan–Lin (KL) algorithm.
- BA's random walk characteristic enables a broader exploration of potential solutions, avoiding premature convergence.
- The proposed circuit partitioning is designed to efficiently discover both promising starting points and the best possible solutions.
- It handles various constraints associated with circuit partitioning, such as balancing the number of nets between partitions or ensuring that specific modules remain together.
- Finally, the scope of this research extends beyond bio-inspired computing, evolving into the realm of natural computing. This field explores natural phenomena, aiming to emulate these processes to address real-world challenges.

The sections of the paper are presented as follows. In Section 2, we discusses the related works in the VLSI circuit partitioning approaches. In Section 3, we present the insights of the proposed BA-based circuit partitioning approach. In Section 4, we present the experimental results and discussions over proposed and existing partitioning approaches. Finally, In section 5, we summarize the conclusions and discuss potential future directions for VLSI circuit partitioning.

2. Related works

A comprehensive examination of relevant research findings on VLSI circuit partitioning utilizing heuristic and metaheuristic algorithms is presented in this section. Traditional algorithms such as KL and Fiduccia Mattheyses (FM) are efficient linear-time heuristics for partitioning VLSI circuits [8]. The KL algorithm works on a graph with $2n$ nodes, creating two balanced partitions. However, the KL algorithm struggles to partition imbalanced graphs effectively and requires excessive computational resources when dealing with large-scale networks. On the other hand, the FM algorithm provides a solution for imbalanced graphs, but its complex nature and tendency to get stuck in local minima can limit its effectiveness in finding the best possible solution.

As an alternative, in comparison to conventional approaches, natural-inspired heuristics demonstrate greater potential for effectively addressing the complexities of VLSI circuit partitioning. The work in [9] demonstrates the hybrid/memetic algorithm, which combines genetic algorithm (GA) and SA to solve the circuit partitioning problem. This memetic algorithm holds enhanced convergence speed and the ability to handle complex optimization problems. However, combining two algorithms may lead to an increased computational overhead, especially for large-scale problems.

In [3], the cluster-based optimization approach is explored for circuit partitioning. It provides another dimension in circuit partitioning to address the complexities of modern partitioning, emphasizing the role of effective clustering in achieving optimal results. By emphasizing the role of clustering, this technique provides the simplest way to address the challenges of modern circuit partitioning. Similarly, the work in [10] paper introduces a unique multilevel memetic algorithm specifically tailored for hypergraph partitioning. Although the algorithm is a significant improvement due to its uniqueness, its applicability may be restricted by computational efficiency concerns, particularly for very large hypergraphs.

The studies in [4] present the discrete particle swarm optimization (DPSO) and discrete firefly algorithm (DFFA) for successful circuit partitioning and it is inspired by the collective intelligence of bird flocks. Further, the DPSO and DFFA can easily get trapped in local minima, preventing it from exploring the search space. The research paper [11] offers a multilevel partitioning algorithm that leverages a refined Kernighan-Lin algorithm. The primary objective of the algorithm is to create an efficient circuit partitioning method to minimize connections between sub-circuits while ensuring balanced partitioning of the graphs. The algorithm could face performance challenges with extremely large datasets. Additionally, its focus on bipartition balancing may limit its versatility for solving diverse circuit problems.

The researchers in [12] explored the effective adaptation of ant colony optimization (ACO) and bee colony optimization (BCO) for VLSI circuit partitioning and demonstrates superior performance in terms of speed compared to Tabu search, GA, and SA. In addition, the support vector machine (SVM) approach is analyzed for the successive circuit partitioning problem. The authors in [13] present an ACO-based partitioning approach and highlight the algorithm's sensitivity to parameter settings. This suggests that careful tuning and experimentation are essential for optimal performance. Furthermore, scalability issues may arise when applying the algorithm to larger circuits, leading to increased computational demands and potentially longer processing times.

The researchers in [14] present a unique satin bowerbird optimization (SBO) algorithm for VLSI Circuit partitioning, which minimizes the cut cost by clustering the circuits into sub-circuits. SBO's performance is highly susceptible to premature convergence, a common limitation among metaheuristics. Additionally, the algorithm's scalability to large-scale circuits remains unexplored. The researchers in [15] investigated the whale optimization algorithm (WOA) and adaptive bird swarm optimization algorithm (ABSO) for the partitioning approach. Compared to traditional methods, this provides better results and adapts well to complex designs. However, both WOA and ABSO can converge prematurely to suboptimal solutions, especially for challenging VLSI designs. The quality of the solution depends on various factors, including the algorithm's parameters, the initial population, and the complexity of the design. On the other hand, developing solutions for physical design challenges in 3D IC becomes more challenging. Accordingly, the researchers in [16] introduce a unique architecture for via alignment and placement, facilitating easy connections between multiple tiers on a 3D FPGA. Moreover, an SA-based unbalanced partitioning method was developed for better utilization of 3D FPGA layers (tires) and to fit more highly active circuit gates onto the initial (bottom) layers to ensure efficient heat transfer.

A comprehensive review of existing literature reveals that the effectiveness of VLSI circuit partitioning and the minimization of associated performance factors are closely tied to the selection of a suitable optimization algorithm. Furthermore, the choice of a natural-inspired heuristic algorithm significantly impacts solution convergence time. As a result, we proposes a BA-based circuit partitioning (BCP), and the random walk characteristic of BA enables a broader exploration of potential solutions, avoiding premature convergence.

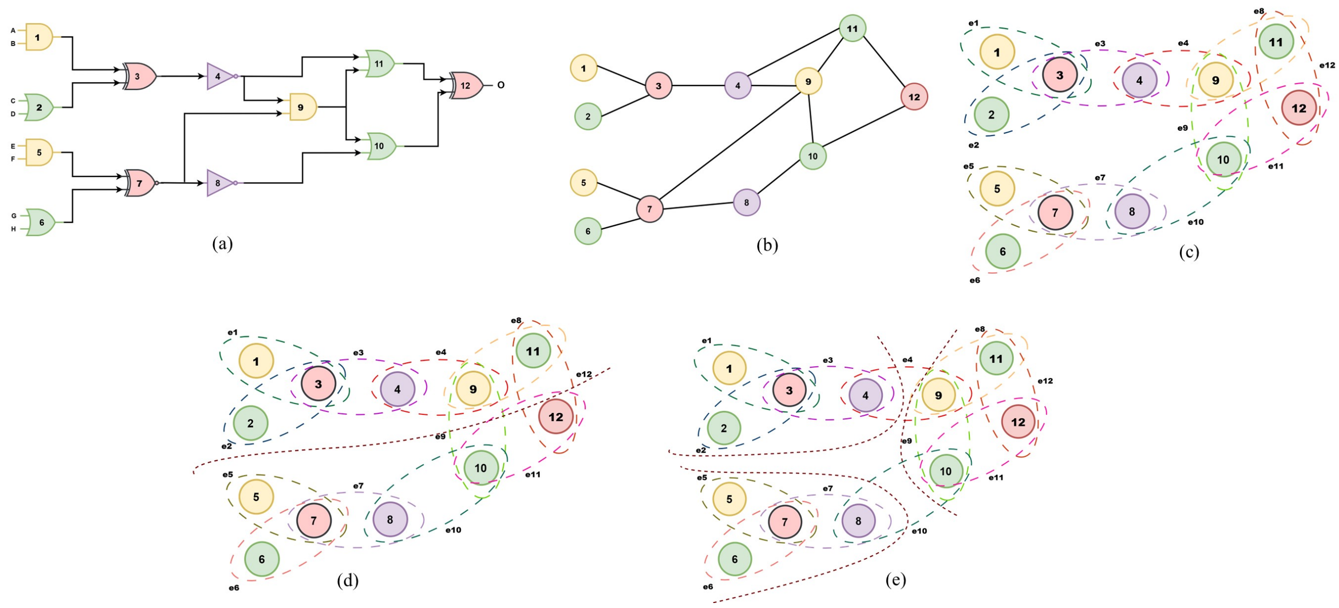

Figure 1. Circuit partitioning: (a) Sample logic circuit, (b) Simple graph representation, (c) Hypergraph via star transformation, (d) 2-Partitions, and (e) 3-partitions.

3. Bat algorithm based circuit partitioning

3.1. Hypergraphs and circuit abstractions

Due to their complex structure with multiple interconnected logic gates, VLSI circuits can be effectively represented by graph theory concepts, such as nodes for components and edges for connections. The structure of VLSI circuits can be perfectly captured by hypergraphs. In summary, the degrees of detail present in the hypergraph vary depending on the different types of level of abstraction in the VLSI circuit. At the gate level of abstraction, logic gates are represented as vertices, and the signal nets linking their input and output pins are represented as hyperedges. Furthermore, focusing on the cell level abstraction, the vertices denote standard/macro cells, and the hyperedge denote the nets connecting their terminals. In view of block level abstraction, the vertices indicate functional block, and hyperedge illustrate the inter communication between the functional blocks.

Although the interconnection structure of digital circuits can be naturally modeled using hypergraphs, where a net forms a hyperedge connecting multiple components, we adopt the star model in this work. In the star model, each net is decomposed into a set of edges from the net's driver to each of its receivers (e.g., a net connecting components {4, 9, 11} becomes edges {4, 9} and {4, 11} (See Figures 1(b) and 1(c)). This abstraction enables us to apply well-established graph-based algorithms while maintaining a faithful representation of signal flow. We clarify that while our approach is inspired by hypergraph connectivity, we use a graph model derived from hypergraphs via this star transformation.

3.2. Problem statement

Let $\mathfrak{G}$ be a hypergraph (star model) representing a benchmark circuit. $\mathfrak{G}(L, C)$ consists of a set of vertices L, representing logic gates, and a set of edges C, representing interconnections between pairs of vertices in L. Figures 1(a) and 1(c) represent the sample logic circuit and its star model equivalence, respectively.

$$L = \{l_1, l_2, \ldots, l_n\}, \quad |L| = n \tag{3.1}$$

$$C = \{c_{ij} = (l_i, l_j) \in L,\ i \neq j\} \tag{3.2}$$

Here, l_i represents a logic gate or component in the circuit, and c_{ij} denotes a connection (net) between gates l_i and l_j. We aim to partition the star model such that the cut size or the number of edges crossing partition boundaries is minimized, while maintaining a feasible and balanced number of partitions. To address the successive partitioning, the cost function for cut size $F_{\text{cut}}(p)$ for s number of subsets is defined as follows:

$$F_{\text{cut}}(p) = \min\left((\alpha + \beta) \sum_{a=1}^{s-1} \sum_{b=a+1}^{s} \left|\{c_{ij} \in C \mid (c_{ij} \cap B_a \neq \emptyset) \cap (c_{ij} \cap B_b \neq \emptyset)\}\right|\right) \tag{3.3}$$

where α is the weight assigned to minimizing cut size, and β is the weight assigned to achieving partition area balance. Both are user-defined constants ranging between 0 and 1. B_a and B_b denote distinct partition subsets. Let $\mathfrak{M}$ and $\mathfrak{N}$ represent the areas of the maximum and minimum allowed subsets, respectively. The balanced partition constraints can be expressed as:

$$\begin{cases} B_1 \cup B_2 \cup \cdots \cup B_s &= B \\ B_i \cap B_j &= \emptyset, \quad \forall i \neq j \in \{1, 2, \ldots, s\} \\ \mathfrak{N} \leq S(B_t) &\leq \mathfrak{M}, \quad \forall t \in \{1, 2, \ldots, s\} \end{cases} \tag{3.4}$$

Here, $S(B_t)$ denotes the size (area) of subset B_t, and the third condition ensures all partitions stay within the specified area bounds. A partitioning configuration $p \in P$ defines an assignment of all vertices in L to one of the subsets $B_1, \ldots, B_s$. Thus, the cost function $F_{\text{cut}}(p)$ evaluates the quality of each partitioning configuration p, guiding the optimization process.

3.3. Bat algorithm (BA): An overview

The BA is one of the popular nature-inspired algorithms that mimics the echolocation and hunting techniques of microbats. The echolocation behavior of microbats is characterized by emitting loud sound pulses and analyzing the returning echoes from surrounding objects. The sound pulse emitted by a microbat is a frequency-modulated signal that falls within the approximate frequency range of 25 kHz to 150 kHz. These frequency-modulated signals are characterized by short, single-burst emissions.

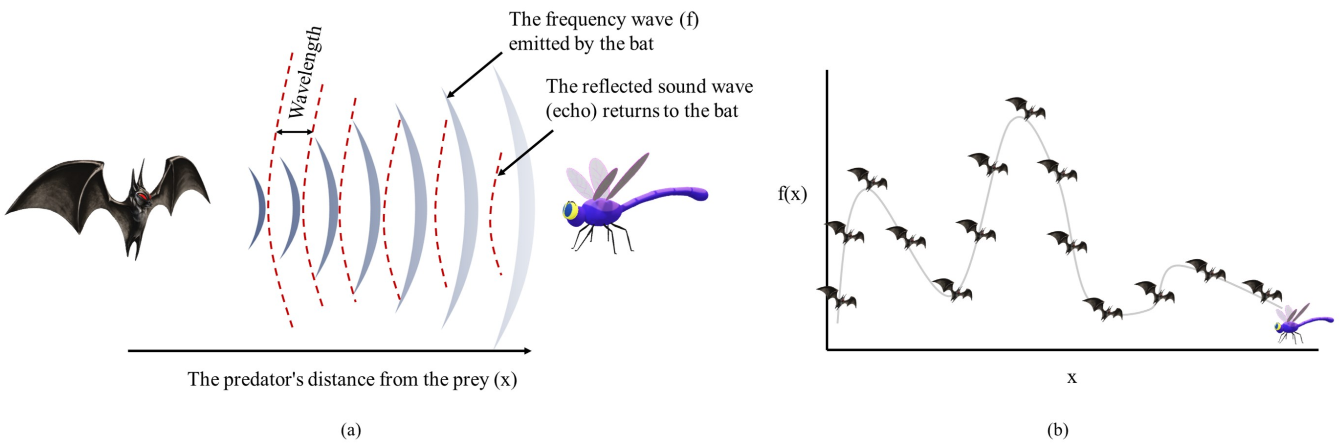
Wavelength
The frequency wave (f) emitted by the bat
The reflected sound wave (echo) returns to the bat
The predator's distance from the prey (x)
(a)
f(x)
x
(b)

Figure 2. Bat algorithm: (a) Echolocation behavior of the microbat and (b) trajectory of the microbat.

In addition, microbats can emit sound at a frequency of 15 to 20 bursts per second. On the other hand, upon approaching prey, the microbat will typically increase its pulse emission rate to approximately 200 pulses, and simultaneously, the sound of these pulses decreases. Accordingly, the microbat starts at a random position and then adjusts its location according to its emitted sound pulses and emission rates, reaching the prey. Upon identifying its prey, the microbat's echolocation changes: Loudness decreases while pulse emission rate increases. Figure 2a and Figure 2b represent echolocation behavior, and trajectory of microbat, respectively.

3.4. BA for circuit partitioning

The work by Xin-She Yang in [5] demonstrated the BA as a computational method inspired by the echolocation techniques of bats to locate their prey. To apply the BA to the s-way graph partitioning problem, it is essential to establish a clear correspondence between the BA's components and the partitioning problem's elements. Mapping of BA with partitioning elements is given below:

- The goal of s-way partitioning is to reduce the sum of weights of cut edges. According to BA, the partitioning cost function is directly used as the fitness function. In BA, the partitioning cost function is directly used as its measure of fitness. Each candidate solution (or "bat") is evaluated based on the total weight of cut edges, thereby guiding the optimization toward solutions with minimal inter-partition connectivity.
- The properties of each bat in a typical BA are determined by its location and velocity in the solution space. In the partitioning problem, a bat's location corresponds to a full assignment of each graph vertex to one of the s-partitions. The bat velocity in the BA acts as a governing factor for changes in the assignment, effectively indicating the rules or deviations applied to the current partition configuration.
- The generation of new solutions in BA (movement of bats) is designed to handle the discrete characteristics of partitioning problems. Reassigning a selection of vertices to different

partitions, informed by random search and the best-known solutions, leads to more effective bat updates. Further, it ensures the proper balance between diversification and intensification of bat populations.

- The completeness of the proposed BA and the stopping criteria are defined by the best fitness value remaining unchanged over a set number of consecutive iterations, thereby guaranteeing sufficient exploration and computational efficiency.

The rest of this section details the concrete implementation of this mapping and explores the step-by-step procedure for applying BA for successive partitioning.

3.4.1. Bat initialization and partitioning problem

Let $\mathfrak{G}(L, C)$ be the star-model graph of the circuit, where L is the set of vertices (logic gates) and C is the set of edges (connections). Let $n = |L|$ denote the total number of vertices in the circuit. For each vertex $l_i \in L$, we introduce a decision variable p_i, whose value indicates the partition to which the vertex is assigned. That is,

$$p = (p_1, p_2, \dots, p_n), \quad \text{where } p_i \in P_i = \{1, 2, \dots, s\},$$

and s is the total number of partitions. Each P_i is the domain of allowable partition indices for the i-th variable. We define the feasible space as the Cartesian product:

$$P = P_1 \times P_2 \times \cdots \times P_n.$$

Thus, when we write $p \in P$, it mean that each component $p_i \in P_i$ for all $i = 1, 2, \dots, n$. The goal of the BA's initialization stage in the partitioning problem is to identify an optimal configuration $p \in P$ that minimizes the partitioning cost. Specifically, the optimization problem is defined as:

$$\min_{p \in P} F_{\text{cut}}(p), \tag{3.5}$$

where $F_{\text{cut}}(p)$ is a cost function that measures the number of cut edges, i.e., those that span more than one partition.

3.4.2. Bat population initialization

In this stage, the random initialization of bat locations for partitioning is performed as outlined below

$$p_i^j = lb_i + (ub_i - lb_i).\Gamma \tag{3.6}$$

where $\Gamma \in [0, 1]$ is a uniformly distributed random number controlling the randomization of the bat's position within the allowable bounds defined by lb_i and ub_i. lb_i and ub_i denote the feasible range of values for the decision variable p_i, i.e., the possible partition index to which a vertex can be assigned. They ensure that bats (solutions) do not exceed valid partition index limits during initialization and movement phases. Furthermore, the randomly generated partitioning population is ranked by its cost function values like $F_{cut}(p^1) \leq F_{cut}(p^2) \leq \cdots \leq F_{cut}(p^N)$ and arranged in a matrix structure.

3.4.3. Bat movements

Each bat, p^j, is flying with a velocity of v^j, influenced by a randomly generated frequency, λ_j. The new updated bat location (p^{j+}) and it is velocity (v^{j+}) are represented as follows

$$\lambda_j = \lambda_{min} + \mid (\lambda_{min} - \lambda_{max}) \mid .\Gamma \tag{3.7}$$

$$v^{j+} = v_i^j + (p_i^j - p_i^{Gbest}).\lambda_j \tag{3.8}$$

$$p^{j+} = p_i^j + v^{j+} \tag{3.9}$$

where λ_{min} and λ_{max} represent the minimum and maximum frequency values, respectively. These are key tuning parameters in the bat algorithm, determining the search step size. The frequency controls the rate of bat movement across the solution space and thus balances exploration and exploitation. p^{Gbest} denotes the best overall solution found among all bats (i.e., the global minimum cut partition configuration discovered so far). It steers the global exploration, affecting the velocity and position update of each bat. The updated offspring bat location is calculated by adding a relatively small value to the coordinates of the existing parent bat location. As the parent bat gets closer to the offspring bat, this small value is inherited from the distance between the global best bat location and the parent bat.

3.4.4. Bat updating

To optimize local search, the algorithm creates new solutions for each bat by randomly exploring the search space using a random walk strategy. The updated bat can be expressed as follows

$$p^{j+} = p_i^{best} + \xi\Lambda^t \tag{3.10}$$

where ξ indicates the weight factor of random walk, with values varying between -1 and 1. The mean loudness of all bats at step time is represented by Λ. Finally, the newly derived bat location, based on Eqs 3.9 and 3.10, is computed as follows

$$p^{j+} = \begin{cases} p_i^{best} + \xi\Lambda^t & \text{if rand()} > \delta_i^t \\ p_i^j + v^{j+} & \text{otherwise} \end{cases} \tag{3.11}$$

where *rand*() is a uniform random number between 0 and 1. δ_i^t is the pulse rate of the *i*-th bat at time t. p^{best} represents the best solution found by a specific bat (individual solution) based on its historical movement and evaluations. It guides the local exploitation strategy through random walks in its neighborhood. The first condition in Eq (3.11) is applied when a randomly generated number is greater than the bat's pulse rate, favoring local search around the best solution. Otherwise, the second formula is adopted, which incorporates the bat's velocity and encourages exploration of the search space.

3.4.5. Update of bat population, pulse rate, and loudness

The objective function of newly updated bat population ($F_{cut}(p^{Gbest})$) is much better than $F_{cut}(p^{j+})$. Furthermore, as the iteration progresses, the loudness and pulse emission rate (δ) are dynamically recalculated as follows

$$\begin{aligned} \Lambda_i^{t+1} &= \eta.\Lambda_i^t \\ \delta_i^{t+1} &= \delta_i^0[1 - exp(-\mu t)] \end{aligned} \tag{3.12}$$

where Λ_i^t is the loudness of the i-th bat at iteration t. δ_i^t is its pulse rate, η is the loudness damping factor, and μ is the pulse rate increase coefficient. These control how aggressively bats exploit the solution space as iterations progress. Bats increase their pulse emission rate as they approach prey, but their loudness decreases. Figure 3 represents the flow diagram of the proposed mapping approach with BA.

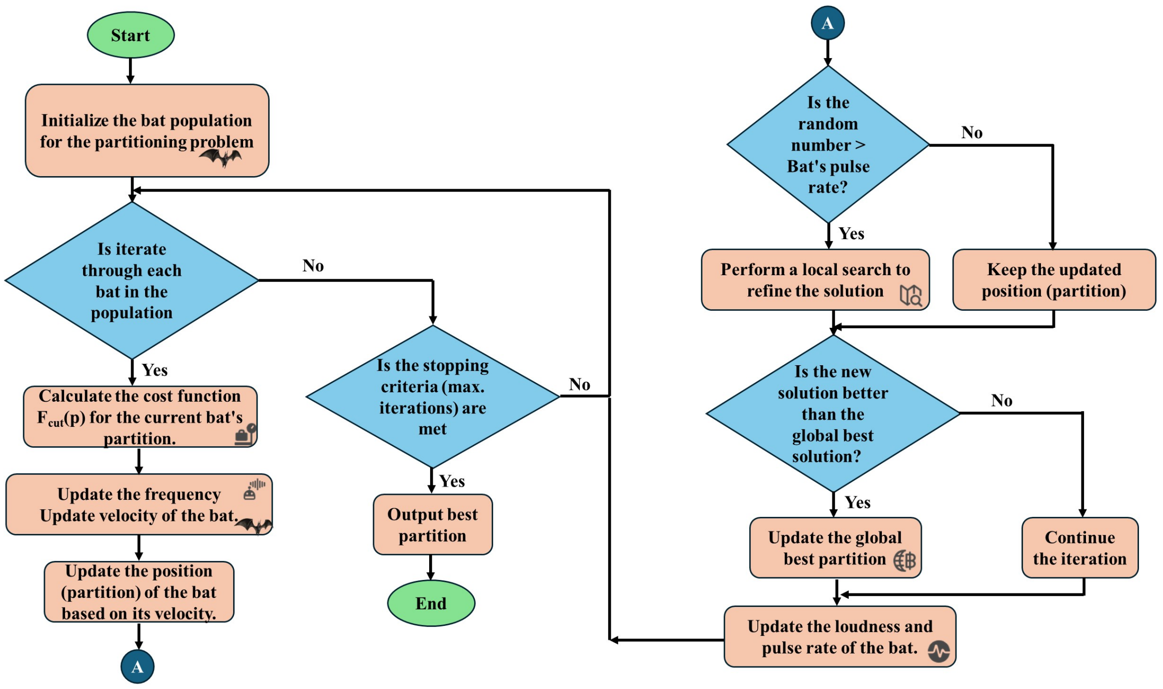
Start
Initialize the bat population for the partitioning problem
Is iterate through each bat in the population
No
Yes
Calculate the cost function $F_{cut}(p)$ for the current bat's partition.
Update the frequency
Update velocity of the bat.
Update the position (partition) of the bat based on its velocity.
A
Is the stopping criteria (max. iterations) are met
No
Yes
Output best partition
End
A
Is the random number > Bat's pulse rate?
No
Yes
Perform a local search to refine the solution
Keep the updated position (partition)
Is the new solution better than the global best solution?
No
Yes
Update the global best partition
Continue the iteration
Update the loudness and pulse rate of the bat.

Figure 3. Flow diagram of the proposed mapping approach with BA.

3.4.6. Post-processing and discretization

While the BA operates in a continuous space for efficient exploration and solution refinement, the final solution must comply with the discrete nature of the partitioning problem. Thus, after the algorithm terminates, the final bat position vector p^{Gbest} is converted into an integer vector p^{final} by rounding each component to the nearest integer and clamping it to the range $[1, s]$ as follows

$$p_i^{final} = min(s, max(1, round(p_i^{Gbest}))) \tag{3.13}$$

where s is the number of partitions. This ensures each vertex is assigned to a valid partition. Intermediate solutions during the optimization are retained as real-valued to facilitate smooth convergence.

4. Results and discussions

The proposed partitioning method is assessed using five microelectronics centers of North Carolina (MCNC) and ISPD-98 benchmark circuits. The key attributes of MCNC and ISPD-98 benchmarks are presented in Table 1 and Table 2, respectively. The proposed partitioning is implemented in python programming and it is executed on a PC with a 5.6 GHz Intel Core i7 processor and 16 GB of RAM.

Each benchmark circuit undergoes partitioning into 2, 4, 8, and 16-way partitioning, ensuring that the area distribution among the partitions is well-balanced. A comparative analysis is conducted between the BA-based circuit partitioning and the existing algorithms SBO [14], WO-ABSO [15], and DPSO [4] to determine their relative performance.

Table 1. Key attributes of the MCNC benchmark.

Benchmark Circuit	# Nodes/Vertices	# Edges/Nets
Fract	149	164
Primary1	833	905
Primary 2	3014	3030
Struct	1952	1920
Biomed	6417	5711

Table 2. Key attributes of ISPD-98 benchmark.

Benchmark Circuit	# Nodes/Vertices	# Edges/Nets
ibm01	12506	14111
ibm02	19342	19584
ibm03	22853	27401
ibm04	27220	31970
ibm05	28146	28446

To reduce computational time and obtain a promising multi-partitioning solution, BA restricts the bat population size to 100 individuals and limits the number of iterations to 50. The cut size factor (α) and balanced weight factor (β) are set to 0.9 to 0.5, respectively. The weight factor used in the random walk (ξ) process is dynamically determined based on the changes in the bat population. This factor can range from -1 to +1. Moreover, it is identified that the loudness factor (η) outperforms the value 0.5.

Figure 4 represents the proposed approach's peak fitness per generation to DPSO, SBO, and WO-ABSO on the ibm 05 benchmark circuit. A comparative analysis of the performance graph reveals that the proposed BA consistently achieves superior results compared to existing approaches. Table 3 highlights the differences in performance factors between the proposed partitioning approach and existing methods. Performance factors include 'cut' (cut size), 'D_{cp}' (critical path delay), and 'T_{cpu}' (computation time). Table 4 indicates the percentage of improvement of the proposed partitioning approach against the existing one.

The proposed partitioning approach outperforms existing methods DPSO, SBO, and WO-ABSO by an average of 76.9%, 52.5%, and 22.7% in terms of cut size, as shown in Table 4. Similarly, The proposed BA-based partitioning method achieves an average delay reduction of 32%, 22.5%, and 9.8% compared to DPSO, SBO, and WO-ABSO, respectively, in terms of critical path delays.

According to computation time, the proposed approach holds an improvement of 28.5%, 31.1%, and 13% over DPSO, SBO, and WO-ABSO, respectively.

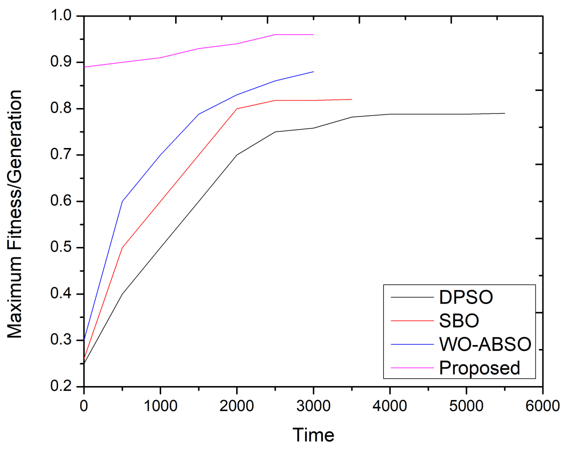

Figure 4. Performance of the peak fitness/generation on the ibm 05 benchmark circuit.

Table 3. Comparison of performance factors between the proposed partitioning approach against DPSO, SBO and WO-ABSO.

Benchmark Circuit	**DPSO**			**SBO**			**WO-ABSO**			**Proposed**		
	Cut	D_{cp} (ps)	T_{cpu} (s)	Cut	D_{cp} (ps)	T_{cpu} (s)	Cut	D_{cp} (ps)	T_{cpu} (s)	Cut	D_{cp} (ps)	T_{cpu} (s)
Fract	32	278	88	23	239	83	18	226	78	12	197	62
Primary1	41	531	191	37	443	177	31	389	167	26	325	153
Primary2	58	566	478	56	578	484	46	486	412	38	435	396
Struct	48	486	332	43	453	374	36	388	276	32	386	249
Biomed	74	564	629	69	593	689	52	579	595	47	567	516

Table 4. Comparison of percentage of improvement between the proposed partitioning approach against DPSO, SBO, and WO-ABSO.

Benchmark Circuit	**Over DPSO**			**Over SBO**			**Over WO-ABSO**		
	Cut	Dcp	Tcpu	Cut	Dcp	Tcpu	Cut	Dcp	Tcpu
Fract	96.7	41.1	41.9	91.7	21.3	33.9	50.0	14.7	25.8
Primary1	57.7	63.4	24.8	42.3	36.3	15.7	19.2	19.7	9.2
Primary2	52.6	30.1	20.7	47.4	32.9	22.2	21.1	11.7	4.0
Struct	50.0	25.9	33.3	34.4	17.4	50.2	12.5	0.5	10.8
Biomed	57.4	-0.5	21.9	46.8	4.6	33.5	10.6	2.1	15.3
Average	**76.9**	**32.0**	**28.5**	**52.5**	**22.5**	**31.1**	**22.7**	**9.8**	**13.0**

4.1. Analysis of the standard and proposed partitioning

This section analyzes the proposed partitioning approach by comparing it to standard methods such as the KL, FM, and SA algorithms. Table 5 shows the minimum and average cut size results and average computation time for the standard and proposed partitioning approaches. The KL and FM algorithms demonstrates the best and average cut sizes over 50 experimental runs. On the other hand, the proposed approach demonstrates improved efficiency, achieving successful partitioning in 20 runs, whereas simulated annealing requires 30.

Table 5. Comparison of performance factors between the proposed and standard partitioning approaches.

Benchmark Circuit	**KL (50 runs)**			**FM (50 runs)**			**SA (30 runs)**			**Proposed (20 runs)**		
	Min. Cut #	**Avg. Cut #**	**Avg. T_{cpu} (s)**	**Min. Cut #**	**Avg. Cut #**	**Avg. T_{cpu} (s)**	**Min. Cut #**	**Avg. Cut #**	**Avg. T_{cpu} (s)**	**Min. Cut #**	**Avg. Cut #**	**Avg. T_{cpu} (s)**
Fract	51	75	121	38	46	136	65	101	98	12	22	68
Primary1	74	96	248	45	65	257	52	76	225	26	38	164
Primary2	141	164	488	118	139	493	145	157	438	38	53	396
Struct	85	108	285	60	78	299	75	85	289	32	40	258
Biomed	96	121	621	78	85	686	83	92	611	47	54	536

Table 6. Comparison of standard deviation (σ) and percentage of improvement of proposed over standard partitioning approaches.

Benchmark Circuit	**Standard Deviation (σ)**				**Improvement of proposed approach (%)**		
	KL	**FM**	**SA**	**Proposed**	**Over KL**	**Over FM**	**Over SA**
Fract	7.45	5.67	2.56	0.64	76.47	68.42	81.54
Primary1	12.56	10.35	5.38	1.14	64.86	42.22	50.00
Primary2	15.78	14.76	8.83	2.89	73.05	67.80	73.79
Struct	17.35	10.76	6.91	2.23	62.35	46.67	57.33
Biomed	6.87	5.43	13.31	3.16	51.04	39.74	43.37
Average	**12.00**	**9.39**	**7.40**	**2.01**	**65.56**	**52.97**	**61.21**

Table 6 presents the standard deviation (σ) of the proposed approach compared to the standard partitioning approach, along with the overall percentage improvement achieved by the proposed approach. Standard deviation in this context measures the spread of cut size values of the proposed and standard partitioning approach across multiple runs, indicating that the typical amount each cut size deviates from the average. The lower standard deviation value indicates that the various cut sizes, across their runs, are tightly grouped around the mean. On the other hand, the higher standard deviation values indicate greater variability in cut sizes around their average. As a result, Table 6 indicates that the proposed partitioning yields the average smallest standard deviation (2.01) when compared to other methods over the 20 experimental runs. Similarly, the KL algorithm exhibits the highest standard deviation (12), compared to 9.39 for FM and 7.40 for SA. Furthermore, the proposed partitioning approach achieves average cut size improvements of 65.56%, 52.97%, and 61.21% compared to KL, FM, and SA algorithms, respectively.

4.2. Analysis of the complexity of the proposed and standard partitioning approaches

For a graph with n vertices, the basic KL partitioning algorithm typically has a time complexity of $O(n^3)$ per pass when dividing it into two parts. Under the condition that the KL algorithm executes m passes to achieve convergence, the total time complexity is given by $O(mn^3)$. The KL algorithm becomes computationally expensive for s-way partitioning when using recursive bipartitioning. Alternatively, the FM algorithm offers a notably better improvement over KL in time complexity. The FM algorithm's time complexity scales approximately linearly with the number of pins (N_p) in the circuit netlist, resulting in an $O(N_p)$ complexity. In the worst case, N_p exhibits an approximate proportionality to the product of the total number of vertices and the degree of a vertex. As a result, for larger circuits, the FM algorithm has a superior time complexity compared to the basic KL algorithm. On the other hand, the BA is a metaheuristic optimization algorithm. A metaheuristic algorithm's time complexity is influenced by factors such as the number of iterations (I), the population size (N), and the search space dimensions (n) (number of circuit components). Further, the time complexity of the BA is expressed as $O(INn)$. Hence, the effectiveness of BA's complexity is evaluated based on the quality of the solutions it finds within a reasonable time.

5. Conclusions

In this paper, A balanced multi-level circuit partitioning technique employing a BA is presented. The s-way partitioning has been adopted in this investigation. The proposed partitioning approach has conducted rigorous testing with MCNC and ISPD-98 benchmark circuits to evaluate its performance and effectiveness. Each benchmark circuit has been partitioned into 2, 4, 8, and 16 ways. The BA verified that the area was evenly distributed across all partitions at each level of the partitioning process. The performance of the proposed BA-based partitioning method compared with the existing algorithms DPSO, SBO, and WO-ABSO. Comparative analysis revealed that BA-based multilevel partitioning achieved an average cut size reduction of 76.9%, 52.5%, and 22.7% in comparison to DPSO, SBO, and WO-ABSO, respectively. In the feature, we aim to investigate the application of the BA for 3D partitioning and evaluate its effectiveness through extensive benchmarks.

Author contributions

Velamala Pavan Kumar: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Funding acquisition, Project administration; Aravindhan Alagarsamy: Methodology, Formal analysis, Writing – original draft.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors thanks DST - FIST for funding the lab facility for supporting this research under grant number S R/FIST/ET II/2019/450.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. Sait SM, Habib Youssef (1995) *VLSI Physical Design Automation: Theory and Practice*, 1 Eds., World Scientific, New York: IEEE Press, 35–75.
2. Yodtean A, Chantngarm P (2004) Hybrid algorithm for bisection circuit partitioning. *Proceedings of IEEE Region 10 Conference* 500: 324–327. https://doi.org/10.1109/TENCON.2004.1414935
3. Manikandan R, Vijay L (2014) Effective clustering algorithms for VLSI circuit partitioning problems. *Contemporary Engineering Sciences* 7: 923–929. https://doi.org/10.12988/ces.2014.4653
4. Rajeswari P, Chandra T (2018) Memetic multilevel hypergraph partitioning. *Proceedings of the Genetic and Evolutionary Computation Conference* 347–354. https://doi.org/10.1145/3205455.3205475
5. Yang XS (2010) A New Metaheuristic Bat-Inspired Algorithm. *Studies in Computational Intelligence* 284: 65–74. https://doi.org/10.1007/978-3-642-12538-6_6
6. Alagarsamy A, Gopalakrishnan L (2018) MBA: A new cluster based bandwidth and power aware mapping for 2D NoC. *Proceedings of Int. Conf. on Circuits and Systems in Digital Enterprise Technology*, 1–5. https://doi.org/10.1109/ICCSDET.2018.8821150
7. Shehab M, Abu-Hashem MA, Shambour MK, Alsalibi AI, Alomari OA, Gupta JN, et al. (2023) A Comprehensive Review of Bat Inspired Algorithm: Variants, Applications, and Hybridization. *Arch Comput Method Eng* 30: 765–797. https://doi.org/10.1007/s11831-022-09817-5
8. Sinha B, Laskar NM, Sen R, Baishnab KL (2015) Heuristics in Physical Design Partitioning: A review. *Int. Conf. on Innovations in Information, Embedded and Communication Systems*, 1–5. https://doi.org/10.1109/ICIIECS.2015.7192900
9. Shanavas IH, Gnanamurthy RK, Thangaraj TS (2010) A Novel Approach to Find the Best Fit for VLSI Partitioning - Physical Design. *Int. Conf. on Advances in Recent Technologies in Communication and Computing*, 330–332. https://doi.org/10.1109/ARTCom.2010.93
10. Andre R, Schlag S, Schulz C (2018) A survey on an optimal solution for VLSI circuit partitioning in physical design using DPSO & DFFA algorithms. *Proceedings of the International Conference on Intelligent Sustainable Systems*, 868–872. https://doi.org/ISS1.2017.8389301

11. Lei X, Liang W, Li KC, Luo H, Hu J, Cai J, et al. (2019) A New Multilevel Circuit Partitioning Algorithm Based on the Improved KL Algorithm. *Proceedings of Int. Conf. on High Performance and Smart Computing*, 178–182. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2019.00041
12. Manikandan R, Parameshwaran R, Prassanna J, Sekar KR (2019) A Study on Specific Computational Algorithms for VLSI Cell Partitioning Problems. *International Journal on Emerging Technologies* 10: 67–70.
13. Pavithra Guru R, Vaithianathan V (2020) Ant Colony Optimization Based Partition Model for VLSI Physical Design. *Proceedings of Int. Conf. on Computer Communication and Informatics*, 1–5. https://doi.org/10.1109/ICCCI48352.2020.9104175
14. Pavithra Guru R, Vaithianathan V (2020) An efficient VLSI circuit partitioning algorithm based on satin bowerbird optimization (SBO). *J Comput Electron* 19: 1232–1248. https://doi.org/10.1007/s10825-020-01491-9
15. Karthick R, Senthilselvi A, Meenalochini P, Senthil Pandi S (2023) An Optimal Partitioning and Floor Planning for VLSI Circuit Design Based on a Hybrid Bio-Inspired Whale Optimization and Adaptive Bird Swarm Optimization (WO-ABSO) Algorithm. *J Circuit Syst Comp* 32: 1–33. https://doi.org/10.1142/S0218126623502730
16. Rahimi H, Jahanirad H (2021) An evolutionary approach to implement logic circuits on three dimensional FPGAs. *Expert Syst Appl* 174: 114780. https://doi.org/10.1016/j.eswa.2021.114780