Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

A Framework for High-Level Synthesis of VLSI Circuits Using a Modified Moth-Flame Optimization Algorithm

M.R. Esmaeili [1], *S.H. Zahiri* [2,*], *S.M. Razavi* [2]

[1]*Department of Electrical Engineering, Faculty of Electrical and Computer Engineering, University of Birjand, Birjand, Iran.*

[2]*Department of Electrical Engineering, Faculty of Engineering, University of Birjand, Birjand, Iran.*

Article Info

Article History:
Received 22 February 2018
Reviewed 03 April 2018
Revised 05 July 2018
Accepted 05 November 2018

Keywords:
Digital VLSI circuits
Datapath
Digital filters
High-Level Synthesis
Moth-Flame algorithm

*Corresponding author's
Email Address:
hzahiri@birjand.ac.ir

Abstract

Background and Objectives: High-level synthesis (HLS) is one of the substantial steps in designing VLSI digital circuits. The primary purpose of HLS is to minimize the digital units used in the system to improve their power, delay, and area.
Methods: In the modified MFO algorithm presented in this paper, a hyperbolic spiral is chosen as the update mechanism of moths. Also, by presenting a new approach, a paramount issue involved in applying meta-heuristic methods for solving HLS problems of VLSI circuits has been disentangled.
Results: By comparing the performance of the proposed method with Genetic algorithm (GA)-based method and particle swarm optimization (PSO)-based method for the synthesis of the digital filters, it is concluded that the proposed method has the higher ability in the HLS of data path in digital filters. The best improvement is 2.78% for the delay (latency), 6.51% for the occupied area of the chip and 6.93% in power consumption. Another feature of the proposed method is its high-speed in finding optimal solutions, in a manner which, more than 21.6% and 12.9% faster than the GA-based and PSO-based methods, respectively on average.
Conclusion: The most important very large scale integration (VLSI) circuits are digital filters and transformers, which are widely used in audio and video processing, medical signal processing, and telecommunication systems. The complex, expansive, and discrete nature of design space in high-level synthesis problems has made them one of the most difficult problems in VLSI circuit design.

Introduction

HLS is one of the important levels in designing digital very large scale integration (VLSI) circuits such as digital filters so that improvements in this level have a higher efficiency than optimization at other lower levels. Besides, high-level optimization reduces the design time and increases the velocity of design at lower levels, leading to better circuit indices [1]. High-level synthesis (HLS) is a level of VLSI design wherein behavioral description is converted into a structural characteristic [2], [3]. This behavioral description is generally shown as a graphical schematic called the data flow graph (DFG). The obtained structural characteristic is a mapping corresponding to the initial behavioral description which describes how to implement the behavioral description.

For a unique and certain behavioral description, there could be some structural characteristics and it is of note that their outputs are to be equivalent to the output being considered in the behavioral description [4]. Typically, the aims of high-level optimization are to improve the circuit performance in terms of speed (delay), the occupied area on the chip, power consumption, construction cost, and the like. These goals are generally in conflict with each other, and any improvement in one might weaken another. Consequently, establishing a compromise between these conflicting goals is oftentimes required depending on the intended use. As an illustration, the performance and speed of digital filters are high-priority whilst the cost and power consumption rank next in devices such as high-powered computer systems used in servers and supercomputers. On the other hand, in high-volume market gadgets such as ASICs (Application-specific integrated circuits) for cell phones, tablets, and laptops, the final cost and power consumption are the main priorities in design, whereas the performance and speed have less priority. Therefore, it can be concluded that according to the type of intended use, the design space should be explored and the optimal and appropriate design should be selected [5].

Due to the complexity and spread of design space in HLS problems, making use of automated methods in solving them is highly indispensable [6], [7]. Moreover, if the dimensions of the problem increase, the design space will also be greatly growing up and it will be more difficult to find the optimal solutions. As a result, utilizing CAD-based automated methods is exceedingly felt.

It should be underscored that in [8], scheduling and binding in HLS have been conducted simultaneously in which priorities and time limits have been formulated under the Integer linear programming (ILP). As in [9], power optimization has been undertaken using the ILP and it is claimed that making use of this method will be practically impossible for large-dimension problems due to its very low runtime in reaching a response [10]. In [11], a scheduling DFG is optimized in terms of power by using the game theory where the operators have been considered as players whereas the fitness function is assumed to be the consumption power. It needs to be asserted that the computational load of this method is very heavy requiring a very large runtime, particularly for real problems having high dimensions. Indeed, this will be tremendously overpriced, increasing exponentially.

Another method to solve HLS problems is to initially use default schedules and then the problem is solved. Afterward, by repeatedly applying the transformations, the initial schedule is gradually improved [12]. The focal demerit of this method is that the obtained responses are highly sensitive to the transformation which is used. Also, [13] addressed HLS problem solving by using the parallel programming language (PPL) method. Moreover, the simulation annealing (SA) algorithm was addressed in [14] while in [15] the registers were simultaneously scheduled, allocated, and reduced using the SA algorithm. Another method based on the SA is SALSA introduced in [16] which has been employed to solve HLS problems. Besides, in [17], an improved firefly algorithm is used. In this method, the combination of firefly algorithm and Tabu search (TS) algorithm is used and finally the problem is optimized in terms of speed and power. The simplest solution is to execute HLS steps independently. In these methods, only one operator is scheduled at a time and the process continues in the same fashion until all the operators are scheduled. Having done the scheduling, functional units (FUs) are allocated. An illustration of this includes the as-soon-as-possible (ASAP) scheduling [18], as-late-as-possible (ALAP) scheduling [18], path-based scheduling (PBS) [19], and force-directed scheduling (FDS) [20]. In [21], the delay and power are optimized using the memetic algorithm. In [22], researchers have also proposed a method based on the bacterial foraging optimization algorithm (BFOA) to solve the HLS problems.

In [23], weighted sum Genetic algorithm (WSGA) method was recommended to optimize the delay and occupied area in datapath synthesis, where the WSGA algorithm was employed to simultaneously schedule and allocate FUs for the DFG synthesis. In [24], another instance of GA was introduced to explore the design space during the DFG scheduling. In [25], a multi-objective evolutionary approach named non-dominated sorting Genetic algorithm–II (NSGA-II) was presented in which scheduling and allocating of the FUs was undertaken simultaneously. In [26], with the intention of minimizing the schedule time, a PSO-based method was introduced by assuming that the problem is resource-constrained. In [27], a PSO-based method was introduced in which scheduling and binding in the filter's datapath is performed simultaneously while a weighted particle swarm optimization (WSPSO) approach was proposed in [25] to solve the HLS problem. In order to optimize the area and delay in HLS problems, a design space exploration method using the NSGA-II algorithm was recommended in [28] in which the linear regression was employed to calculate the area and delay objective functions. Moreover, the researchers in [29] introduced the ant colony optimization (ACO) algorithm whereas the Tabu search meta-heuristic algorithm was utilized to solve the HLS problem in [30].

The spread of the search space in the HLS problems as well as the three steps of scheduling, binding, and allocation, which are completely interdependent together, make attractive the use of population-based metaheuristic algorithms to solve such problems.

Algorithms such as GA, PSO, ACO and Tabu search methods are among the methods used in this field. These heuristic algorithms can provide a large group of solutions in an implementation.

This paper is pioneering in that it has scrutinized using a modified meta-heuristic MFO method when solving a complex VLSI circuit engineering problem for the first time. In the MFO algorithm, the hyperbolic spiral function is used to move the moths toward its corresponding flame. The obtained results not only confirm the efficiency of the MFO method in the complex problem of HLS of VLSI circuits (digital filters as an instance here) but also validate its superiority over other renowned evolutionary methods (such as GA) and swarm intelligence (such as PSO). It is overstated that based on the existing literature, large dimensions of the exploration design space have been an outstanding feature of the HLS problems causing troubles in the methods. This and many design constraints such as the limited number of FUs and the operators' execution priority have not only lowered the algorithm runtime, but also influenced the final results. In this research work, we have overcome the limitations by introducing a novel solution and by applying the optimization algorithm separately for each operator and updating the priorities at each stage. When applying this solution, the problem limitations are mitigated while the meta-heuristic method is directed towards more effective and useful areas of the exploration space. This implies that the agents do not incline towards the unbeneficial areas which fail to fulfill the requirements. Plus, improving the obtained response, this also increases the convergence speed of the algorithm.

The structure of this paper is organized as follows: Section 2 is intended to explain the high-level synthesis of VLSI circuits. In Section 3, the meta-heuristic MFO-based proposed method is presented, and Section 4 includes the results of the simulation. In this section, a comparison of the proposed method with the GA-based method and PSO-based method is also established. Finally, Section 5 is dedicated to the conclusion.

High-level Synthesis of VLSI Circuits

Synthesis is in front of the analysis, in which the features and behavior of the circuit are taken as inputs and according to the existing resource constraints and intended objectives, the circuit is designed and implemented on the chip.

The synthesis of a digital filter involves different levels and stages. The first level in synthesis is to define the behavioral description of the circuit, which describes how to implement the commands and arrange the placement of operators. It is commonly characterized as a graph called DFG, in which the inputs and outputs and, interconnections between them have been specified. Indeed, all the operators in a behavioral description and all the relationships between them are represented graphically. The next level is to convert the behavioral description into a structural characteristic. This level is called register transfer level (RTL). Thereafter, there is the logical level in which the RTL design obtained from the previous level is converted to the logical gates level by logical synthesis methods and, according to the obtained design, the logic gates are placed.

The next step in designing a digital filter is called the layout level which helps us to determine the type of transistors and the required technology to implement the circuit. Furthermore, implementation of the logic gates on the chip area can also be considered at this level. To end with, the final step in designing a digital filter is the implementation of the circuit on the chip via available technologies.

HLS is the first step in synthesizing a circuit where the behavioral description is converted into a structural characteristic. As stated earlier, the behavioral description is generally represented by a graph called DFG. Fig. 1 illustrates a hypothetical DFG for (1).

$$Y = \Big(\big((a+b)\times(c\times d)\big) \\ + \big((e+f)\times(g\times h)\big)\Big) \\ + \big((e+f)\times(g\times h)\big) \tag{1}$$

where *a*, *b*, …, *h* are the output and *Y* is the input.

In each DFG, the nodes represent the operators and, the edges determine the dependencies between these operators. For example, in Fig. 1, there are four adder operators and four multiplier operators to perform and calculate (1). The outputs of operators 1 and 2 are considered as the inputs of operator 5 and, the outputs of operators 3 and 4 are considered as the inputs of operator 6. This can also be considered for other nodes and operators.

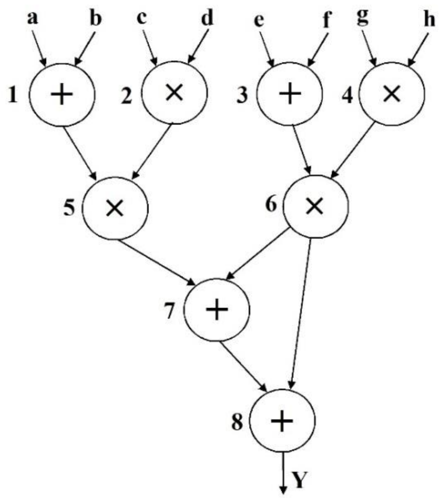


Fig. 1: Data flow graph for (1).

The simplest but not necessarily the most optimal way to implement each DFG is to map each operator or node one by one into a distinct FU. For instance, in Fig. 1, the desired description can be implemented and executed by 4 adder and 4 multiplier units. Nonetheless, the fact is that this may be feasible and affordable for DFGs with a low number of nodes, but by increasing the number of nodes and increasing the number of operational units, the final cost (in terms of power consumption or area) greatly increases. As a result, for reducing the number of FUs, another approach can be employed in which those nodes doing the same operation can be executed by a specific FU albeit at different time steps.

Datapath synthesis involves scheduling, allocation, and binding steps. Fig. 2 shows these three steps in HLS.

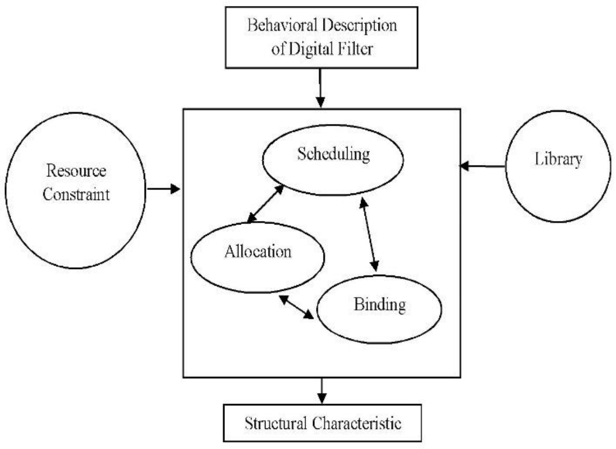

Behavioral Description of Digital Filter
Resource Constraint
Scheduling
Allocation
Binding
Library
Structural Characteristic

Fig. 2: HLS subtasks.

Scheduling is a stage of HLS that determines the time step in which a particular node in the DFG is to be executed. The number of all steps required for executing all the operators is called schedule length.

A sample of the scheduling for the DFG presented in Fig. 1 is shown in Fig. 3. By having a closer look, it is observed that the schedule length is equal to 4 because four steps are required for all the operators to be executed leading to an output. Nodes 1 to 4 are executed in the first time step, nodes 5 and 6 in the second time step, node 7 in the third time step, and finally, node 8 is executed in the fourth time step.

Allocation is a stage of HLS in which the number of hardware resources required to execute the operators as well as the number of required registers is determined. Binding assigns an FU for executing each node. For example, two possible ways for DFG scheduling in Fig. 1 can be seen in Fig. 3 and Fig. 4.

Mobility allows some nodes to be executed at different time steps without affecting the overall nature and output of the problem. For example, in the scheduled DFG in Fig. 4, node 3 is moved to the second time step, node 6 is transported to the third time step, node 7 is moved to the fourth time step, while node 8 is sent to the fifth time step. As can be seen, both Fig. 3 and Fig. 4 exhibit the same DFG but with different scheduling. In Fig. 3, the schedule length is equal to 4 while it is 5 in Fig. 4. Despite the increase in the schedule length in Fig. 4 compared to Fig. 3, the number of FUs has decreased compared to Fig. 3. In the scheduled DFG in Fig. 3, the number of required adders and multipliers is 2 and 2, respectively, which is reduced to 1 and 2 for Fig. 4, respectively.

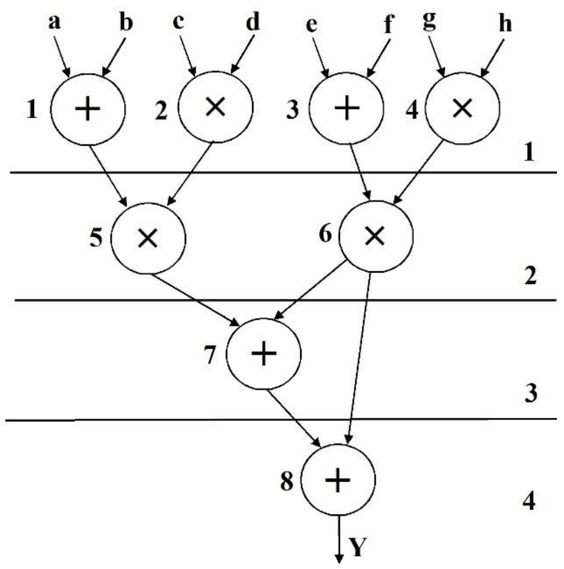


Fig. 3: Illustration of a scheduled DFG presented in Fig. 1.

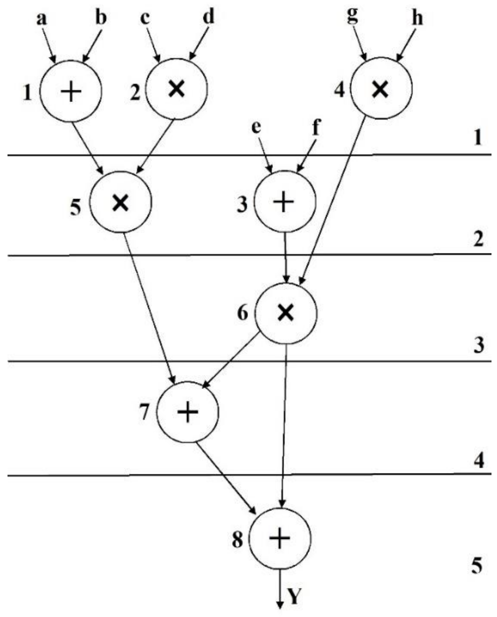
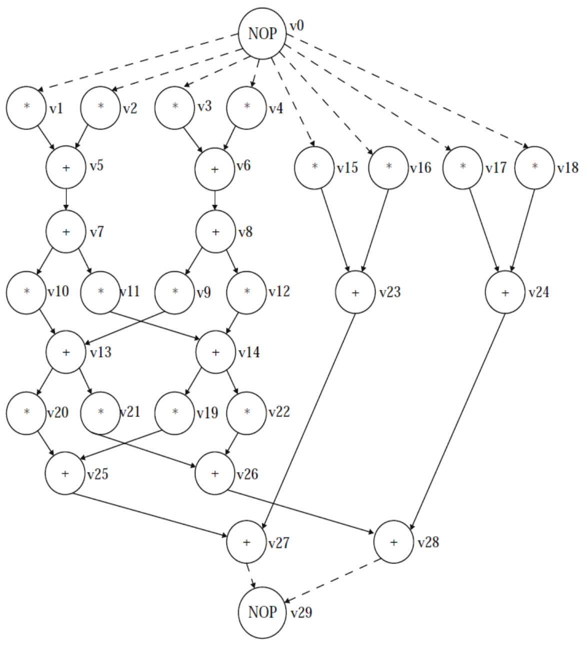


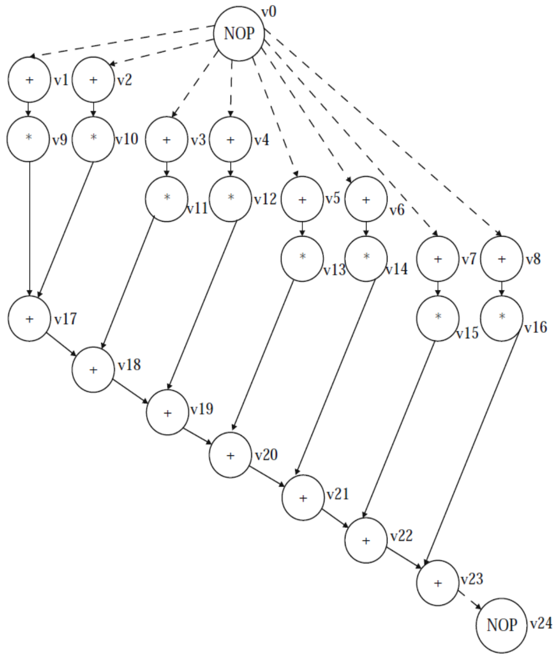
Fig. 4: Another way of a scheduled DFG for Fig. 1.

Digital transformers and digital filters have a special place in the VLSI circuits. So that in most circuits and applications we can find a snare of them [31]-[34]. However, digital filters are the most commonly used circuits in digital circuit design which are widely used in signal processing applications.

Digital filters are abundantly used to process signals, images, and videos, communication applications, and so on. The infinite impulse response (IIR), the finite impulse response (FIR), the auto regressive filter (ARF), the band-pass filter (BPF), the elliptic wave filter (EWF) and the wave digital filter (WDF) are the digital filters used in this paper. The data flow graph of the two ARF and FIR filters used in this paper has been shown in Fig. 5 [35].



(a)



(b)

Fig. 5: (a) ARF data flow graph, (b) FIR data flow graph.

The Proposed Method

The nature of HLS problems in which the search space is highly widespread and complex and discreteness of this space, has led the researchers to take advantage of the metaheuristic methods in this field. In this paper, the modified MFO-based approach is used to simultaneously optimize delay, area, and power consumption in HLS of datapaths in digital filters. Finally, the proposed method is compared with the GA-based and PSO-based methods. The MFO algorithm is one of the metaheuristic algorithms which was proposed by [36]. This algorithm, like other metaheuristic methods, is inspired by nature to find optimal solutions. The idea of the MFO algorithm is originated from the moths and their instinctive navigation system and their flights towards light sources.

In the MFO, moths can be considered as the chromosomes in GA, the swarm in PSO, and the ants in ACO. The variables and dimensions of the problem are the same as the position of the moths in the design space. The moths' position can be defined as (1):

$$M = \begin{bmatrix} m_{1.1} & m_{1.2} & \cdots & m_{1.d} \\ m_{2.1} & m_{2.2} & \cdots & m_{2.d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n.1} & m_{n.2} & \cdots & m_{n.d} \end{bmatrix} \tag{1}$$

where n is the number of moths, d is the number of variables of the problem and $m_{i,j}$ is the i-th variable of the j-th moth. The corresponding fitness values of the moths can also be stored in a matrix as follows:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{2}$$

where n is the number of flames and OM_i is the fitness value of i-th moth. Two other matrices can be defined for flames in the form of (4) and (5).

$$F = \begin{bmatrix} f_{1.1} & f_{1.2} & \cdots & f_{1.d} \\ f_{2.1} & f_{2.2} & \cdots & f_{2.d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n.1} & f_{n.2} & \cdots & f_{n.d} \end{bmatrix} \tag{3}$$

where n is the number of moths, d is the number of variables of the problem and $F_{i,j}$ is the i-th variable of the j-th moth.

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{4}$$

where n is the number of moths and OF_i is the fitness value of i-th flame.

Here, both the moths and the flames are feasible solutions of the problem; yet, the difference is that the flames are the best position of the moths to this moment, and the moths are the solutions to move in the search space for reaching to the optimal solution.

Indeed, each moth moves around a flame until it finds a better position and replaces it with the flame position. To escape from the local optima and increase the exploration capability, each moth updates its position only with its corresponding flame in the flame sorted matrix so that more space would be explored. For instance, the first moth in the moths' matrix is updated with the best flame and, the last moth is updated with the worst flame in the sorted matrix of the flames.

In addition to the basic MFO algorithm, other methods have been developed based on this algorithm. In [37], an improved version of MFO algorithm based on Lévy-flight strategy, which is named as LMFO, is proposed. Lévy-flight can increase the diversity of the population against premature convergence and make the algorithm jump out of local optimum more effectively. In [38], Chaos theory and crossover processes are introduced in MFO algorithm which increases the randomness or diversity. Chaotic systems have properties like randomness, certainty and ergodicity which help the solution to jump out of local minima. In [39], a Cauchy distribution function is added to enhance the exploration capability, influence of best flame has been added to improve the exploitation and adaptive step size and division of iterations is followed to maintain a balance between the exploration and exploitation.

A. Defining the samples position

An example of the position of a moth for the DFG shown in Fig. 1 can be indicated as {4 1 3 6 2 5 7 8: 1 2}.

In this definition, the position of each moth (or flame) has been divided into two parts. The left part indicates the order and priority to execute the nodes in the DFG. The right part characterizes the number of the FUs of each type. In this example, there are one adder and two multiplier FUs. The order for the number of the nodes in the priority part specifies the node which has to be executed earlier. As in the example above, nodes 4, 1, and 3 are executed respectively, and this process lasts until all the nodes are executed. It should be noted that the priority to execute the nodes must be taken into account in all moths and flames. For instance, in DFG shown in Fig. 3, node 6 has to be executed after the execution of nodes 3 and 4 because the outputs of the operators 3 and 4 are required for its execution, but the same node (node 6) is independent of nodes 1, 2, and 5, and it can be executed before or after them.

In order to schedule the mentioned moth, we start from the first node and place the nodes in the time steps according to the available resources. In this example, first, the node 4, which is a multiplier, is placed at the first time step and one of the two multiplier functional units is occupied. Then, node 1 is checked and because there is an unused adder FU, this node can also be executed at this time step. Since the only adder unit in this time step is being used to execute node 1, node 3 as an adder cannot be executed in this time step. Now it is time to execute node 6, but since the execution of this node depends on the output of nodes 3 and 4, these outputs are not yet available, so this node may not be executed at this time step. Then, node 2 as a multiplier needs to be executed; nonetheless, because there is still an unused multiplier unit and this node is not dependent on the other nodes, so node 2 is executed in this time step. This process continues until the executable nodes in the first step are identified. The same process is repeated for the rest of the nodes so that an FU will be allocated to each node. As such, the entire nodes are scheduled and the obtained scheduling for the mentioned example has been illustrated in Fig. 4. This process is carried out for all moths and flames. In the next step, for updating the position of the moths using the position of the corresponding flames, the method described in the following is used and the new position of the moth is calculated. The MFO algorithm is applied separately to both the priority and the FUs part. In the former part, the first node on the left is considered as the starting point. Afterward, the successor and the predecessor allowable location of this node is found which is saved as the upper and lower bounds of the node, as indicated by *ub1* and *lb1*, respectively. Subsequently, using (6) which indicates the moths' movement towards the flame, the new location of the given node is calculated.

$$m_{i.l(New)} = \left(d_{i.l} \times \frac{\cos(2\pi t)}{t}\right) + f_{j.l} \qquad (5)$$

where $m_{i,l}$(New) is the new position of *l*-th variables of *i*-th moth, *t* is a random number in [-1, 1], $f_{j,l}$ is the position of *l*-th variables of *j*-th flame and $d_{i,l}$ indicates the distance between the position of *l*-th variables of *i*-th moth and the position of *l*-th variables of *j*-th flame that is obtained by (7).

$$d_{i.l} = \left|f_{j.l} - m_{i.l}\right| \qquad (6)$$

where $d_{i,l}$ indicates the distance between the position of *l*-th variables of *i*-th moth and the position of *l*-th variables of *j*-th flame, $m_{i,l}$ is the position of *l*-th variables of *i*-th moth, and $f_{j,l}$ is the position of *l*-th variables of *j*-th flame.

Such a process applies in turn to all the nodes in the moth's priority part. As for the FUs part, an equation is used similar to (6), except for the fact that instead of the location of the variables, the number of the FUs is calculated. Moreover, the lower and upper limits of each FU, which are indeed the lowest and highest allowable numbers of that type of FU, are predefined and constant. Lastly, the moth's new position is attained after determining all the variables, whether in the priority or the FU parts. Also, in order to improve the exploitation of the algorithm, the number of flames in each iteration is updated through (8).

$$Flame_{Number} = Round(N - I \times \left(\frac{N-1}{Max_i}\right)) \tag{7}$$

where *FlameNumber* is the number of flames, *N* is the maximum number of flames, *I* is the current iteration and *Max*$_i$ is the maximum number of iterations. Therefore, all the moths move around the best remaining flame in the final steps of iterations. Other methods have been proposed to increase the exploration and exploitation of the MFO algorithm.

B. The fitness function

In order to evaluate the parameters of delay, area, and power in the proposed method and compare its performance against the previous methods, the following Fitness function is used:

$$Fitness = W_1 \times \frac{L_t}{L_{Max}} + W_2 \times \frac{A_t}{A_{Max}} + W_3 \times \frac{P_t}{P_{Max}} \tag{8}$$

where *Fitness* is the fitness function, L_t is the schedule length of sample evaluated, L_{Max} is the longest schedule length in the current generation, A_t is the total number of transistors in the operators and registers, A_{Max} is the largest area in the current population, P_t is the power consumption of the all operators and P_{Max} is the highest power consumption in the current population. W_1, W_2, and W_3 are the weights of the delay, occupied area, and power consumption terms, respectively. These three coefficients according to which of the parameters of time, occupied area or power consumption is optimization priority, are selected in such a way that their sum is always equal to one.

The number of operators was obtained directly from the number of FUs and the number of registers was obtained by the LEA method [20].

Results and Discussion

The results of the simulation of the proposed method as well as the algorithm-based method of GA [23] and PSO [40] and a comparison of these three methods are presented in this section.

All three methods are implemented in MATLAB software (R2015b) under the system with Core i7 6700HQ processor and 8GB of RAM. The initial population and the maximum number of iterations, in all three methods, is equal to 30 and 100, respectively. In GA, the mutation probability is equal to 0.1 and the crossover probability is equal to 0.9. Each algorithm is executed 50 times and the average of the obtained responses is given as the final response. The maximum number of resources and operational units are assumed to be equal to 5 for the entire cases.

A. Results

Tables 1 to 6 show the results of the synthesis of the digital filters. In these tables, the delay term is the same as the schedule length that represents the time steps required to execute the DFG. The area represents the total number of transistors needed to implement the operators and registers, and the power is the total power consumption of the FUs [41]. In all the Tables, W_1, W_2, and W_3 vary in three different modes notwithstanding having a total sum being always equal to 1. For each mode, the average of the acquired responses for a 50-time execution of the algorithm has been tabulated along with their relevant standard deviations. Due to a large amount of the occupied surface area and the power consumption, standard deviation of the response logarithms has been used to better represent and comparison of the data.

Table 1: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on IIR DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W_1=0.8 W_2=0.1 W_3=0.1	Ave.	5	5868.8	6324.75	5.08	6369.28	6917.47	5	6442.88	7007.66
	Std.	0	0.007	0.007	0.274	0.095	0.107	0	0.062	0.068
W_1=0.1 W_2=0.8 W_3=0.1	Ave.	7.02	3076.48	3147.24	7.46	3209.92	3222.89	7.42	3199.68	3217.85
	Std.	0.318	0.010	0.006	0.503	0.023	0.018	0.538	0.025	0.021
W_1=0.1 W_2=0.1 W_3=0.8	Ave.	7.12	3090.88	3147.24	7.3	3228.16	3207.76	7.3	3210.24	3187.59
	Std.	0.328	0.011	0.006	0.580	0.02	0.018	0.544	0.017	0.015

Table 2: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on FIR DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W_1=0.8 W_2=0.1 W_3=0.1	Ave.	9.1	7029.44	7198.70	9.36	7923.52	8152.76	9.3	7855.04	8084.97
	Std.	0.303	0.060	0.069	0.485	0.072	0.083	0.462	0.077	0.088
W_1=0.1 W_2=0.8 W_3=0.1	Ave.	14.8	3542.08	3162.37	15.06	3763.2	3232.98	15	3742.72	3222.89
	Std.	0.606	0.013	0.010	0.712	0.025	0.024	0.782	0.024	0.021
W_1=0.1 W_2=0.1 W_3=0.8	Ave.	15.14	3704.32	3147.24	15.12	3733.44	3157.33	15.16	3723.2	3152.29
	Std.	0.350	0.010	0.006	0.773	0.009	0.009	0.618	0.01	0.008

Table 3: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on ARF DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W_1=0.8 W_2=0.1 W_3=0.1	Ave.	8.18	11172.16	12119.97	8.36	11348.8	12419.74	8.3	11340.8	12394.53
	Std.	0.388	0.005	0.004	0.563	0.045	0.053	0.505	0.023	0.027
W_1=0.1 W_2=0.8 W_3=0.1	Ave.	18.2	3534.08	3182.54	18.38	3658.88	3300.77	18.44	3627.2	3258.2
	Std.	0.404	0.010	0.013	0.923	0.035	0.041	0.787	0.016	0.017
W_1=0.1 W_2=0.1 W_3=0.8	Ave.	18.4	3555.84	3142.2	18.72	3704	3182.55	18.6	3688.32	3187.59
	Std.	0.494	0.011	0.004	0.757	0.012	0.013	0.670	0.014	0.014

Table 4: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on EWF DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W1=0.8 W2=0.1 W3=0.1	Ave.	14	6848.64	6617.25	14.08	6834.88	6597.68	14	6924.8	6702.99
	Std.	0	0.006	0.007	0.396	0.044	0.056	0	0.006	0.007
W1=0.1 W2=0.8 W3=0.1	Ave.	25.06	3857.28	3202.72	23.52	4125.76	3483.95	22.76	4039.04	3388.73
	Std.	4.037	0.014	0.019	4.841	0.059	0.077	5.057	0.043	0.057
W1=0.1 W2=0.1 W3=0.8	Ave.	25.24	3949.76	3167.41	25.16	4152	3403.26	24.46	4070.08	3323.16
	Std.	2.924	0.014	0.011	3.247	0.057	0.077	3.95	0.043	0.057

Table 5: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on BPF DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W_1=0.8 W_2=0.1 W_3=0.1	Ave.	8.06	6546.24	6699.58	8.24	7304.64	7549.95	8.22	6996.48	7203.75
	Std.	0.239	0.034	0.038	0.555	0.069	0.078	0.545	0.062	0.069
W_1=0.1 W_2=0.8 W_3=0.1	Ave.	17.7	3605.12	3197.67	16.76	3835.84	3298.54	15.94	3884.48	3333.84
	Std.	3.688	0.018	0.014	3.426	0.035	0.028	3.449	0.033	0.027
W_1=0.1 W_2=0.1 W_3=0.8	Ave.	16.92	3744	3217.85	16.7	3969.92	3346.16	17.42	3870.72	3263.24
	Std.	3.212	0.028	0.015	3.37	0.039	0.049	3.038	0.025	0.025

Table 6: Comparison of the proposed MFO-based method with GA-based and PSO-based methods on WDF DFG

		MFO			**GA**			**PSO**		
		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
W_1=0.8 W_2=0.1 W_3=0.1	Ave.	8.06	6546.24	6699.58	8.24	7304.64	7549.95	8.22	6996.48	7203.75
	Std.	0.239	0.034	0.038	0.555	0.069	0.078	0.545	0.062	0.069
W_1=0.1 W_2=0.8 W_3=0.1	Ave.	17.7	3605.12	3197.67	16.76	3835.84	3298.54	15.94	3884.48	3333.84
	Std.	3.688	0.018	0.014	3.426	0.035	0.028	3.449	0.033	0.027
W_1=0.1 W_2=0.1 W_3=0.8	Ave.	16.92	3744	3217.85	16.7	3969.92	3346.16	17.42	3870.72	3263.24
	Std.	3.212	0.028	0.015	3.37	0.039	0.049	3.038	0.025	0.025

As shown in Tables 1 to 6, as the weight of each parameter increases, a significant improvement has been observed to find the optimal response of the same parameter. For example, by considering 0.8 for W_1 which is the weighting factor related to the delay parameter, and 0.1 for W_2 and W_3 which are the coefficients of the occupied surface area and power consumption respectively, the best delay compared to the other two modes will be obtained. The same applies to the other two modes. The delay in the first row of each table shows the best delay, while the area in the second row is the best occupied surface and the obtained power in the third row shows the best power consumption compared to the other two rows.

In Table 1, which presents the synthesis of the IIR filter, the proposed method and the PSO-based method obtain an average value of 5 for the delay. But the GA-based method performed slightly weaker and presented an average of 5.08. In the case of the lowest occupied area and power consumption, the proposed method performed better than the other two methods with average values of 3076.48 and 3147.24, respectively. In the FIR synthesis shown in Table 2, the proposed method performs better than the other two methods in all three modes. In this filter, the delay, area, and power consumption are 9.1, 3542.08 and 3147.24, respectively. In the ARF, the proposed algorithm yielded better results of 8.12, 3534.08, and 3142.2, respectively for the delay, occupied area, and power consumption.

With reference to the EWF synthesis, both MFO-based and PSO-based methods have obtained the same average delay. But in the other two cases, finding the best area and power consumption, the method based on the MFO algorithm performed better with the values of 3857.28 and 3167.41, respectively.

In Table 5, which presents the synthesis of the BPF, the results show that the proposed method performs better. The best average delay, occupied area and, power consumption obtained in this filter were 8.06, 3605.12 and 3217.85, respectively.

Finally, in the WDF filter, the proposed method still performs better. The proposed method has a delay of 14.08 as the best delay, which is better than the other two methods with values of 14.28 and 14.12 for GA-based and PSO-based methods, respectively. The best area and power consumption in the MFO-based algorithm are also obtained 4144 and 3172.46, which is better than the other two methods.

Using the information in Tables 1 to 6, although the delay, occupied area, and power consumption in the proposed method are better than the other two methods, at each mode with the improvement of one parameter, the other two parameters increase significantly.

For example, as the delay improves, two other parameters, namely power consumption and occupied area increase at first mode. It seems that the kind of decrease in the number of flames after each iteration causes these drastic changes. In this paper, the number of flames in each iteration is updated through (8).

Fig. 6 shows a representation of the best-averaged responses for each filter.

The percentage of improvement obtained by the proposed method based on MFO algorithm compared to the other two methods based on GA and PSO algorithm in the synthesis of each digital filter is presented in Table 7.

As shown in Table 7, in the IIR and EWF filters, the proposed method together with the PSO-based method calculated the same delay. But in other cases, the proposed method based on the MFO algorithm has shown better performance.

In order to obtain optimal delay, the proposed method showed the best performance in FIR filter synthesis with 2.78% and 2.15% improvement compared to the GA-based and PSO-based methods, respectively.

Regarding the optimal occupied area, the best performance has been found in the BPF synthesis with a 7.19% improvement compared to the PSO-based method and in the EWF filter synthesis with a 6.51% improvement compared to GA-based method.

The highest improvement in power consumption was also achieved in the EWF, with 6.93% and 4.69% improvement compared to GA and PSO, respectively.

Fig. 7 shows some examples of the proposed method and the two GA-based and PSO-based methods when calculating the best power of the 6 filters (W_3=0.8 and W_1=W_2=0.1). While the Continuous black lines depict the data obtained from the proposed method based on the MFO algorithm, the blue dotted lines show the data from the PSO-based method and the red lines illuminate the data of the GA-based method.

Then, by considering three objective functions (delay, area, and power), the Pareto front display estimated by the proposed method for all three functions simultaneously will be three-dimensional and will not be an efficient and appropriate criterion for the visual measurement. Therefore, by considering the weight of power is constant (W_3 = 0.3 which means that the importance of this target is never less than 30% compared to the other objectives, the Pareto front has been shown in two dimensions for contrasting the other two objectives (delay and occupied area).

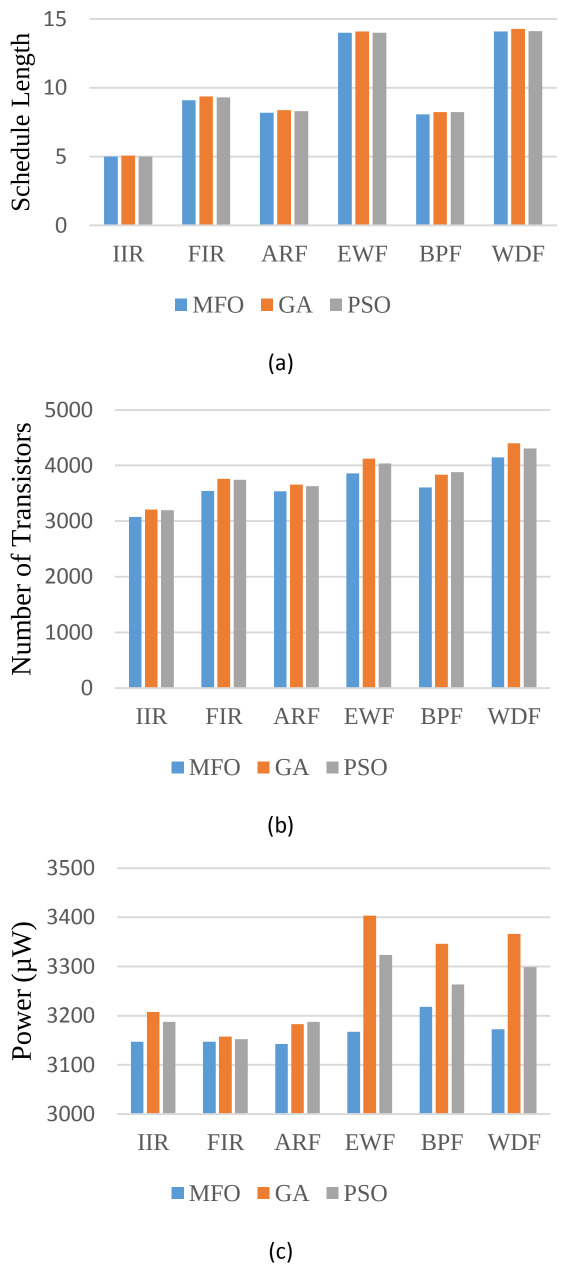


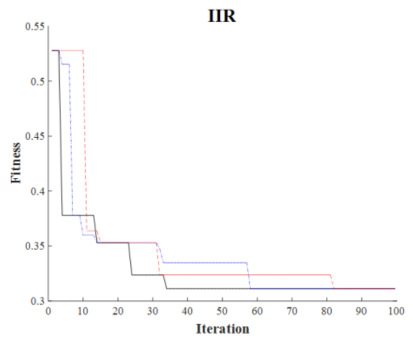
Fig. 6: The best results for a) Delay, b) Occupied area, and c) Power in digital filter synthesis.

Table 7: Improvement of the MFO-based method compared to GA-based and PSO-based methods

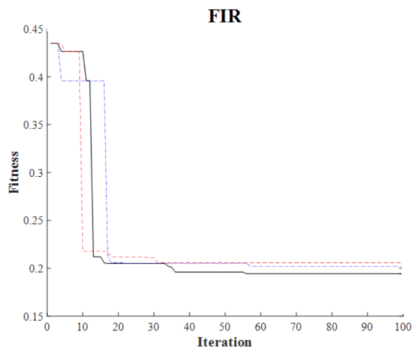
		Performance improvements (%)		
		Delay	Area	Power
IIR	GA	1.57	4.16	1.89
	PSO	0	3.85	1.26
FIR	GA	2.78	5.87	0.32
	PSO	2.15	5.36	0.16
ARF	GA	2.15	3.41	1.27
	PSO	1.44	2.57	1.42
EWF	GA	0.57	6.51	6.93
	PSO	0	4.5	4.69
BPF	GA	2.18	6.01	3.83
	PSO	1.95	7.19	1.39
WDF	GA	1.4	5.82	5.76
	PSO	0.28	3.85	3.82

Table 8 demonstrates the final response of the investigated methods. In this table, W_3 is considered to be 0.3 while W_1 and W_2 have been changed from 0.1 to 0.6 with the interval of 0.1 so that the sum of all three coefficients will be always equal to 1. It can be therefore inferred that the proposed method based on the MFO algorithm performs better than the other two methods based on GA and PSO algorithm with improvements in delay, area and, power consumption. It can be seen that by changing the coefficients related to delay, and area, W_1 and W_2, there are significant changes in the obtained responses. It is observed that there will be a diminishment in the delay and a rise in the occupied area by increasing the delay coefficient (W_1) and decreasing the occupied area coefficient (W_2). For instance, by increasing W_1 and decreasing W_2 in the proposed method for the synthesis of the IIR filter, there will be a reduction in the delay from 7.38 to 5.9 and a rise in the area from 3192.64 to 4194.56. In the same vein, in the GA-based method, there is a fall in the delay from 7.58 to 6.18 and an upsurge in the area from 3377.92 to 4241.6. In the PSO-based method, the delay was decreased from 7.42 to 6.06 and the occupied area increased from 3232.64 to 4229.76. It is of note that the proposed method had a better performance on the other filters, too. Fig. 8 shows the Pareto fronts obtained using the data in Table 8 for the studied filters, where the continuous black lines show the data obtained from the proposed method based on the MFO algorithm while the blue dotted lines depict the data from the PSO-based algorithm and the red lines show the data related to the GA-based algorithm. Besides, to better represent the data, in the vertical axis, the logarithm of the obtained area has been used to better comparison of the three methods.

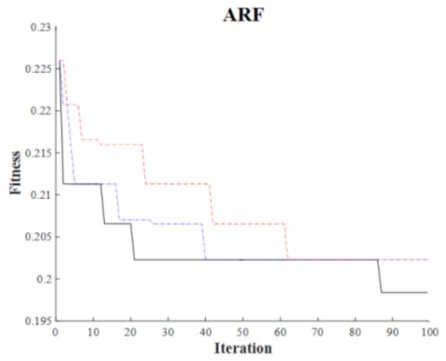
It can be seen from Fig. 8 that in all the benchmarks, the lower diagram of the proposed method and hence the lower area under this curve compared to the red and blue curves of the GA-based and the PSO-based methods, respectively demonstrate the better performance of the proposed method in finding optimal responses. After all, to prove the improvement of the proposed method based on the MFO algorithm over the other two methods, statistical hypothesis test has been conducted. Table 9 shows the results of the statistical hypothesis test.



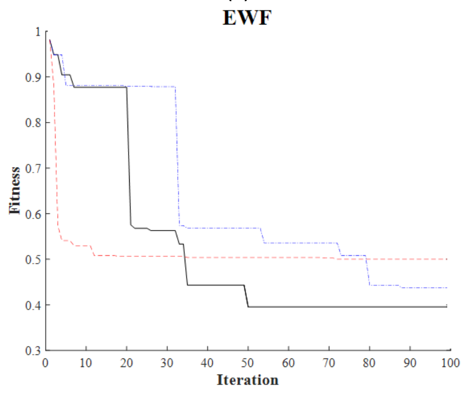
(a)



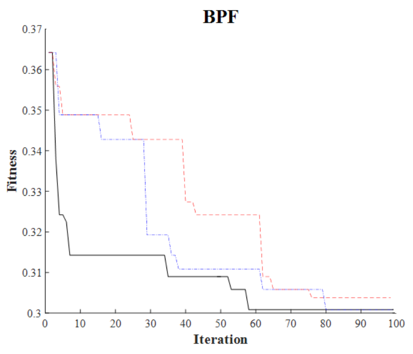
(b)



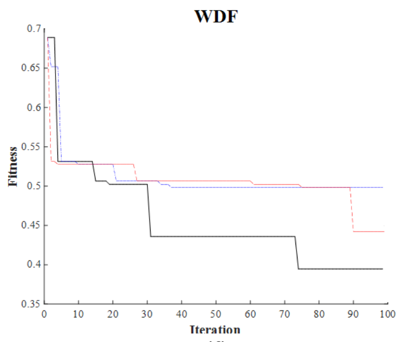
(c)



(d)

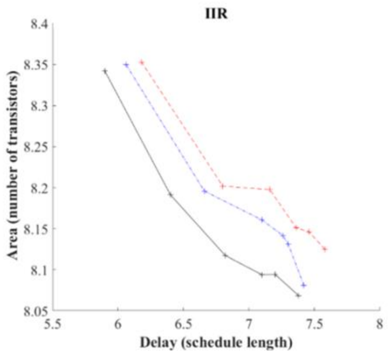


(e)

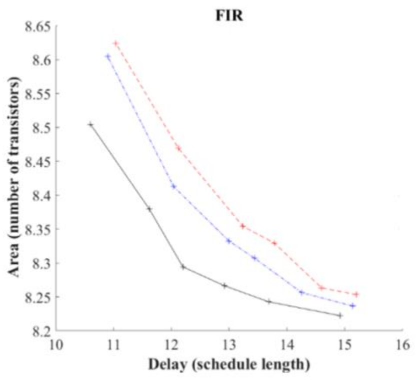


(f)

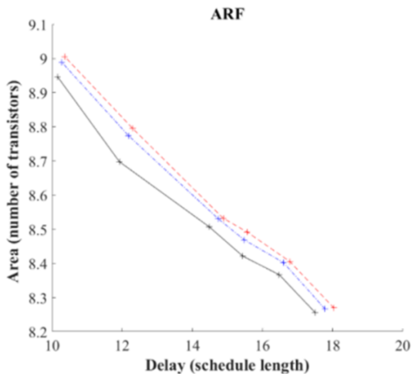
Fig. 7: An example of the implementing MFO, PSO, and GA-based methods for W_2=W_1=0.1 and W_3=0.8 in the synthesis of a) IIR, b) FIR, c) ARF, d) EWF, e) BPF, and f) WDF.



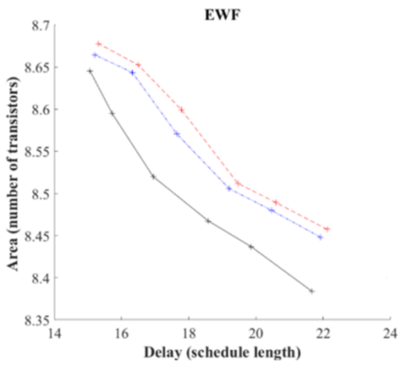
(a)



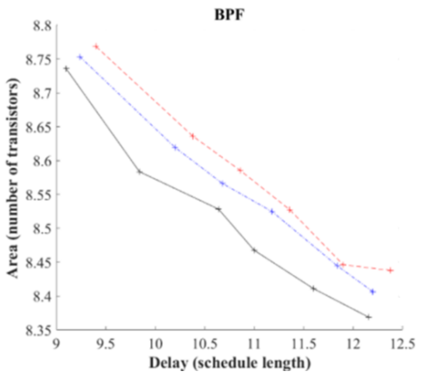
(b)



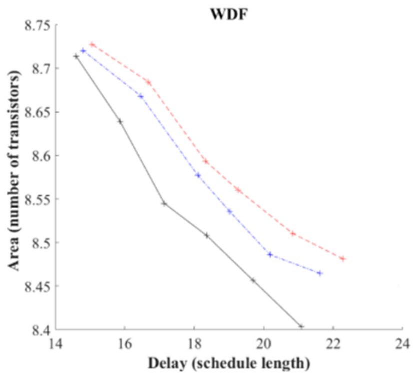
(c)



(d)



(e)



(f)

Fig. 8: The pareto front by coefficients changes of W_1, W_2, and W_3 = 0.3 for a) IIR, b) FIR, c) ARF, d) EWF, e) BPF, and f) WDF.

Table 8: Results on benchmarks. delay (schedule length), area (number of transistors), and power (µw)

		MFO			**GA**			**PSO**		
W_3=0.3		Delay	Area	Power	Delay	Area	Power	Delay	Area	Power
IIR	W_1=0.1 W_2=0.6	7.38	3192.64	3177.50	7.58	3377.92	3343.93	7.42	3232.64	3212.81
	W_1=0.2 W_2=0.5	7.2	3276.16	3265.46	7.46	3450.24	3439.16	7.3	3399.68	3388.73
	W_1=0.3 W_2=0.4	7.1	3274.56	3263.24	7.36	3467.52	3444.79	7.26	3434.56	3404.45
	W_1=0.4 W_2=0.3	6.82	3352	3341.11	7.16	3632.32	3638.07	7.1	3500.48	3482.32
	W_1=0.5 W_2=0.2	6.4	3609.28	3632.43	6.8	3647.68	3665.51	6.66	3625.28	3640.29
	W_1=0.6 W2=0.1	5.9	4194.56	4345	6.18	4241.6	4378.08	6.06	4229.76	4367.99
FIR	W_1=0.1 W_2=0.6	14.92	3723.2	3197.68	15.2	3840.64	3268.28	15.14	3776.64	3202.72
	W_1=0.2 W_2=0.5	13.68	3800.64	3320.94	14.6	3879.04	3341.11	14.26	3851.2	3325.98
	W_1=0.3 W_2=0.4	12.92	3891.2	3364.10	13.78	4144.96	3592.68	13.44	4052.8	3504.72
	W_1=0.4 W_2=0.3	12.2	4001.28	3475.05	13.24	4247.36	3688.5	13	4156.16	3588.23
	W_1=0.5 W_2=0.2	11.62	4357.44	3881.77	12.12	4768	4320.97	12.04	4504.32	4015.12
	W_1=0.6 W2=0.1	10.6	4939.52	4626.83	11.04	5564.48	5279.48	10.9	5456.96	5164.08
ARF	W_1=0.1 W_2=0.6	17.5	3851.84	3516.44	18.04	3901.44	3581	17.78	3893.76	3566.87
	W_1=0.2 W_2=0.5	16.48	4301.76	4055.91	16.78	4472	4241.91	16.6	4456.96	4221.74
	W_1=0.3 W_2=0.4	15.44	4542.4	4364.58	15.56	4875.84	4693.43	15.48	4765.12	4567.94
	W_1=0.4 W_2=0.3	14.48	4950.72	4823.96	14.88	5076.48	4969.62	14.74	5067.52	4959.53
	W_1=0.5 W_2=0.2	11.92	5989.44	6053.6	12.3	6601.92	6761.14	12.18	6464.32	6608.21
	W_1=0.6 W2=0.1	10.16	7678.08	7987.96	10.36	8150.4	8595.22	10.28	8013.76	8427.17
EWF	W_1=0.1 W_2=0.6	21.66	4376.96	3807.76	22.12	4709.76	4201.57	21.92	4666.24	4148.91
	W_1=0.2 W_2=0.5	19.86	4614.08	4013.93	20.6	4861.76	4314.74	20.46	4818.24	4262.09
	W_1=0.3 W_2=0.4	18.58	4757.76	4222.34	19.46	4974.08	4490.66	19.2	4943.68	4443.05
	W_1=0.4 W_2=0.3	16.94	5014.4	4499.12	17.78	5428.16	4988.16	17.64	5275.52	4815.06
	W_1=0.5 W_2=0.2	15.72	5404.48	4970.8	16.5	5723.2	5332.13	16.32	5673.6	5269.39
	W_1=0.6 W2=0.1	15.06	5684.48	5297.42	15.32	5870.72	5528.23	15.2	5793.92	5427.95
BPF	W_1=0.1 W_2=0.6	12.16	4309.76	4062.14	12.38	4618.24	4418.43	12.2	4475.84	4250.37
	W_1=0.2 W_2=0.5	11.6	4496	4064.96	11.9	4657.6	4248.15	11.84	4650.24	4243.1
	W_1=0.3 W_2=0.4	11	4758.08	4411.16	11.36	5051.52	4704.7	11.18	5038.4	4699.66
	W_1=0.4 W_2=0.3	10.64	5055.68	4745.05	10.86	5352.32	5091.25	10.68	5250.56	4975.85
	W_1=0.5 W_2=0.2	9.84	5343.04	5159.03	10.38	5630.4	5442.49	10.2	5537.92	5347.26
	W_1=0.6 W2=0.1	9.1	6219.2	6260.38	9.4	6428.16	6481.09	9.24	6329.6	6375.78
WDF	W_1=0.1 W_2=0.6	21.08	4462.72	3574.73	22.28	4824	4001.03	21.62	4744.32	3900.76
	W_1=0.2 W_2=0.5	19.7	4704.96	3854.87	20.84	4965.76	4151.73	20.18	4847.04	4003.85
	W_1=0.3 W_2=0.4	18.36	4954.24	4111.98	19.26	5221.12	4427.92	19.02	5093.44	4269.95
	W_1=0.4 W_2=0.3	17.14	5140.48	4348.42	18.32	5396.8	4649.22	18.1	5309.76	4543.91
	W_1=0.5 W_2=0.2	15.86	5649.28	4931.05	16.68	5910.08	5236.9	16.46	5815.68	5126.55
	W_1=0.6 W_2=0.1	14.6	6086.08	5425.73	15.06	6168.64	5526	14.8	6125.12	5473.34

According to the findings reported in Table 9, it can be seen that by performing the statistical hypothesis test, in all cases except power consumption of the FIR filter, the proposed method performs better compared to the GA method with a confidence level of 99%. Compared to the PSO method, in all cases except power consumption of the FIR, delay of the ARF, and delay of the WDF, the proposed method performs better with a confidence level of 99%. As such, in the case of delay of the ARF with a confidence interval of 97.5%, the proposed method is better. As shown in Tables 1 and 4, that represent the synthesis of the IIR and EWF, standard deviation of the delay of the proposed method is equal to 0. Therefore, the result of T-test is negative infinity and has been shown with the "-∞" symbol in Table 9.

The use of the proposed new approach, which applies the optimization algorithm separately for each operator and updating the priorities at each stage, has made the algorithm escape the local optima and increase the exploration and exploitation simultaneously. Also, as mentioned, in MFO algorithm each moths searches around a unique flame. This significantly increases the exploration at the beginning of the execution of the algorithm, because the flames are located in different parts of the search space and the moths can search for larger space by moving around these flames.

Table 9: The Z value obtained in the statistical hypothetical test

		GA	PSO
	Delay	-∞	-∞
IIR	Area	-12.24	-11.3
	Power	-8.57	-5.71
	Delay	-7.22	-5.68
FIR	Area	-13.52	-12.27
	Power	-1.43	-0.71
	Delay	-3.28	-2.19
ARF	Area	-9.93	-7.41
	Power	-8	-9
	Delay	-∞	-∞
EWF	Area	-14.09	-9.54
	Power	-20.15	-13.3
	Delay	-5.31	-4.72
BPF	Area	-10.66	-12.91
	Power	-7.64	-2.70
	Delay	-5.16	-1.03
WDF	Area	-14.1	-9.15
	Power	-15.51	-10.87

At the end of the execution of the algorithm, the number of flames decreases according to (8) and the best flame remain. So, all the moths move around the best flame, which greatly enhances the exploitation of the algorithm at the end of the execution.

But for example in PSO algorithm, if the inertia weight is low and the leader of the group moves toward local optima, the whole particle also converges to local optima and the algorithm is trapped in that point. It is also difficult in the PSO and GA to accurately determine the parameters of the algorithm. However, the low number of parameters in the modified MFO algorithm makes that significantly less sensitive to the parameters. In addition, because of the simpler equation of the movement in the modified MFO algorithm, the runtime of the proposed algorithm is greatly increased.

B. Computational complexity

The computational complexity of the algorithm depends on the number of variables, the number of moths, the maximum number of iterations, and the flame sorting algorithm in each iteration. Using the Quicksort algorithm, the computational complexity of sorting is at the best as O(nlogn) and at the worst as O(n2). The total computational complexity is also calculated by (10).

$$O(MFO) = O\big(t\big(O(Quick\ sort) + O(Position\ Update)\big)\big) \tag{9}$$

where O is the computational complexity order and t is the maximum number of iterations. The computational complexity is, at the worst case, equal to (11).

$$O(MFO) = O\big(t(n^2 + n \times d)\big) = O(tn^2 + tnd) \tag{10}$$

where O is the computational complexity order, t is the maximum number of iterations, n is the number of moths and d is the number of variables.

C. Runtime

One of the prominent feature of the proposed method is its fast runtime in comparison with the other two methods in obtaining the solutions. Table 10 shows a comparison of the average runtime of the three methods to achieve the solution. According to Table 10, the runtime of the proposed MFO-Based method is faster than the other two GA-based and PSO-based methods. An average improvement of more than 21% in the runtime of the proposed method compared to the GA-based method and more than 12% compared to the PSO-based method guarantees the fast runtime of our approach. The maximum improvement in the runtime of this method over the PSO-based method in the EWF was 27.14%, while in the FIR filter it was 25.45%, and in the IIR filter being 21.43%. Likewise, the highest improvement over the PSO-based method was observed in the synthesis of the EWF, BPF, and ARF filters being 19.68%, 17.78%, and 17.65%, respectively. Finally, in Table 11 the advantages and disadvantages of the proposed method are compared to the other two methods.

Table 10: Execution times

	MFO	GA		PSO	
	Runtime (s)	Runtime (s)	Reduction (%)	Runtime (s)	Reduction (%)
IIR	2.2	2.8	21.43	2.4	8.33
FIR	4.1	5.5	25.45	4.3	4.65
ARF	7	8.9	21.35	8.5	17.65
EWF	10.2	14	27.14	12.7	19.68
BPF	7.4	9.3	20.43	9	17.78
WDF	12.3	14.3	13.99	13.6	9.56

Table 11: The advantages and disadvantages of the proposed method compared to two other methods

Advantages	**Disadvantages**
Good solutions	
Fast runtime	
Proper Exploitation	Weaker Exploration
Low number of the parameters	Slower Convergence
Easy implementation	
Escape from local optima	

Conclusion

One of the most important and influential steps in the design of a digital VLSI filter is high-level synthesis. Due to the vast and discrete search space and the priorities for executing the operators, high-level synthesis problems have their complexity and are one of the most difficult problems in engineering. However, using metaheuristic methods that have already demonstrated their performance in solving such problems [42], [43] may improve the performance of the synthesis and find optimal solutions.

In this paper, a novel method based on MFO metaheuristic algorithm has been presented that after applying this method to synthesize the tested digital filters, it was found that this method has a higher ability to find the optimal solution compared to the GA-based and PSO-based methods. In the MFO algorithm, moths use a system called transverse orientation to move toward the flame. This transverse orientation system ensures that the moths move in a spiral direction toward the flame.

In the method presented in this paper, the hyperbolic spiral function is used to move the moths toward its corresponding flame. Finally, the obtained results were compared in terms of delay, occupied area, and power consumption. The results indeed showed an improvement in all the above-mentioned parameters. The greatest improvement was observed in the delay with a rate of 2.78% in the FIR synthesis compared to the GA-based method. A 7.19% improvement in the area in the synthesis of the BPF and a 6.93% improvement in the power consumption in the synthesis of the EWF were also obtained compared to the PSO-based and the GA-based methods, respectively. Also, the better performance of the proposed method has been proved based on the statistical hypothesis test.

Then, by plotting the Pareto fronts for a certain mode of the problem, it is observed that in the proposed method, the area left below the curve in all the cases was lower than the other two methods.

The fast runtime of the proposed method to achieve the appropriate solutions is another striking feature of the MFO-based method proposed here. The results showed an average improvement of more than 21% in the runtime of the proposed method compared to the GA-based method and more than 12% compared to the PSO-based method. This improvement in runtime accelerates the design speed, especially in very large-scale problems having a high number of operators.

Although the obtained delay, area, and power in the proposed method are better than the other two methods, with the improvement of one parameter, the other two parameters increase significantly.

For example, as the area improves, two other parameters, namely power consumption and delay increase at first mode. It seems that the kind of decrease in the number of flames after each iteration causes these drastic changes. Therefore, as a suggestion, different methods can be used to reduce the number of flames. This directly increases the exploitation of the algorithm in the final iterations.

Author Contributions

M. R. Esmaeili collected the data and benchmarks. M. R. Esmaeili and S. H. Zahiri carried out the data analysis and wrote the code in MATLAB. M. R. Esmaeili, S. H. Zahiri, and S. M. Razavi interpreted the results and wrote the manuscript.

Conflict of Interest

The author declares that there is no conflict of interests regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancy have been completely observed by the authors.

References

[1] N. S. Kim, J. Xiong, W. W. Hwu, "Heterogeneous computing meets near-memory acceleration and high-level synthesis in the post-moore era," IEEE Micro., 37(4): 10-18, 2017.

[2] C. Pilato, S. Garg, K. Wu, R. Karri, F. Regazzoni, "Securing hardware accelerators: A new challenge for high-level synthesis," IEEE Embedded Systems Letters, 10(3): 77-80, 2018.

[3] A. Sengupta, S. Bhadauria, S. P. Mohanty, "TL-HLS: methodology for low cost hardware Trojan security aware scheduling with optimal loop unrolling factor during high level synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36(4): 655-668, 2017.

[4] A. Mahapatra, B. C. Schafer, "VerilIntel2C: Abstracting RTL to C maximize high-Level synthesis design space exploration," Integration, 64): 1-12, 2019.

[5] S. Das, R. Maity , N. P. Maity, "VLSI-based pipeline architecture for reversible image watermarking by difference expansion with high-level synthesis approach," Circuits, Systems, and Signal processing, 37(4): 1575-1593, 2018.

[6] J. Zhao, L. Feng, S. Sinha, W. Zhang, Y. Liang, B. He, "COMBA: A comprehensive model-based analysis framework for high level synthesis of real applications," in Proc. 2017 IEEE/ACM International Conference on Computer-Aided Design, Irvine, CA, USA.

[7] P. Fezzardi, C. Pilato, F. Ferrandi, "Enabling automated bug detection for IP-based design using high-level synthesis," IEEE Design & Test, 35(5): 54-62, 2018.

[8] X. Tang, T. Jiang, A. Jones, P. Banerjee, "Behavioral synthesis of data-dominated circuits for minimal energy implementation," presented at the 18th International Conference on VLSI Design, Kolkata, India, 3-7 2005.

[9] N. Chabini, W. Wolf, "Unification of scheduling, binding, and retiming to reduce power consumption under timings and resources constraints," IEEE Transactions on VLSI Systems, 13(10): 1113–1126, 2005.

[10] A. Kumar, M. Bayoumi, "Multiple voltage-based scheduling methodology for low power in the high level synthesis," in Proc. 1999 the International Symposium on Circuits and Systems (ISCAS): 371–379.

[11] A. K. Murugavel, N. 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.

[12] R. K. Brayton, R. Camposano, G. De Micheli, R. Otten, J. van Eijndhoven, "The Yorktown silicon compiler system," in Silicon Compilation, D. D. Gajski, Ed. Reading, MA: Addison-Wesley): 204–310, 1988.

[13] O. V. Nepomnyashchiy, I. V. Ryjenko, V. V. Shaydurov, N. Y. Sirotinina, A. I. Postnikov, "The VLSI high-level synthesis for building onboard spacecraft control systems," in Proc. 2018 the Scientific-Practical Conference “Research and Development: 229-238, 2016.

[14] S. P. Mohanty, R. Velagapudi, E. Kougianos, "Physical-aware simulated annealing optimization of gate leakage in nanoscale datapath circuits," in Proc. 2006 the Conference on Design, Automation and Test in Europe: 1191–1196.

[15] S. Devadas, A. R. Newton, "Algorithms for hardware allocation in data path synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 8(7): 768-781, 1989.

[16] J. A. Nestor, G. Krishnamoorthy, "SALSA: A new approach to scheduling with timing constraints," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 12): 1107–1122, 1993.

[17] S. Rajmohan, N. Ramasubramanian, "Group influence based improved firefly algorithm for design space exploration of datapath resource allocation," Applied Intelligence, 49(6): 2084-2100, 2019.

[18] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York 1994.

[19] R. Camposano, "Path-based scheduling for synthesis," IEEE Trans. Comput. -Aided Des., 10: 85–93, 1991.

[20] S. H. Gerez, Algorithms for VLSI Design Automation, Wiley, 2004.

[21] S. Rajmohan, N. Ramasubramanian, "A Memetic algorithm based design space exploration for datapath resource allocation during high level synthesis," Journal of Circuits, Systems and Computers.

[22] S. Bhadauria, A. Sengupta, "Adaptive bacterial foraging driven datapath optimization: Exploring power-performance tradeoff in high level synthesis," Applied Mathematics and Computation, 269): 265-278, 2015.

[23] V. Krishnan, S. Katkoori, "A genetic algorithm for the design space exploration of datapaths during high-level synthesis," IEEE Trans. EComput., 10(3): 213–229, 2006.

[24] A. Sengupta, R. Sedaghat, "Integrated scheduling, allocation and binding in high level synthesis using multi structure Genetic Algorithm based design space exploration," in Proc. 2011 The 12th International Symposium on Quality Electronic Design: 1-9.

[25] D. S. Harish Ram, M. C. Bhuvaneswari, S. S. Prabhu, "A novel framework for applying multiobjective GA and PSO based approaches for simultaneous area, delay, and power optimization in high level synthesis of datapaths," VLSI Design, 2012: 1-12, 2012.

[26] R. F. Abdel-kader, "Particle swarm optimization for constrained instruction scheduling," VLSI Design, 2008(4): 1-7, January 2008.

[27] S. A. Hashemi, B. Nowrouzian, "A novel particle swarm optimization for high-level synthesis of digital filters," in Proc. 2012 The 25th IEEE International Symposium on Circuits and Systems, pp 580–583, 2012.

[28] C. Pilato, D. Loiacono, A. Tumeo, F. Ferrandi, P. L. Lanzi, D. Sciuto, "Speeding-up expensive evaluations in high-level synthesis using solution modeling and fitness inheritance," Computational Intelligence in Expensive Optimization Problems, 2: 701–723, 2010.

[29] G. Wang, W. Gong, B. DeRenzi, R. Kastner, "Design space exploration using time and resource duality with the ant colony

optimization," in Proc. 2006 The 43rd ACM/IEEE Design Automation Conference): 451–454, 2016.

[30] C. Gopalakrishnan, S. Katkoori, "Tabu search based behavioral synthesis of low leakage datapaths," in Proc. 2004 IEEE Computer Society Annual Symposium on VLS): 260–261, 2004.

[31] R. Kianzad, H. M. Kordy, "Automatic sleep stages detection based on EEG signals using combination of classifiers," Journal of Electrical and Computer Engineering Innovations (JECEI), 1(2): 99-105, 2014.

[32] A. Khalili, A. Rastegarnia, V. Vahidpour, Md. K. Islam, "Adaptive-filtering-based algorithm for impulsive noise cancellation from ECG signal," Journal of Electrical and Computer Engineering Innovations (JECEI), 4(2): 169-176, 2016.

[33] A. Ghanbari, A. Sadr, M. Nikoo, "High speed delay-locked loop for multiple clock phase generation," Journal of Electrical and Computer Engineering Innovations (JECEI), 1(1): 19-27, 2013.

[34] M. Moradi, M. R. Sadeghi, "Combining and steganography of 3-d face textures," Journal of Electrical and Computer Engineering Innovations (JECEI), 5(2): 93-100, 2017.

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

[36] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," Knowledge-Based Systems, 89: 228-249, 2015.

[37] Z. Li, Y. Zhou, S. Zhang, J. Song, "Levy-flight moth-flame algorithm for function optimization and engineering design problems," Mathematical Problems in Engineering, 2016.

[38] N. Muangkote, K. Sunat, S. Chiewchanwattana, "Multilevel thresholding for satellite image segmentation with moth-flame based optimization," presented at the 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 2016.

[39] K. Kaur, U. Singh, R. Salgotra, "An enhanced moth flame optimization," Neural Comput. & Applic., 2018.

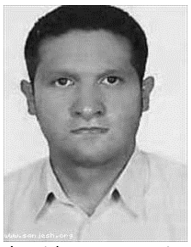
[40] M. C. Bhuvaneswari, Application of Evolutionary Algorithms for Multi-Objective Optimization in VLSI and Embedded Systems, Springer, India 2015.

[41] M. Jhamb, Garima, H. Loohani, "Design, implementation and performance comparison of multiplier topologies in power-delay space," Engineering Science and Technology, an International Journal, 19(1): 355-363, 2016.

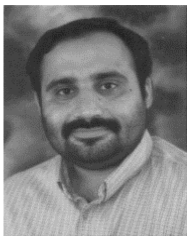
[42] S. S. Choong, L. P. Wong, C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the traveling salesman problem," Swarm and Evolutionary Computation, 44): 622-635, 2019.

[43] A. C. Parker, J. Pizarro, M. Mlinar, "MAHA: A program for datapath synthesis," presented at the 23rd ACM/IEEE Design Automation Conf., Las Vegas, USA, 29June-2 July, 1986.

Biographies



Mohammad Reza Esmaeili received the B.Sc. degree in Electronics Engineering from University of Birjand, Birjand, Iran in 2011 and the M.Sc. degree in Electronics Engineering from University of Birjand, Birjand, Iran in 2013. He is currently working toward the Ph.D. degree in Electronics Engineering at University of Birjand, Birjand, Iran. His research interest includes signal processing, evolutionary algorithms, swarm intelligence algorithms, and VLSI design.



Seyed Hamid Zahiri received the B.Sc., M.Sc., and Ph.D. degrees in Electronics Engineering from Sharif University of Technology, Tehran, Tarbiat Modares University, Tehran, and Mashhad Ferdowsi University, Mashhad, Iran, in 1993, 1995, and 2005, respectively. Currently, he is a Professor with the Department of Electronics Engineering, University of Birjand, Birjand, Iran. His research interests include pattern recognition, evolutionary algorithms, swarm intelligence algorithms, and soft computing.

Seyed Mohammad Razavi was born in Birjand, Iran. He received the B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran, Tarbiat Modares University, Tehran, Iran, and Tarbiat Modares University, Tehran, Iran, in 1994, 1997, and 2006, respectively. Currently, he is an Associate Professor with the Department of Electronics Engineering, University of Birjand, Birjand, Iran. His research interests include text and character recognition, pattern recognition, and Image processing.

How to cite this paper:
M. R. Esmaeili, S. Hamid Zahiri, S. M. Razavi, “A framework for High-Level Synthesis of VLSI Circuits Using a Modified Moth-Flame Optimization Algorithm,” Journal of Electrical and Computer Engineering Innovations, 7(1): 95-111, 2019.

DOI: 10.22061/JECEI.2020.5992.272

URL: http://jecei.sru.ac.ir/article_1190.html