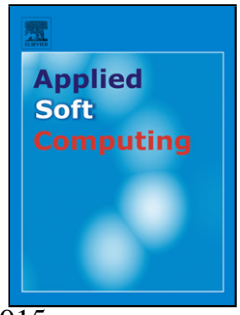


Title: FPGA based hardware implementation of Bat Algorithm

Authors: Mohamed Sadok Ben Ameur, Anis Sakly

PII: S1568-4946(17)30187-4
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2017.04.015
Reference: ASOC 4148

To appear in: *Applied Soft Computing*

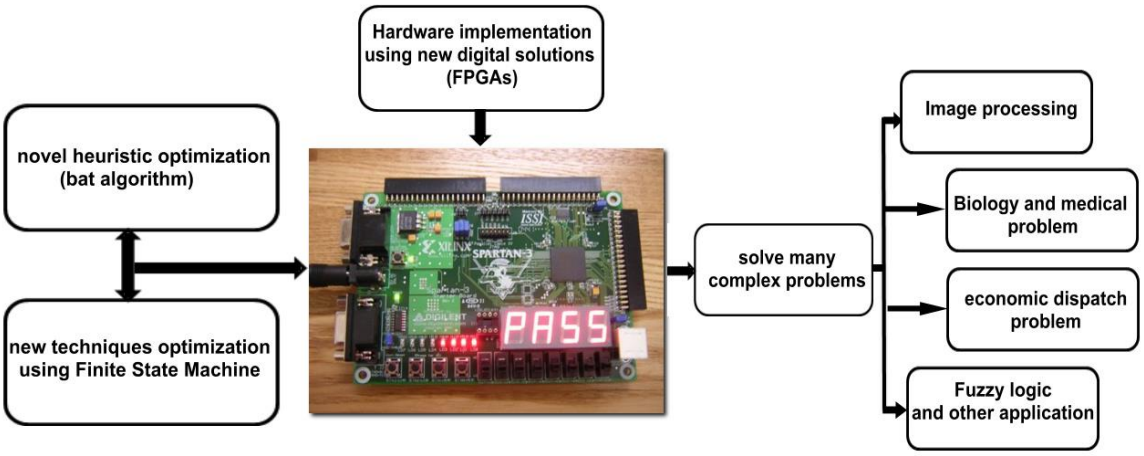
Received date: 4-5-2016
Revised date: 8-3-2017
Accepted date: 10-4-2017

Please cite this article as: Mohamed Sadok Ben Ameur, Anis Sakly, FPGA based hardware implementation of Bat Algorithm, Applied Soft Computing Journalhttp://dx.doi.org/10.1016/j.asoc.2017.04.015

FPGA based hardware implementation of Bat Algorithm

Mohamed Sadok BEN AMEUR[1,2], Anis SAKLY[2],
[1:] Laboratory of Electronic and Microelectronic, University of Monastir, Tunisia.
Mohamed sadok ben ameur, msba2014@gmail.com
[2:] Research unit ESIER, National Engineering School of Monastir, University of Monastir, Tunisia.
Anis sakly, Sakly_anis@yahoo.fr,

Graphical abstract



Highlights

- **New hardware architecture of bat algorithm based on Finite State Machine (FSM) is implemented into FPGA.**
- **New approach for global engineering optimization using different search strategies to diversify the bat population.**
- **A modification for the local search of BA is executed to improve the bats' movement.**
- **A parallel bat algorithm was developed to solve the well-known benchmarks function using Field-Programmable Gate Arrays (FPGAs) which give better solutions compared to other algorithms in terms of execution time and material resources.**

Abstract— **Several meta-heuristics algorithms are used mainly for research of global optimal solutions for real and non-convex problems. Some of them are the Genetic Algorithms (GA), Cuckoo Search algorithms (CS), Particle Swarm Optimization (PSO). Some algorithms have achieved satisfactory results but not all of them. Therefore, new algorithms give better optimization to solve many problems having continuous search space like Bat Algorithm (BA). That's why we proposed a new hardware implementation on Field Programmable Gate Array (FPGA) of bat algorithm, it is a new proposed meta-heuristic for global optimization. The work presented in this article is designed to use new digital dedicated hardware solutions such as FPGAs that are available to generate a better implementation of bat algorithm. This circuit is well adapted to many applications because its material structure is molded with the requirements of calculations. Moreover the inherent parallelism of these new hardware solutions and their large computing capabilities makes the computing time negligible despite the complexity of these algorithms.**

Keywords: BAT algorithm; Finite state machine; FPGA; Parallele programming; optimisation

I. Introduction

Over the last decade, several researches of meta-heuristic algorithms are proposed to solve hard and complex problems in optimization. There are two main concepts in computation: first one the evolutionary algorithms like genetic algorithm (GA) [1][2], these algorithms based on Darwin's principle of survivor and they are distinguished by their representation of solutions.

Although they were developed in the last few years, the two algorithms share some characteristics like crossover operators and selection operators. The second concept is swarm intelligence algorithms like particle swarm optimization (PSO) [3][4], artificial bee colony and bat algorithm. Bat algorithm is a new meta-heuristic algorithms based on echolocation which is an important feature of bat behavior. The ability and the simplicity to solve complex problems make the studies active in this area compared with many others optimization techniques. The effectiveness of this algorithm give satisfactory to solve the most difficult problems for many algorithms related to real optimization problems. The first implementations of digital bat algorithms were performed using microcontrollers, microprocessors and DSPs. These solutions did not give a good optimization at execution time. That's why the implementation of Bat Algorithm (BA) cannot be prepared by conventional design techniques, but requires high-level synthesis tools to Control a Dynamic Approach (CAD). Therefore, some digital solutions such as Field-Programmable Gate Arrays (FPGAs) were needed to solve many complex problems in computing and are used as numerical targets for the implementation of bat algorithm. The basic steps of bat algorithm are given in the section one. Next, we will show the capacity to produce competitive results to find the maximum or minimum of an objective function. So, in this article we will propose a novel hardware implementation of a bat algorithm into FPGA circuit.

The organization of this paper is described as follow: In the first section, the mechanism of bat algorithm is introduced. Particularly, a brief introduction of generating the initial population of bats and their move. Section 2 presents the proposed novel implementation of bat algorithm. The objective of this section is to describe the general steps performing the BA. Next section, illustrates the experimental results of some benchmarks functions applied into the bat algorithm. Finally, section 4 concludes the work and makes some implications and directions for future studies

II. Bat algorithm

Bat algorithm is one of new heuristic optimization algorithm was recently proposed by Yang [5][6]. The latest addition of bat algorithm is the Bat Inspired Search (BIS). The new algorithm use echolocation behavior of bats in searching for an optimum solution. Indeed, bats use several methods to detect theirs preys or other shapes around them and even in the dark places. Actually they achieve the prey by emitting a sound pulse to the air and they listen for the echoes reflect back from them. Generally echolocation calls are presented by three important features; the pulse emission rate, the frequency and the loudness. After that the information are collected and calculated in the brain to make a virtual image of their surroundings.

The bat algorithm give satisfactory results in solving many dispatch problems related to biology medical, finance, 3d graphics, image processing and others.

The bats and some other animals use echolocation; it is an advanced search based on navigation system to detect any objects in their surroundings by emitting a sound to the environment and returns to them as an echo. The direction and intensity of the returned signal enables them to locate direction and distance of potential prey. In addition, they have a surprising ability to quickly differentiate between an obstacle and a prey.

At first, the bat flies blindness around the search space while emitting sound wave of certain amplitude (intensity) and pulse rate. Between the pulse rates, it receives feedback signals (its own signal and possibly signals from other bats swarm) by echolocation and interprets these sound waves. If the received signals have a low intensity and / or a strong rate then it is very likely that prey is detected and the bat should run toward it. Gradually, as the bats approach to the prey, its increase the amount of pulses (pulse rate) and at the same time decreases the intensity of these pulses (loudness). But if the received signals are very low levels, it continues its flight blindly, without changing the intensity of the emitted sound wave.

The emitted sound travels in zones which has the same atmospheric air pressure and in a constant speed to follow time delay of the returning echoes. Moreover, bats use the loudness of the received signals each time to identify the direction of the shapes. These informations allow the bats to make a virtual image of their environment.

A. Position vector

Generally, the objective of swarming of bats is to indicate by their positions the distance to the best prey. Indeed, each bat is associated with a point in the search space and the position of each bat is depending on the intensity and frequency of the emitted sound wave (by itself and by its companions).

To simplify the design of our BA, the algorithm focused on the main characteristics of bats to accelerate finding the prey [7]. In fact, the artificial bats use three vectors; the first vector is the position the second one is the velocity vector and the third one is the frequency vector. All vectors are updated throughout the race of iterations by the following equations (1) and (2):

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

$$v_i(t+1) = v_i(t) + (x_i(t) - Gbest)\, f_i \tag{2}$$

Where $x_i(t+1)$ is the position of bats at iteration "$t+1$" and $x_i(t)$ at iterations "t", "Gbest" is the best value reached and it is obtained via the optimization process which represents our global best solution found. f_i is the i[th] bat frequency and $v_i(t)$ is the velocity of bat.

Considering that bats move by flying and that the solutions of the search space "S" are positions in space. At each time t, each of the "N" bats in the population has a position in space "S", and a velocity v_i . (In the tested benchmark functions "N" is fixed with 20 bats).

B. Velocity vector

Each artificial bat has velocity vector which is updated during the course of iterations: At initialization, the bats are uniformly distributed in the search space. The initial velocity is zero in general. Each bat flies using an equation of velocity and the movement of bats are reduced around the prey. The prey is detected when the amplitude and the pulse rate perceived by the bat are less than the emitted loudness, respectively, higher than the emitted rate.

The moving of bat obeys a simple rule: either it continues its current path, or it changes its direction. In the first case, the rule is similar to that of the particle swarm optimization (PSO) [8][9]. The new speed is obtained by adding the current velocity and external velocity vector. This external velocity is generally obtained by multiplying the frequency f_i with the current position and the position of the best solution.

In the second case, the position is obtained from the current position of a randomly chosen bat. This position obtained by adding a random perturbation proportional to the average power of the loudness emitted by all bats.

C. Frequency vector

The frequency f_i is generated uniformly in the range [$f_{\min}$, $f_{\max}$] and it allows controlling the rhythm of the movement. The new position is obtained by adding the new velocity to the current position. Thus, the distance to the prey is estimated by the doppler effect by varying the frequency of the emitted signal wave.

f_i is the i[th] frequency bat and it is updated after every iteration using the following equation:

$$f_i = f_{\min} + (f_{\max} - f_{\min})B_i \tag{3}$$

"*B*" is a value chosen randomly from a uniform repartition between [0,1]. It's clear to see from the equations (2) and (3) that several frequencies allow artificial bats to have a various positions to the optimum solution. The three equations allow the diversity of the BA. On the

other hand, if there are no detected preys, a global search used in order to allow the exploitation of the search space using the following equation:

$$X_{new} = X_{old} + eA^t \tag{4}$$

In the equation below, "e" is the ability to perceive emitted sound by other bats from the swarm and it is a random value between [-1,1], "A" is the amplitude of emitted sound allowing the bats to perform a global exploration of its space [8]. Note that we fixed "$e = 0{,}9$" as a good factor.

D. Loudness and pulse rate module

In most implementations, the loudness is general chosen in the interval [0, 1] that is to say, $A_{\min} = 0$ and $A_{\max} = 1$. The initial value of the loudness and pulse rate are generally set to a value close to 0.5. In this case, a bat has a 50% chance to choose a random movement. It is also possible to initialize A_i and r_i randomly for each bat.

In fact, bats tend to increase pulse emission rate and to decrease the loudness of emitted sound when they are closer to its prey. That way is modeled using the two equations (5) and (6) as follows [10][11]:

$$A_i(t+1) = \alpha A_i(t) \tag{5}$$

$$r_i(t+1) = r_i(0)\left[1 - \exp(-\delta t)\right] \tag{6}$$

An approval of what we have said previously, the bat algorithm incorporates a combination between PSO algorithm and completive local search. So, these techniques are checked by loudness wave A_i and pulse rate r_i and they have to be updated from iteration to another. r_i is the pulse emission rate and it is varied once a solution is improved. The bat is moving towards optimal solution according to this equation. After each iteration, the loudness (A_i) usually decreases when a bat finds its prey. Where, $A_i(t+1)$ is the updated number of the loudness for each bats and $A_i(t)$ the previous value, "t" is iteration number, $r_i(t+1)$ is the pulse emission rate of bats at iteration "$t+1$", $r_i(0)$ is the number of the pulse rate of the first iteration and finally α and δ are the coefficient parameters [12] of loudness wave and pulse rate, respectively.

In the simplest case, we can use $\alpha = \delta$ and in the standard BA we can use $\alpha = \delta =$ 0.9 to 0.975 in most cases, though we have used, $\alpha = \delta$ = 0.95 in our simulations.

Whereas in equation (6), the pulse rate immediately approaches $r_{\max}$ in a few iterations and remains stationary at this value. The following figure present a graphical comparison of equation (6) by choosing $r_{\max} = 1.0, r_i(0) = 0.5,\ \alpha = \delta = 0.95$ over an iteration number of 150.

III. Hardware implementation of bat

A. Bat architecture

The essential bloc of the bat is described in figure 8. There are several parameters that should be initialized before running and must be set:

- The size of the search space.
- The position of the bats in our search space.
- The Velocity of bats.
- The best global fitness achieved of the microbats.
- The position of the microbats to the solution.
- The pulse rate and loudness of each microbats.

To initialize all parameters (position, velocity and frequency) each bats in the current generation must be initialized with initial seed using the module of pseudo random number after that, this seed is compared with the pulse rate r_i of each bat and the (rand) is random function implemented in the bat algorithm and it is a value between 0 and 1.

B. Pseudo-random generator

The pseudo random generator is an algorithm that generates a sequence of numbers with certain properties of chance. The numbers are assumed to be sufficiently independent of each other, generators programs are particularly suitable for implementation, so more easily and effectively used. Most pseudo random algorithms try to produce outputs that are uniformly distributed. A common class generator uses a linear congruence. Others are inspired by the Fibonacci sequence by adding the two previous values. Most popular and fast algorithms were created in 1948 D. H. Lehmer introduced linear generators congruentiels and will eventually become extremely popular. In the bat algorithm the pseudo random generator [13] are used at the initial position of bats and in the velocity, loudness and frequency vector. The frequently used pseudo-random generator in this algorithm called the linear congruent of Lehmer:

$$F_{n+1} = (AF_n + B) \bmod C \quad (7)$$

Where:

- F_{n+1} : is the random number obtained from the function F_n
- F_n : is the previous number obtained
- A and B : are multiplicative and additive value, respectively
- C : the modulo number

In general, the seed must be a prime number. The period of the generator is equal to "C", that is to say, "C" is generally chosen to be equal of word length (here C=16).

So, if ($rand > r_i$) then, a new local bat is created by flying randomly to another position in the search space. If not, another bat is created through random flying when we adjust its frequency and update its position and velocity. From the following architecture, it's plain to see that BA [14, 15] shares a lot of common factor with PSO algorithm. In fact, the two algorithms begin with a random generation using the module of random generator; both of them evaluate the population using a fitness module. Also the two algorithms update the position of population using the position module to search for an optimum solution. They exploit the random generator to update themselves using the internal module of velocity. The PSO and bat algorithm use bloc memory which is very important.

C. The finite state machine

In this work a parallel BA implemented to be applied into large optimization problem. This algorithm increases the convergence to a local minimum and to increase the performance of this algorithm a Finite State Machine (FSM) is used to exploit the maximum of parallelism. The dynamical process of the proposed bat's architecture is presented as follow:

Indeed, every state may have at every time a position of many possible finite states. In the beginning, a number of fixed states are proposed; every transition may have one or more around their state. In this way, states which have only one state and have no other possible move we named it the final states. The basic of BA was programed in order to updates the optimum remembered microbats, the fitness module and their correlated positions and velocities. The algorithm performs updating the optimum fitness number after the evaluations of all the bats. Here, when their positions, velocity and loudness are updated, it is possible to obtain a good convergence rates after evaluating each microbats. In a dynamic parallel computing, the main factor of performance is the communication latency after each transition between states. The goal of parallel dynamic computing is to produce optimal results even

when we use multiple processors to reduce the running time. In this architecture a pair memory modules are used to compound the bandwidth and thus, the capabilities of the algorithm can be effective that's why it's recommended to use dual channel bloc RAM. In that way, it is possible to access to the data memory in two modes: write or read at the same frequency. There are problems with the dual RAM. In fact, the reading time of the content of memory is delayed by one clock comparative to the last reading. Luckily, new advances in processor technology are capable and available to compute a complex program and use low cost power beyond clusters of mid-range performance computers. So, the dynamic process implemented in the bat could be separated in many operations which update the position, velocity and loudness of microbats using dynamic process giving the same results. So many states are used with the aim to reduce the executing time. In this paper, the soul of the parallel processing was used to generate a dynamic bat algorithm [16] and the aim of using parallel computing to the bat algorithm, is to speed up the algorithm processing using a uniform distribution method to achieve optimum solutions with a significant execution time. Figure 9 presents the finite state machine of the global control module; especially presents step by step the code of the BA to keep the algorithm more practical. From its bat-inspired algorithm Yang had demonstrate that the choice of upper and lower bounds for echolocation parameters might have some significant influence on convergence characteristics of the bat algorithm. In this paper, the bounds $f_{\min}$, $f_{\max}$, $r_{\min}$ and $A_{\max}$ are initialized to the following value: $f_{\min}$ = 0; $f_{\max}$ = 1; $r_{\min}$ = 0,5 and $A_{\max}$ = 1. Here every bat includes some parameters: β = (f_i, r_i, A_i), in which contains the pulse rate (r_i), the frequency (f_i) and the amplitude parameter (A_i). All parameters are positive dynamic values with the following value interval:

$$r_{\min} < r_i < r_{\max}, \; f_{\min} < f_i < f_{\max}, \; A_{\min} < A_i < A_{\max} \tag{8}$$

Where $r_{\min}$ is the lower bound and $r_{\max}$ is the upper bound of the pulse rate (r_i), $f_{\min}$ presents the lower bound and $f_{\max}$ is the upper bound for the frequency parameter (f_i) and $A_{\min}$ and $A_{\max}$ are the limited lower and upper bound of the intensity parameter (A_i). In this algorithm, the initial frequency value $f_i(0)$ is fixed randomly between $f_{\min}$ and $f_{\max}$ even the setting of initial loudness $A_i(0)$ is fixed initially to its maximum number $A_{\max}$ = 1 and the initial pulse rate $r_i(0)$ is fixed to its minimum value $r_{\min}$ = 0.5 for each bat.

IV. Experimental results

Most researchers use a number of population size between 10 to 50 for the performance comparison between algorithms. In this work, the population is fixed at 20 bats for the bat algorithm and the same for PSO algorithm and each bat is coded in 16 bits to represent each variable in binary. It should be noticed that the last bit is reserved for the sign of each functions' variable. So, 15 bits only are used. For some example, we search to minimize the desired function. So, from iteration to another, the bats that are near to the optimal solution are selected to be a possible solution of that function. To test the proposed architecture of bat algorithm and to compare its performance with other algorithm like PSO algorithm [17][18] and genetic algorithm, some standard benchmark functions are used which are described as below:

A. Sphere function

$$f_1(x) = \sum_{i=1}^{n-1} x_i \tag{9}$$

$$f_2(x, y) = \sum_{i=1}^{n-1} (x_i^2 + y_i^2) \tag{10}$$

Sphere function is useful to evaluate the characteristics of our optimization algorithms, such as the robustness and velocity of convergence. This function has a local minimum and it is unimodal and continuous. The interval of search space is between [-1,1]. Figure 10 and figure 11 present the results of simulation using "modelsim" of the sphere function and it describes the changed code of global best from iteration to another by updating the position and the velocity equations. It should be noticed that these benchmark presents a good function to test the convergence of the bat algorithm. Moreover this algorithm achieves significant improvement in all the remaining unimodal benchmark functions compared to other algorithms. The detailed result of simulation show that our solution converges to zero from iteration to another.

These microbats work together in a parallel dynamic state to get the best solution of any function, they update positions, frequencies and velocity even if the algorithm has a lot of microbats and this cannot make a hard impact on the global execution time speed. Indeed, the number of microbats in this algorithm is limted by the size of embded features of FPGA.

Table 1-3 present the number of device utilization of each benchmark function used in the bat algorithm. after configuration of the directives for bat algorithm, the kit spartran 3 estimate the device utilization and the performance of the architecture and their configuration. Here, the column number 1 presents the used logic LUT (Look Up Table), the number of bloc RAM (BRAM), the number of input/output block (IOBs), the number of multiplcator unit (MULT)

the number of global clocks (GCLKs) and all the materials resource used in this function and the column 2 presents the available logic in the device xc3s200.

B. Rastrigin function

The function is described as follows:

$$f_4(x) = 10n - \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \tag{11}$$

The Rastrigin function contains several local minima. But it has just one global minimum and it is highly multimodal and the locations of the minima are regularly distributed. The following table summarizes the material resource used in this function.

C. Rosenbrock function

Rosenbrock function is a non-convex function of two variables used as a test for mathematical optimization problems. It was introduced by Howard H. Rosenbrock in 1960. It is also known as the banana function name. In this function the global minimum of search algorithms converge easily. The function is defined as follow:

$$f_3(x, y) = \sum_{i=1}^{n-1} \left[(1 - x_i)^2 + 100(y_i - x_i^2)^2 \right] \tag{12}$$

The global minimum is obtained at point $(x, y) = (1,1)$, for which the function is 0. A different coefficient is sometimes given in the second term, but that does not affect the position of the global minima.

To increase the performance of bat algorithm a bloc memory is added to save the local best of bats and in this case the operating speed of the algorithm is increased. The results illustrate this.

D. Zakharov function

The next benchmark problem is the zakharov function whose global minimum occurs at ($x = 0$). It has no local minima except the global one. It is two-dimensional form.

$$f_5(x) = x_i^2 + (0.5ix_i)^2 + (0.5ix_i)^4 \tag{13}$$

E. Implementation test

The kit of Spartan-3 FPGA is from Xilinx. The Spartan-3 is one of the best low cost generation of FPGAs and the board can offers a choice of many platforms which deliver a unique cost optimization balanced between programmable logic, connectivity and devoted hard IP for many hardware applications. It creates a PROM file and this latter can be written to the non volatile memory.

The kit of Spartan-3 contain a lot of I/O signals, logic cell, 12 of hardware multipliers, 200k gate the following, 4 Digital extern clock and other features. (Figure 12):

The implementation of bat algorithm in kit xc3s200 is done with a specific Digilent cable which is straight compatible with the Xilinx iMPACT software and it loads the bit-stream into the prom of FPGA. During the experimental test of bat algorithm of example f_1 a photo was taken as shown in Figure 13. This photo presents two modes of displays for the tested algorithm; the first one is four seven segments module to display the iteration number used in the algorithm when optimal solution tends to "0" and in the second display, there are diodes LEDs used for the code of global best which composed from 16 bits.

F. Comparison with other algorithms

To make a comparison of this new algorithm to deliver better solution in a significant time especially its robustness and speed, this algorithm was tested against other meta-heuristic algorithms, like particle swarm optimization and genetic algorithms. For GA, the basic model with elitism method is used and the probability of mutation equal 5%. In the PSO algorithm an optimized version made by myself in another paper are used with modified parameters. The simulations have been carried out using spartran-3 of Xilinx with 50MHz. The population is fixed with N = 20 for all simulations.

In the following table, the system easily finds the optimum point (sphere function) after 178 iterations when $f(x)$ =0.

The results are favorable and proved that bat algorithm can be effective for many problems related to any algorithms used. The experiment results was carried out at minimum 5 % which allow judging whether the results of the bat are acceptable and optimized in execution time compared to the best results of other algorithms like PSO or GA. In fact, tests were done with the example of sphere function and the three algorithms are implemented on same kit of FPGA (spartran3 xc3s200). The results clearly shows that the genetic algorithm took (380) iterations to achieve the solution "0" while PSO algorithm need a little more iterations (420) but the bat algorithm need only (178) iteration to achieve the solution. In the three algorithms, the time

allocated to execute one iteration differs from algorithm to another. In bat algorithm, it takes (1590) clock cycle for the processing time of one iteration and it is acceptable compared to PSO algorithm. So the execution time depends on the number of iterations allocated to achieve the solution. Here in this example, the bat algorithm need only (5,66 ms) to get the optimal solution while (9,91 ms) for PSO algorithm and (74,02 ms) for genetic algorithm.

Figure 13 presents an example of tested function using seven segments module to display the number of iterations when we achieve the optimal solution. Eight diodes DEL are used to show the global best coded in 16 bits. The 8 most significant bits are displayed first and then an action on the pushbutton allow us to display the 8 low significant bits of this global best.
We can go further with this work using another kit to ameliorate and fix the problem of display like virtex 6 which has a good clock speeds.

V. Conclusion

Our work contributes the implementation on FPGA a novel optimization algorithm called BA experimented in many benchmark functions. In fact, this article develops a dynamic process of the Bat Algorithm and presents its hardware architectures' implementation in a Xilinx spartran3. In this algorithm, a finite state machine is used to exploit all the parallelisms that make the program converge very quickly. The bat algorithm has superior features, including quality of solution, good computational efficiency and stable convergence characteristics. The comparison shows that bat algorithm performs better than the mentioned methods. The proposed architecture of bat algorithm proves that it has a favorable convergence speed compared to the most of other meta-heuristic techniques depends on the size of design space; it means the number of bats allocated and the complexity of the problem. So, the BA's robustness is attached to its enhanced ability to achieve a satisfaction between two requirements, the size of memory and the processing time of algorithm to solve complex problems. Therefore, bat optimization is a promising technique for solving complicated problems in the real world applications.
For future studies, it is recommended to apply the BA to different practical applications. we can also ameliorate the processing time of our algorithm using another kit from Xilinx like virtex 6 in which it consists of several memories with very fast operating speed, then the processing time of the algorithm will be reduced. The effects of different transfer functions on the performance of BA are also worth investigating. It supports the usage of the BA in further experiments and in further real world applications.

VI. References

[1] Ben Ameur M.S; Sakly A.; Mtibaa A, implementation of real coded Genetic Algorithms using FPGA technology. 10th International Multi-Conference on Systems, Signals and Devices (SSD) 2013.

[2] Holland JH. Adaptation in natural and artificial systems. Arbor, MI: The University of Michigan Press; 1975.

[3] Ben Ameur M.S; Sakly A.; Mtibaa A. Implementation of real coded PSO algorithms using FPGA technology; 15th international Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2014

[4] Hardware/software co-design for particle swarm optimization algorithm Shih-An Li, Chen-Chien Hsu b, Ching-Chang Wong, Chia-Jun Yu, Information Sciences 181, Elsevier (2011) 4582–4596

[5] Yang X-S; Gonzalez JR. A new meta-heuristic bat Inspired algorithm. In: Nature inspired cooperative strategies for optimization (NISCO 2010) Studies in computational intelligence. Berlin: springer; 2010.

[6] Yang X-S. Bat algorithm for multiobjective optimization. Int J Bio-Inspired Comput 2011;3:267–74.

[7] Seyedali M; Xin-She Yang. Binary bat algorithm. Neural Comput & Applic (2014) 25:663–681

[8] J. Peña, A. Upegui, A population-oriented architecture for particle swarms, in: Second NASA/ESA Conference on Adaptive Hardware and Systems, (AHS 2007), pp. 563–570

[9] Tewolde G.S.; Hanna D.M., Haskell R.E. Multi-swarm parallel PSO: hardware implementation in: Proceedings of the 2009 IEEE Symposium on swarm intelligence

[10] Yang X-S; Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Eng Comput 2012;29:464–83.

[11] Xin-She Y. A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO). Studies in Computational Intelligence, 284:65-74, 2010.

[12] Serdar C; Oguzhan H. Optimum Design of Steel Space Frames via Bat-Inspired Algorithm, 10th World Congress on Structural and Multidisciplinary Optimization May 19 -24, 2013, Orlando, Florida, USA

[13] Lehmer D.H. Mathematical methods in large-scale computing units, Ann. Computing Lab. Harvard Univ. 141-146, 1951

[14] Gandomi AH, Yang X-S, Alavi AH, Talatahari S. Bat algorithm for constrained optimization tasks. Neural Comput Appl 2012. http://dx.doi.org/10.1007/s00521-012-1028-9.

.[15] D. Bratton, T. Blackwell, Understanding particle swarms through simplification: a study of recombinant PSO, in: Proceedings of the 9th 1013 Annual Conference on Genetic and, Evolutionary Computation (GECCO'07), 1014 pp. 2621–2627.

[16] Teke T; Hasançebi O; Pekcan O. A bat-inspired algorithm for structural optimization: Computers and Structures 128 (2013) 77–90

[17] Mingchang C. Self-adaptive Check and Repair Operator-based Particle Swarm Optimization for the Multidimensional Knapsack Problem, Applied Soft Computing, Vol. 26, pp.378-389, 2015.

[18] Mingchang C; Chin-Jung L; Maw-Sheng C; Tsung-Yin O. Particle Swarm Optimization with Time-Varying Acceleration Coefficients for the Multidimensional Knapsack Problem, Applied Mathematical Modelling, Vol. 38, No. 4, pp.1338-1350, 2014.

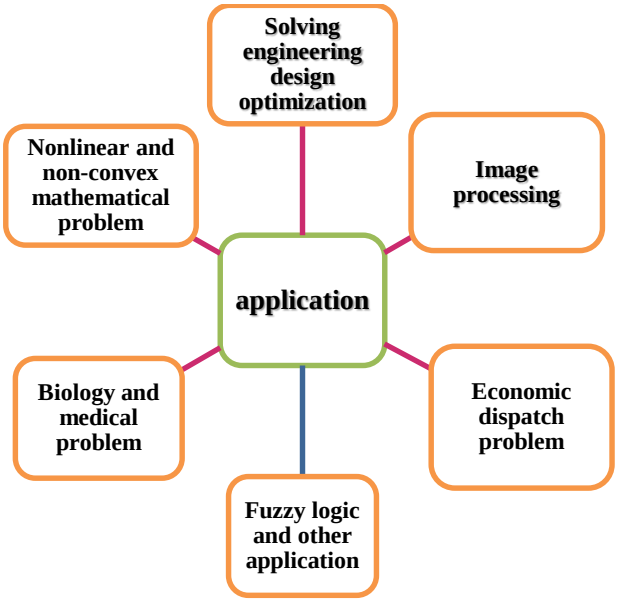


Figure 1. *Some applications of bat algorithms*

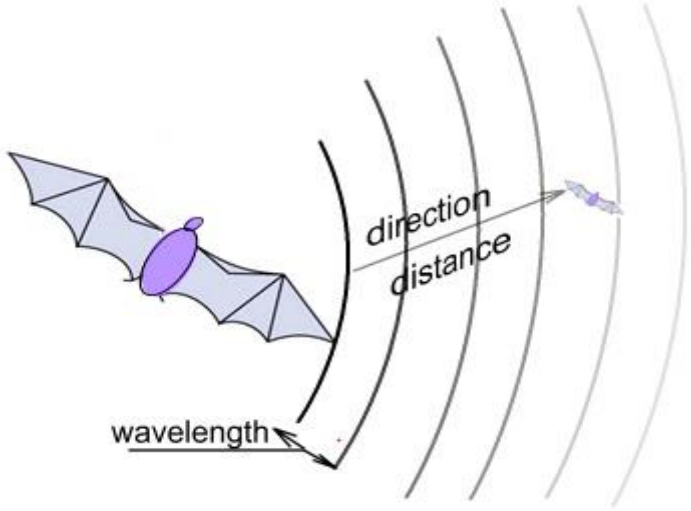


Figure 2. *The direction and intensity of the returned signal*

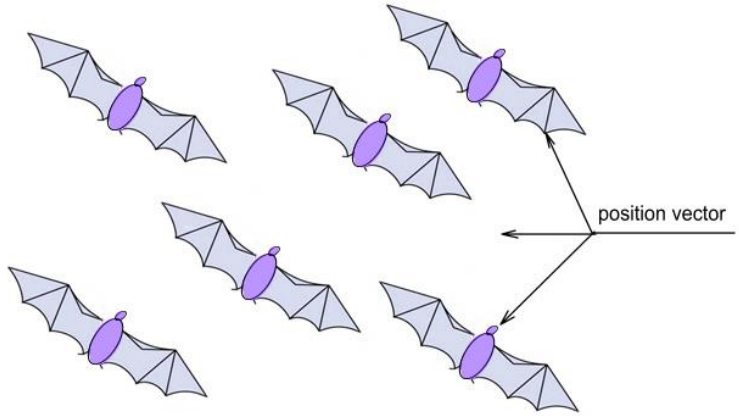


Figure 3. *Position of bats in the search space*

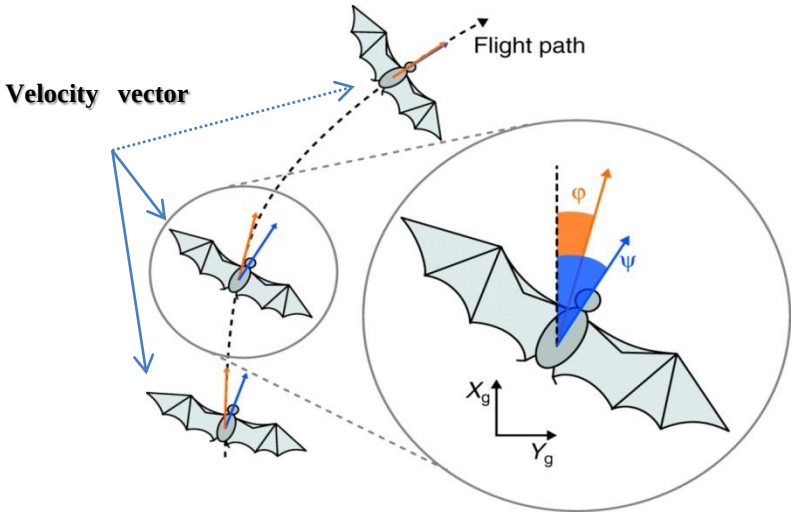


Figure 4. *The velocity vector of bats*.

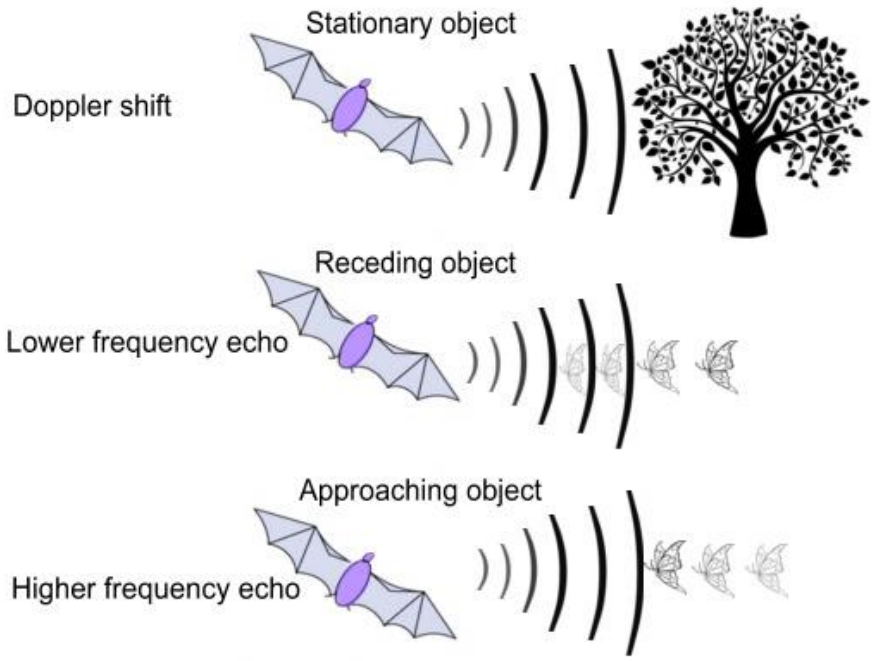


Figure 5. *Frequency vector of the emitted signal*

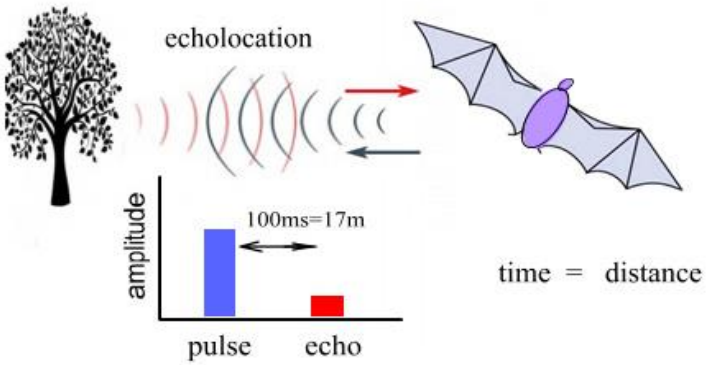


Figure 6. *The amplitude of pulse rate*

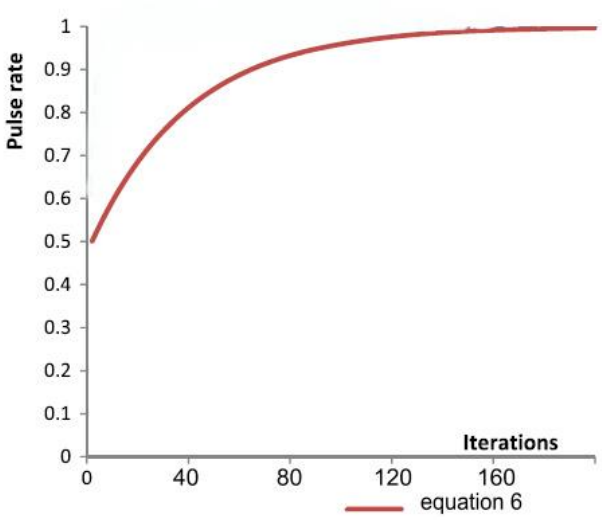


Figure7. *Comparison of pulse rate adaptation strategies.*

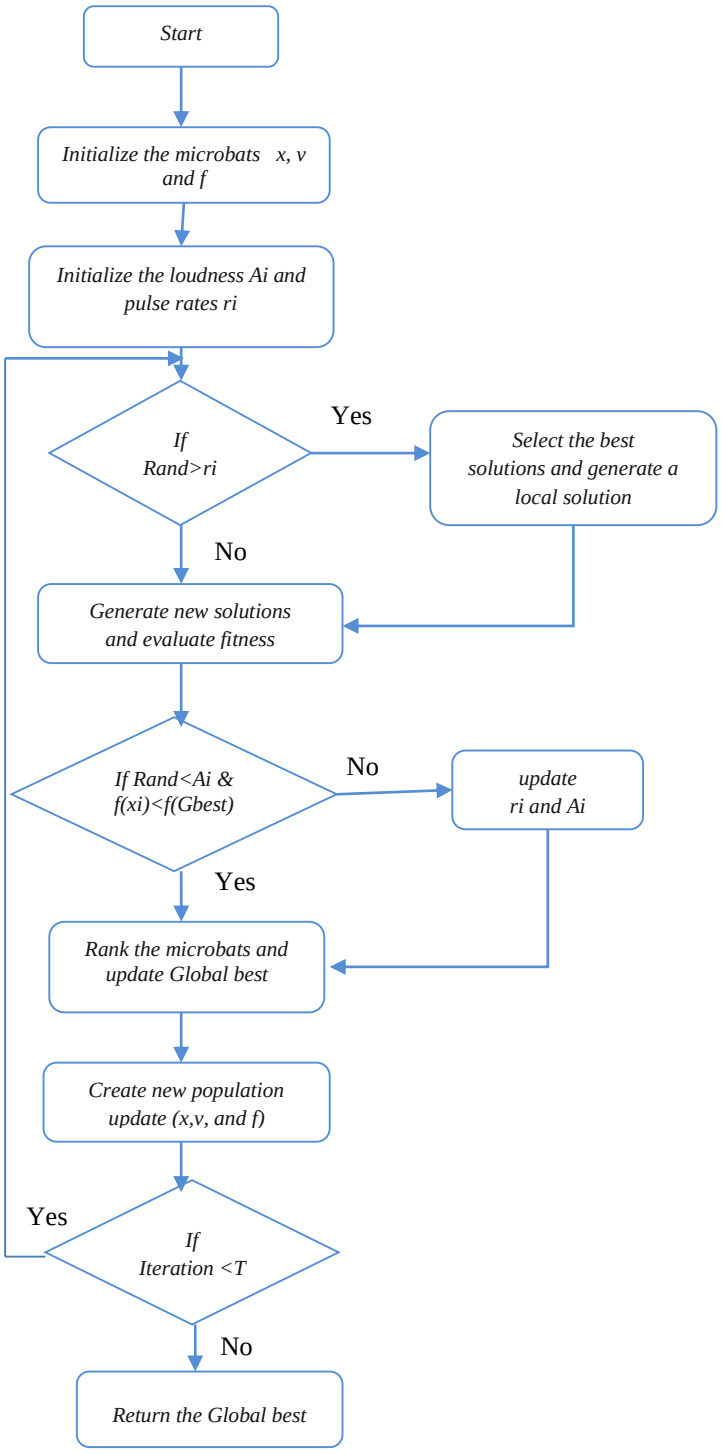


Figure 8. *Proposed architecture of Bat Algorithm*

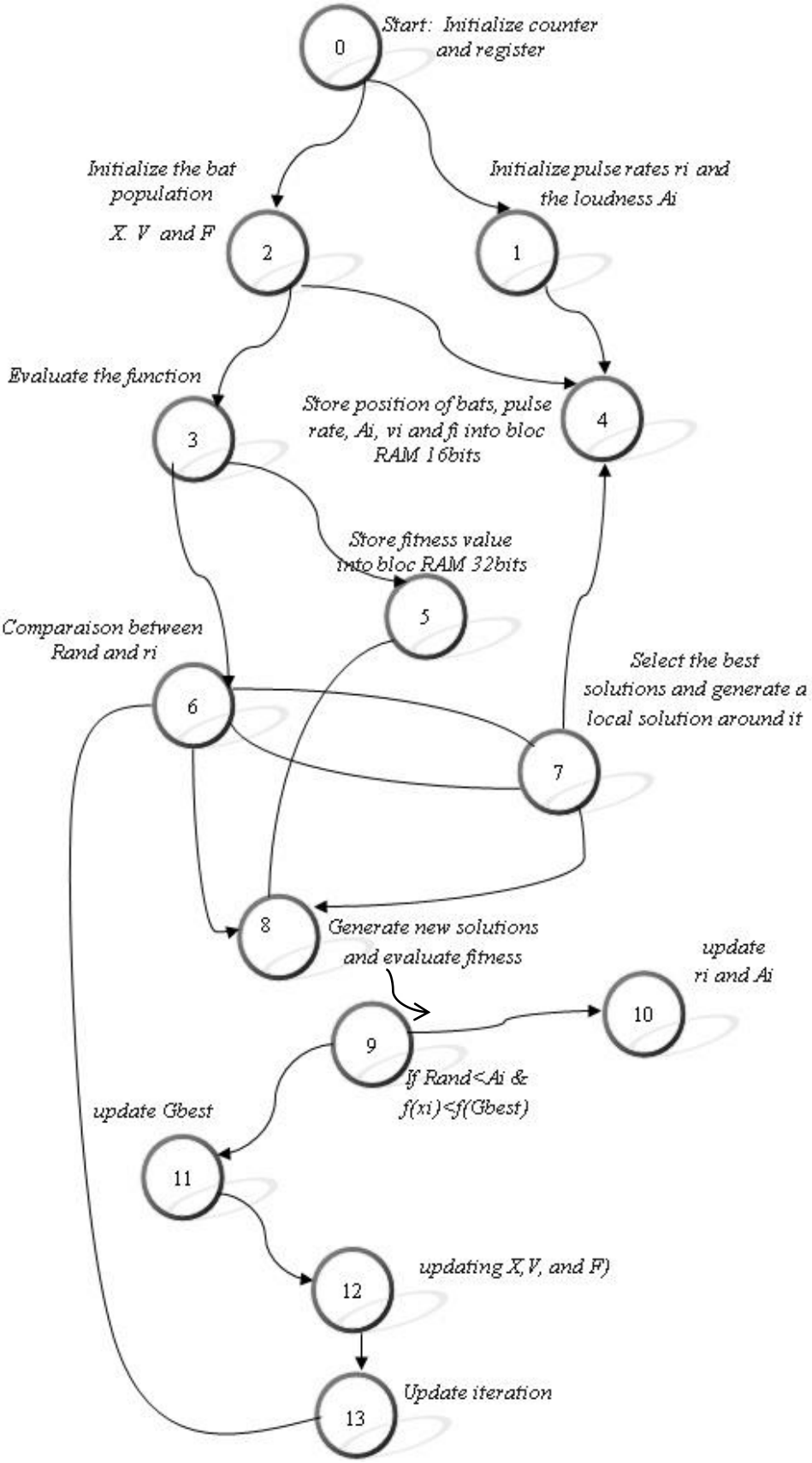


Figure 9. *The finate state machine of the proposed algorithm.*

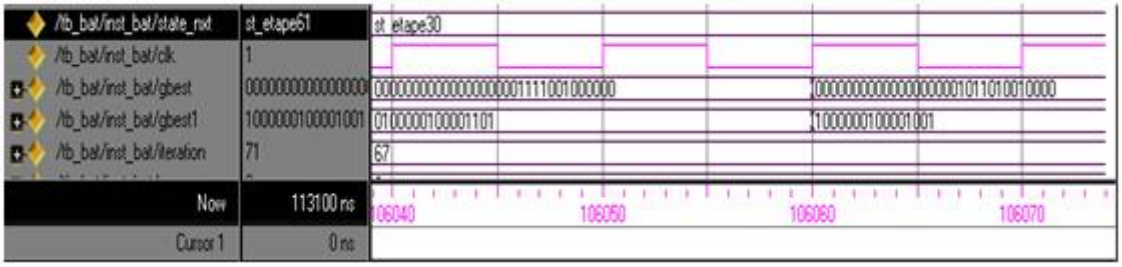


Figure 10. *Expriment results of function* $f(x)=x^2$

Figure 11. *Simulation results in modelsim of function* $f(x)=x^2+y^2$

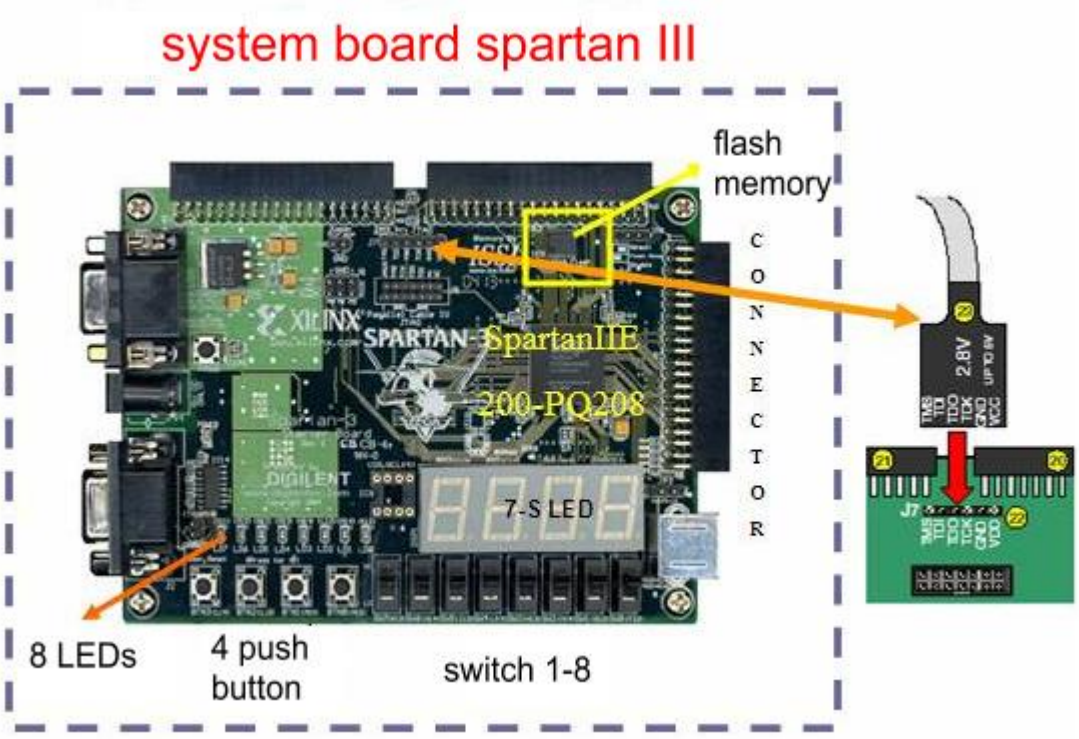


Figure 12. *The block diagram of SPARTAN-3*

Figure 13. *Display of the number of iteration of sphere function*

Table 1. *Device utilisation of material resources*

Device utilization summary (estimated values)			
Logic utilization	***Used***	***Available***	***Utilization(%)***
Number of slices	602	1920	31%
Number of slice flip flops	568	3840	14%
Number of 4 input LUTs	1010	3840	26%
Number of IOBs	18	173	12%
Number of BRAMs	5	12	41%
Number of MULT18X18s	4	12	33%
Number of GCLKS	7	8	87%

Table 2. *Material resources of rastrigin function*

Device utilization summary (estimated values)			
Logic utilization	***Used***	***Available***	***Utilization(%)***
Number of slices	1162	1920	60%
Number of slice flip flops	1001	3840	26%
Number of 4 input LUTs	1690	3840	44%
Number of IOBs	18	173	10%
Number of BRAMs	5	12	41%
Number of MULT18X18s	5	12	41%
Number of GCLKS	8	8	100%

Table 3. *Material resources of rosenbrock function*

Device utilization summary (estimated values)			
Logic utilization	***Used***	***Available***	***Utilization(%)***
Number of slices	1152	1920	60%
Number of slice flip flops	1097	3840	28%
Number of 4 input LUTs	1774	3840	46%
Number of IOBs	18	173	10%
Number of BRAMs	7	12	58%
Number of MULT18X18s	7	12	58%
Number of GCLKS	8	8	100%

Table 4. *Comparison between GA, PSO and Bat algorithms of sphere function*

algorithm	Genetic	PSO	BAT
clock cycle of one iteration	9740	1180	1590
Number of iteration to achieve "0" (STD)	380 (42)	420 (38)	178 (24)
Execution time (ms) (STD)	74,024 (8,18)	9,912 (0,89)	5,660 (0,76)