A Genetic Algorithm Processor Based on Redundant Binary Numbers (GAPBRBN)

Miss: KIRTI JOSHI

Abstract

A Genetic Algorithm (GA) is an intelligent search strategy supported by operations inspired by biological evolution. A typical GA for a particular application is a robust optimization algorithm that mimics the Darwinian theory of evolution. Although a GA is able to find very good solutions for a variety of applications, it typically requires many computations and iterations to be effective, and the amount of time consumed by these computations and iterations is enormous. Thus, software implementations of GAs applied to increasingly complex problems and large search spaces can cause unacceptable delays. An alternative to this approach is the hardware implementation of GAs in order to achieve tremendous speed up over software counterparts by exploiting the inherent parallelism of the GA paradigm. This paper presents the design of libraries of hardware modules for a GA system. The different modules of the GA system, i.e., fixed number generator, selection modules (roulette wheel selection), crossover modules (one-point crossover, two-point crossover, arithmetic crossover and uniform crossover) and the mutation module are described in hardware using the HDLs. These VHDL RTL models are simulated on the Modelsim 6.5e to check for functionality and performance.

***Index Terms*— Genetic Algorithm, Redundant binary number Modelsim 6.5e.**

1 INTRODUCTION

GAs, as powerful and broadly applicable stochastic search and optimization techniques, are perhaps the most widely known types of evolutionary computation methods today. GA of evolutionary computation is inspired by Darwin's theory of evolution. A GA generates a population of possible solutions encoded as chromosomes, evaluates their fitness and creates a new population by applying genetic operators which are crossover and mutation. By repeating this process over many generations, the genetic algorithm has five basic components which are (Gen and Cheng, 2000)

- A genetic representation of solutions to the problem.
- A way to create an initial population of solutions.
- An evaluation function rating solutions in terms of their fitness.
- Genetic operators that alter the genetic composition of children during reproduction.
- Values for the parameters of genetic algorithms

Manuscript received Nov.-2014
***Miss Kirti joshi** , Electronics and communication, RGPV(Bhopal)/ Swamivivekanand college of engineering, Indore/ Ujjain,India, Mobile No-9770053748*

GAs uses fitness values of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes and the mutation operator changes the gene value in some randomly chosen location of the chromosome. After a number of successive reproductions, the less fit chromosomes become extinct, while those best fit gradually come to dominate the population. John Holland, from the University of Michigan began his work on genetic algorithms at the starting of the 60s. A first accomplishment was the publication of Adaptation in Natural and Artificial System in 1975. Holland had a double enhance the understanding of natural adaptation method, and to design artificial systems having properties similar to natural systems. The basic idea for this process is as follows: the genetic pool of a given population potentially comprises the solution, or an improved solution, to a given adaptive problem. This solution is not "active" because the genetic combination on which it relies is split between various subjects. Only the association of different genomes can lead to the solution. Simply speaking, we could by example consider that the shortening of the paw and the extension of the fingers of our basilosaurus are controlled by 2 "genes" [1]. No subject has such a genome, but during reproduction and crossover, new genetic arrangement occur and, finally, a subject can inherit a "good gene" from both parents: his paw is currently a flipper. Holland method is particularly effective because he not only considered the role of mutation (mutations improve very seldom the algorithms), but he also used genetic recombination,(crossover): in these recombination, the crossover of partial solutions significantly improve the capability of the algorithm to approach, and eventually find, the optimum. Recombination or sexual reproduction is a key operator for natural evolution. Technically, it takes two genotypes and it produces a new genotype by mixing the gene found in the originals. In natural science, the most communal form of recombination is a crossover, two chromosomes are cut at one point and the halves are spliced to create new chromosomes. The outcome of recombination is very important because it allows characteristics from two different parents to be asserted. If the father and the mother possess different good qualities, we would suppose that all the good qualities will be passed onto the child. Thus the offspring, just by combining all the best features from its parents, might exceed its ancestors. Many people believe that this mixing of genetic material via sexual reproduction is one of the most powerful features of Genetic Algorithms. As a quick digression about sexual imitation, Genetic Algorithms representation usually does not differentiate male and female individuals (without any perversity). As in many living species (e.g., Snails) any individual can be either a male or a female. Actually, for almost all recombination operators, mother and father are interchangeable [2]. Mutation is the other way to get new genomes. Mutation contains in changing the value of genes. In natural evolution, mutation frequently engenders non-viable genomes. Actually mutation is not a very frequent operator in natural evolution. Nevertheless, is optimization, a few random changes can be a good way of exploring the search space

quickly.According to Darwinism, inherited variation is characterized by following properties:

1. Variation be essentially copying for the reason that selection does not create directly anything, but presupposes a large population to work on.
2. Variation must be small-scaled in practice. Species do not appear suddenly.
3. Variation is undirected. This is also known as the blind watchmaker paradigm.Then, the genetic algorithm

loops over an iteration process to make The population evolve. Each iteration consists of the following steps:

Selection:
The first step consists in selecting individuals for reproduction. This assortment is completed randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.

Reproduction*:*
In the second step, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm is able to use both recombination and mutation.

Evaluation*:*
Then the fitness of the new chromosomes is calculated.

Replacement*:*
Through the last phase, individuals from the old population are killed and replaced by the new ones [5].The algorithms stopped when the population come together to the optimal solution.The elementary genetic algorithm is as follows:

1. **Start -** Genetic random population of n chromosomes (suitable solutions for the problem)

2. **Fitness -** Evaluate the fitness f(x) of each chromosome x in the population

3. **New population -** Create a new population by repeating following steps until the New population is complete

4. **Selection -** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).

5. **Crossover -** With a crossover probability, cross over the parents to form fresh offspring (children). If no crossover was achieved, offspring is the exact copy of parents.

6. **Mutation -** With a mutation probability, mutate new offspring at each locus (position in chromosome)

7. **Accepting -** Place new offspring in the new population.

8. **Replace -** Use new generated population for a further sum of the algorithm.

9. **Test -** If the end condition is satisfied, stop, and return the best solution in current population.

10. **Loop -** Go to step2 for fitness evaluation [4].

The Genetic algorithm procedure is deliberated through the GA cycle

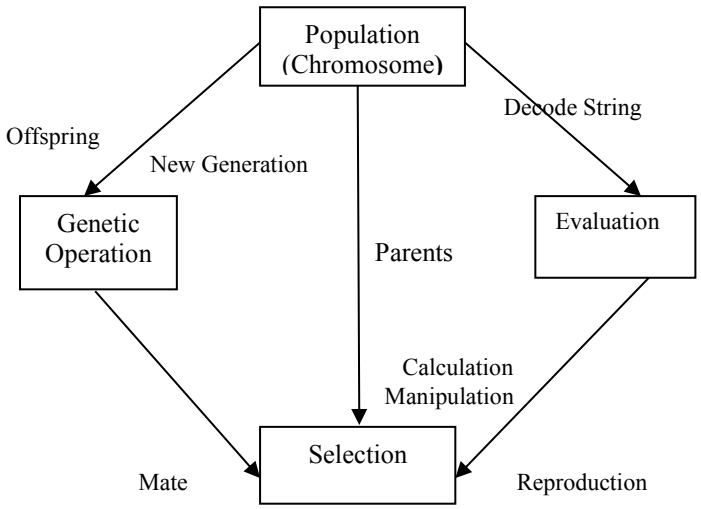

Figure 1.1: Genetic Algorithm Cycle

Reproduction is the process by which the genetic material in two or more parent is combined to obtain one or more offspring. In fitness evaluation stage, the individual's superiority is measured. Mutation is achieved to one individual to produce a new version of it where some of the original genetic material has been randomly changed. Selection process helps to decide which individuals are to be used for reproduction and mutation in order to produce new search points [4].

2 REVIEW

Carrying out literature review is very significant in any research project as it clearly establishes the need of the work and the background development. It generates related queries regarding improvements in the study already done and allows unsolved problems to emerge and thus clearly define all boundaries regarding the development of the research project. Plenty of literature has been reviewed in this chapter in connection with Genetic Algorithm. We start with basic overview of Genetic Algorithm
Genetic Algorithms are heuristic search algorithms based on natural genetics. They represent an intelligent exploitation of random search to solve optimization problems. Although randomized, GAs are not totally random, since they exploit historical information to direct the search into regions of better performance within the search space [9].

3 PROPOSED METHODOLOGY

This is describes the various modules of the Genetic Algorithm system. These GA modules are implemented in VHDL and consist of; Fixed Number Generator, Selection modules, Crossover modules, Mutation module and Fitness modules.

4 FIXED NUMBER GENERATOR

In this research work, to reduce the hardware complexity, we have used fixed number generator rather than the conventional Random Number Generators. In this approach we have fixed the initial population.

4.1 SELECTION MODULES

There are various selection schemes for Genetic Algorithms, each with different characteristics. An ideal selection scheme is

one that is easy to code, and efficient for both non-parallel and parallel architectures. Furthermore, a selection scheme should be able to adjust its selection pressure to adapt to different domains, where the selection pressure is the degree to which better individuals are favoured [9].

The main task of the selection module is to select individuals from the population so that these individuals can be sent to the crossover and mutation modules in order to attain new offspring. There are numerous selection methods such as roulette wheel selection, rank selection, tournament selection etc., and they are usually biased towards more highly fit individuals. Selection is one of the key operators on GAs that ensures survival of the fittest [7]. The

different types of selection modules implemented in this research work are described below:

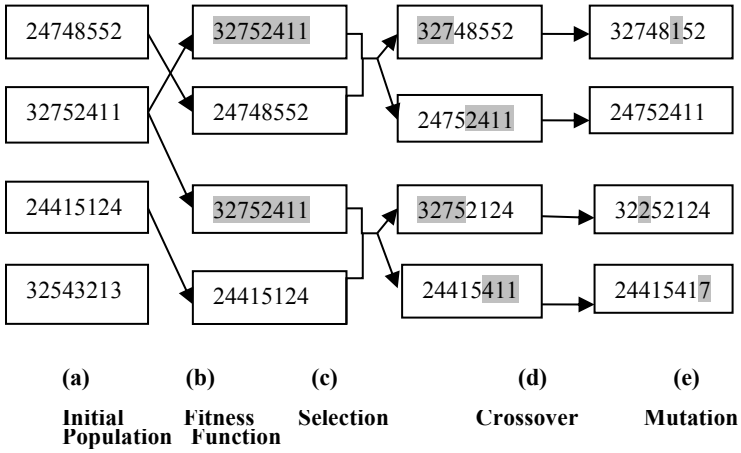

Figure 4.1: The Various Processes of a GA System [9]

Roulette Wheel Selection This is one of the more popular fitness-proportional selection methods. Here, the parents or individuals are selected based on their fitness values, and thus more fit chromosomes have a higher probability of being selected. The analogy to a roulette wheel can be envisaged by imagining a roulette wheel in which each candidate solution represents a section on the wheel; the size of the sections are proportionate to the appropriate fitness measure [9].

An example of the Roulette Wheel Selection for a GA process is illustrated below. This process has an initial population of six members.

Individual	Chromosome	Fitness	Cumulative Fitness
X1	101100	20	20
X2	111000	7	27
X3	001110	6	33
X4	101010	10	43
X5	100011	12	55
X6	011011	9	64

Table 4.1: Example Population to Illustrate Roulette Wheel Selection

Random Number Generated	Individual Selected
24.3	X2
42.8	X4
54.88	X6
28	X2
19.78	X1

Table 4.2: Individuals Selected for the GA Population by the Roulette Wheel Selection Me

4.2 CROSSOVER MODULES

Crossover is the fundamental mechanism of genetic rearrangement in Genetic Algorithms. The purpose of crossing individuals in GAs is to test new parts of the target regions instead of testing the same string repeatedly in successive generations [10]. Once the Selection Module selects two chromosomes, the Crossover Module is used to generate two new offspring. This is mainly done by the exchange of strings between the two parent chromosomes. If the best strings of the parents are combined then the resultant offspring may be better than the two parents. The amount of crossover is controlled by the crossover probability, , which is defined as the ratio of the number of offspring produced in each generation to the population size. A higher crossover probability allows proper exploration of the search space and reduces the chances of settling for a false optimum while a lower crossover probability enables exploitation of existing individuals in the population that have relatively high fitness [7]. Thus, the performance of the GA depends greatly on the type of crossover operator used. There are many crossover schemes such as the one-point crossover, two-point crossover, uniform crossover, arithmetic crossover etc.

4.2.1 One-point Crossover

In this crossover scheme, a single crossover point is chosen at random and the first offspring is equal to the first parent up to the crossover point and identical to the second parent after the crossover point. The second

offspring is identical to the second parent till the crossover point and to the first parent after the crossover point. This scheme is illustrated with an example in Figure 4.2.1. 'Parent1' and 'Parent2' represent the two parent chromosomes and a crossover point between bits 3 and 4 is selected. The two resulting offspring obtained after the process of crossover are also shown.

7654|3210 bit position

Parent1: ***1010***|*1011*

Parent2: 1101|1111

Offspring1: ***1010***|1111

Offspring2: 1101|***1011***

Figur4.2.1: One-point Crossover

4.3 MUTATION MODULE

Mutation is the process which consists of making small alterations to the bits of the chromosomes. In the binary-encoded GA, mutation is merely the process of flipping random bits of the chromosomes that have undergone crossover. The mutation process is performed with a small probability of mutation, which is defined as the probability of mutating each chromosome. It controls the rate at which new bit values are introduced into the population. If Pm is too high, the offspring will be too random and hence lose their resemblance to the parents. Hence, low mutation rates are preferred so that new values are introduced that may prove to be useful.

The mutation process is illustrated in Figure 4.3, where bit 4 undergoes mutation. Although the crossover operator is the most efficient search mechanism, it does not independently guarantee exploration of the entire search space with a finite population size. This gap is filled in by the mutation operator [7].

76543210 bit position

Before Mutation: 10110111

After Mutation: 10100111

Figure 3.4: Mutation Operation

4.4 FITNESS MODULE

The fitness module is application specific and is designed according to the function to be optimized or evaluated. In this dissertation, two separate functions have been presented for optimization. The two mathematical functions are given below:

$$(X) = | x_2\text{-}25|$$

$$(X) = | x_2\text{-}40000|$$

These functions are optimized by using different combinations of modules from the library developed in this research work.

5 SIMULATION AND RESULTS

5.1 RTL SCHEMATIC

Synthesis has been carried out using Xilinx13.1 ISE. Figure 5.1 shows the pin diagram for top level block

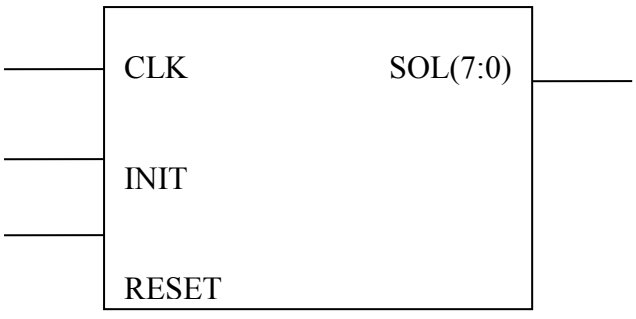

Figure 5.1 : Pin diagram for top level block

5.2 VHDL SIMULATIONS

Simulation results for mathematical functions $f(x) = |x\ 2-\ \ 25|$ and $f(x) = |x\ 2-\ 40000|$, has been shown in figures 5.1 and 5.2 respectively. Simulation is carried out on Modelsim 6.5e simulator.

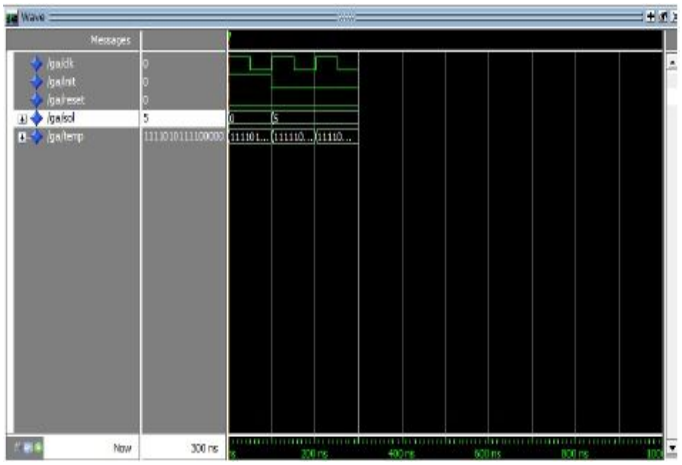

Figure 5.1 : Simulation waveform for mathematical functions $(x) = |x\ 2- 25|$

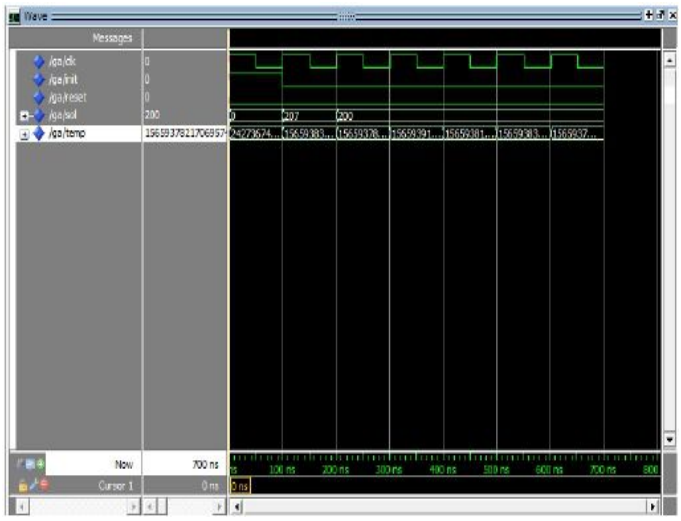

Figure 5.2: Simulation waveform for mathematical functions $(x) = |x\ 2- 40000|$

6 CONCLUSION AND FUTURE SCOPE

This research work presents the design of a library of modules to enable hardware implementation of Genetic Algorithms. The various modules of the GA system are described in Hardware Description Language (VHDL). This enables reuse and flexibility for the modules and achieves higher speed for the GA than its software counterpart. The GA requires very intensive computations to perform optimization. Hence, a dedicated VLSI implementation is necessary.

Each module was verified to be functionally correct through numerous simulations. The modules were also simulated to analyze the performance. After the correct functionality of each module was tested individually, the modules were connected together and the system was functionally verified. The VHDL simulations on the ModelSim 65e. platform tool confirmed that each of these modules and the different GA systems were functionally correct.

The performance results for the GA system maximizing the function,$(x) = |x\ 2 - 25|$ and$(x) = |x\ 2 - 40000|$ were good. Both the systems were able to find the maximum attainable value for the function in most of the cases. This demonstrates that GAs are efficient systems for optimizing specific functions. The work could be further improved through architectural changes to optimize the time and area requirements. Lesser no. of processing elements could be utilized to decrease the hardware requirements and strike a balance between the delay and hardware requirements tradeoff.

7 ACKNOWLEDGEMENT

The path to the successful completion of this project has gone through various ups and downs. Patience, determination and perseverance coupled with help and encouragement from several quarters have paved the path to success.

8 REFERENCES

1http://www.saylor.org/site/wp-content/uploads/2011/06/CS411-3.1.pdf

2 Xavier Hue, "Genetic Algorithms for Optimization Background and Applications", The University of Edinburgh, Version 1.0, February 1997.

3 Jarmo T. Alander, "Genetic Algorithms and other "natural" optimization method to solve hard problems — a tutorial review", Department of Electrical Engineering and Energy Technology University of Vaasa,Finland, 2012.

4 Rohini V, "A Phased Approach to Solve the University Course Scheduling System", International Journal of Computational Engineering Research, Vol. 03, Issue 4, 2013.

5 Nitin S. Choubey, Madan U. Kharat, "Grammar Induction and GeneticAlgorithms: An Overview", Pacific Journal of Science and Technology,Volume 10. Number 2. November 2009.

6 Goldberg, G.A., "Genetic Algorithms in Search", Optimization and Machine Learning, Pearson Education, Inc., 2002.

7 Mazumder, P. and Rudnick, E. M.,"Genetic Algorithms for VLSI Design",Layout and Test Automation, Prentice Hall, 1999.

8 Houck, C., Joines, J., and Kay, M., A Genetic Algorithm for Function Optimization: A MATLAB Implementation, NSCU-IE Technical Report, 1995.

9 Miller , B. L. and Goldberg, D.E., "Genetic Algorithms, Tournament Selection and the effects of Noise", Technical Report, University of Illinois,1995.

10 http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm

Name: Kirti joshi
Education: M.E. (Pursuing)
College Name: Swami Vivekanand College of Engineering, indore (M.P.)
Residential address: 55 Abdalpura, Ujjain(M.P.)
Mob. No.- 9770053748

