Introduction to Genetic Algorithms
Dr. Mohammad Najim Abdullah

Genetic Algorithms (GAs) are robust search and optimization algorithms that mimic the theory of evolution and natural selection. GAs were originally developed by John Holland in 1975 and have since been applied to many applications, including the Traveling Salesman Problem (TSP,and circuit partitioning problems. Since GAs are heuristic methods, they are not guaranteed to find the optimum, but various applications have shown that they are able to find very good solutions to hard optimization problems. But GAs implemented in software, when applied to increasingly complex problems and large search spaces can cause unacceptable delays. Hence, an alternative to this approach is to implement the GA in hardware. The combination of pipelining and parallelism possible in a hardware GA implementation increases the speed of the system when compared to its software counterpart. This allows the hardware implementation to be applied to various complexes applications. Since GA operations are relatively simple, GAs are good candidates for FPGAs and reconfigurable systems.

Motivation

GAs are extensively used in many applications across a large and growing number of disciplines. Although a GA is able to find very good solutions for a variety of applications, the amount of time consumed for large computations and iterations is enormous. Hence, software implementation of GAs for increasingly complex applications can cause unacceptable delays. Also, GAs lend themselves easily to pipelining and parallelization. All these factors make GAs good candidates for hardware implementation. This thesis presents the design of a library of modules to support hardware implementation of GAs, providing a rich set of modules and example architectures which users can easily be customized for specific applications.

Introduction to Genetic Algorithms

Genetic Algorithms are a class of evolutionary algorithms that use biologically-derived techniques such as inheritance, natural selection, crossover (or recombination) and mutation. They work on the principle of “survival of the fittest”, where the less fit members of a particular generation are replaced by new members formed by combining parts of highly fit members. Traditionally, solutions to GAs are represented in binary as strings of 0s and 1s, but different encoding schemes are also possible.
GAs differ from other conventional optimization processes in the following ways:
- GAs work with a coding of the parameter set, not the parameters themselves;
- GAs search from a population of points, not a single point;

- GAs use objective function information, not derivatives or other auxiliary knowledge;
- GAs use probabilistic transition rules, not deterministic rules.

Genetic Algorithms work by evolving a population of members over a number of generations. First, the initial population of members is generated. After this, the fitness of each member is evaluated, where the fitness is a function of the application. Two members or individuals are then selected simultaneously. The selection is usually proportional to the fitness value, i.e., the members with higher fitness have a greater probability of being selected. These members are crossed to form new individuals and these new members are occasionally mutated to avoid convergence to local optima. The resulting members replace the less fit members of the old population. This process is repeated until a stopping criterion is met. The stopping criterion could be either a predetermined number of generations or lack of improvement in the fitness values after a certain number of generations. Thus, the fitness value of the population is typically improved with every new generation.

Overview of Genetic Algorithms

Genetic Algorithms were developed by John Holland in 1975 in an attempt to explain the adaptive processes of natural selection and to thus design an artificial system's software to retain the important mechanisms of natural systems. A typical GA is an intelligent search strategy supported by operations inspired by biological evolution. The theory of evolution was introduced by Charles Darwin to explain his observations of plants and animals in the natural ecosystem. He observed that as every new generation was associated with some changes, the less-fit individuals tended to lose the battle for survival in the competition for food. Thus, this principle of "survival of the fittest" leads to improvement in the species with every new generation and this was the basic principle implemented in all GA systems.

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

Terminology of Genetic Algorithms

Because Genetic Algorithms are rooted in both natural genetics and computer science, the terminology used in the GA literature is a mix of the natural and artificial. All Gas work on a *population* or a collection of solutions to the given problem. Each individual in the population is called a string or *chromosome*, in analogy to chromosomes in natural systems. These individuals are coded as binary strings and the individual characters in the strings are referred to as *genes*. In each iteration of the GA, a new

generation is evolved from the existing population in an attempt to obtain better solutions.
The *fitness* function i.e., the evaluation function is used to determine the fitness of each chromosome. The fitness function is problem specific and user defined. In the selection process, two chromosomes are chosen simultaneously from the population for reproduction. These selected strings are called *parents*. Crossover is the main operator used for reproduction. It combines properties of both parents to create two new chromosomes called *offspring*.
Mutation is an incremental change made to each member of the population, with a very small probability. Mutation enables new features to be introduced into a population. The various processes of the GA system are shown below [figure 2.1].

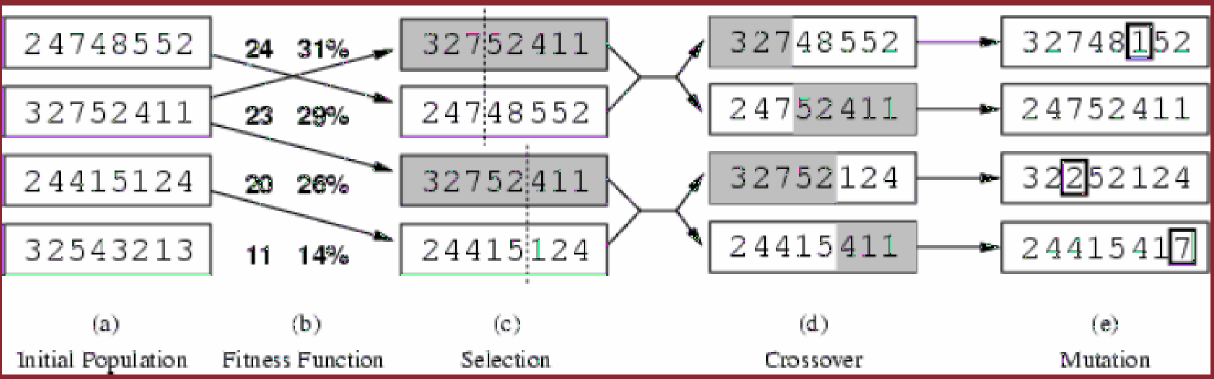

Figure 2.1. The Various Processes of a GA System [18]

GAs basically use two types of population replacement schemes:
- a total replacement scheme or the simple GA
- a partial replacement scheme or the steady state GA.
These are described in detail in the following sections.

A Simple Genetic Algorithm

Two of the most compelling reasons for using a GA are simplicity of operation and power of effect . The simple GA is also known as the total replacement algorithm and this is illustrated in figure 2.2.
A simple GA is basically composed of three operators – selection, crossover and mutation. These three operators primarily involve random number generation, copying and swapping partial strings, thus making these GAs simple to implement. Figure 2.3 below illustrates the simple GA process in flowchart form.

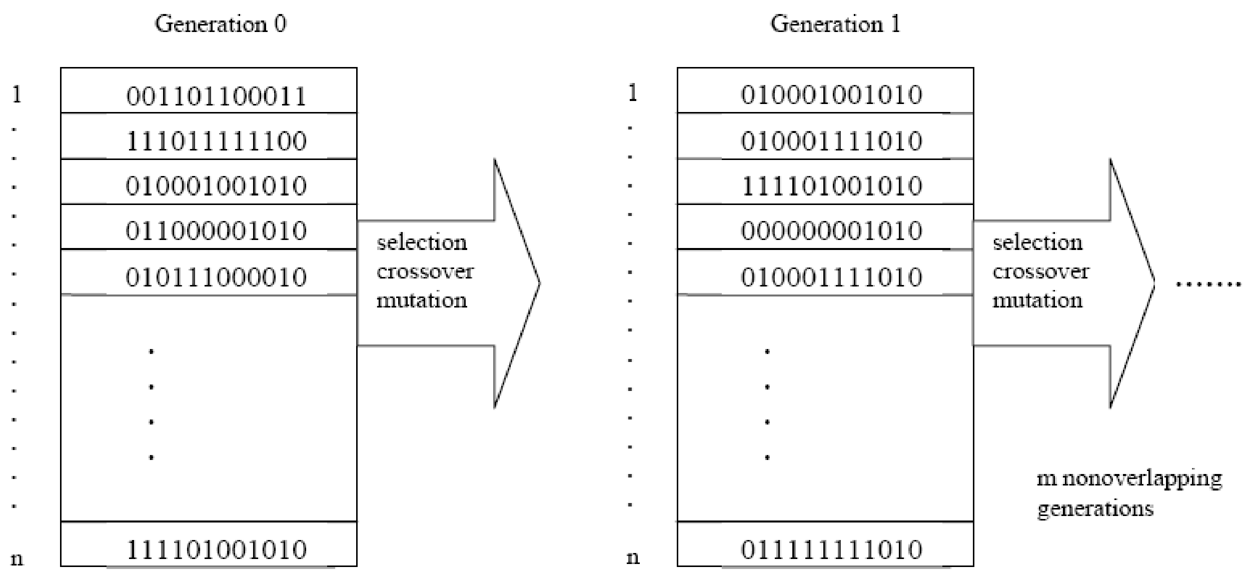

Figure 2.2. A Simple Genetic Algorithm [4]

The GA process starts with the generation of the initial population, which is usually generated randomly, but may also be supplied by the user. Each chromosome of the population is evaluated according to some fitness function that is application specific. This is followed by the selection process where two individuals are selected at a time. These two parents then undergo crossover to form two new offspring, which are then mutated with a low mutation probability. A highly fit population is evolved through several generations of selection, crossover and mutation until some stopping criteria are met.
The GA can be terminated according to several stopping criteria:
- A fixed number of generations have occurred;
- All individuals in the population converge to the same string;
- No improvements in the fitness value are found after a certain number of generations.
Once the stopping criterion is met and the new generation is evolved, the old generation can be discarded.

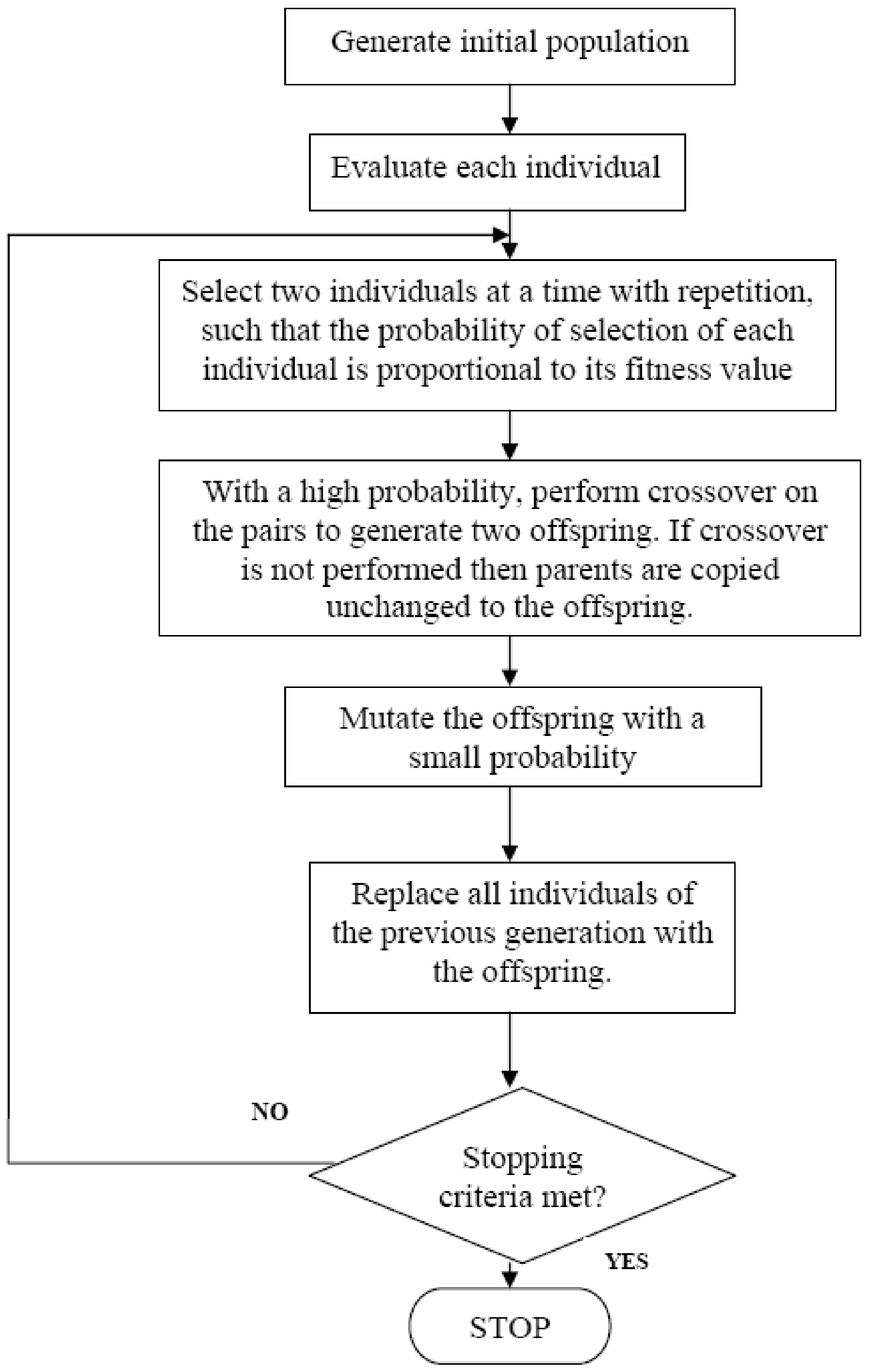
Generate initial population
Evaluate each individual
Select two individuals at a time with repetition, such that the probability of selection of each individual is proportional to its fitness value
With a high probability, perform crossover on the pairs to generate two offspring. If crossover is not performed then parents are copied unchanged to the offspring.
Mutate the offspring with a small probability
Replace all individuals of the previous generation with the offspring.
NO
Stopping criteria met?
YES
STOP

Figure 2.3. Flowchart of a Simple Genetic Algorithm [4]

GA: Disadvantages

1. No guarantee for optimal solution within a finite time
2. Weak theoretical basis
3. Interdependency of genes
4. Parameter tuning is an issue
5. Often computationally expensive, i.e. *slow*

GA: Advantages

1. A robust search technique
2. No (little) knowledge (assumption) the problem space
3. Fairly simple to develop: low development costs
4. Easy to incorporate with other methods
5. Solutions are interpretable
6. Can be run interactively, i.e. accommodate user preference
7. Provide many alternative solutions
8. Acceptable performance at acceptable costs on a wide range of problems
9. Intrinsic parallelism (robustness, fault tolerance)