Chapter 2

Simple Genetic Algorithms

In this chapter, I will show the simple genetic algorithms (SGA) (Goldberg, 1989). The GA is a powerful search and optimisation algorithm. It is used in many EHW applications because the solution representation, which is a binary string, exactly matches to the configuration bits of programmable logic device (PLD). Therefore the GA can search for the configuration bits which satisfy the design constraints (e.g. circuit function, silicon area, power consumption). The following sections describe how to apply GA to a particular problem.

2.1 Solution Representation

The GA is applied to an optimisation problem in which the solution can be represented by a fixed-length binary string. Since everything in computer is stored in binary, most optimisation problems can be solved by GA (Ronald, 1997). I will give an example of applying GA to solve one-max problem. The goal is simply finding a binary string composed of all 1-bits. In this example, the string length is set at 8, and therefore the number of possible solutions is 2^8. In practice, the string length may be longer than 100 bits, resulting more than 2^{100} possible solutions. The exhaustive search on a state-of-the-art computer is relatively impossible. For that reason, the GA has been used because it performs effectively in a large search space.

2.2 Fitness Function

The fitness function is a function to be optimised. The fitness means the quality of solution, or how much the solution is close to the optimum. In this example, the fitness function is defined as $f(x)$ where x is a binary string and $f(x)$ returns the number of 1-bits in x. For example, $f(01101011) = 5$, $f(10010000) = 2$, and $f(00000000) = 0$.

2.3 Initialisation

The GA starts with a random *population*, which is a set of possible solutions. A solution, in the population, is called *individual*. Each bit of an individual is created by flipping an unbiased coin. The initial population, of which the size is 4, is shown in Figure 2.1.

No.	Individual	Fitness
1	11001000	3
2	00100010	2
3	01000100	2
4	00001000	1

Figure 2.1: Initial population.

2.3 Selection

The GA explores a large number of solutions in the search space simultaneously by reproducing the population. The genetic operators, namely *crossover* and *mutation*, are employed to produce the offspring. By simulating the natural selection, the fitter individuals have more chance to survive to the next generation. This causes the fitness improvement because the fitter individuals seem to reproduce the more fitter ones.

The individuals are probabilistically selected to be performed genetic operators. This method, in simple GA, is called *roulette-wheel selection*. The n-hole roulette, where n is the number of individuals, is constructed. A hole size is proportional to the fitness of an individual. The roulette wheel, built from the population in Figure 2.1, is shown in Figure 2.2. The fitter individuals have more chance to be selected.

2.4 Reproduction

A pair of individuals, called *parents*, is selected using the roulette wheel. With the probability of P_c, the crossover operation is performed to produce a pair of *children*, otherwise two parents are copied to be children. The crossover is shown in Figure 2.3.

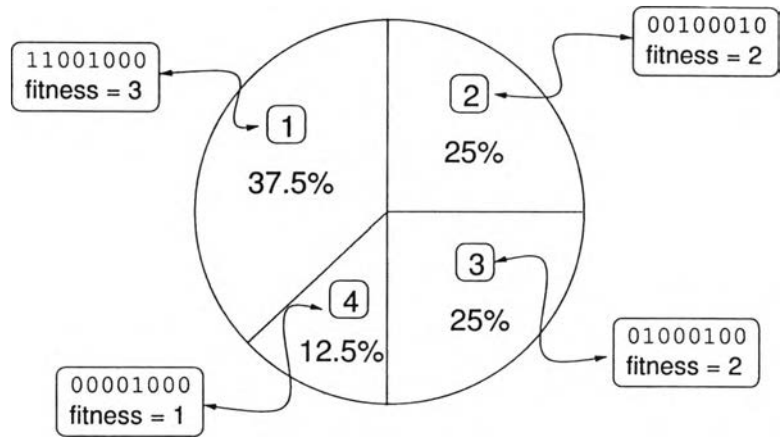

Figure 2.2: Roulette-wheel selection.

The cut point is randomly selected, then the children are produced by the recombination of partial solutions. Following that, the mutation operator is applied to the children. Each bit of a child is flipped with the probability P_m. The mutation operator is shown in Figure 2.4. In general, P_c and P_m are set at 0.06 and 0.001 respectively. A pair of parents are selected by the roulette wheel until the number of children equals to parents. The next generation begins. All parents are discarded, then the children become parents. The reproduction is repeated until the optimal solution is found or the maximum number of generations is reached. The reproduction is shown in Figure 2.5. With this artificial evolution, the optimal solution emerges after a number of generations.

2.5 Theory of Genetic Algorithms

The theory of GA is based on the *schema* – a template representing a set of fixed-length strings. The schema H consists of symbols "0", "1", and "*", where "0" and "1" are the fixed characters and "*" is *don't care* symbol (similar to a wildcard character). Thus a schema $H = 10 * 0$ represents strings 1000 and 1010. The *order* of schema, $o(H)$, is defined as the number of "0" and "1" in schema H, for example, $o(*0 * 1) = 2$. The defining length, $\delta(H)$, is the distance between the first and the last fixed characters in schema H, for example, $\delta(0 * 1*) = 3 - 1 = 2$.

It can be shown that short, low-order, above-average fitness schemata increase exponentially from generation to generation. This is named as *Schemata Theorem* or *Fundamental Theorem of Genetic Algorithms*. It can also be shown that the GA processes

$O(n^3)$ schemata, where n is the population size. This important property is named as *implicit parallelism*. Thus GA finds the optima effectively in a large search space. Although the Schemata Theorem is now being debated, the success of GA over the past years guarantees that it is a powerful optimisation algorithm.

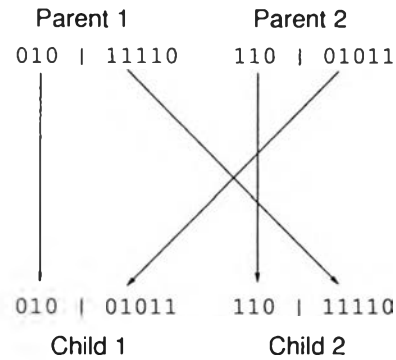

Figure 2.3: Single-point crossover.

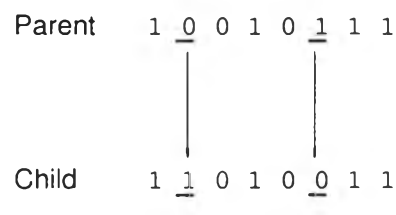

Figure 2.4: Mutation.

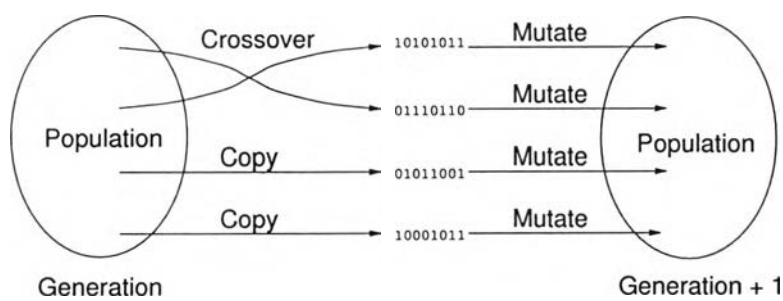

Figure 2.5: Reproduction.