Genetic Algorithms for solving the placement and routing problem of an FPGA with area constraints

Manuel Rubio del Solar
Dpto. Informática
Universidad de Extremadura
Cáceres 10071
Spain
mrubio@unex.es

Juan Manuel Sánchez Pérez
Dpto. Informática
Universidad de Extremadura
Cáceres 10071
Spain
sanperez@unex.es

Juan Antonio Gómez Pulido
Dpto. Informática
Universidad de Extremadura
Cáceres 10071
Spain
jangomez@unex.es

Miguel Ángel Vega Rodríguez
Dpto. Informática
Universidad de Extremadura
Cáceres 10071
Spain
mavega@unex.es

Abstract – **In this work a method based on Genetic Algorithms for solving the placement and routing problem of a Boolean function in a Field Programmable Gate Array (FPGA) is presented. The case of constraints defining a restricted area in the FPGA is also considered. We have faced the problem using a Simple Genetic Algorithm (SGA) and a Parallel Genetic Algorithm (PGA) using the MPI Standard. With the proposed method cuasi-optimal solutions in reasonable time are found.**

I. INTRODUCTION

The Field Programmable Gate Arrays (FPGAs) are integrated devices used for implementing as logical circuits Boolean functions after a programming process with the appropiated tools[8].

Although there are in the market several types of FPGAs, in this work we use the FPGA model based on islands [2] (FIG. 1). In this type of FPGA there are three kind of elements: Configurable Logic Blocks (CLBs) by the user, that allow to implement Boolean functions of a given variable number, Input/Output Blocks (IOBs) that communicate the FPGA with the outside world and Interconection Resources (IR) as Interconection Arrays and Channels. Through the IR go the interconnection lines used for connecting the CLBs or the CLBs with the IOBs.

In this work, we pose and solve the placement and routing problem with area constraints of a Boolean function in an island based FPGA [2].

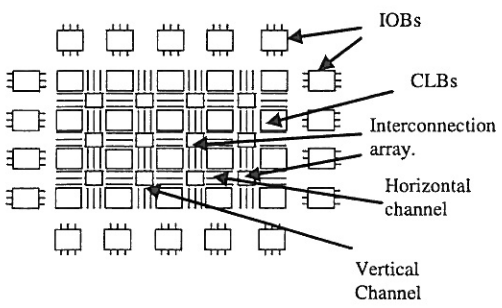

FIG 1. Island based FPGA.

The rest of the paper is structured as it follows: In section 2 the problem is presented. In section 3 the GA design and implementation are explained. In section 4 some found results are shown and finally in section 5 the conclusions and future work are shown.

II. PROBLEM STATEMENT

The departure point is a Boolean function optimized from the logical point of view [1]. This function will be implemented in a based island FPGA [2].

The Boolean function may be represented as product sum (FIG. 2) or as a circuit (FIG. 3) or as a block diagram (FIG. 4).

Each Boolean function term, subcircuit or block is mapped with a FPGA CLB. It is necessary to take into account that CLBs and FPGA interconnection resources are limited and by that the different subcircuits are placed

$$f = A\cdot\overline{B} + \overline{A}\cdot B$$

FIG 2. Product Sum.

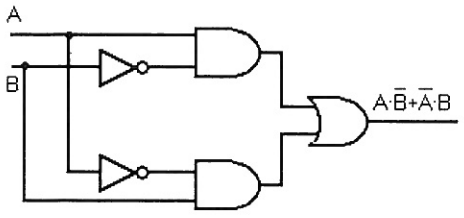

FIG 3. Circuit.

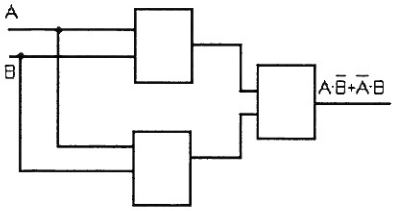

FIG 4. Block Diagram.

in CLBs in order to the FPGA used area will be the lower possible with the aditional consideration that the interconnection channel density (number of occupied connection lines / number of total connection lines) will not be very high.

We consider also the case in which a part of the FPGA will be occupied by other circuits, that is interesting for designing several circuits in a FPGA of great size. In order to tackle this problem we impose an area constraint specifying a forbidden area for CLBs and its corresponding interconnections. This area may have any size and may be placed in any FPGA part. For simplifying the problem we consider only rectangular areas.

This is an optimization problem with many solutions. For finding a cuasi-optimal solution in a reasonable time we have used firstly a Simple GA and after a Parallel GA.

III. DESIGN OF GENETIC ALGORITHMS

3.1 Simple Genetic Algorithm

We start with an initial population of eight chromosomes. Each chromosome represents a problem possible solution. Each solution consists on a CLB list in which the CLB coordinates are known. So the chromosome length depends on the represented circuit size. Each chromosome has also information about the connections between its CLBs so as its associated cost. (See FIG. 5).

In order to each population chromosome will be evaluated we propose to use the following fitness function:

$$f = \alpha \cdot dis + \beta \cdot den$$

Where `dis` represents the dispersion, that is, the general distance between all CLBs. Higher distance higher occupied FPGA area [4]. The X and Y coordinates of each CLB are known in each time. The value of `dis` factor is given by:

$$\sum_{i=0}^{n} \sqrt{\left(cX_i - cX_{i+1}\right)^2 + \left(cY_i - cY_{i+1}\right)^2} \qquad (1)$$

`Dens` is the interconnection channel occupation density.

The α and β coefficients are configurable parameters by the user in order to assign more importance to dispersion or density.

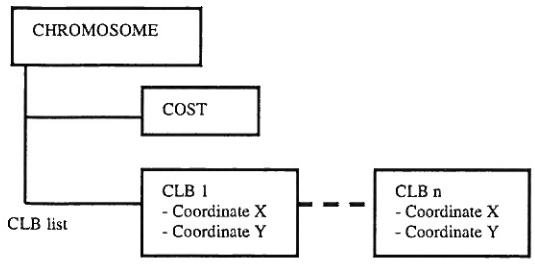

FIG 5. Chromosome representation.

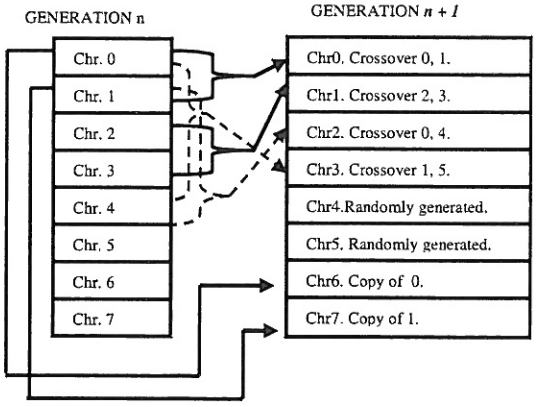

FIG 6. Scheme of the replacement politics.

After to establish the fitness function we explain the selection mechanism for passing from a generation to the next one (the initial population corresponds to the 0 generation).

In each generation, chromosomes are distributed according to the increasing order of its cost. The chromosome population is replaced and selected using the scheme of FIG. 6.

The offsprings 0 and 1 of the new generation are the result of crossing the chromosomes 0, 1 and 2, 3 of the previous generation, respectively.

The offsprings 2 and 3 of the new generation are the result of crossing the chromosomes 0, 4 and 1, 5 of the previous generation, respectively. So, we introduce some diversification crossing no optimal chromosomes.

The offsprings 4 and 5 of the new generation have been randomly generated. They are not consequence of the crossover between two chromosomes. So, more diversification is introduced for exploring the solution space.

The offsprings 6 and 7 of the new generation are an exact copy of chromosomes 0 and 1 of previous generation. This is a mode for preserving the best solutions of the previous generation.

When all chromosomes of the new generation has been found, its costs are computed and then, they are ordered according to the increasing order of its weigths. So a new population is obtained in which its best chromosomes are in the first positions. So the population is ready for creating a new generation.

For implementing the crossover operator we have used the crossing in the middle point of each chromosome, that is, an offspring of two chromosomes will have as CLBs those of the first half of the first chromosome and the second half of the second chromosome.

With the algorithm used until the 2500th generation, mutations are produced with a frecuency of 3%. When this number is passed then the mutation probability increases to the 25%.

The mutation is performed assigning a random coordinate to any CLB of the mutant chromosome (CLB randomly chosen).

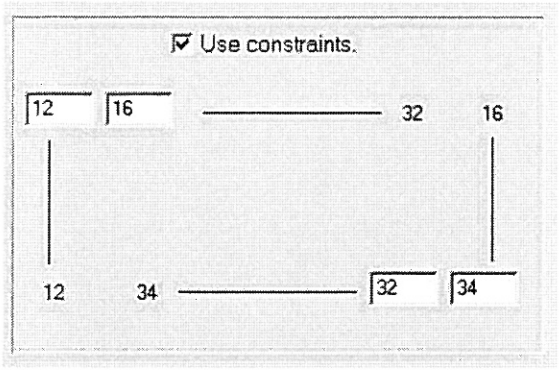

FIG 7. Constraint specifications

3.2 Constraints

For implementing the constraints [3] (see FIG. 7) it is neccesary to indicate the restricted area coordinates.

When the initial population is generated, a checking is made for assuring the CLB coordinates are not in the forbidden area. (FIG. 8) The crossing operation have not any problem with the constraints because chromosomes with valid positions are being crossing. During the offspring selection it is necessary to check the chromosomes 4 and 5 are valid. In the mutation operation it necessary to check that the resultant chromosome will be valid.

In the routing step the constraints must be considered because the connection lines are not able to cross the restricted area. (See FIG.9)

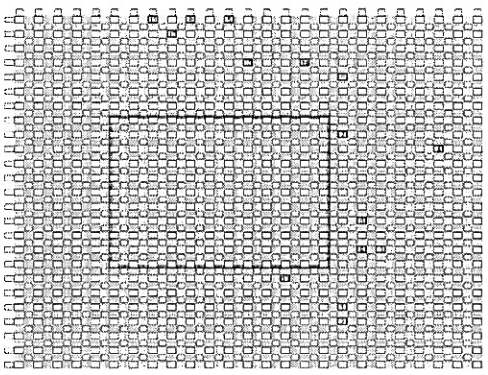

FIG 8. Placement with constraints

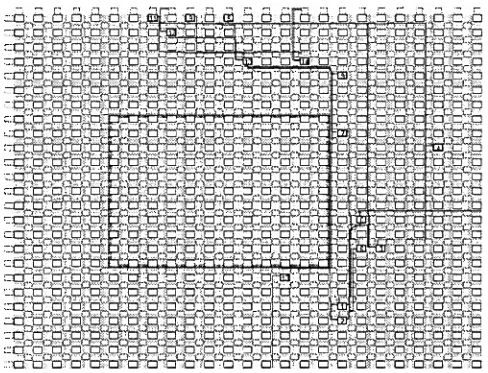

FIG 9. Placement and routing with constraints.

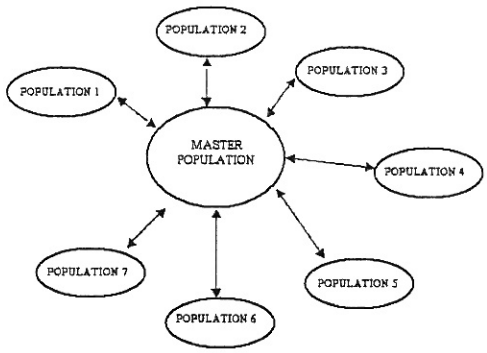

FIG 10. Parallel Algorithm in star.

3.3 Parallel Genetic Algorithm

The parallel execution offers better results than the sequential execution even in a single processor machine [2].

The used parallel genetic algorithm joins in a star [7] eight populations. (FIG. 10) One is the master population, the other ones are the slaves. Each slave population send its best chromosome to the master population. At the same time the master population sends its best chromosome to each of slave populations.

After each interchange of best chromosomes, all populations are ordered according to the cost of each chromosome as it was made in the simple GA.

3.4 Parallel GA Using MPI

In order to parallel the mentioned GA we have used the MPICH [8] tool. This is one of the free issues [9] more spreaded of the MPI Standard [10]. The parallel implementing consists of executing eight processes simultaneously. These processes are based on the simple GA. Each process is a population random initialization so as its evolution. The eight processes are executed simultaneously with MPI. The data interchanged between the processes are chromosomes; that is solutions. The solutions are stored in a non standard data structure (Fig 5) for MPI. So, before the process communication they are coded as a character string. After, they are decoded for its interpretation. A schema of the parallel implementation in MPI is:

```
- Initiate parallel execution
  - Find the process number in execution
  - Find the process ID.
  - If it is a slave process (Id < 8) then send
    the solution to master process.
  - If it is the master process (Id = 8) then
    Receive the solutions of the remain
     processes, store them in an array and show
    the result.
- End parallel execution.
```

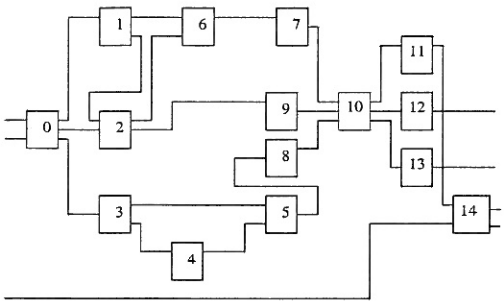

FIG 11. Circuit to implement.

IV.RESULTS

In this point we show the found results using the 15 CLBs circuit represented in FIG. 11. This is a circuit of medium complexity that we have used for demonstrating the Genetic Algorithms works.

With the simple GA we start with a randomly generated population. FIG. 12 shows the best chromosome of the initial population in the FPGA the implementation of this solution in a FPGA. We can see it is a bad solution because the CLBs are spreaded in the FPGA and in the FPGA top there are a great interconnection line density.

FIG. 13 and FIG. 14 represent the best solutions in the 2000th and 5000th generations, respectively. After 2000 generations, for the best chromosome the CLB area is more compacted and the interconnection line density is a bit better than that of the best chromosome of initial population. After 5000 generations the CLB area and interconnection line density of the best chromosomes are similar to best of the 2000th generation.

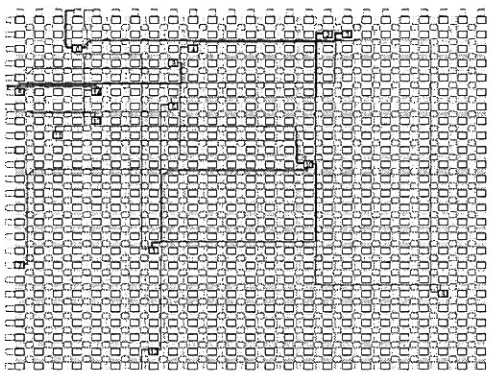

FIG 12. Best chromosome of initial population.

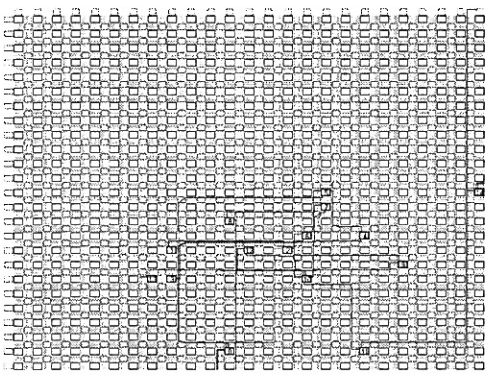

FIG 13. Best chromosome of the generation 2000

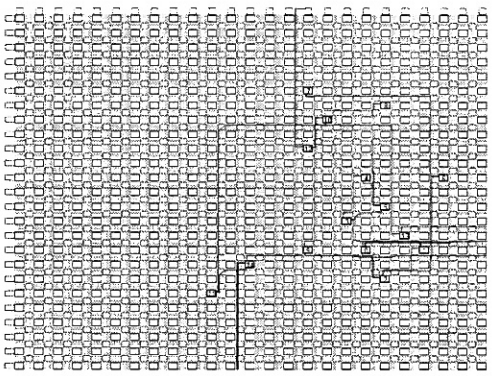

FIG 14. Best chromosome of the generation 5000.

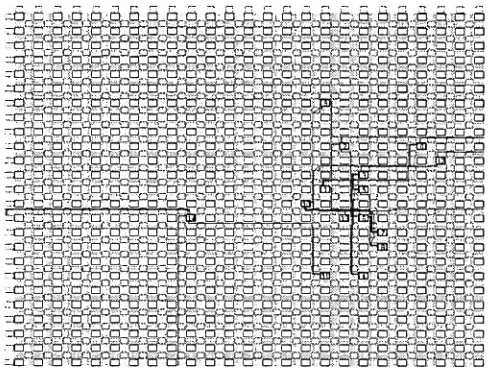

FIG 15. Best chromosome of master population in the 2000 generation.

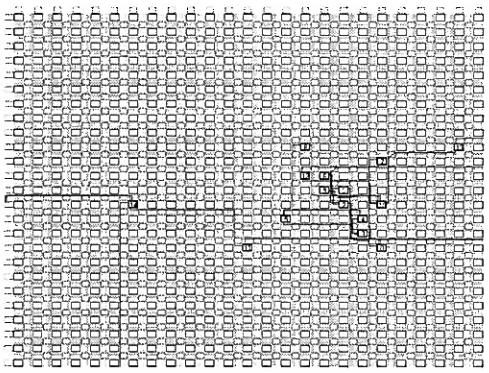

FIG 16. Best chromosome of master population in the 5000 generation.

When we use the parallel GA the best chromosomes found in the 2000th and 5000th generations are shown in FIG. 15 and FIG. 16. The best chromosome of the master population of the 2000th generation is better than the best chromosome found in the 5000th generation obtained with the simple GA. We can conclude that with the parallel GA the CLB area and interconnection line density found for the best chromosomes are better than the found results with the simple GA.

The population evolution when we use the simple GA is shown in FIG. 17 and when we use the parallel GA is shown in FIG. 18.

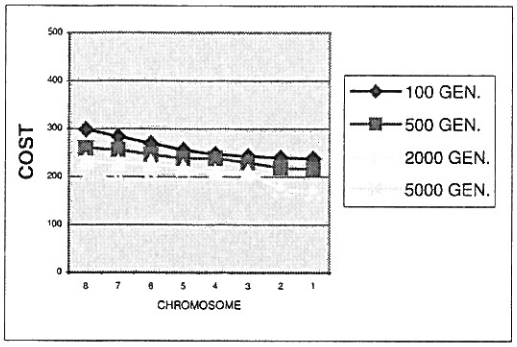

FIG 17. Simple GA evolution.

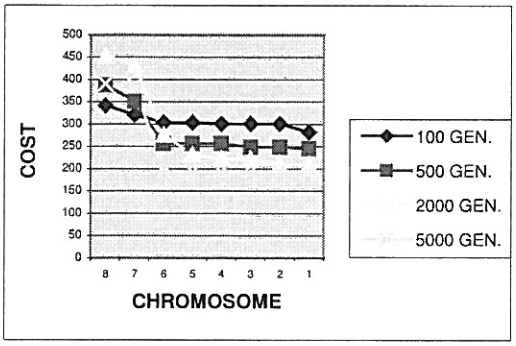

Fig 18. Parallel GA Evolution.

V.CONCLUSIONS

In this work we have used a a simple and a parallel GA (using MPI) for solving the placement and routing problem with constraints of Boolean functions in a FPGA. A graphical interface to show by the computer screen the problem solutions has been also built. In the future we will try of improving the area constraints implementation . Also we will use benchmarks in order to compare the found results with these algorithms.

This work has been partially supported by the TRACER project (TIC-2002-04498.C05-1) of the Spanish Technology and Science Ministery.

V.REFERENCES

[1] Xilinx Programmable Logic Devices. www.xilinx.com

[2] Fernández de Vega, F. Modelos de programación genética distribuida con aplicación a la síntesis lógica de FPGAs.Ph. D. Thesis Universidad de Extremadura. 2000

[3] Hidalgo, J.I, Fernández, F., Lanchares, J., Sánchez, J.M. Multi-FPGA Systems Synthesis by Means of Evolutionary Computation. GECCO. LNCS 2724 pp. 2109-2020. 2003

[4] Dinesh P. Mehta, Naveed A. Sherwani: On the use of flexible, rectilinear blocks to obtain minimum-area floorplans in mixed block and cell designs. pp 82-97 .2000

[5] Koza, J.R. (1992) Evolution of subsumption using genetic programming. In Varela, F.J. and Bourgine, P. editors. 1992

[6] Holland, J.J. Adaptation in natural and artificial systems. Ann Arbor: The University Michigan Press. 1975

[7] Lanchares, J., Desarrollo de metodologías para síntesis y optimización de circuitos lógicos multinivel. Ph. D. Thesis.Universidad Complutense. 1995.

[8] MPICH Implementation. www- unix.mcs.anl.gov/mpi/mpich/

[9] List Of MPI Implementations. http://www-unix.mcs.anl.gov/mpi/implementations.html

[10] MPI Standard Definition. http://www-unix.mcs.anl.gov/mpi/mpi-standard/mpi-report-1.1/mpi-report.htm