Analysis of Genetic Algorithm for synthesis digital systems modeled in finite state machine

Mateus Ferreira da Silva, Warley Gramacho da Silva, Rafael Lima de Carvalho, Edeilson Milhomem da Silva, Tiago da Silva Almeida

Department of Computer Science, Universidade Federal do Tocantins, Palmas/TO, Brazil

***Abstract*—*To achieve smaller digital systems, like microprocessors, controller, etc., it is require design them with a small area and the treats the power dissipation. These issues are important because can prolong the time of use of the equipment and reduce the manufacturing costs. To do so, digital circuits can be modeled as finite state machines with a large amount of states for most practical problems. To achieve a minimum result, you need to optimize a state assignment. Find a solution that meets these characteristics, i.e., find the optimal state assignments is a complex task, because it is an NP-Complete problem. Thus, this research analyzed the Genetic Algorithm to obtain an optimization in the state assignment in a reasonable time. The experiments showed good results, however, the adjusts of the parameters of GA must be investigated to find better results.*

***Keywords*—Digital Systems, Finite State Machine, Genetic Algorithm, Metaheuristic, Synthesis of Circuits.**

I. INTRODUCTION

Nowadays, transistor fabrication technology has reached such a scale of miniaturization that physical limitations can become an obstacle to continuous evolution proposed in Moore's law [1]. The production scale of the transistors is currently very small, about 10 nm, reducing more than this is challenge, because as the transistors get smaller, they are subject to more imperfections during the manufacturing process. Which makes it increasingly difficult maintain the integrity of the electrical signals. Therefore, it is necessary to plan more efficient logical structures, able to reduce the area in silicon. Hence the importance of the work in revisiting the area of Finite State Machine (FSM) optimization with the goal of designing efficient logic circuits.

Minor digital circuits occupy a small area and additionally have low power dissipation. These characteristics can be obtained with the optimization of the state assignment in FSM.

FSM is a sequential circuit design technology widely adopted at the system level [2],FSMs are composed of inputs, outputs and states. A state is a record of the main information of a system, each state must receive a specific binary code (which is named assignment state). According to [3], FSM is a generic sequential circuit consisting of a section made of combinational logic and a section of memory (usually flip-flops).

An FSM called X can be defined as the mathematical model $X = (I, O, S, \delta, \omega)$, where I, O and **S** are finite and non-empty sets of inputs, outputs and states, respectively.$\delta: I \times S \rightarrow S$is the next state function, $\omega: I \times S \rightarrow O$ is the output function for Mealy Model and $\omega: S \rightarrow O$ is the output function for Moore Model.

Note that minimal FSMs are desirable because they consume less resources, that is, transistors, which in turn will consume less power, occupy less area on the viable surface of the silicon chip, dissipate less heat and can work in higher frequencies. To optimize the combinational circuit of FSMs a very important factor is the assignment of states. Optimizing the state assignment is a complex task, since all possible arrangements must be tested to obtain the optimal assignment, the total number of possible single assignment for an FSM is given by.

$$A(n, b) = \frac{(2^b - 1)!}{(b!\,(2^b - n)!)} \tag{1}$$

where$n$ is the number of states and b is the number of bits needed to represent each state, therefore, it is an NP-Complete problem.

The general objective of this work is to evaluate the results of the application of metaheuristic (Genetic Algorithm - GA) algorithms in the search for an optimization in the assignment of states in polynomial time.

II. RELATED WORK

Algorithms specializing in heuristic searches are becoming popular for solving complex optimization problems, which are in the class of NP-Complete problems, and many researchers have used these algorithms to try to solve the problem of assigning states. [4], used a GA that adopts the Pareto Ranking scheme, to find the state assignment that minimizes both area and power dissipation. The applied multi-objective GA allows the hardware designer to prioritize power dissipation or logical complexity that can reduce the circuit area, or select a solution that reduces both, but does not guarantee the absolute minimum in any of them.

Another research that applies a genetic algorithm seeking to optimize state allocation is performed by [5], to reduce power dissipation and circuit area. In the same way [6], presents an approach based on the GA for the synthesis of an FSM, with the objective of reducing power dissipation. [7]uses an Immunological Algorithm. The proposed algorithm combines the immunological operator with a local search. In addition, this algorithm has modified mutation and crossover operators. Local search is adopted to avoid premature convergence of the algorithm and to maintain population diversity.

The immune algorithm takes into account two objective functions, the first is related to the energy dissipation of the sequential circuit that is proportional to the switching activity between the states of the circuit, and the second objective function is the minimization of the number of terms because the sequential circuit area is proportional to the number of terms. This minimization is obtained by an external tool, called ESPRESSO, this tool generates the minimized circuit, and the number of terms provided by it represents the size of the circuit area.

[8]has its research focused on finding the minimum number of state variables in an FSM, for which a GA was used. However, the FSM presented by it has a different characteristic because it uses the output signal as a state variable.

In another recent work, [9] proposed an Evolutionary Strategy (ES), based on a state assignment model called ESSA. According to the study, this method has selection and mutation operators specifically designed based on analyzes of the state assignment problem. Their conclusions show that this approach achieves a significant reduction of area.

III. METHODOLOGY

In general, a basic AG consists of the following steps: coding is the way the solution is represented, the initial population is randomly generated, the objective function determines the cost of each individual (solution), selection of individuals that will participate in the production of the later generation and the mutation that is performed in order to avoid that the algorithm is stuck in a local minimum.

The implemented AG has an array of size $2 \times n + 1$, with n being the number of states of the FSM. In the first line each position from 0 to n corresponds to a read state of the file, position $n + 1$ refers to the cost of this individual. The second line contains the values assigned to each state respectively, and the cost value, these values are in the decimal basis, as shown in Fig. 1.

A	B	C	D	E	Cost
2	5	7	4	1	37

Fig. 1: Possible assignment of states to FSM of five states.

The coding was performed using decimal numbers, because in this way the treatment of the infeasible of the problem of assignment of states (PAS) becomes simpler, infeasibility occurs when two or more states are allocated with equal values.

In this step, the fitness value of each individual of the population is calculated through a given function. This is the most important component of any GA. It is through this function that you measure how close an individual is to the desired solution or how good this solution is.

It is essential that this function be very representative and differs in the correct proportion the bad solutions of the good ones. If there is little accuracy in the evaluation, a great solution can be set aside during the execution of the algorithm, in addition to spending more time exploring less promising solutions.

The objective function of the algorithm implemented takes into account the cost of each individual. It seeks to better assignment states to an FSM, in order to minimize area size and energy dissipation, so the best individuals are those who have the lowest cost.

The logical module of the SymPy (from Python language[1]) library is used to calculate the cost of each individual. After the generation of the individual, we obtain the list of mintermsand don'tcare states, which are passed as a parameter to the function SOP (Sum of Product) form, of the mentioned module. This function returns the minimum Boolean expression, as previously quoted to obtain the cost value, the weight "1" is assigned for each of the terms and literals present in the expression.

The generation of the initial population is performed randomly, since the quality of the solutions presented must be independent of the initial population.

[1]https://www.sympy.org/pt/index.html

The number of individuals forming the population is obtained by

$$I = 3 \times n \times v \qquad (2)$$

where, I is number of individuals in population, $n$number of states, and $v$the minimum number of state variables necessary to assignment the states$n$, and v the smallest integer greater than or equal to $\log_2 n$.
The state assignment problem has a constraint, all states must have distinct binary representations, i.e., two or more states cannot have the same assignment.
Due to randomness in the generation of the initial population, infeasible individuals may arise. To avoid this problem a resource has been used, so that individuals do not violate the restriction.
In short, this feature consists of assembling a map that contains all the possible values that state variables can assume, to determine the size of the map the equation is used,

$$t = 2^v \qquad (3)$$

with each assignment of a value to a state of the individual, this value is taken off the map to prevent another state of the same individual from assuming it.
A demonstration of the operation of the strategy used to generate the initial population is shown in Fig. 2.

Map	0	1	2	3	4	5	6	7

	A	B	C	D	E	Cost
Individual						

Fig. 2: Representation of the strategy of generation of the population, used in the proposed algorithm.

Initially the map is complete, and no value has been assignment to a particular stateindividual. A map position is chosen randomly and the value of this position is assignment to state A, so the cell is excluded from the map to prevent its value from being assigned to another state. This procedure is observed in Fig.3.

Map	0	1	2	3	4	5	6	7

	A	B	C	D	E	Cost
Individual	2					

Fig. 3: Example of an iteration in the production of an individual, value 2 is removed from the map and assigned to the individual.

The process is repeated until all states receive a value, a new map is generated for each individual.
The cost is calculated later using SymPy's SOPform function.
In this work we used the roulette selection method, this method works as follows, each individual of the population is represented in roulette in proportion to his fitness index. Thus, for individuals with high fitness a larger portion of the roulette wheel is given, whereas for individuals of lower fitness, a relatively smaller portion is given.
As the interest is to find the best assignment of states, the individuals with greater aptitude and which consequently will occupy a greater portion of the roulette, are the ones that have the lowest costs.
Through crossover, new individuals are created by mixing features of two individuals, “parents”. This mixture is made by trying to imitate (at a high level of abstraction) the reproduction of genes in cells. The result of this operation is an individual who potentially combines the best characteristics of the individuals used as a basis.
The crossover implemented in this algorithm was at one point. After the execution of the crossover, the generated individuals can be infeasible, that is, having two or more states with the same value, to solve this to the same procedure applied in the generation of the initial population, and one of the valuesrepeated by some value contained in the map is changed. The following example illustrates how this process works.
The Fig. 4, presents a demonstration of the execution of the crossover operator proposed in this work.

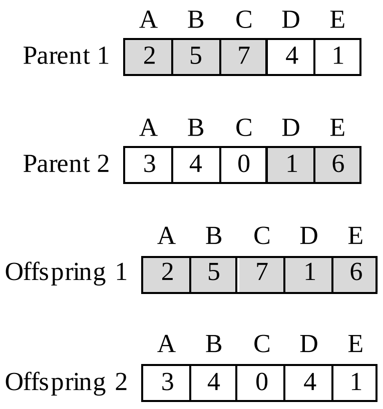

Fig. 4: Example of a crossover operator application.

Parent 1 and Parent 2 represent two possible allocations for the FSM, and the gray color indicates the crossover point.
Note that in Offspring 2 states B and D were assigned with the same value, which makes the individual

infeasible. To solve this problem we use the map, which contains the possible values that have not yet been assigned, a random value is chosen on the map and allocated to one of the states whose values are the same. This procedure is exemplified in Fig. 5.

Map				
	2	5	6	7

	A	B	C	D	E
Offspring 2	3	4	0	7	1

Fig. 5: Resolution of infeasibility in an individual.

As seen in Fig. 5 the decimal representation of the D state has been changed from 4 to 7, thereby inserting a random information, and a new infeasibility is avoided.

The individual with the greatest aptitude of each generation, that is, the one with the lowest cost is stored. The solution is the most fit individual after *x* iterations.

The mutation operator is important to introduce and maintain the genetic diversity of the population, and prevent the algorithm from being stuck in a local optimal. The mutation function implemented in this algorithm randomly selects a gene from the chromosome and changes its value by some available value on the map, also randomly selected. If the map is empty, another gene from the chromosome itself is chosen randomly, and the values of these two genes are exchanged. This strategy was adopted to avoid the problem of infeasibility mentioned previously.

IV. EXPERIMENTAL RESULTS

The proposed GA has been implemented in Python and applied to MCNC benchmark circuits [10]. Test results are given in Fig 6.

Benchmarks	In / Out / Num. States	GA
		Ter / Lit
beecount	3/4/7	20/66
dk14	3/5/7	43/130
ex3	2/2/10	26/81
mc	3/5/4	16/58
tav	4/4/4	21/62
train11	2/1/11	28/100

Fig. 6: Results obtained during the experiment

SOPform SymPy library function is used to minimize the expression of each state assignment. From the expression it is possible to extract the number of terms and literals.

The second column in Fig 6 denotes the number of inputs bits, number of output bits and number of states for the given benchmark in the first column. The set of results produced by the GA is giving in column 3. "ter" denotes to number of minterms and "lit" the number of literals.

The results were obtained using population size defined by Equation (2), mutation rate=0.02, crossover rate = 0.4, selection rate = 0.6 (percentage of individual will be select to compose the new generation) and 300 for the number of generations. All rates were used in total values and the specific individual and gene are selected randomly.

The graph shown in Fig. 7 shows the curve of the behavior of the algorithm during the 300 iterations, we notice the evolution in search of lower costs during the execution. Each curve refers to a specific benchmark as indicated in the legend.

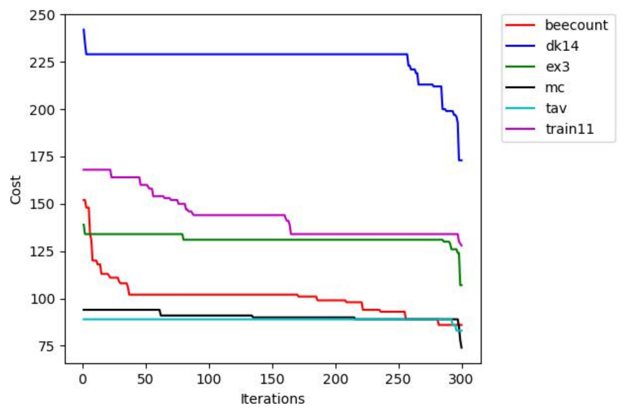

Fig.7: Costs of each benchmark during execution of GA proposed.

In the experiments was noticed the decrease of costs in final generations of the GA. Perhaps, it will be more suitable adjust the criterion stop to increase the results. The same way, FSMs with more states could require more generations to achieve better results.

The graph of Fig. 8, shows the normal distribution of the costs of the best individual of each generation. The curves are dislocated because of the different values of costs.

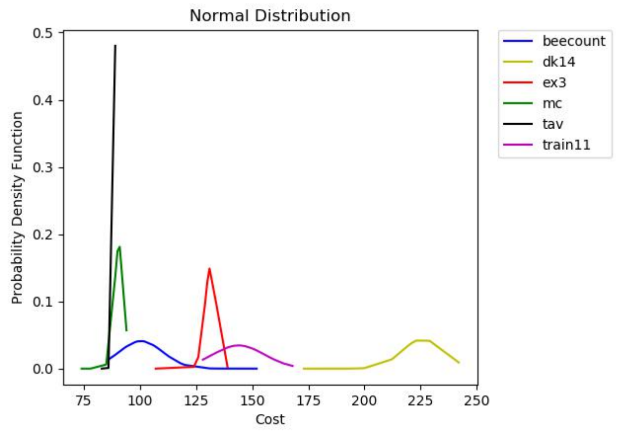

Fig.8: Normal distribution of results of each benchmark analyzed.

This graph was built to show the probability density of the solution to be a good solution. To plot the graph was used,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (4)$$

where x is value of cost, μ is the mean of distribution and σ is the standard deviation of distribution [11].

From Fig. 8, benchmarks `train11`, `beecount` and `dk14` curve is very low, meaning the result can be improved. So, this fact corroborated with hypothesis of use more generations to achieve better results. The best result was with `tav` benchmark.

Another hypothesis is to use different parameters in the GA, other rates of crossover, mutation and selection.

V. CONCLUSION

In this paper was analyzed an implementation of GA in Python to solve the assignment state problem to FSM. In the experiments, the GA was able to find a good solution to the benchmarks tested. However, it is necessary to study a better criterion stop to GA.

Another issue is that performance of the GA. It was observed the long time to reach the solution for FSM with large number of states. Thus, the complexity of algorithm is exponential.

The great problem for such time is the evaluation of fitness function, i.e., the FSM cost. In our experiments was used an external library and there is not control about its implementation, which is probably not heuristic. This means all combination of expression is tested for each bit, in each assignment, in each chromosome, in each generation.

One possible solution is divided the in many thread and in others unit functional (CPU) to divide the overload e achieve more performance. Or to try implement a better cost function with another criterion or another algorithm.

REFERENCES

[1] Moore, G. E (1965). Cramming more components onto integrated circuits, reprinted fromelectronics, vol.38, n. 8, pp.114 _. IEEE Solid-State Circuits Society Newsletter, v. 11, n. 5, pp. 33-35, Sept 2006.

[2] Gerstlauer, A. et al (2009). Electronic system-level synthesis methodologies. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v. 28, n. 10, pp. 1517-1530.

[3] Floyd, T. L.(2007).Sistemas Digitais: fundamentos e aplicações. Bookman Editor.

[4] Jassani, B. A. A., Urquhart, N. and Almaini, A. E. A. (2011). Stateassignment for sequentialcircuitsusingmulti-objectivegeneticalgorithm. IET Computers Digital Techniques.

[5] Xia,Y., Almaini, A. and Wu, X. (2003). Power optimization of finite state machine based on genetic algorithm, Journal of Electronics, vol. 20, n. 3, pp. 194–201.

[6] Chattopadhyay,S. (2005). Area conscious state assignment with flip-flop and output polarity selection for finite state machine synthesis: a genetic algorithm approach. Comput. J., vol. 48, pp. 443–450.

[7] Zhou, C., Li, C., He, A. and Yang, Y. (2016). An efficient solution of circuit state assignment with immune algorithm, 2016 6th International Conference on Electronics Information and Emergency Communication (ICEIEC),pp. 38–41.

[8] Curtinhas, T. et al. (2017).Stateassignmentofdirect output synchronous fsms using geneticalgorithm. In: 2017 IEEE XXIV International Conference on Electronics, ElectricalEngineering and Computing (INTERCON), pp. 1-4.

[9] Tao, Y., Zhang, L. and Zhang, Y. (2015). An evolutionary strategy based state assignment for area-minimization finite state machines, 2015 IEEE Symposium Series on Computational Intelligence, pp. 1491–1498.

[10] Yang, S (1988). LogicSynthesisandOptimization Benchmarks. https://ddd. t.cvut.cz/prj/Benchmarks/ index.php?page=download.

[11] Martins, G. A. and Domingues, O (2017). Estatísticageral e aplicada. Gen Editor, ed. 6.