Automatic Design of Synchronous Sequential

Logic Circuit Using Genetic Algorithm

Hiba B. Alwan

Al- Mansour University College

Abstract:

The design of synchronous sequential logic circuits is of great interest to a number of researchers and designers, and there are many different researches dealing with this subject.

This paper deals with the design of synchronous sequential logic circuits using Genetic Algorithms (GA). GAs is search algorithms based on the mechanism of natural selection and natural genetics and use natural genetics operators such as crossover and mutation.

The design of synchronous sequential logic circuits starts from a set of specification or a list of Boolean functions from which a logic design can be obtained. The first step in the design of sequential circuit is to obtain state table or an equivalence representation, such as state diagram.

In this work GAs are used to design synchronous sequential logic circuit with minimum number of gates. A population of candidates is maintained, and goes through a series of generations. For each new generation, some of the existing candidates survive, while others are created by types of reproduction and mutation from a set of parents.

The input to this work is the user requirement which is represented as circuit specification which we need to design. GA takes this specification and applies genetic operation to it to construct state assignment.

The circuit is represented in such a way that the genetic operations can be carried out. The architecture of the circuits design has three stages. In stage one; the circuit specification is represented using State Transition Table (STT). In stage two; genetic algorithm uses STT to generate state assignment to assign binary code for each state. The third stage gives the minimum circuit using tabulation method. Different counters circuits have been selected and applied to show the result of GA.

1. Introduction

Design is the process of coming up with a solution to a problem. To do this, we not only have to understand precisely what the problem is, but also the constraints our solution must meet [1].

We can also define the design as the process of translating an idea into a product that can be manufactured [2].

Logic circuits for digital systems may be combinational or sequential. A combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. Sequential circuits on the other hand employ memory elements in addition to logic gates [3].

The design of a synchronous sequential circuit starts from a set of specifications in a logic diagram or a list of Boolean functions from which a logic diagram can be obtained. In contrast to a combinational logic, which is fully specified by a truth table, a sequential circuit requires a state table for its specification. The first step in the design of sequential circuits is to obtain a state table or an equivalence representation, such as a state diagram. The complexity of logic circuit can be defined as a function of the number of gates in the circuit [2].

Genetic search has the ability to locate very good approximate solutions in extremely large search spaces with a reasonable amount of computational efforts. The basic difference between genetic search and traditional method is given in Table (1) [4].

Table (1) Differences between genetic search and traditional search

	Genetic search	Traditional search
Work with	Coding of a parameter set	Parameters directly
Search	A population of points	A single point
Use information	Payoff (objective function)	Payoff + derivatives
Rules	Probabilistic in nature, random population with random crossover and mutation	Fully deterministic

The basic iterative genetic search operators work in three fundamental stages:

1. Produce of a random initial population with M members (or candidates), and determination for each of them the fitness (valuation), i.e., a measurement of its performance.

2. Generate by reproduction of a new population of identical size favoring the members of highest fitness.

3. Apply of crossover and mutation operators to some members to introduce some diversity in the population, favoring thus a good exploration of the often-large space of potential solutions.

GA is used in this work to design the synchronous sequential logic circuit; the stages of the design process are given in Figure (1).
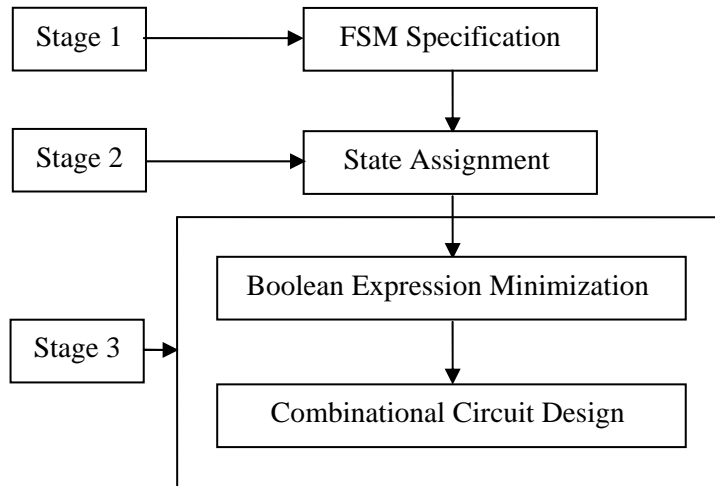
Figure (1) the stages of the design synchronous sequential logic circuits

Stage1: Represent the target circuit specification using symbolic state transition table.

Stage2: The genetic algorithm uses this state transition table (STT) to generate state assignment to assign binary code for each state.

Stage3: The obtained Boolean expression would be minimized using tabulation method and the circuits would be created at the gates-level by using function set of logic device such as AND, OR, and NOT.

2. What is Genetic Algorithm

GAs is search algorithms based on the mechanics of natural selection and natural genetics [5]. From a mechanistic view, GA is a variation in the generated and test method. GA derives their name from the fact that they are loosely based on models of genetic change in a population of individuals. These models consist of three basic elements:

1. **A Darwinian notion of "fitness" which governs the extent to which an individual can influence future generations.**
2. **A "mating operator" which produces offspring for the next generation.**
3. **A "genetic operators" which determine the genetic makeup of offspring from the genetic material of the parents.**

The key components of GAs are [6]:

1. Genotype: **This is an encoding which describes the Phenotype.**
2. Phenotype: **It is the objects that "live" and interact with the environment and defined by GA.**
3. Fitness Function: **A performance measure of the Phenotype induced by a Genotype.**
4. Selection Criteria: **These are the criteria used to select individuals for reproduction and creation of new individual. It is common to select pairs of parents on the basis of fitness so that the probability of an individual being chosen for reproduction is proportional to its fitness value.**
5. Breeding Operators: **The breeding operators take individuals chosen by selection criteria and use these as "parents" to create new offspring individuals.**
6. Population Stabilization **The offspring created by the breeding operators are added to the population, increasing its size. To keep the population size stable, some members must be added. This is carried out on the basis of fitness by choosing members at random from a distribution that favors those with lowest fitness values.**

3. Logic Circuit

There are two types of logic circuits: combinational circuit and sequential circuit. A combinational logic circuit consists of logic gates whose outputs at any time are determined from the present combination of inputs. A combinational circuit performs an operation that can be specified logically by a set of Boolean functions [7].

Therefore, a combinational logic circuit is one in which two or more gates are connected together to combine several Boolean inputs. A combinational circuit also can be described by (m) Boolean function one for each output variable. Each output function is expressed in terms of the input variables [7]. The block diagram of combinational logic circuit is shown below:
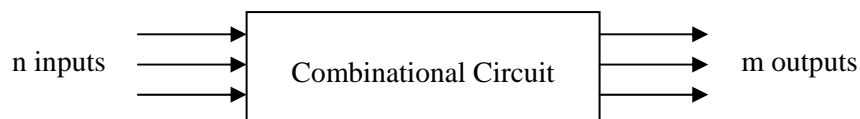
n inputs → | Combinational Circuit | → m outputs

**Figure (2) Block diagram of combinational logic circuit**

Sequential logic circuits is a combinational logic circuit included a storage element. It consists of a combinational circuit to which storage elements are connected to form a feedback path. The storage elements are device capable of storing binary information.

The most basic sequential circuit elements are flip-flops. A flip-flop circuit has two outputs, one for the normal value and other for the complement value of the bit stored in it [8]. There are four main types of latch and flip-flop, these types are SR (Set-Reset), D (Data), JK and T (Toggle). There are two main types of sequential logic circuits and their classification depends on the timing of their signals, and these types are Synchronous Sequential Logic Circuit and Asynchronous Sequential Logic Circuit [7].

A synchronous sequential circuit is sometimes referred to as clocked system. A clock is just a signal that alternates (over time) between 0 and 1 at a regular rate [9]. Synchronous circuit is the general term for collection of logic gates and flip-flops that are controlled by a common clock. A synchronous circuit has all of its flip-flops transition at the same time so that they settle at the same time, with resultant improvement in performance. Another benefit of synchronous logic is easier circuit analysis, because all flip-flops change at the same time [10]. The block diagram of a synchronous clocked sequential circuit is shown in Figure (3).
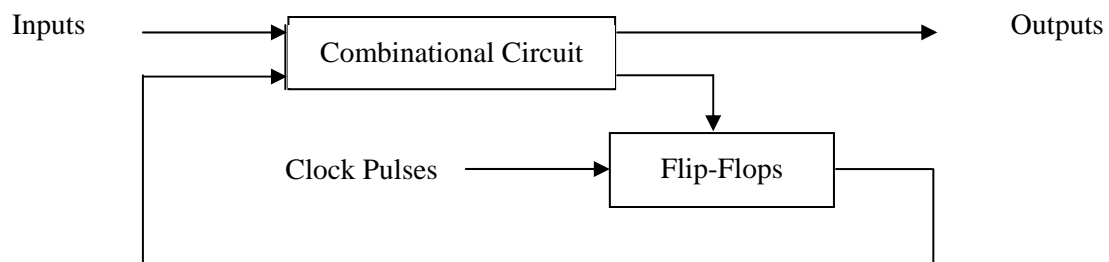
a. Block diagram
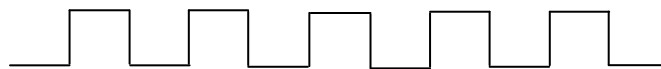
b. Timing diagram of clock pulse

Figure (3) Synchronous clocked sequential circuits

A) Synchronous Counters [8]

In synchronous counters, all the flip-flops change their state simultaneously, the operation of each stage being initiated by the clock. This is done by connecting the input clock to each flip-flop of the counter; hence, all flip-flops are clocked simultaneously. These counters are classified according to:

1. Sequence of states.
2. Number of states or
3. Number of flip-flops (stages) used in the counter.

Here, all flip-flops are triggered simultaneously. Hence, these counters are called "parallel counters".

B) Design of Synchronous Counters

Designing a synchronous counter requires the addition of logic to calculate the new count value based on the current count value. Synchronous circuits consist of state-full elements (flip-flops), with combinatorial logic providing feedback to generate the next state based on the current state. Combinatorial logic is the term used to describe logic gates that have no state on their own. Inputs flow directly through combinatorial logic to outputs and must be captured by flip-flops to preserve their state [10].

In synchronous counters, all flip-flops are clocked at the same time and input of each flip-flop in the counter must be at the correct level to ensure that each flip-flop goes to the correct state. The following steps can be followed to design any desired sequence counter [11].

1. Determine the number of flip-flops needed, i.e., desired number of bits and the desired counting.
2. Choose the type of flip-flop to be used, SR, JK, D or T etc.
3. Draw the state transitions showing all possible states, including those that are not a part of the desired counting sequence.

4. Use state transitions to obtain the state table that lists all present states and next states.

5. From the state table, derive the circuit excitation table.

6. Use K-map or any other simplification method to derive the circuit as flip-flop input functions i.e., design the logic circuits to generate the levels required at each flip-flop input.

7. Draw the logic diagram, i.e., implement the final expression.

4. System Architecture

The proposed system named "Automatic Design of Sequential Circuit (ADSC)" is composed of a number of modules, which are:

1. User specification module.

2. Initial population module.

3. Population alternation module.

4. Minimization module.

The user specification module is used for specifying the problem requirements which are given as the target circuit. The setup sub-module is used for setting a number of genetic parameters according to the user needs. These are probability of mutation (P_m), probability of crossover (P_c), population size (M), number of generation (Gen), genes on chromosome, and fitness. After that the initial population module is used for generating initial population. Alternation module is then used which is considered as the heart of the system.

Fitness calculation, which is used for computing fitness value of each individual, if the result, is not satisfied the current population is modified using selection operator to select a superior individual for the new mating pool. Then genetic operators are employed to identify new solutions based on the individual in the new mating pool. The process continues for the next generation until the stopping criteria are reached.

After getting the satisfactory solution and formed the State Transition Table of the obtaining circuit, the STT would be minimized using tabulation method. The goal of the minimization is to get the minimum circuit which means less cost and complexity.

a.   Initializing the Population

The first important aspect of GA is initializing the population. The initial population is created randomly. The selection of the population size is the most important choice. The population size must be chosen with the complexity of the problem in mind. In general, the larger the population, the better, but the improvement due to a large population may not be proportional to the increased computational resources required. Selection of the population size is an external decision that must be made by the user.

The method followed to create the population is reading the computer's time and generating the random number. This method is illustrated in following equation:

(hour × 3600) + (minutes × 60) + (second × 60) + millisecond

Finally take the mod population size operator to the summing number. We take the mod operator to the final number because the final number becomes too large so we need to normalize it by taking mod operator to it. This method is useful because we can get a diversity population because there are not two instances of time are similar.

After this, the generated chromosome would be mapped to a number of states. The length of the chromosome is equal to the number of the states used for the sequential machine. This mapping can be summarized as follows:

1. **Assume that the random numbers generated are 2, 4, 5, 2, 4, 2**

2. **Compute number of states as follows: n + the maximum number generated randomly.**

If we assume that the max. number generated randomly is 5 and the n is 6, then the number of states is equal to 11, i.e., (0,1,2,3,4,5,6,7,8,9,10); the list starts with zero and contains all possible assignments for states using minimum length code, i.e. $b=[\log_2 n]$ bits to encode the set of n states. The procedure is interpreted as follows:

1. **The first random number is 2, take the second number from the possible state list 2 as first code for the initial state, and this state   will be deleted from the list of state and the other states in the list will be shifted to the left.**

2. **The next random number is 4, take the fourth number from the possible state list and remove it from the list by shifting the other states in the list to the left.**

3. **The next random number is 5, take the fifth number from the possible state list and remove it from the list by shifting the other states in the list to the left.**

The procedure continues in the same way for the remaining numbers in the list. It can be seen in Figure (4) the random number (2, 4, 5, 2, 4, 2) would map the states (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10) to the assignment (2, 5, 7, 3, 8, 4) respectively. This method is applied to generate randomly the initial generation.
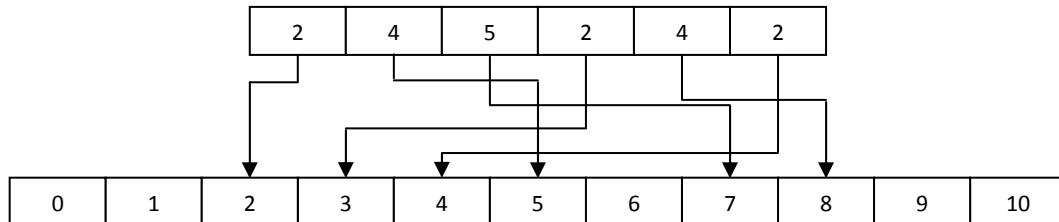
Figure (4) Chromosome mapping

b.   Evaluating the Population (Fitness Calculation)

Fitness calculation is used for computing the fitness values. In each generation fitness value is assigned to each individual in the population. This value is used to control the application of the operations that modify the population.

$$Fitness\ Function = \frac{100}{No.\ of\ gene\ in\ each\ chromosome}$$

In this way we give each gene a specific percentage so if the gene satisfies what we want (which type of circuit you want) this percentage is accumulated with other percentages of genes to the same chromosome that satisfies that we want. If the gene does not satisfy what we want this percentage would be subtracted from the total percentage 100%. This fitness is the percentage that reaches a circuit in the real world.

c.   **Population Alternation**

To decide which individual in the population is to survive and possibly reproduce offspring in the next generation, selection algorithm is used. Copying individuals according to their fitness values means that individuals with a higher value have a higher probability of contributing two offspring in the next generation. In this work Roulette Wheel selection method is chosen.

At each generation a number of genetic operations are applied to the current population to generate a new population.

There are several primary and optional secondary operators in GA; each operator has a role to evaluate the solution to the problem.

Crossover is the creation of two new offspring from the parents selected in the pairing process for testing new offspring rather than testing the same offspring over again in successive generations.

These types of crossover are:

1. **Single-point crossover at random point on both parent.**

   Parent 1   Child 1

   Parent 2   Child 2

2. **Two-point crossover at random point on both parent.**

Parent 1 ☐☐☐   **Child 1** ☐▦☐

Parent 2 ▦▦▦   **Child 2** ▦☐▦

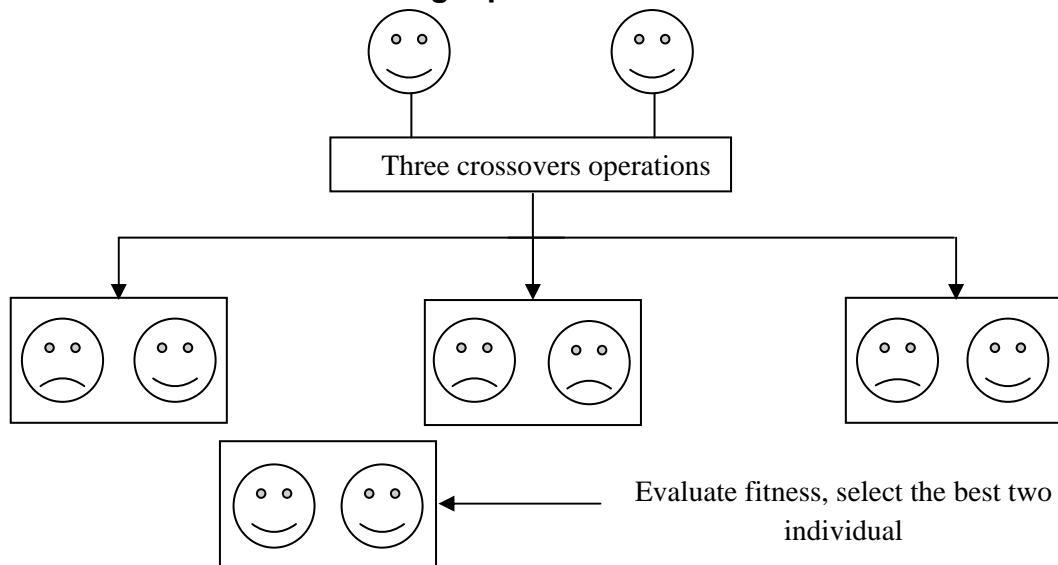3. **Brood crossover with Single-point crossover.**

Figure (5) Brood crossover with B=3

Mutation operator is fundamental because of its ability to overcome some loss of potentially useful information. Mutation randomly alters genes according to some small probability.

Two types of mutation are used with this system.

1. **One bit mutation.**

2. **Two bits mutation.**

d.   Search Convergence

The convergence or termination criteria finally bring the search to a halt. There is always a limit to the maximum number of generations. Reaching this limit would indicate that the termination criteria are too stringent or that there is no satisfactory solution. In this work, the termination considering the search to have satisfactory convergence is one of the following criteria:

1. When the fitness value of an individual in the population is equal to 100%.
2. When the maximum number of generation is reached.

After satisfaction of the termination criterion, the single best individual in the population produced during the run (the best - so - far individual) is designed as the result of the run and the STT of the target circuit is formed, and then it would be minimized using circuit drawer module. In this work, T flip-flop is used. The excitation table for T. F. F. is given in Table (2).

Table (2) T.F.F. Excitation table

Q_A	Q_{A+1}	T. F. F.
0	0	0
0	1	1
1	0	1
1	1	0

 e. Minimization

After we get the fittest chromosome, we find the state assignment to it. The tabulation method would be used to minimize the Boolean expression. This method is more efficient than K-map because it can deal in simple way with many variables unlike K-map, which becomes very difficult to deal with five or more variables. After doing this minimization we gain the Boolean expression of the circuit.

5. System Requirements

The requirements of the system are given in terms of the following cases:

1. Data requirement: The information needed by this system is the target circuit that the user wants. Thus, the data required for this proposed system is very low.

2. Time requirement: If Gen is the number of generation, M is the initial population size, and n length of solution vector (chromosome). Then the time is required and its complexity is given by O (Gen × M × n), such that GA consumes amount of processor time evaluating a number of candidate chromosome.

6. Experimental Results

Different types of counters would be used in our experiments. These counters are: Random Counter, Odd Counter, Even Counter, Random Odd Counter, Random Even Counter, Binary Up counter, Binary Down Counter, and finally BCD Counter.

Some information is needed by the ADSC system. Table (3) shows the parameters used through the experiments. A user can modify this information by running the setup module.

The results of applying ADSC are examined in terms of:

1. Experimental results on choosing genetic parameters.
2. Experimental results on choosing genetic operators.

Table (3) ADSC information

M	25	100
Gen.	50	
P_c	0.5	0.9
P_m	0.1	0.05
Selection method	Roulette Wheel Selection	
Best of run	Fitness = 1.0	

The population size depends on the problem. From the results here, we see that the random; random odd and random even counters do not need to a large population and a population size equal to 25 is enough to find correct solution. This is in contrast with other counters circuit which needs a population size equal to 100 to find a correct solution.

6.1. <u>Example: Odd Counter</u>

In this example we want to design a counter that counts odd number. In this case, we must get a chromosome which contains odd alleles and the difference between these alleles must be stable and equal to -2.
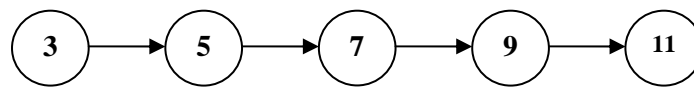
$$3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 11$$

Figure (6) Odd Counter

6.2. <u>The Solution in ADSC</u>

The first part in using ADSC is to generate a population of chromosomes. From the information in Figure (6) we can see that the counter which is considered to be modeled consists of five states. Table (4) illustrates ADSC operations and the individual in the table with an appropriate fitness represents the desired solution. Table (5) illustrates the S.T.T.

Table (4) ADSC operation on odd counter example

Action	Individuals	Fitness
Current Population	1 3 5 9 14	0.5
	3 5 9 12 6	0.25
	7 11 4 0 13	0.0
Selection (1, 2)		
Crossover	1 3 5 12 6	0.5
	3 5 9 9 14	0.25
Mutation	1 3 5 10 6	0.5
	3 5 7 9 14	0.75
New Population	3 5 7 9 14	0.75
	1 3 5 10 6	0.5
	1 3 5 9 14	0.5
Selection (1, 3)		
Crossover	1 5 7 9 14	0.5
	3 3 5 9 14	0.25
Mutation	1 5 2 9 14	0.0
	1 3 5 9 14	0.5
New Population	3 5 7 9 14	0.5
	1 3 5 9 14	0.5
	3 5 7 9 14	0.75
	Selection (2, 3)	
Crossover	1 3 5 9 14	0.5
	3 5 7 9 14	0.75
Mutation	4 3 5 9 14	0.25
	3 5 7 9 11	1.0

Table (5) S.T.T. for chromosome (3 5 7 9 11)

Present State	Next State	T.F.F.
Q_A Q_B Q_C Q_D	Q_{A+1} Q_{B+1} Q_{C+1} Q_{D+1}	T_A T_B T_C T_D
0 0 1 1	0 1 0 1	0 1 1 0
0 1 0 1	0 1 1 1	0 0 1 0
0 1 1 1	1 0 0 1	1 1 1 0
1 0 0 1	1 1 1 1	0 1 1 0
1 0 1 1	0 0 1 1	1 0 0 0

$T_A = \sum (7, 11)$

$T_A = \overline{Q}_A Q_B Q_C Q_D + Q_A \overline{Q}_B Q_C Q_D$

$T_B = \sum (3, 7, 9)$

$T_B = \overline{Q}_A Q_C Q_D + Q_A \overline{Q}_B \overline{Q}_C Q_D$

$T_C = \sum (3, 5, 7, 9)$

$T_C = \overline{Q}_A Q_C Q_D + \overline{Q}_A Q_B Q_D + Q_A \overline{Q}_B Q_C Q_D$

$T_D = 0$

6.3. <u>Experimental Results of Choosing Genetic Parameters</u>

This experiment is concerned with selecting the crossover and mutation rate. Two experimental values for crossover and mutation probability have been selected to compare in ADSC. From these results, we can conclude that crossover rate (0.9) and mutation rate (0.1) are the suitable choices, because it gives the highest number of correct solution (chromosome gain 100% fitness).

6.4. Experimental Results of Choosing Genetic Operators

This experiment is concerned with selecting the genetic operations used to apply with ADSC. The performance of mixing between different types of crossover and mutation is presented. 1-Point crossover and 1-bit mutation is the best, because it gives the highest number of correct solution (chromosome gain 100% fitness).

7.   Conclusion

In conclusion we may note the following:

1. Crossover rate 0.9 and mutation rate 0.1 are the suitable choices, because it gives the highest number of good chromosome that gain fitness score equal to 100%.

2. 1-Point crossover and 2-bit mutation are the suitable choices, because it gives the highest number of good chromosome that gain fitness score equal to 100%.

3. Some types of counters we could not get it according to some probability of crossover and mutation and some type of combine crossover type and mutation type.

4. In general the aim of design synchronous sequential logic circuit given its input/output specification has been achieved.

5. A minimization is performed on the output chromosome to give the minimized expression which contains the minimum number of logic gates.

6. Determining genetic parameter rate is an important part of the work with GA. The genetic operators used also have an important role in the program efficiency.

7. Some problems may require a large population and since GA is a probabilistic method, then not all problems have 100% successful rate, but in experiments of this paper we have fixed the fitness at 100% some cases have been given a null solution. It was found that it is necessary to increase the size of initial or generational population to achieve a solution in some cases. However deciding upon the size of the initial and generation population is very difficult. There is more trade-off between population size and number of generations needed to converge. Increasing the population size will work more slowly but will eventually achieve better solution and in these cases the number of generations needed to find a solution decreases. So in brief, we can conclude that the most effective population size is dependent on the problem being solved, and the operators manipulating this population.

8. Suggestions for Future Work

There are a number of directions in which this work could be extended. Some of these directions are:

1. Using Genetic Programming GP for the circuit designing and comparing the result from GP and GA to decide which is better.

2. Using Automatic Define Function ADF with GP. ADF approach assumes that problem solution can be specified in terms of a main program and a hierarchical collection of subroutines and it is based on the idea of evolution of reusable subroutines.

3. Using gates number as a parameter in fitness evaluation.

4. Finding a program that is capable of taking the minimization of expression and drawing the circuit according to the minimum expression.

5. Including some equations to control the generation of number to find the correct solution in a fast way.

6. Designing combinational logic circuit and large sequential logic circuit (both types are synchronous and asynchronous).

7. Use Evolvable Hardware EHW to design a circuit.

9. References

[1]. Katz R., Gaetauo, "Contemporary Logic Design", Prentice Hall of India, 2005.

[2]. Ali B., Almaini A., Kalganova T., "Evolutionary Algorithms and Their Use in Design Sequential Logic Circuits", Kluwer Academic, 2004.

[3]. Ankenbrandt C., Buckles B., Petry F., "Scene Recognition using Genetic Algorithms with Semantic Nets", North-Holland, pp. 285-293, 1990.

[4]. Matti R., "Automatic Synthesis of Petri Nets using Genetic Programming", Ph. D. Thesis, Dept. of Computer Science and Information Systems, University of Technology, March 2002.

[5]. Goldberg D. E., "Genetic Algorithm in Search, Optimization, and Machine Learning", Addison Wesely Publishing Company, 1989.

[6]. Sharman K., E. A., Esparcia, "Genetic Evolution of Symbolic Signal Models", Procs. Of the 2nd Intl. IEEE Workshop on Natural Algorithms in Sp, Vol. 2, pp. 29, Nov., 1993.

[7]. Mano M., "Digital Design", Prentice Hall, 2002.

[8]. Subramanyam M., "Switching Theory and Logic Design", LAXMI Publications LTD.

[9]. Marcovitz A., "Introduction to Logic Design", McGraw Hill, 2002.

[10]. Balch M., "Complete Digital Design: a Comprehensive Guide to Digital Electronics and Computer System Architecture", Tata McGraw-Hill Publishing Company Limited, 2003.

التصميم المؤتمت لدائرة متزامنة متسلسلة منطقيه باستخدام الخوارزمية الجينية

م.م.هبه باسم علوان

كلية المنصور الجامعه

المستخلص :

أهتم العديد من الباحثين و المصممين بتصميم الدوائر المتزامنه المتسلسله المنطقيه, و هنالك العديد من البحوث المختلفه التي تتعامل مع هذا الموضوع.

في هذا البحث تم تصميم الدوائر المتزامنة المتسلسلة المنطقية باستخدام الخوارزميات الجينية والتي هي عبارة عن خوارزميات بحث مبنية على أساس ميكانيكيه الأنتقاء (الأختيار) الطبيعي و علم الوراثه الطبيعي و تستخدم عمليات علم الوراثه الطبيعي مثل التزاوج و الطفره الوراثيه.

إن تصميم الدوائر المتزامنة المتسلسلة المنطقية يبدأ من مجموعة من المواصفات و التحديدات أو من قائمة من الدوال التي من الممكن للتصميم المنطقي الحصول عليها.

أول خطوة في تصميم الدوائر المتزامنة المتسلسلة المنطقية هو الحصول على جدول الحالة أو مايكافئه من تمثيل مثل مخطط الحالة.

هذا العمل اقترح طريقه لتصميم الدوائر المتزامنة باستخدام الخوارزميات الجينية حيث أن تعقيد الدائرة المنطقية من الممكن أن يعرف كدالة لعدد البوابات الموجودة في الدائرة. الدائرة سوف تمثل بطريقة تستطيع العمليات الجينية من الحصول عليها.

إن معمارية تصميم الدوائر تمثلت بثلاث مراحل:

المرحله١: تمثيل مواصفات الدائره المطلوبة باستخدام جدول انتقال الحاله.

المرحله٢: استخدام جدول انتقال الحالة من قبل الخوارزمية الجينية لتوليد مخصص الحالة لتخصيص شفرة ثنائيه لكل حالة.

المرحله٣: توليد الدائرة المصغرة باستخدام طريقه الجدولة.

مجموعة مختلفة من دوائر العدادات تم اختيارها و تطبيقها لتبيان نتيجة الخوارزمية الجينية.