Double Helix Structure and Finite Persisting Sphere Genetic Algorithm in Designing Digital Circuit Structure

Nurzanariah Roslan, Karmila Kamil and Chong Kok Hen
College of Engineering, Universiti Tenaga Nasional,
Jalan Ikram-Uniten, 43000 Kajang, Selangor, Malaysia
Email: nurzanariah@uniten.edu.my

Abstract

This paper proposes a new approach of chromosome representation in digital circuit design which is Double Helix Structure (DHS). The idea of DHS in chromosome representation is inspired from the nature of the DNA's structure that built up the formation of the chromosomes. DHS is an uncomplicated design method. It uses short chromosome string to represent the circuit structure. This new structure representation is flexible in size where it is not restricted by the conventional matrix structure representation. There are some advantages of the proposed method such as convenience to apply due to the simple formation and flexible structure, less requirement of memory allocation and faster processing time due to the short chromosomes representation. In this paper, DHS is combined with Finite Persisting Sphere Genetic Algorithm (FPSGA) to optimal the digital circuit structure design. The experimental results prove that DHS uses short chromosome string to produce the flexible digital circuit structure and FPSGA further optimal the number of gates used in the structure. The proposed method has better performance compared to other methods.

Key words: Digital circuit, Genetic Algorithm, FPSGA, Double Helix Structure.

1. Introduction

Nowadays, researchers apply the optimization approach in various fields of studies. These methods represent the problem in the form of coding and to be processed by the computer. Simpler design flow is more convenience to the user as well as to minimize the task to the processor. In digital circuit design, the representation of the chromosomes using Cartesian Genetic Programming (CGP) is first introduced by Miller and Thomson [1] [2].They proposed a chromosome representation in the form of linear string of integers from an indexed graph. After that, they developed another approach which is Developmental Cartesian Genetic Programming (DCGP) [3]. DCGP has a complex transformation between the genotype and encoded CGP graph from a defined program in the genotype. Then, it will be run inside the note of the phenotype [4].
Slowik in [5][6] explained about the representation of multi- layer chromosome in designing digital circuit. The author attempts to prove that the representation from single- layer to multi-layer chromosome can help to improve the algorithms. In other approaches, Coello et al, 2000 used the idea from Louis in encoding the chromosome. The bi-dimensional matrix was used where the element in the matrix is the gate type where it receives input from the previous array [7] [8]. In this paper, a new approach Double Helix Structure (DHS) is proposed which use short chromosome string representation in the digital circuit structure design and the structure is further optimized by Finite Persisting Sphere Genetic Algorithm (FPSGA).

2. Double Helix Structure in Digital Circuit Design

2.1. Double Helix Structure of DNA

DNA present in the chromosome processes the genetic information in all living things. The biology field revealed that the structure of the DNA is modeled by the double-helix arrangement [9].
Double helix structure consists of two identical helices tangled about a common axis. The DNA is making up of four types of molecules, adenine, thymine, guanine and cytosine (A, T, G and C). The term of "base pairs" is used to characterize the pair combination of A with T and G with C [9] [10] as shown in Figure 1.

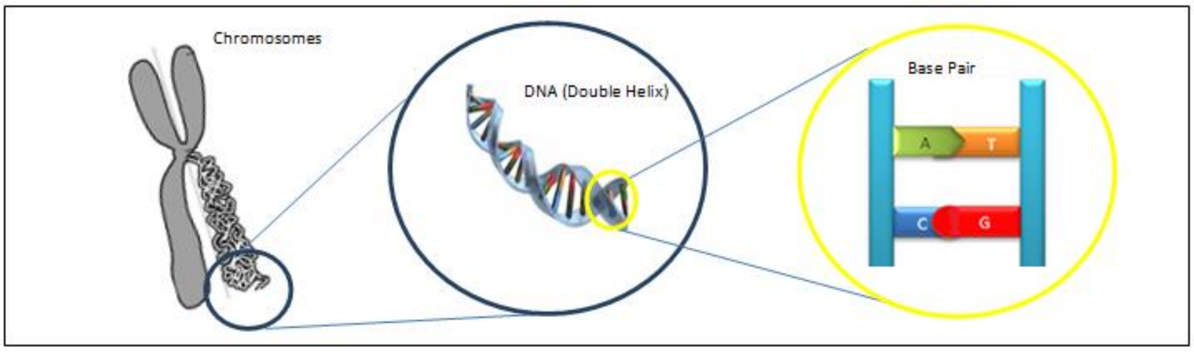


Figure 1. Double Helix Structure of DNA in the Chromosomes

The reproduction of DNA is called DNA replication where biological inheritance in all living organism occur. In the process of DNA replication, two identical molecules will be produced. Each strand of the double stranded DNA serves as template for the production of the paired strand [10]. The illustration of the process is shown in Figure 2.

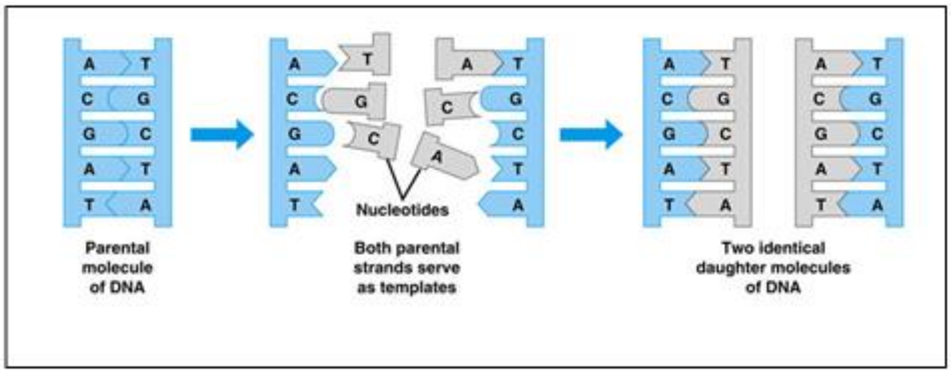

Parental molecule of DNA
Nucleotides
Both parental strands serve as templates
Two identical daughter molecules of DNA

Figure 2. DNA Replication [11]

2.2. Double Helix Structure

The idea of DHS in digital circuit design is inspired from the natural formation of DNA structure in human body. However, only two molecules are responsible in forming a base pair. They are static and dynamic components. The illustration of the proposed idea is shown in Figure 3.

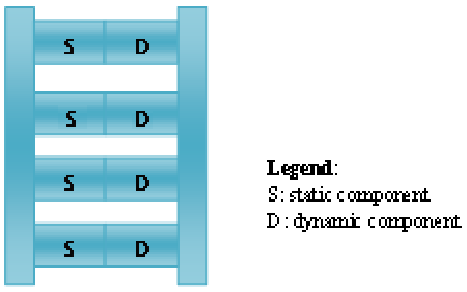


Figure 3. Base Pair in DHS

The dynamic component of the base pair comprises of bit string where it is used to encode the type of logic gates in the design of the digital circuit. It represents in the form of real number from 0 to 7. Each bit has different function of logic gates and can be defined from Table 1 [12]. The representation of number 6 and 7 in table 1 show the function of wire. The output of wire 1 depends on the input 1 while the output of wire 2 will depend on the input 2.

Table 1. Logic Gates Function and Bit Representation

Chrom	Operation	Truth Table		
0	OR	A	B	X
		0	0	0
		0	1	1
		1	0	1
		1	1	1
1	AND	A	B	X
		0	0	0
		0	1	0
		1	0	0
		1	1	1
2	NOR	A	B	X
		0	0	1
		0	1	0
		1	0	0
		1	1	0
3	NAND	A	B	X
		0	0	1
		0	1	1
		1	0	1
		1	1	0
4	XNOR	A	B	X
		0	0	1
		0	1	0
		1	0	0
		1	1	1
5	XOR	A	B	X
		0	0	0
		0	1	1
		1	0	1
		1	1	0

6	Wire1	Output=input1
7	Wire2	Output=input2

For the offspring production, crossover and mutation operation will be performed on the dynamic structure only while the static structure will serve as template during the process. After the process of crossover and mutation, it will be combined together to their respective static pair for the fitness evaluation process.
Static structure functions as the input selector for the input of the gate; it is coded in the dynamic component. The static structure will not be affected by the process of genetic operators. One static bit provides a pair of input combination for a gate where the code is flexible and can be designed by the programmer.
The static part will not perform any transformation and the location is static in its generation. The representation of the static bit is in the form of real number and the total number of the bit is similar with the number of bit in the dynamic part where one input selector is denoted for one logic gate. In this approach, the total length of the chromosome is twice the number of gates.
The size of the chromosome can be started from as small as 6 bits. However, if there is no feasible solution found, the size can be increased by two bit which is one bit for static bit and one bit for dynamic bit. Figure 4 shows the link between the bits in the chromosome. S1 is responsible to supply input for D1, S2 for D2, S3 for D3 and it is applied to the entire structure.

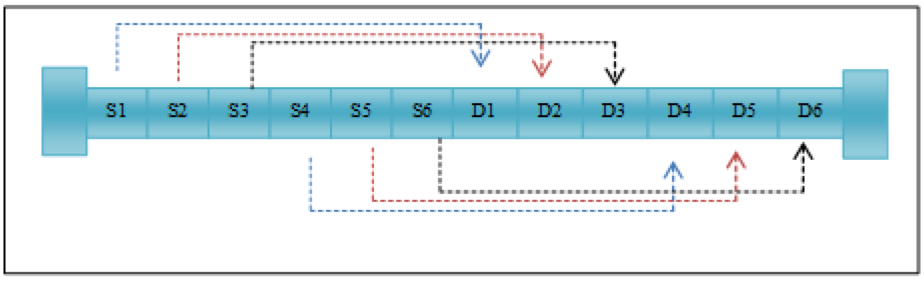


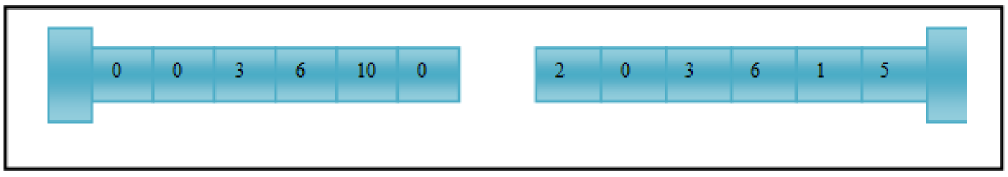
Figure 4. Link between Bits

Table 2 represents the input combination in the static structure for the input of a 3-bit *f*(a,b,c) digital circuit. The rule of the input combination is that a combination of input pairs can only be repeated two times, for example, if it firstly exists in S1, therefore it can only be repeated in S2 and S3.
For the configuration in Table 2, the maximum bit of S1 is 2, S2 is 5, S3 is 9, S4 is 8, S5 is 10 and 13 for S6 where it provides different selections of input for the gate. The programmer can propose the input combination based on their need in the circuit design.

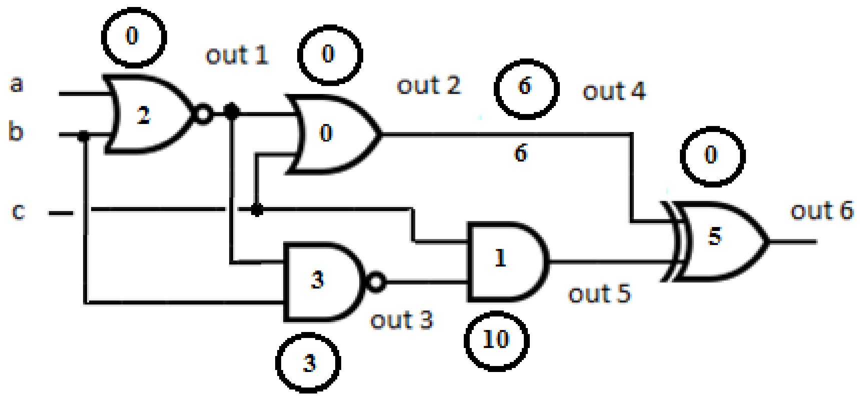
Table 2. The configuration of Input Combination in Static Component of DHS

Bit	S1		S2		S3		S4		S5		S6	
	I/P1	I/P2	I/P1	I/P2	I/P1	I/P2	I/P1	I/P2	I/P1	I/P2	I/P1	I/P2
0	a	b	out1	c	out1	out2	out3	out2	out4	out3	out5	out4
1	a	c	out1	b	out1	a	out3	out1	out4	out2	out5	out3
2	b	c	out1	a	out2	a	out3	a	out4	out1	out5	out2
3	-	-	a	b	out1	b	out3	b	out4	A	out5	out1
4	-	-	a	c	out2	b	out3	c	out4	B	out5	a
5	-	-	b	c	out1	c	out1	out2	out4	C	out5	b
6	-	-	-	-	out2	c	out2	a	out3	out2	out5	c
7	-	-	-	-	a	b	out2	b	out3	out1	out4	out3
8	-	-	-	-	a	c	out2	c	out3	A	out4	out2
9	-	-	-	-	b	c	-	-	out3	B	out4	out1
10	-	-	-	-	-	-	-	-	out3	C	out4	a
11	-	-	-	-	-	-	-	-	-	-	out4	b
12	-	-	-	-	-	-	-	-	-	-	out4	c

The example of the genotype-phenotype mapping can be seen in Figure 5. The numbers in the circles of figure 5b are denoting the static component of the DHS chromosome while the numbers in the gate denote the dynamic component of the chromosome.



(a)



(b)

Figure 5. (a) Genotype and (b) Phenotype Mapping of DHS Chromosome

In the proposed method, during the process of crossover and mutation, the static and dynamic components will be separated. Referring to Figure 6a to 6c the bits in the static part of the parent operate as template for the base pair while the dynamic part proceeds with the crossover and mutation process. After the process, the dynamic part will be combined back to their respective static pair.

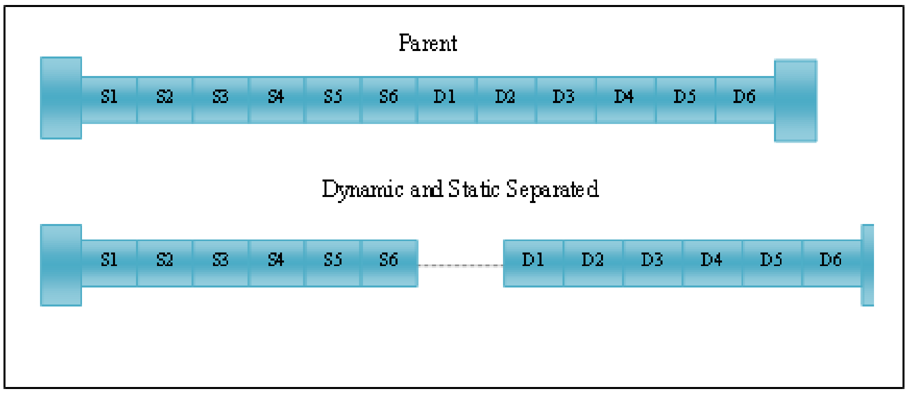

Parent
S1 S2 S3 S4 S5 S6 D1 D2 D3 D4 D5 D6
Dynamic and Static Separated
S1 S2 S3 S4 S5 S6
D1 D2 D3 D4 D5 D6

Figure 6a. Separation of Dynamic and Static to Prepare for Crossover and Mutation Process

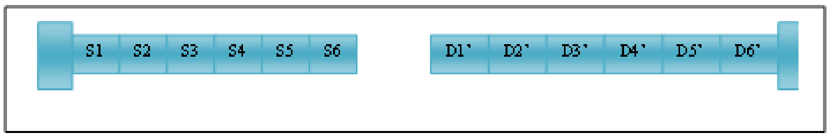


Figure 6b. The DHS chromosome after the Process of Crossover and Mutation

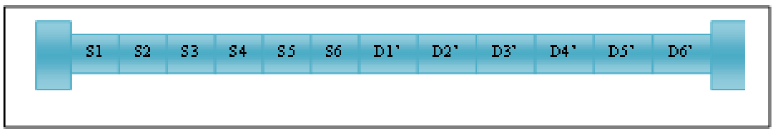


Figure 6c. Dynamic and Static Component Combined Together to Form Offspring

3. Finite Persisting Sphere Genetic Algorithm (FPSGA)

The process of FPSGA is comprised of uni-chromosome crossover and Finite Persisting Sphere.

3.1. Uni-chromosome Crossover

All selected chromosome from the process of Roulette Wheel Selection will have similar chances to produce child. In the uni-chromosome crossover, the number of child is depended on the number of Finite Persisting Sphere loop. Child produced from the uni-chromosome crossover process is formed from the revolution of the chromosomes in the horizontal orientation of the chromosomes [13].
The uniqueness of the uni-chromosome crossover is that one single parent can be very productive in producing child. In Genetic Algorithm (GA), children produced from the crossover will inherit various genes and information from its parent. In uni-chromosome crossover, the children will compete to each other, be ranked and only the fittest child will represent its family and bring the best genes to compete with other child in other family from the crossover process of different individuals. This is how the synergy is increased for all individuals in a same population.
There are several types of crossover operator such as one point crossover and two point crossovers. From [14], the process of one point crossover can be summarized in the following format:

Chromosome of Parents:

Parent 1= $\{a_1, a_2,\ldots..a_n\}$ (1)
Parent2 = $\{b_1, b_2,\ldots.b_n\}$ (2)

Chromosome of Offspring:

Offspring 1 = $\{a_1, a_2,\ldots.a_i', b_{i+1},\ldots\ldots\ldots,b_n\}$ (3)
Offspring2 = $\{b_1, b_2,\ldots.b_i', a_{i+1},\ldots\ldots\ldots,a_n\}$ (4)

where:

$a_i' = \alpha_i a_i + (1-\alpha_i)b_i$
$b_i' = \alpha_i b_i + (1-\alpha_i)a_i$

The process of this type of genetic recombination will swapped the contents of the chromosomes in order to produce child. In one point crossover, two parents will produce two children. Uni-chromosome crossover is proposed to improve the productivity of parent in producing child. The aim of uni-chromosome crossover is as the number of child is increased; the probability to have high-quality genes is also increased. It will in addition widen the diversity of the chromosome distribution where it can prevent the possible solution from being trapped in local optimum area. Depending on the number of Finite Persisting Sphere, the process of uni-chromosome crossover can be represented as follows [13]:

Chromosome of 1 Parent:

$$= \underset{i=0}{\overset{i=P}{\mathbf{X}}} ,x_1, x_2,x_3, x_4\ldots..x_{n-1}\} \quad (5)$$

Let Number of Finite Persisting Sphere= 3;
Chromosome of offspring:

Offspring 1= $\{x_1', x_2', x_3'' x_4',\ldots..x_{n-1}',x_0'\}$ (6)
Offspring 2= $\{x_2',x_3', x_4',\ldots..x_{n-1}',x_0',x_1'\}$ (7)
Offspring 3= $\{x_3', x_4'\ldots..x_{n-1}',x_0', x_1', x_2'\}$ (8)

where

P = No of population
n= No of variables/allele

Instead of producing 2 children from a couple of parents, uni-chromosome crossover can produce greater number of children from an individual parent. Equation 5 represents a parent from a population of possible solution in a problem. The chromosome is built up of n number of genes. Depending on the number of Finite Persisting Loop, the number of child produced is equivalent to it as shown in Equation 6 to 8. Each child will be evaluated based on their fitness value.

3.2. Finite Persisting Sphere Process

Finite persisting sphere is introduced in order to maintain the diversity of good genes in a population. It can be defined as a loop where it contains a number of processes and this loop will be terminated as the predefined number of loop is achieved. This loop is to ensure that each child produced from the uni-chromosome crossover process will be evaluated and ranked according to their fitness value. The chromosomes in this area are not affected by the mutation process. The best offspring from each parent will be retrieved and the rest will continue the next process of the FPSGA [13].
The pseudo code of the process of finite persisting sphere can be explained as follows:

```
1.0 Defined the number of Finite Persisting Sphere loop, $N_{FPS}$
2.0 As $N_{FPS}$ did not achieve:
do
                2.1 Uni-chromosome crossover
                2.2 Evaluate the fitness function of each child
        2.3 Rank the chromosome according to their fitness
        2.4 Store the best chromosome from Finite Persisting Sphere process as $F_C$
        end
```

The complete procedure of FPSGA in designing digital circuit is shown in Figure 7. The FPSGA process is shown in the highlighted box.

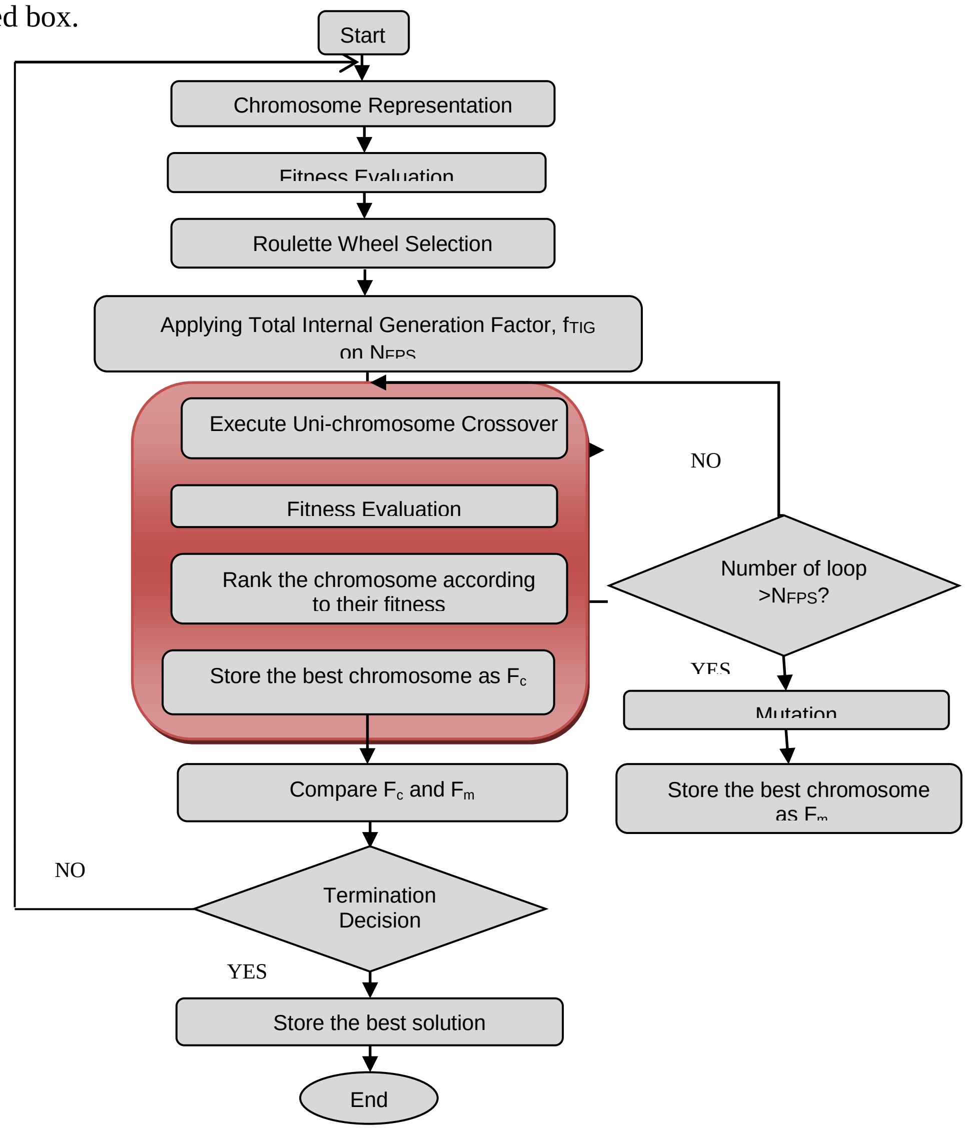

Start
Chromosome Representation
Fitness Evaluation
Roulette Wheel Selection
Applying Total Internal Generation Factor, fTIG on NFPS
Execute Uni-chromosome Crossover
Fitness Evaluation
Rank the chromosome according to their fitness
Store the best chromosome as Fc
NO
Number of loop >NFPS?
YES
Mutation
Store the best chromosome as Fm
Compare Fc and Fm
NO
Termination Decision
YES
Store the best solution
End

Figure 7. Full Process in FPSGA

4. The Digital Circuit Design Using FPSGA and Double Helix Structure of Chromosomes

As discussed earlier, the chromosome used for the digital circuit is encoded using the DHS technique. The chromosomes are then being used in designing digital circuit design and the engine of optimization which is FPSGA will optimize the number of gate in the circuit. There are four examples of circuits with different minterms designed and analyzed in this paper. The GA parameters applied in the experiments are listed in Table 3.

Table 3. GA Parameters in designing digital circuit.

Experiment	**1**	**2**	**3**	**4**
Population Size	55	55	50	500
Length of Chromosome (bits)	8	8	8	16
Probability of crossover	1.0	1.0	1.0	1.0
Probability of mutation	0.1	0.1	0.1	0.3

4.1. Example 1

The first design in the experiment involves three inputs and one output of digital circuit. The minterm and the truth table of the circuit given as follows:

Minterm:

$$f(a,b,c) = \sum (3, 5, 6) \qquad (9)$$

Truth table:

Table 4. Truth table of the circuit in Example 1

INPUT			**OUTPUT**
A	B	C	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The designed and optimized circuit is retrieved at generations 101. The genotype and phenotype of the circuits is shown in Figure 8.

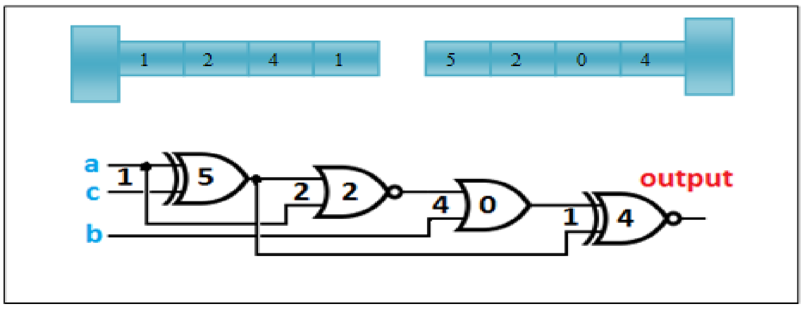


Figure 8. The genotype and phenotype of Example 1

4.1.1. Analysis I

Table 5. Comparison of the result for Example 1

Criteria	**Proposed Method**	**MLCEA-TC** [6]	**GA with bidimensional Matrix** [7]
Number of Gates (optimized)	4	4	4
Length of Chromosome/ Pattern of Gates	8	25	75
Population Size	55	100	300

Referring to Table 5, the number of gates of the designed circuits in Example 1 for all the methods are the same which is 4. However, it is found that the length of the chromosomes designed by the DHS is only 8 bits while by using MLCEA-TC [6], 25 patterns of gates are used and 75 bits are used in normal GA with bi-dimensional matrix [7]. By the proposed method, the number of population used is only 55 while MLCEA-TC used 100 population and GA with bidimensional matrix used 300 populations. It is not only promising less memory usage and less data to be processed but it also helps in reducing the processing time of the design. It can be concluded that by combining the chromosome representation using DHS and optimizing the number of circuit using FPSGA, the proposed method are very reliable in designing digital circuit structure.

4.2. Example 2

The second design in the experiment involves four inputs and one output of digital circuit. The minterm and the truth table of the circuit given as follows:
Minterm:

$$f(a,b,c,d) = \sum (2,3,5,6,8,9,12,15) \qquad (10)$$

Truth table:

Table 6. Truth table of the circuit in Example 2

INPUT				**OUTPUT**
A	B	C	D	z
0	0	0	0	0

0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	-1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The designed and optimized circuit is retrieved at generations 51. The genotype and phenotype of the circuits is shown in Figure 9.

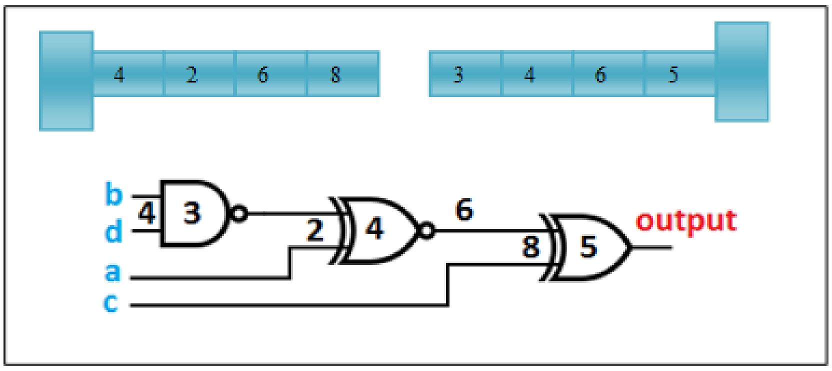


Figure 9. The genotype and phenotype of Example 2

4.2.1. Analysis II

Table 7. Comparison of the result for Example 2

Criteria	**Proposed Method**	**GA with matrix representation** [15]
Number of Gates (optimized)	3	3
Length of Chromosome/ Pattern of Gates	8	10
Population Size	55	1000

In the second example of the experiment, Table 7 shows that the optimized number of gates is the same for both methods. However, only 8 bits of chromosomes are being used by using the DHS method while 10 bits used by GA with matrix representation. It also shows that the proposed method only needs 55 number of population in making the circuit functioning while the GA with matrix in [15] needs 1000 populations to achieve the objectives. Again the proposed method shows an outstanding performance of the method in designing the digital circuit as well as optimizing the number of gates.

4.3. Example 3

The third design in the experiment also involves four inputs and one output of digital circuit. The minterm and the truth table of the circuit given as follows:
Minterm:

$$f(a,b,c,d) = \sum (10, 11, 12, 13, 14, 15) \qquad (11)$$

Truth table:

Table 8. Truth table of the circuit in Example 3

INPUT				OUTPUT
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	-1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

The designed and optimized circuit is retrieved at generations 51. The genotype and phenotype of the circuits is shown in Figure 10.

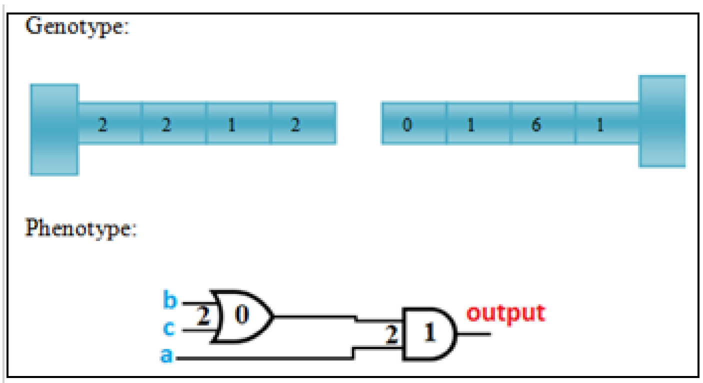


Figure 10. The genotype and phenotype of Example 3

4.3.1. Analysis III

Table 9. Comparison of the result for Example 3

Criteria	Proposed Method	GA with binary bit string representation [2]
Number of Gates (optimized)	2	4
Length of Chromosome/ Pattern of Gates	8	36
Population Size	50	100

Table 9 shows that the number of gates in the designed circuit after it has been optimized is 2 by using the proposed method and 4 by using the GA with binary bit string representation. It shows that FPSGA is more efficient in optimizing the number of circuit compared to GA with binary bit string representation. The length of the chromosomes designed by the DHS is only 8 bits while the compared method used 36 bits which is too long and wasting the memory application. It also shows that the number of population used by the proposed method is only half of the number used in [2]. Again, it shows the capability of the FPSGA and DHS in designing and optimizing the digital circuit structure.

4.4. Example 4

The fourth design in the experiment also involves four inputs and one output of digital circuit. The minterm and the truth table of the circuit is given as follows:
Minterm:

$$f(a,b,c,d) = \sum (0, 1, 3, 6, 7, 8, 10, 13) \tag{12}$$

Truth table:

Table 10. Truth table of the circuit in Example 4

INPUT				OUTPUT
A	B	C	D	Z

0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	-1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

The designed and optimized circuit is retrieved at generations 20. The genotype and phenotype of the circuits is shown in Figure 11.

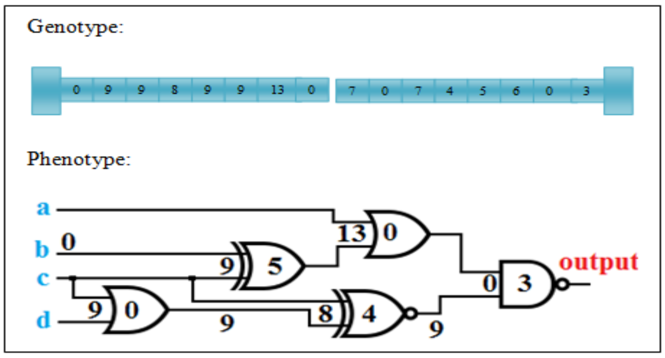


Figure 11. The genotype and phenotype of Example 4

4.4.1. Analysis IV

Table 11. Comparison of the result for Example 4

Criteria	Proposed Method	MLCEA-TC [6]	GA with bidimensional Matrix [7]
Number of Gates (optimized)	5	6	10
Length of Chromosome/ Pattern of Gates	16	25	75
Population Size	500	100	1000

Table 11 shows the result of the digital circuit structure design for Example 4. The proposed method gives the least number of gates after the design has been optimized which is 5 while the MLCEA-TC and GA with bidimensional matrix used 6 and 10 gates respectively. Although the population size of the proposed method is quite large compared to MLCEA-TC, the proposed method only used 16 bits of chromosome and the MLCEA-TC used 25 bits of chromosome. GA with bidimensional matrix used too many bits in representing the chromosome and at the same time, it used a large number of population in designing the circuit.

5. Conclusion

From the experiments that have been conducted and analyzed, the results show that DHS is very efficient in representing the chromosomes of digital circuit structure. The representation is short, simple and convenient to conduct. The programmer can creatively create their database of input combination and freely use it to depend on their needs. Since the length of the chromosome is short and simple, it can help to reduce usage of the memory. It will also reduce the processing time in designing the circuit. FPSGA, which is the optimization engine in this proposed method, proved that it is reliable in reducing the number of gates compared to other evolutionary method as can be seen in part 4 of this paper. In conclusion, designing the chromosome of digital circuit using DHS and optimizing the number of gates using the FPSGA is very effective, efficient and reliable.

6. References

[1] Miller, J.F. and P. Thomson, “Cartesian Genetic Programming”, Proceedings of European Conference on Genetic Programming, Edinburgh, April 15-16 2000. pp: 121-132.

[2] Yan, X., Q. Wu, C. Hu and Q. Liang.,“Design electronic circuit using evolutionary algorithms”, J. Next Gener. Inform. Technol., 2010, 1: 127-139.

[3] Shanthi, A.P. and R. Parthasarathi.,Practical and scalable evolution of digital circuit. Applied Soft Comput., 2009, 9: 618-624.

[4] Miller, J.F. and P. Thomson, “A developmental method for growing graphs and circuits”, Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware, Trondheim, Norway, March 17-20, 2003, pp: 93-104.
[5] Slowik, A. and M. Bialko, “Evolutionary design of combinational digital circuits: State of the art, main problems and future trends”,
Proceeding of the 1st International Conference on Information Technology, Gdansk Poland, May 18-21, 2008, pp: 1-6.

[6] Slowik, A. and M. Bialko, “Evolutionary design and optimization of combinational digital circuits with respect to transistor count”, Bull. Pol. Acad. Sci., 2006, 54: 437-442.

[7] Coello, C.A.A., A.D. Christiansen and AH. Aguirre, “Use of evolutionary techniques to automate the design of combinational logic circuits”, Int. J. Smart Eng. Syst. Des., 2000, 2: 299-314.

[8] Coello, C.A.A., A.D. Christiansen and A.H. Aguirre, Towards automated evolutionary design of combinational circuits, Comput. Electr. Eng., 2001, 27: 1-28.

[9] Voet, D. and J.G. Voet, *Biochemistry*, 2nd Edn., John Wiley and Sons Inc., New York, USA., 1995, pp: 850-868.

[10] McKee, T. and J.R. McKee, *Biochemistry: An Introduction*, 2nd Edn., McGraw-Hill, New York, USA., 1999, pp: 467-469.

[11] Scott H., The Double Helix, Lab Notes of Biology from Course Websites of Department of Life Science, Riverside Community
College, http://faculty.rcc.edu/herrick, retrieved on April 2011.

[12] Mano, M.M., *Digital Design*, 3rd Edn., Prentice Hall, USA., 2002, pp: 53-61.

[13] Kamil, K., K.H. Chong, S.K. Tiong and K.H Yeap, "Finite persisting sphere genetic algorithm in solving multi objectives problem", Proceeding of the IEEE Student Conference on Research and Development, December 13-14, 2010, Putrajaya, Malaysia, pp: 183-186.

[14] Hwang, S.F. and R.S.He, Improving real-parameter genetic algorithm with simulated annealing for engineering problems, Adv. Eng. Software, 2006, 37: 406-418.

[15] Chong, K.H., I.B. Aris, M.A. Sinan and B.M. Hamiruce, "Digital Circuit Structure Design via Evolutionary Algorithm Method", J. Applied Sci., 2007, 380 -385.