Performance Analysis of Combinatorial Digital Circuit Evolution with Cluster Growth Technique

Atul Kumar Srivastava and Hariom Gupta

Department of Electronics and Communication Engineering, Jaypee Institute of Information and Technology, NOIDA, U.P. India 201307

Abstract: An interesting approach of using genetic programming is to synthesize circuit with least possible number of logic gates and the same is presented in this paper. The genetic algorithm (GA) is one of the optimization techniques of evolutionary algorithm used to design evolvable hardware. This paper proposes the applicability of cluster based growth approach with genetic programming. Since in previous related literature of evolvable hardware the drawback stated is scalability, evolution time and redundancy. With Clustering technique presented in this paper there is an effort to overcome these drawbacks. The method of evolution is also modified so that the evolution time is reduced. Since in this paper the clustering evolution is used that reduces the size of chromosome in genotype and this somewhat over comes the problem of scalability. Our main contributions are: 1) Adaptation of genetic operators in a way suitable for clustering growth, 2) Clustering with genetic programming of digital circuits and show its performance, 3) To obtain the desired functionality with least number of logic gates such that the interconnection of nodes in the earlier defined architecture is modified with clustering, so that the fast convergence is obtained. This in turn reduces stalling effect in evolvable hardware.

Key words: Evolvable hardware · Evolutionary algorithm · Mutation · Tournament selection · Cluster growth

INTRODUCTION

Evolvable hardware (EHW) [1-3] is a special technique to avoid manual engineering. It is a software based field that can give an answer to a problem more easily and efficiently than manual or traditional methods. Earlier it has been outlined that how EHW can be applied for hardware design of real-world applications. The circuit here is designed using evolutionary algorithm. Genetic programming [4, 5] is the most commonly used evolutionary algorithm and is discussed in this paper.

Nowadays, for the reconfiguration of electronic circuits Cartesian genetic programming [6] is used for converting phenotype to genotype representation and the same is used in this paper also. The phenotype represents the physical nature of the chromosomes while genotype is the chromosome’s hereditary information. Success depends on fitness where successful individuals are able to reproduce and pass on their genes to the new generation using two important methods: mutation and crossover. The fit strings are selected to perform operations-crossover and mutation [4, 7, 8] the complete

![Fig. 1: Genetic algorithm cycle](image-url)

GA cycle is also shown in Fig. 1. These cycles are performed till a desired result, i.e. 100% fit individual is not obtained or in other word we can say the error between obtained solution and expected solutions is zero or within presumed limits. Since in every field and area, optimum solution to a complex problem is required and this is where genetic algorithm comes into play to find the optimum solution with high efficiency. In digital circuit
design, circuit with least number of gates is favorable to reduce the size of design, to decrease the complexity and side by side increase the efficiency and finally make the design cost efficient.

Clustering [8-10] based on genetic programming gives a far better result than K-means clustering. Cluster growth is a method of grouping data that share similar nature or pattern [8, 9]. It partitions the set into subsets or clusters such that each subset shares common traits. The notion of similarity is problem dependent. In this paper, clustering using genetic programming has been used to find the optimum solutions.

**Problem Formulation:** Evolvable hardware consist of predefined hardware consisting of nxm programming elements (PE’s), that are programmed and reconfigured as shown in Fig. 5. Many times the evolved circuits consist of redundant PE’s. This though is helpful in fault tolerant systems but sometimes the number is quite high. We have used clustered based growth technique [9] for evolution so that the minimum numbers of logic gates are used to obtained the functionality of digital circuits. Fewer gates reduce the area complexity of hardware and power/area dissipation also.

**Genetic Algorithm:** It is class of algorithm inspired by biological evolution process. With a problem in hand, GA makes a population of some optimum solutions and a desired result is obtained by various operations. These operations include selection, crossover and mutation. Cell crossover [11-13] crosses the cells to get good sub solutions. A general cycle of genetic algorithm is shown in Fig. 1.

Purpose of doing selection is to search in a promising population and to reach earliest the solution. Tournament selection, selects individuals from the population with some probability, is used in this paper. A chromosome wins according to its fitness. Mutation is done with the aim to decrease the effect of error with less probability occurred during duplicity and thereby helps in removing some small loss of information. It works according to the following procedure:

- One offspring is selected for mutation.
- Mutation points are selected according to mutation probability $P_m$.
- Selected points are changed randomly and new individual is obtained.

Mutation process is represented in Fig. 2 with mutation point 3, 5 and 8 (in italics).

![Mutation](image)

**Fig. 2: Mutation**

**Offspring:**

Before: 0 1 0 1 1 1 1 0 0

After: 0 1 1 1 1 1 1 0 0

---

![Cluster growth](image)

**Fig. 3: Cell crossover**

![Cell crossover](image)

**Fig. 4: Cluster growth**

Similar to biological mating, crossover selects two chromosomes to mate with each other according to a crossover probability. In cell crossover, the complete cell is exchanged as shown in Fig. 3 the chromosome 1, cell2 is exchanged with cell 5.

The design methodology for any circuit requires that the size of circuit has least possible number of gates. This can be achieved by a clustering growth. Population is grouped into clusters of smaller population of similar chromosomes and genes (i.e. similar data). This algorithm helps to find circuit with least number of gates and also to get different possible circuits for a desired output.

**Clustering:** Formally clustering can be said as technique in which clusters are connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a low density of points [9, 10]. Large set of data is grouped into clusters of smaller sets of similar data. Fig. 4 shows a small representation of how clustering can be done.

Clustering technique is applicable in engineering sciences, life sciences and also in practical life [8-10].
Clustering Has Been Performed Majorly in Two Ways:

- K-means [9, 10]: It is distance based clustering which can be used to make a given number of clusters from the data.
- Genetic programming [8, 10]: Every chromosome is a cluster and each gene is an object. Gene with same allele value is in same cluster.

K-means has drawbacks which includes necessity of knowing the number of clusters and also it does not give a global result but a result from which a complete different final cluster can be made. But this drawback can be corrected by using clustering in genetic algorithm.

**Problem Solution:**

**Genetic Encoding for Digital Circuit:** The architecture is viewed as a matrix; the individual blocks or cells are represented by logic gate along with its n inputs [11, 12]. Initially circuit is viewed as single logic gate and step by step numbers of gates are increased to maximum n x m number in a hardware matrix. Fig. 5 shows a representing a cell.

Each cell represents here a basic logic gate that has input from previous column. If cell is in first column then input is represented by negative number else it is given a positive number.

For example $a_{ij} = \{-1 -1 3\}$ implying the cell is in $i^{th}$ row and $j^{th}$ column with inputs given externally and 3 represents gate type 3.

$a_{ij} = \{1 2 4\}$ where $a_{ij}$ is an intermediate column cell with internal inputs and gate type as 4, i.e. NAND gate. Along with digital gate, a cell can be a wire also as mentioned in the look up Table 1.

**Fitness Function:** Fitness provides the percentage correctness of evolved circuit.

Each and every bit in the truth table of evolved circuit is compared with the corresponding bit of required in the truth table. If the corresponding bits are equal then fitness is increased by a certain constant factor and for mismatch in the bits then fitness remains unchanged. For an efficient data structure, fitness value can be stored as a last gene (last column) of chromosome. This is for maintaining the simplicity of algorithm.

\[
F_t = \frac{\text{number of right outputs}}{\text{total number of test inputs}} \times 100 \quad (1)
\]

Fitness for complexity of the circuit is defined as a factor always less than or equal to one.

\[
F_c = \frac{1}{N} \sum_{i \in N} \sum_{f=1}^{G_c} G_c
\]

Where N is the total number of transistor required to realize a circuit, N is the number of transistor required to realize $k_{th}$ gate. Gc is the type of gate available in the library. If $F_c \leq 1$ then the complexity constraint is satisfied else the whole process is repeated.

**Algorithm for Generating Circuit:** 1. Initially the matrix as shown in fig. 4 is assumed such that there exists only one gate and rest all are wire, represented by integer ‘7’.

A truth table is made for the required and obtained output.

Let the number of inputs be n. Therefore the possible combinations will be equal to $2^n$. Number of rows in the truth table will be $2^n$. Input is represented in binary form as shown in Table 2. ‘1’ or ‘0’ implies wrong output.
Each bit is compared with that of output and is calculated according to the formula.

**Algorithm for next Generation:** Population generated is used to check fitness of the chromosomes [11, 12].

**Chromosomes Are Arranged According to the Fitness:** Less fit chromosomes are rejected and fit chromosomes are used to generate a new population. Healthy chromosomes pass on their genes to the new population. This process of inheritance is carried out by cell crossover and mutation.

**Two Chromosomes Are Selected by Tournament Selection:** Tournament selection selects two good chromosomes to perform cell crossover according to crossover probability. New chromosomes are generated which carry forward the genes of their parents. Cells with high fitness are taken to the next generation and bad ones are eliminated at present stage or whichever may be the criterion of next generation.

**Before Mutation, Elite Chromosomes Are Selected:** Elite chromosomes are good chromosomes and mutation is not applied on them. Probability of mutation is low and change the bits of the chromosomes randomly. High probability of mutation is not effective. It can be applied to whole population or on selected chromosomes. High probability of mutation is not effective. It can be applied to whole population or on selected chromosomes. 

**RESULTS AND DISCUSSION**

The complexity of circuits is increasing day by day. It is required to have hardware within limited available resource. This in turn reduces the power dissipation. In the proposed work two techniques are clubbed together that are 1) Clustering growth, 2) genetic programming for the evolution of circuit. The desired combinatorial digital circuit performance of [11] along with few other digital circuits are evolved in this paper using CGP method of evolution and clustering technique of [8, 9, 10]. Our work combines the two tools for the synthesis of arithmetic circuits with a cell crossover operator and a fast fitness function also as given in [3, 16].

Results of our method for full adder and odd/even parity are shown in Table 3 and compared with the simulation without using cluster technique in Table 4.

The simulation for full adder is for two different hardware matrix i.e. 3x3 and 4x4. Whereas for odd/even parity, it is of 4x4 and 5x5 matrix. But the numbers of generations are different for the two circuits. The programming is performed in MATLAB and the synthesized circuit is shown in Fig.7a-7d. Each rectangular box in the figure represents a gate and the arrow represents interconnection between the two gates. In all the CGA programming techniques the matrix for the synthesis is always defined when the population is generated. This means that the only criterion left for a new hardware is the number of generations required to produce a 100% fit circuit. In our technique, the hardware n x m matrix is consisted of only the wires initially and gates are replaced by the wires in each iteration. The fitness is evaluated till the desired fitness value is obtained and new generation is evolved by adding new types of gate combinations in the given matrix, replacing the wires from it. Our paper conducts the validation of it in arithmetic circuits. It unites the knowledge of user of system with its implementation. Genetic operators are discussed in details.
Table 3: Simulation table for full adder and odd/even parity circuit without clustering technique

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of generations</th>
<th>Number of rows</th>
<th>Number of columns</th>
<th>Number of Output ckts obtained</th>
<th>Minimum number of gates obtained in a circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Adder</td>
<td>15</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Odd/even parity</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>21</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4: Simulation table for full adder and odd/even parity circuit without clustering technique

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of generations</th>
<th>Number of rows</th>
<th>Number of columns</th>
<th>Number of Output ckts obtained</th>
<th>Minimum number of gates obtained in a circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Adder</td>
<td>15</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Odd/even parity</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>30</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig. 6: Flow chart for cluster genetic programming

Fig. 7 (a)

Fig. 7 (b)

Fig. 7 (c)
CONCLUSION

Clustering using genetic programming is an effective technique where a population is stored so as to evolve it with time. Genetic programming uses operations like mutation and crossover to get a new generation. K-mean clustering is not as effective as clustering with genetic algorithm. Programming can be performed in a way so that instead of taking fixed clusters or a fixed matrix size as in Fig 5, it can be modified to a form such that initially population size is small with only single gate in consideration and then population size is increased by increasing the number of gates. They can be easily adapted to an output. In this proposed method though it’s in a preliminary stage and in future we would like to perform analysis with other limitations of evolvable hardware.

REFERENCES

9. Ujjwal Maulik, Sanghamitra Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern recognition 33(2000), received on 24 June 1998; received in revised form 29 April 1999; Accepted 29 April 1999.