I J C T A, 9(5), 2016, pp. 2369-2375
© International Science Press

Evolution of larger digital arithmetic circuits with smaller modules for improving scalability

V.P. Kolanchinathan* and G. Saravana Kumar**

ABSTRACT

The structure and functions of traditional hardware once created cannot be changed which makes it inflexible. In order to overcome such real world problems we need to create different hardware structures. To solve the problem dynamically we use EHW approach. The hardware structure is fixed in design phase for conventional systems. But in our system the hardware changes its structure dynamically. For this purpose some effective search algorithms are used while designing a circuit. The algorithm used is referred as evolutionary algorithm. The problem exhibited in the design of combinational circuit is problem of scale. The gates involved for designing optimal circuits is too high. The reasons being search space size is more as there are enormous gates and as the truth table size increases the time taken to calculate the fitness of circuit also increases. Thus our proposed system involves designing combinational circuits in which the basic building blocks are small sub circuits. The design of digital arithmetic circuits using digital modules are not feasible and hence arithmetic circuits with modules evolved. But this also has a disadvantage because as modules increase the number of gates used also increases.

Keywords: Evolutionary algorithm, scalability, Evolutionary Cycle, CGP and arithmetic circuits.

1. INTRODUCTION

Earlier engineers designed physical systems based on rules and principles. Top down approach is used in design process. Later design process was based on natural selection process. The design starts with DNA which has a set of instructions and then transferred into the RNA of cell nucleus and finally translated into the proteins of cell cytoplasm. Sequence of amino acids present in DNA carries the set of instructions and later on various biochemical reactions living organisms are created. Creation process is followed by assemble and test. The top down process is shown as small sub regions in large available space. Assumptions are required while using parts within the space. The use of assemble and test along with evolutionary algorithm [1] may utilize the entire design space and here larger space are utilized due to the absence of rules involved in design process. The quality of design is improved largely by the use of assemble and test concept along with the evolutionary algorithm and the design is made in the Evolvable Hardware [2-6] field and the task is to design an electronic circuit.

2. MOTIVATION

The design of digital [7] circuits is based on rules and principles to create a large and efficient electronic circuit. The intrinsic digital circuit is based on evolutionary process where testing of hardware[8] takes place and extrinsic digital circuits are implemented entirely over the software. The main disadvantage is

* Research Scholar, Department of Electronics and Communication Engineering, St. Peter's University, Avadi, Chennai, India, *Email: vpknathan@gmail.com*

** Dean and Professor Department of ECE, Vel Tech High Tech Dr. Rangarajan Dr. Sakunthala Engg. Coll., Chennai, India, *Email: shawn_pooja2003@yahoo.co.in*

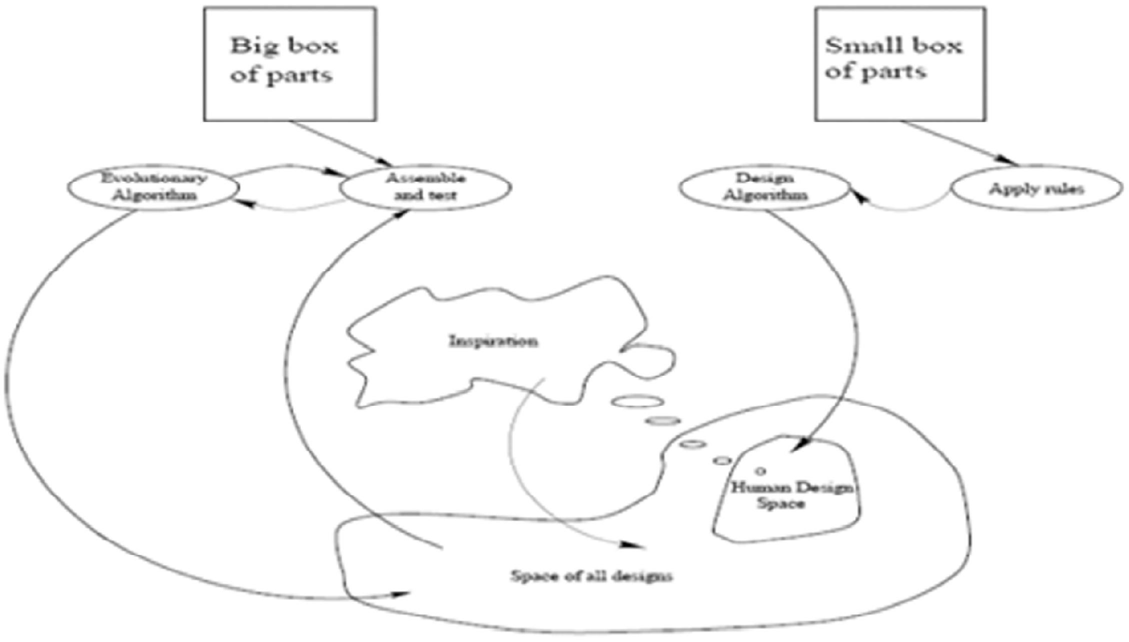

Figure 1: Conventional design versus evolutionary design with assembles and test

the problem of scale. The number of logic gates increase as the logic function increases [9-10]. The search becomes difficult with evolutionary techniques. The drawback is the time for calculating the fitness and it is shown in fig. 1 of circuit. Thus the size of truth table of circuit increases. To solve the scalability problem is a challenging task and its applicability is discussed here.

3. PROBLEM STATEMENT

Here we are going to identify how to create large circuits using evolutionary algorithm and the solution is to clear the problem of scalability in evolutionary design of digital circuits. A new methodology is applied for designing large digital arithmetic circuits. The design is made on digital domain but it can be extended in analog domain. The tasks involved are in two phases.

1. A set of digital components are used as blocks in the design of digital circuits which are larger in size. And blocks are bit sliced in nature.
2. Library of components are used to design large circuits and also genetic algorithm [11-13] are involved. The problems are shown through chromosomal representation and genetic operators are also customized.

4. APPROACH

4.1. Task 1

The task1 involves the library components to be developed which are later used as building blocks of larger circuits. The approach used here is discussed here.

1. The digital circuits are identified and studied and later used as building blocks for the design of larger circuits. Thus we have designed scaled digital circuits.
2. After finding the building blocks the next step is to design an electronic circuit with two input gates and two input multiplexers.

The final step is to create a set of components. These libraries of components are used for the generation of larger circuits.

4.2. Task 2

Here we use the library of evolved components as building blocks and the designed electronic circuit rather than two input gates.

The approach is outlined here:

1. Here we just create a representation for electronic circuit using chromosomal approach and is evolved as the chromosomal genes.
2. Next step is to develop appropriate genetic operators for the above represented chromosomal approach.

4.3. Task 3

Then the evolutionary algorithms are implemented by using operators and finally it is concluded that the evolutionary design of circuit is easier when compared to original one where gates are the building blocks.

5. SUMMARY OF RESULTS

The design of larger digital arithmetic circuits is enhanced by using a set of library components in the design of a circuit. These library components set are used along with two input gates and multiplexers. The next step is the use of evolutionary algorithm for designing larger circuits by the use of library sets. Then comes the representation of circuit using chromosomal approach and evolutionary algorithm are used for generation of genetic operators. The library set of components used for generating larger circuits becomes infeasible due to the use of two input gates and multiplexers.

6. EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms (EAs) are a broad class of stochastic optimization algorithms [14], inspired by Biology and in particular by those biological processes that allow populations of organisms to adapt to their surrounding environment: genetic inheritance and survival of the fittest. Charles Darwin[15] introduced this concept during 19th century and till today it has been widely accepted even though there may some complementary decisions. At first John Holland and Lawrence Fogel and his colleagues made a work which was based on the strategies for human evaluation.

7. THE EVOLUTIONARY CYCLE

The evolutionary cycle is based on the evolutionary algorithm and this algorithm is based on picking up a population of randomly generated individuals even though we can make use of previously saved populations. Also some heuristic algorithms may be used for this cycle. Bit strings that are generated randomly may be used for initial population. After creating this initial population the next process is the entry to a loop. By the use of some operators over the old population that are generated we get a new set of population. One such iteration is referred to as generation. And it is shown in fig. 2.

8. APPROACH TO THE EVOLUTION OF LARGE DIGITAL ARITHMETIC CIRCUITS

8.1. Scalability Problems of digital circuit evolution

Here the goal is to design a large and efficient electronic circuit by using some general set of rules and principles. Intrinsic and extrinsic evolution of digital arithmetic circuits takes place here. The intrinsic circuit is based on hardware testing and building of circuits whereas extrinsic is implemented purely on the software. The disadvantage here seen is the scalability of the circuits.

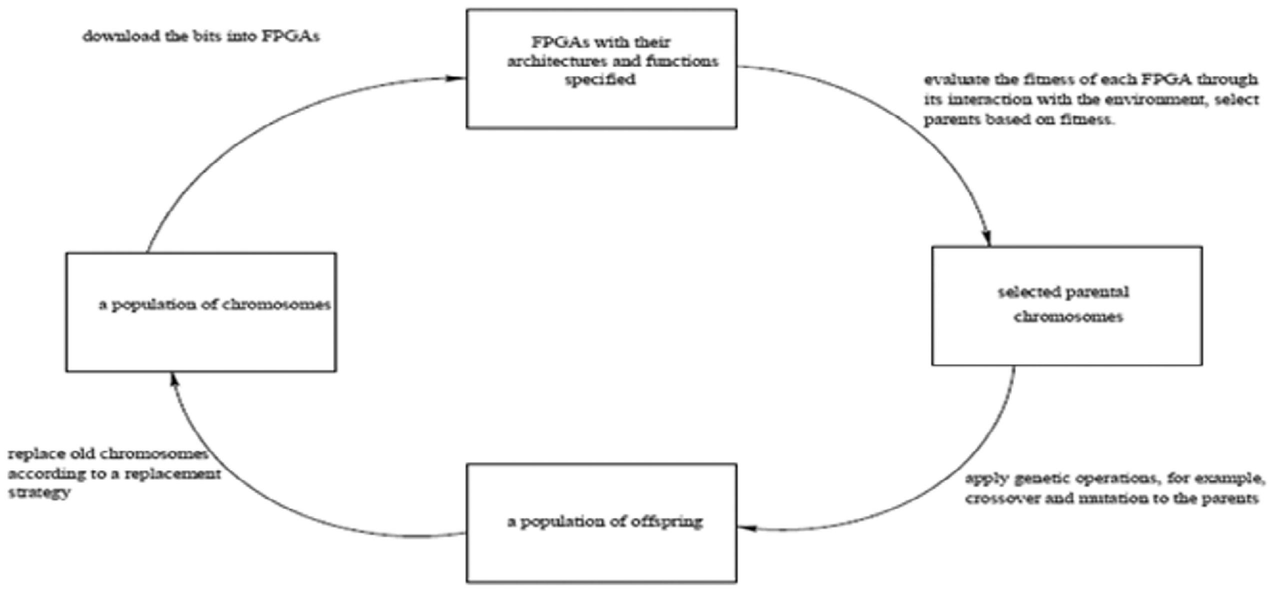
download the bits into FPGAs
FPGAs with their architectures and functions specified
evaluate the fitness of each FPGA through its interaction with the environment, select parents based on fitness.
selected parental chromosomes
apply genetic operations, for example, crossover and mutation to the parents
a population of offspring
replace old chromosomes according to a replacement strategy
a population of chromosomes

Figure 2: Major steps in an evolutionary cycle of evolvable hardware (for example, FPGAs).

As the number of gates used in the design of the circuit increases the logic function involved also increases. This results in large amount of evolutionary space for search even if we use good evolutionary techniques. Also the problem of scale incurs a disadvantage of time calculating for the fitness of circuits. This time is directly proportional to the truth table of the circuit designed.

The circuit can be designed effectively by breaking into smaller modules or sub circuits and this is referred as building blocks. This relates to the concept of automatically defined functions studied in this thesis studies the evolutionary design of combinational circuits in which the basic building blocks are small digital components inferred from the conventional digital design methodology for larger circuits.

Here we design small circuit first and from which larger circuits are designed by identifying the building blocks for the circuit required. Then we have to apply the evolutionary algorithm to the designed circuit.

8.2. Evolution of the Digital Modules

8.2.1. Encoding a Digital Circuit as an Indexed Graph

The encoding of a digital combinational circuit into a genotype treats a digital logic circuit as a particular case of a more general graph based computational model called Cartesian Genetic Programming[16] (CGP). Programs are considered as an array of nodes on CGP[17]. This approach has some similarities with other graph based genetic programming.

Here some instance of program is seen as a circuit and they perform some binary data based tasks. A program may be represented as an array of nodes in CGP.

The nodes represent any operation on the data seen as its inputs. Programming constructs (if, switch, OR, * etc.) may be implemented on each node. All the inputs, whether primary data, node inputs, node outputs, and program outputs are sequentially indexed by integers. Indexing applies on the node functions eventually.

We apply the parameters over the output evolved. The cells and outputs are maximally connectable when the number of rows is one and levels-back is equal to the number of columns. Each cell is connected to its neighbors on left when number of rows is one and levels-back is one. Cells within any particular column cannot be connected together. All cells having two inputs and one output use a particular form of CGP and all connections over the cell are feed-forward. In general CGP the cells may have multiple inputs

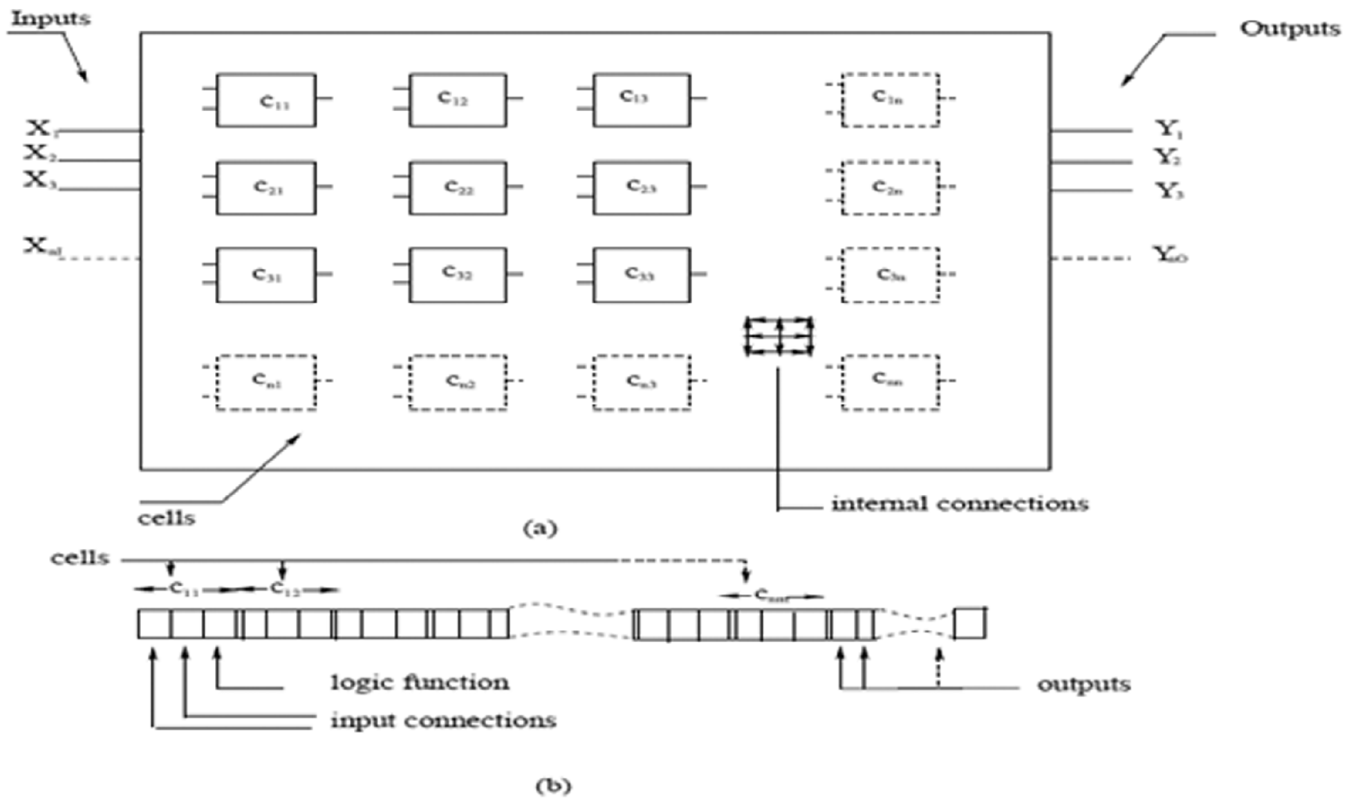

Figure 3: The genotype-phenotype mapping: (a) a n _ m geometry of logic cells with nI inputs and nO outputs, and (b) the genotype structure of the array.

Table 1
Allowed cell functions

Letter	*Function*
1.	0
2.	$a \cdot b$
3.	$a \cdot \overline{b}$
4.	a
5.	$\overline{a} \cdot b$
6.	b
7.	$a \oplus b$
8.	$a + b$
9.	$\overline{a} + \overline{b}$
10.	$\overline{a} \oplus b$
11.	$\overline{b}$
12.	$a + \overline{b}$
13.	$\overline{a}$
14.	$\overline{a} + b$
15.	$\overline{a} \cdot \overline{b}$
16.	1

and outputs and the numbers of these would be encoded into the genotype for the cell. Clocked inputs may be recognized for general primary outputs thus allowing the CGP programs to possess internal states. The genotype and the mapping process of genotype to phenotype are illustrated in Figure 3. The nI primary circuit inputs $X1$; $X2$; _ _ _ ; XnI are allowed to be connected to the input of any cell or any of the nO primary circuit outputs $Y1$; $Y2$; _ _ _ ; YnO. The cells cij may implement the functionality of any two input logic gate, or, alternatively a 2-1 multiplexer (MUX) with single control input.

The allowed two input functions are as shown in the table 1

9. RESULTS

The resulting decision is that the circuits are designed to provide scalability.

The Table 2 represents the number of generations required, the time taken for the evolution of the circuits discussed above. A very large number of generations and time required for evolutionary algorithm was exponential when tried to evolve the circuits in the above table

Also, the evolutionary algorithm did not converge for some circuits for the specified number of generations (109). For example, the three bit adder, the number of generations using the two input cells as building blocks required was 629540. The number of generations using the library of digital modules as the building blocks was 55291. In terms of number of generations required to evolve the circuit, the scaled evolutionary design appeared to be approximately 12 times more efficient.

Even for a scaled evolution of designed circuit the improvement is significant. Note that the building blocks are bigger than the two input cells. They may have two inputs and one output, while the cells in the scaled scenario have four inputs and four outputs.

Possible reasons for this significant improvement are the following: firstly, the building blocks are chosen which are most extensively used in the conventional design of combinational circuits, and secondly, they allow the reuse of gates from inside the modules.

10. CONCLUSIONS

Here we have used evolutionary algorithm to the space required for the design of the circuit and to improve the scalability of the circuit. This algorithm has been implemented in the arithmetic circuit which is considered to be one of the digital circuits. Here we built a large circuit from smaller building blocks. To implement a

Table 2
Number of Generations and Time taken for Convergence

Circuit	*Number of Generation*	*Time taken for Convergence*
2-bit adder	1177	4 min 22 sec
3-bit adder	55291	4 hrs 41 min 21 sec
4-bit adder	94444	10 hrs 23 min 22 sec
2-bit subtractor	1552	7 min 18 sec
3-bit subtractor	91725	11 hrs 34 min 53 sec
4-bit subtractor	8424	22 min 34 sec
3-bit comparator	127035	32 hrs 21 min 56 sec
3-bit decoder	22321	8 hrs 22 min 55 sec
4-bit decoder	204052	43 hrs 48 min 19 sec
A + B – C circuit	274155	49 hrs 39 min 12 sec
A + B + C – D – E circuit	371852	47 hrs 31 min 38 sec

larger circuit we use smaller circuits which reduce the scalability. Since we are using smaller blocks it is easy to evolve larger circuits. Evolution of such large circuits would have never been possible with two input gates as the basic building blocks. This implies that the principle of evolving digital circuits is scalable.

REFFERENCES

[1] Keymuelen D. and Kuniyoshi Y. and Higuchi T. Durantez M. An evolutionary robot navigation system using gate-level evolvable hardware. Lecture Notes in Computer Science, 1259:195 {209, 1997.

[2] Julian F. Miller, Dominic Job, and Vesselin K. Vassilev. Principles in the evolutionary design of digital circuits-i. Journal of Genetic Programming and Evolvable Machines, 1:8 {35, 2000.

[3] Murakawa M., Yoshizawa S., and Higuchi T. Adaptive equalization of digital communication channels using evolvable hardware. Lecture Notes in Computer Science, 1259:470 {481, 1997.

[4] Salami M., Sakanashi H., Tanaka M., Iwata M., Kurita T., and Higuchi T. Online compression of high precision printer images by evolvable hardware. Proceedings of Data Compression Conference (DCC98), pages 219 {228, 1998.

[5] Salami M., Murakawa M., and Higuchi T. Lossy image compression by evolvable hardware. Proceedings of IJCAI-97 Workshop on Evolvable Systems, pages 53 {59, 1997.

[6] Daniel Mange, Moshe Sipper, and Edurado Sanchez. Quo vadis evolvable hardware. Communications of the ACM, pages 50 {56, 1999.

[7] Mange D. Wetware as a bridge between computer engineering and biology. Preliminary Proceedings of the 2nd European Conference on Artificial Life (ECAL93), pages 658 {667, 1993.

[8] Murukawa M., Yoshizawa S., and Kajitani I. Hardware evolution at function level. Lecture Notes in Computer Science, 1141, 1996.

[9] Marchal P., Piguet C., and Mange D. Embryological development in silicon. Artificial Life IV, pages 365 {370, 1994.

[10] Marchal P., Nussbaum P., and Piguet C. Embryonics: The birth of synthetic life. Lecture Notes in Computer Science, 1062:166 {196, 1996.

[11] de Garis H. Evolvable hardware: Genetic programming of the darwin machine. Arti_cial Neural Nets and Genetic Algorithms: Proceedings of the Internationl Conference in Innsbruck, Austria, pages 441 {449, 1993.

[12] Takanashi E., Murakawa M., Toda K., and Higuchi T. An evolvable hardware based clock timing architecture towards gigahz digital systems. Proceedings of the Genetic and Evolutionary Computation Conference, 1999.

[13] Julian F. Miller, Dominic Job, and Vesselin K. Vassilev. Principles in the evolutionary design of digital circuits-i. Journal of Genetic Programming and Evolvable Machines, 1:8 {35, 2000.

[14] de Garis H. Evolvable hardware: Genetic programming of the darwin machine. Arti_cial Neural Nets and Genetic Algorithms: Proceedings of the Internationl Conference in Innsbruck, Austria, pages 441 {449, 1993.

[15] Takanashi E., Murakawa M., Toda K., and Higuchi T. An evolvable hardware based clock timing architecture towards gigahz digital systems. Proceedings of the Genetic and Evolutionary Computation Conference, 1999.

[16] Mange D. Wetware as a bridge between computer engineering and biology. Preliminary Proceedings of the 2nd European Conference on Artificial Life (ECAL93), pages 658 {667, 1993.

[17] Mange D., Staufier A., and Sanchez E. Designing programmable circuits with biological-like properties. Annales du Groupe CARNAC, EPFL et UNIL, Lausanne, 6:53 {71, 1993.