Improved Generalized Disjunction Decomposition for Circuit Evolutionary Design

Yong Lin[1 *]

[1] School of Medical Instrument and Food Engineering, University of Shanghai for Science and Technology, Shanghai 200090, China

Keywords: Evolvable Hardware; Evolutionary Design; Scalability; Short Link Priority; Generalized Disjunction Decomposition

Abstract. Evolvable Hardware can be used to design circuits automatically. The main difficulty of applying real-world applications is its scalability. This paper focuses on improving the scalability of combinational logic circuit evolutionary design. We proposed a complex circuit design method combined Generalized Disjunction Decomposition (GDD) with Short Link Priority mutation evolutionary strategy. The proposed method can evolve complex circuit and use fewer logic gate compared with GDD. The experimental results validated the performance of our method.

Introduction

Evolvable Hardware (EHW) [1] is a novel technique introduced to design circuits, which can evolve circuits only supplying the demand function and without human intervention. It designs the configuration of the programmable hardware under the control of evolution algorithm. Using EHW, researchers have designed many circuits such as filters [2], multiplier [3], finite state machine[4] and so on. EHW can even explore some designs unexpected by human being [5,6].

However, there are few real-world applications developed by EHW. This is mainly due to its limitation of scalability[7-9]. The evolution design of logic circuits is based on the true table. When the inputs increases, the number of input-output combinations increases drastically. In addition, larger logic cell array is needed to evolve the circuits, which lead to the increasing of chromosome length and search space. So to improve the scalability problem, some researchers focused on decrease input-output combination. They have proposed many methods such as function level evolution [10], divide-and-conquer [11], incremental evolution [11,12]. Divide-and-conquer and incremental evolution methods decompose a complex system into many simpler sub-systems, evolve these sub-systems and then assemble them. While some researchers focused on decreasing the computation complexity, they proposed the variable length chromosome [13] and dynamic mutation rate [14], and so on. In fact, if we decrease the computation complexity of the circuit evolution, and combine with the divide and conquer method, it will be effective to improve the scalability problem.

Our research in this paper focuses on combinations of these two methods. The SLP (Short Link Priority) mutation Evolutionary Strategy (ES) we proposed in [15] was introduced in Generalized Disjunction Decomposition (GDD) to accelerate the evolutionary design. The Circuits we design are FPGA (Field Programmable gate Array)-based circuits and the design method are based on CGP (Cartesian Genetic Programming) [3, 14]. During the evolution of the cell array using CGP, SLP-Mutation ES is able to decrease the search space and accelerate the generation of full functional circuit, but it can't evolve more complex circuits. Here, a design method combine with GDD (Generalized Disjunction Decomposition) [8] and SLP-Mutation ES are proposed, which aimed to design complex circuit and improve the scalability problem.

Generalized Disjunction Decomposition combined with SLP-Mutation Evolutionary Strategy

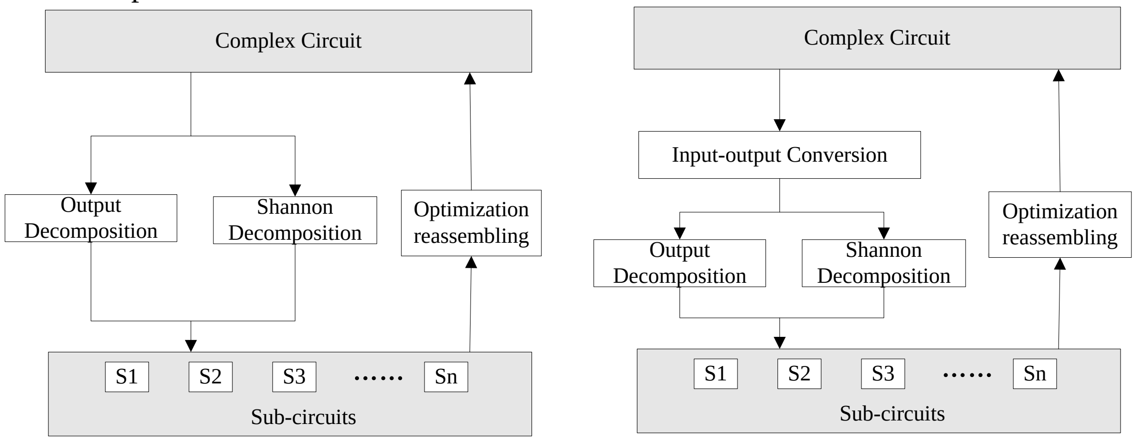
As mentioned above, SLP-Mutation ES can't evolve more complex circuits. Here, a design method combine with GDD and SLP-Mutation ES are proposed, which aimed to design complex circuit and improve the scalability problem.

GDD in Circuit Design. GDD proposed in [8] is a novel method to evolutionary design complex circuits, which has successfully design several combinational circuits that never previously evolved[12]. GDD combines a pretreatment of new input-output conversion with BIE (Bi-direction Incremental Evolution) design method. It is based on the statement that:

(1) The number of generations required to successfully evolve logic circuit is mainly dependant on the number of inputs instead of number of outputs [8].

(2) The decomposition of a complex system into small ones in BIE is done by using output decomposition and Shannon decomposition [12].

The input-output conversion is aimed to reduce input number and increase output number. The conversion detail is described in [8]. The design flow of GDD and BIE is shown in figure 1. The principle of BIE design method is to divide a complex circuit into simple sub-circuits using output decomposition and Shannon decomposition. Then evolve each of these sub-circuits and merge incrementally these sub-circuits, reassembling a new evolved complex circuit. During the reassembling procedure, each sub-circuit is optimized to reduce the gate it uses. Output decomposition applies to the situation where a part of the outputs can be fully evolved. If Output decomposition does not work well, Shannon decomposition is applied. Shannon decomposition reduces one input number and divides the evolved circuit into two sub-circuits. When emerging these two sub-circuits, a multiplexer is needed and the reduced one bit became the selection signal of the multiplexer.



（a） BIE Design Method （b） GDD Design method

Fig1. Design flow of GDD and BIE [8]

GDD method divides a complex circuit into many sub-circuits, evolves them and then merges them into the required circuit. Every sub-circuit's logic is very simple, which needs not many gates to realize it. But there are a lot of the sub-circuits, so the total gate number is very large. Furthermore, it required many multiplexers during the merging procedure. The cost of gate number for multiplexers is also very large. For the input-output conversion before BIE, it converses an n-input m-output circuit into a (n-i)-input ($m \cdot 2^i$)-output circuit. The number of logic gate for the conversion generic multiplexer [8] is calculated as

$$N_{Coversion\_mutiplexer} = 4m \cdot (2^i - 1) \tag{1}$$

For the Shannon decomposition, it needs a multiplexer to merge the two divided sub-circuits. The logic gate number of merging two (n-1)-input m-output sub-circuits into one n-input m-output circuit is calculated as

$$N_{Shannon\_multiplexer} = 3 \cdot m + 1 \tag{2}$$

GDD combined with SLP-Mutation ES. Though GDD method can design complex circuit and improve the scalability problem of circuit evolutionary design, it also exist some deficiencies. In order to facilitate the description, “stalling effect” mentioned in [8] is explained here. It means that the fitness has not improved for a lot of continuous generation. The deficiencies are as follows.

(1) Using of decomposition only depends on the emergence of “stalling effect” [8] defined the emergence of “stalling effect” as fitness not having improved for 2000 generations in the experiments). The result is that the size of divided sub-circuit becomes too small, the number of sub-circuit becomes too much, and the number of logic gate for realizing these sub-circuits becomes too large, though we have optimized the cost of gate number during the reassembling procedure.

(2) Because there are too many sub-circuits, the number of corresponding multiplexer is also very large, which also increases the cost of logic gate.

(3) The small size of sub-circuit is not conducive to the creativity of the evolution design.

Therefore a design method to improve the GDD method is proposed in this paper. We set a threshold too control the size of sub-circuit being decomposition and introduce the SLP-Mutation ES into the evolving of the sub-circuits. When the decomposition circuit size reaches the threshold, the evolution of sub-circuit does not stop with the emergence of the “stalling effect”, but is allowed to evolve continuously in a maximum generation. If the circuit is not fully evolved in maximum generation, we make the sub-circuit output decomposition. Because SLP-Mutation ES can accelerate the fully evolution of relatively complex circuit, it is conducive to evolve the required sub-circuit successfully. Limitation the size of sub-circuit being decomposition can reduce the number of sub-circuit and multiplexer, so the cost of logic gate will correspondingly reduce. In addition, limitation the size of sub-circuit benefits the creativity of the evolution design.

Experimental results and analyses

In this section some multipliers and combinational circuits from MCNC benchmark [16], which is often used as the test bench for circuit evolutionary design, are evolved by using the proposed design method. Each circuit is evolved by three methods. They all simulate the GDD method mention in [8].The difference between these method and GDD includes the chromosome presentation, the fitness values calculation method and some parameters of ES. Chromosome presentation method is the same as the method in [3]. The parameters of (1+λ)ES are set as: λ=4, population size is 50, Number of runs per each experiment is 10 and mutation rate is 0.06. Fitness calculation encourages the fully evolved output and then makes output decomposition. If there is less output fully evolved, we make Shannon decomposition. Three methods are described as follows.

(1) nl_STD_GDD: Doing decomposition only depends on the emergence of “stalling effect”.

(2) l_STD_GDD: This method sets a threshold to limit the size of evolved sub-circuit and uses standard evolution strategy.

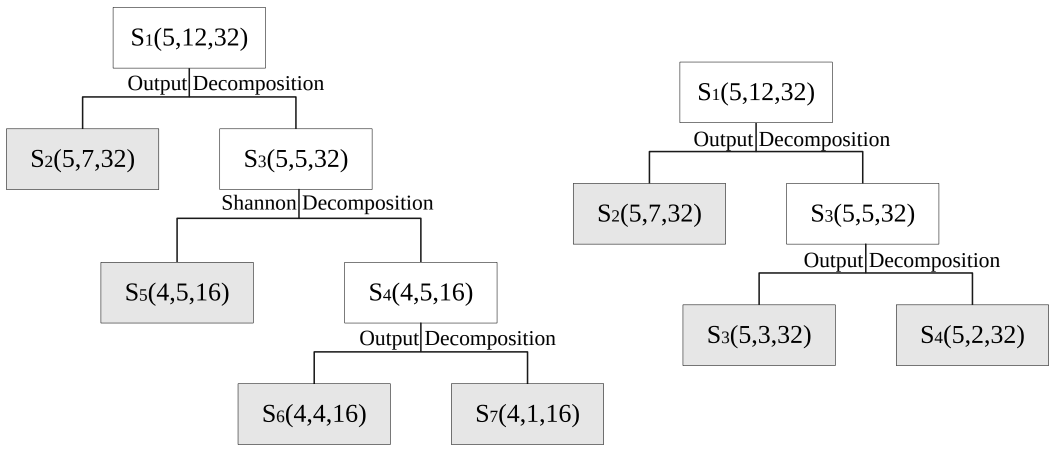
(3) l_SLP_GDD: This method sets a threshold to limit the size of evolved sub-circuit, but it uses SLP-Mutation ES.

The experimental results are shown in table 1, where NSc is the number of sub-circuits, NMp is the number of multiplexer, and TG is total number of generation. During the dividing procedure, Maximum number of generation to evolve the sub-circuits is set 1,000,000, the size of cell array is 12*6, the threshold to limit the size of evolved sub-circuit is 5-input 3-output, and the sub-circuits no more complex than the threshold are evolved in 50,000 generation. After the sub-circuit is successfully evolved, the optimization mentioned in [12] is adopted to reduce the cost of logic gate, and the data in the bracket of “gate using” column in table 1 is the number of gate using after the optimization.

In table 1, l_SLP_GDD and l_STD_GDD method both use more generation to evolve the circuit than nl_STD_GDD method, but they cost much fewer gates because they limit the size of sub-circuit and need fewer sub-circuits and multiplexers. Compared with the l_STD_GDD, l_SLP_GDD evolves the circuits using less generation, and the numbers of using gate of these two methods are very close. So the proposed method in this paper, l_SLP_GDD, is conducive to evolve the circuit using less gate efficiently. In addition, the sub-circuit generated by l_SLP_GDD is relatively complex, it will conducive to the creativity of evolutionary design. Figure 2 is the comparison of evolving 3-3 multiplier procedure between nl-STD-GDD and l_SLP_GDD.

Tab.1 Comparison of evolution design methods: nl_STD_GDD, l_STD_GDD and l_SLP_GDD

Circuits Infomation				Evolution Result Information				
Circuit	in	out	p	Method	NSc	NMp	Gate Using	TG
3-3mul	6	6	64	nl_STD_GDD	4	2	129(118)	9523
				l_STD_GDD	3	1	80(76)	29421
				l_SLP_GDD	3	1	84(78)	21647
4-4mul	8	8	256	nl_STD_GDD	29	13	827 (730)	83872
				l_STD_GDD	27	7	685(633)	676496
				l_SLP_GDD	25	5	671(625)	584522
5-5mul	10	10	1024	nl_STD_GDD	125	62	3569(3026)	475766
				l_STD_GDD	120	42	3147(2733)	4154682
				l_SLP_GDD	113	31	3040(2690)	3753358
6-6mul	12	12	4096	nl_STD_GDD	738	332	22267 (19367)	2351678
				l_STD_GDD	691	161	20421(17247)	18347726
				l_SLP_GDD	665	132	19378(16944)	16734599
9sym	9	1	512	nl_STD_GDD	10	6	315 (267)	41025
				l_STD_GDD	8	3	171 (161)	94844
				l_SLP_GDD	7	2	185(170)	85221
rd84	8	4	256	nl_STD_GDD	17	8	561 (497)	62096
				l_STD_GDD	16	5	433 (390)	327340
				l_SLP_GDD	15	3	416(381)	284139
misex1	8	7	256	nl_STD_GDD	9	4	231(201)	8628
				l_STD_GDD	8	4	215(191)	8571
				l_SLP_GDD	8	4	227(198)	8405
5xp1	7	10	128	nl_STD_GDD	20	10	644(573)	50528
				l_STD_GDD	18	6	448(391)	424125
				l_SLP_GDD	18	4	429(388)	365273



Evolution procedure of 3-3 multiplier using nl_STD_GDD

Evolution procedure of 3-3 multiplier using l_SLP_GDD

Fig.2 Comparison of evolving 3-3 multiplier procedure between nl_STD_GDD and l_SLP_GDD

Summary

In this paper, After analyzing the GDD method and its deficiencies, we proposed a complex circuit design method based on GDD to improve the scalability of circuit evolution design. Setting a threshold and introducing the SLP-Mutation ES in the GDD make the reduction of using logic gate, though it cost more generation. The proposed method is also conducive to the creativity of evolutionary design.

References

[1] X. Yao and T. Higuchi, "Promises and Challenges of Evolvable Hardware," IEEE Transactions on Systems, Man, and Cybernetics, Part C, 29(1998)87-97.

[2] J. R. Koza, F. H. B. III, D. Andre, and M. A. Keane, "Four Problems for Which a Computer Program Evolved by Genetic Programming is Competitive with Human Performance," in the 1996 IEEE Int. Conf. on Evolutionary Computation (ICEC'96), Piscataway, NJ, USA, 1996.

[3] J. F. Miller, D. Job, and V. K. Vassile, "Principles in the Evolutionary Design of Digital Circuits- Part I," Genetic Programming and Evolvable Machines, vol. 1(1), pp. 8-35, 1999.

[4] T. Higuchi, H. Iba, and B. Manderick, "Evolvable Hardware," Massively Parallel Artificial Intelligence, pp. 398-421, 1994.

[5] T. Higuchi, "Evolvable Hardware and it's Application to Pattern Recognition and Fault-Tolerant Systems," in Toward Evlovable hardware: The evolutionary Engineering approach, 1996, pp. 118-135.

[6] T. Higuchi, T. Niwa, and T. Tanaka, "Evolvable Hardware with Genetic Learning," in Simulation of Adaptive Behacior, 1993, pp. 417-424.

[7] T. G. W. Gordon and P. J. Peter, "Development Brings Scalability to Hardware Evolution," in Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware, Washington DC,USA, 2005, pp. 272-279.

[8] E. Stomeo, T. Kalganova, and C. Lambert, "Generalized Disjunction Decomposition for Evolvable Hardware," IEEE Transactions on Systems, Man and Cybernetics, Part B pp. 1024-1043, 2006.

[9] V. K. Vassilev and J. F. Miller, "Scalability Problems of Digital Circuit Evolution Evolvability and Efficient Designs," in The Second NASA/DoD Workshop on Evolvable Hardware, Palo Alto, CA, USA, 2000, pp. 55-64.

[10] T. Higuchi, "Evlovable Hardware at Function Level," in 1997 IEEE Int.Conf.Evolutionary Computat.(ICEC'97), 1997, pp. 187-192.

[11] J. Torresen, "A Divide-and-Conquer Approach to Evolvable Hardware," in the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), 1998, pp. 57-65.

[12] T. Kalganova, "Bidirectional Incremental Evolution in Evolvable Hardware," in the Second NASA/DoD Workshop on Evolvable Hardware, Palo Alto, California, USA, 2000, pp. 65-74.

[13] M. Iwata, I. Kajitani, H. Yamada, and T. Higuchi, "A Pattern Recognition System Using Evolvable Hardware," in Parallel Problem Solving from Nature Ⅳ, Berlin,Germany, 1996, pp. 761-770.

[14] E. Stomeo, T. Kalganova, and C. Lambert, "A Novel Genetic Algorithm for Evolvable Hardware " in the 2006 IEEE Congress on Evolutionary Computation, Canada, 2006, pp. 441--448.

[15] Y. Lin, " A Novel Mutation Strategy to Accelerate Evolutionary Design of Circuits " in the ICNIS 2008, China, 2008, pp. 358-362.

[16] S. Yang, "Logic Synthesis and Optimization. Benchmarks, Version 3.0," Microelectronics Center of North Carolina 1991.