Evolutionary Design of Approximate Circuits

Amad Ul Hassen
College of Engineering and Computer Science
University of Central Florida
Orlando, USA
amad@knights.ucf.edu

Abstract **In this paper we have surveyed recent approaches for design of approximate circuits using Evolvable Hardware. We have discussed some challenges faced during evolution of digital circuits. Then we have proposed spatially varying mutation and cross over rates as possible remedy and discussed its superiority on uniform mutation and crossover rate for digital circuits.**

Keywords**—Evolvable hardware, approximate circuits, Cartesian genetic programming, power efficiency multi objective evolutionary optimization.**

I. Introduction

With the boom of hand held battery powered devices, there is ever increasing demand for power efficient computing solutions. This has created need for exploring new areas for energy efficient designs. For example, Carrol [1] has done detailed analysis of power consumption of different components in mobile phones, and found that multimedia applications are the biggest strain on battery of mobile phones. Fortunately, because of limitations of human perception, multimedia applications can tolerate inaccuracies without adversely affecting performance. Such fault tolerant applications are the biggest beneficiary of approximate computing.

Although evolvable hardware techniques have been used to successfully design some circuits [2], [3], but their performance suffers from very long convergence time for complex digital circuits such as multiplier. Despite this, EHW have extensive potential in design of approximate circuits because in approximate circuit design (i) requirements of ideal performance is relaxed (ii) evolution is able to search for novel designs which is not possible with conventional design approaches (iii) they can search for solutions both at functional level and gate level (iv) EHW approaches iteratively search for better approximations of target circuits until it becomes acceptable.

The rest of paper is organized as follows; first we describe some non evolutionary approaches to for approximate circuit design. In section III, we have first described Cartesian Genetic Programming, commonly used platform for evolutionary design of approximate circuit. In section IV, we have provided of an over view of most recent work in evolutionary design of approximate circuits. This includes different heuristic seeding for CGP, design of different fitness functions and their performance, design of ESD immune circuits using evolvable hardware and Multi Objective Evolutionary algorithm for design of approximate circuits. After comparison of recent approaches in section VI, we have concluded that MOEA is most promising approach for approximate circuit design. In section VII, we have discussed some of the challenges faced during evolution of digital circuits and how they can be solved using spatially variable mutation and cross over operators.

II. Related Work

Approximations can be introduced at any level e.g, circuit level, architecture level, operating system level. In the broad context, approximation methodologies can be divided into two categories (i) Evolutionary (ii) Non Evolutionary. Next we will briefly discuss these approaches.

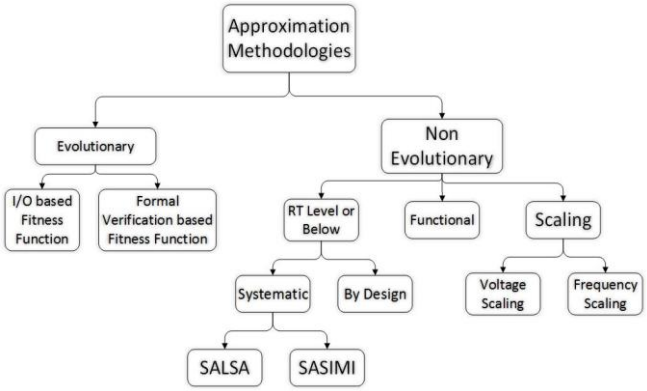


Figure 1: Taxonomy of different approaches for design of approximate circuits

A. *Non Evolutionary Approaches to Approximate Circuits*

1) Functional Approximation:

One approach for trading power with accuracy is implementation functional approximation instead of original function. Examples of such approach are truncating least significant bits in arithmetic operations, or decreasing the length of storage elements in FIR filters, and or replacing expensive computation with safe fixed constants (e.g step size in gradient algorithm).

Here are other approaches for circuit approximation which are independent of target applications

2) Voltage Over-scaling: Semiconductor devices need sufficient voltage difference between logic 0 and logic 1. Probability of error be reduced exponentially by increasing this gap.Conversely, we can save some power by operating our electronics at lower voltage at the expense of increased probability of erroneous result.

3) Frequency scaling: Just like voltage, error free operations of digital circuits is limited by the upper bound on clock frequency. In digital circuits, major of power dissipation occurs during transition between logic 1 and 0, we can reduce clock frequency to conserve power when our device will be in dormant state for long periods of time. Conversely, over clocking can be used to get quick results at reduced accuracy.

The combination of scaling the supply voltage and clock frequency is known as dynamic voltage scaling. Although voltage scaling and frequency scaling are attractive and easy to implement without additional modifications to original system, one drawback is that whole we cannot accomplish this on individual modules, therefore critical/error sensitive modules will be adversely affected by scaling, rendering the whole system useless.

We have seen shortcomings of circuit approximations obtained from functional approach and scaling.

To overcome these shortcomings, there are two more recent approaches (intermediate/synthesis level) of circuit approximations discussed below.

4) SALSA: The Systematic methodology for Automatic Logic Synthesis of Approximate circuits (SALSA) creates approximate circuits by modifying RT level description of fully functional circuits. It starts from original circuit, creates approximate version of circuit and checks if it satisfies error constraints, if it does then it goes ahead with further approximation otherwise, it reverts most recent modification and creates another approximation by modifying other parts of circuit.

5) SASIMI: Substitute-And-SIMplIfy (SASIMI), looks for signal pairs which have similar values with high probability and substitutes both with one of them thus creating an approximate version of the original circuit. Circuitry behind unused signal is removed thus result in power/area efficient version of circuit.

B. *Evolvable Hardware for approximate Circuit Design:*

Genetic algorithms are nature inspired universal optimizers. They have been successfully used to solve complex problems in several fields of science. Due to their ease of application, these algorithms have been successfully used to evolve digital circuits. Seminal work in this field was done by Thompson who evolved tone generator and tone discriminator on XC6216 FPGA using only 100 gates. Due to widespread availability of FPGAs (reconfigurable logic), their potential can be easily harnessed to accomplish digital circuit design. In next section, we have methodically discussed platform for evolution of digital circuits.

III. METHODOLOGY

A. Cartesian Genetic Programming

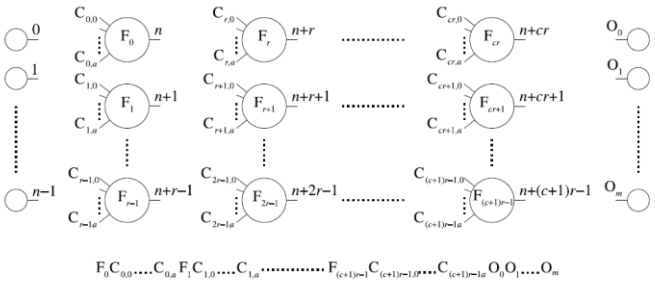
CPG was developed by Julian Miller [4] to evolved digital circuits. Basic structural flow of CGP is same any other generational genetic algorithm irrespective of the optimization objective.

It is a elitist algorithm where individuals are represented as a grid of processing elements as follows.

1) CGP representation

In CGP, genotype consists of a grid of processing elements as shown in figure [2]. There are n_r rows and n_c columns. Each processing element is encoded by a block of genes. One gene is used to represent function encoded by processing element. In addition one gene per input line is used for encoding addressing information.

For example, when these elements consist of simple two input gates, we need three genes for each processing element; two genes for encoding address information and one gene encoding the function (AND, OR, NAND etc) implemented by the gate.



General form of CGP. It is a grid of nodes whose functions are chosen from a set of primitive functions. The grid has n_c columns and n_r rows. The number of program inputs is n_i and the number of program outputs is n_o.

Figure 2: Chromosome Representation in CGP[Miller 2009]

Number of inputs is represented as n_i, number of outputs is represented as n_o. Connection scheme of CGP is a little

restrictive to facilitate faster evolution. The input of each gate can be connected either to the outputs of gates in the previous columns, or to circuit inputs. Only forward connections are allowed in CGP. Genotype has all nodes in the grid. Phenotype consists of only those nodes which affect the output column directly or indirectly. Floating or unconnected gates are not part of phenotype. Due to these differences in genotype and phenotype, circuits' size can be different for different individuals despite having genotypes of same sizes.

a) Generational Steps:

Seed initial population of (1+λ) individuals
Repeat until convergence

- Compute fitness of all individual
- Chose individual with highest fitness as parent.
- Create λ offspring by applying genetic operators to this parent.
- Select best (1+λ) for next generation
- Terminate search if fitness criteria is satisfied

2) Fitness Function

This is the most crucial element for good performance of algorithm. It is measure of how close the performance of individual circuit is to the ideal behavior. Every evolutionary algorithm tries to maximize/minimize (as required) this fitness function.

Unlike CGP representation, fitness function can vary from problem to problem. Poorly designed fitness function may adversely affect the performance of evolutionary algorithm by stagnating search or causing premature convergence to suboptimal circuit.

For approximate circuits, most commonly used fitness functions involve exhaustive input/output testing to measure discrepancy between outputs of evolved circuit and reference circuit,

For single output circuits, discrepancy based fitness function boils down to hamming distance between outputs of reference (t) and evolved circuits(y)

$$Fitness\ Value = \sum_{i=1}^{n} |y - t|$$

Here n is total number of inputs. y_i is total output of circuit being evolved. t_i is desired output.

For multi-output circuits, the generalized form of fitness error for one input is of the following form

$$Fitness\ Value = p_i\ |y_i - t_i|$$

Here p_i is the importance of i^{th} bit in output. When p_i is constant, above fitness function optimizes sum of hamming distance. For Absolute differences, p_i is exponential.

Incorporating problem specific information can improve performance of evolution e.g fitness function based on sum of absolute distance is superior than sum of hamming distance for circuits with arithmetic operands such as adders and multipliers.

3) *Seeding:*

Seeding refers to mechanisms for creating individuals in first generation. They can be generated randomly or by pruning functional circuits (approximations of fully functional circuits). There are advantages of either scheme. Random seeds can harness power of evolution more efficiently but they take very long time to converge. Hand seeded population can result in faster convergence but the solutions obtained are not as innovative as those obtained for random seeds. Due to these reasons, random seeding is avoided in evolution of complex circuits.

IV. RECENT APPROACHES

Using above described platform CGP, several circuits have been evolved using different fitness function. Here we discuss some of the most recent work on approximate circuits using EWH.

A. Comparison of seeding mechanism on evolution of multipliers and median filters

Sekenina and Vasciek's experiments [5] are the most comprehensive recent work on evolution of approximate circuits.

The authors has studied effects of different seeding mechanisms on the evolution of 2, 3, 4 bit multipliers and 9 – 25 input median filters under different area constraints. The goal of approximate evolution was to find power efficient versions of approximate circuits. Measuring power of each individual circuit is time consuming process.Fortunately there is positive correlation between area and power consumption of circuits, therefore author has minimized area of evolved circuit. This is based on the assumption of positive correlation between area and power. Author has also provided evidence for the validity of this assumption.

Circuit evolution with random seeding takes very long for convergence. To improve convergence time, author has introduced two heuristic mechanisms for generating seeds.

1) Heuristic Seeding Mechanisms

Author has compared the performance of evolution for following three seeding mechanisms.

a) Random Seeding:

In random seeding, initial population was created randomly.

b) HS1

In HS1, individual seeds of size (m-1) are created by from optimized circuits of size m. A gate is removed from the circuit of size m to obtain circuit of size m-1.

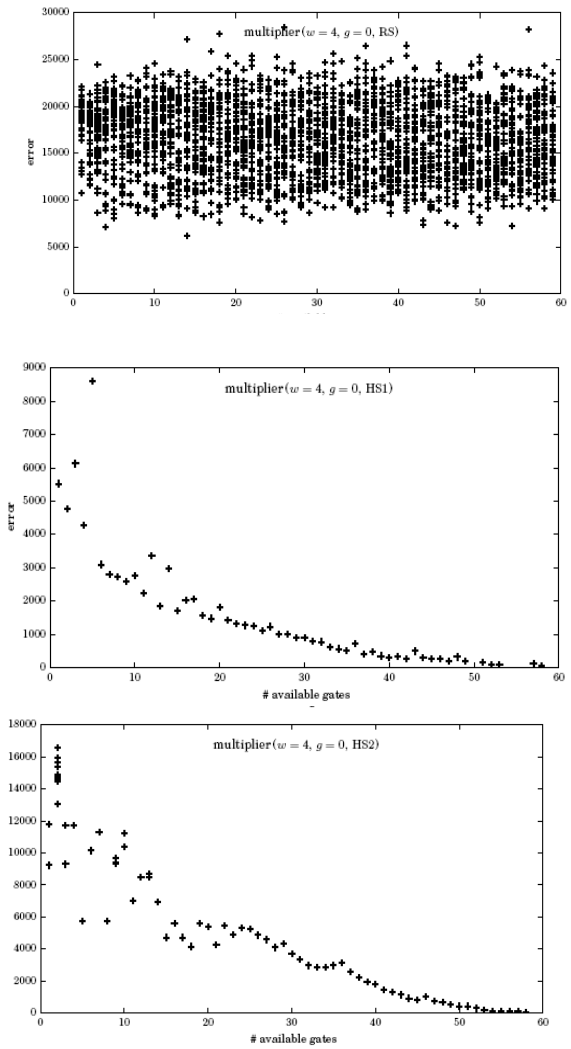


Figure 3: Error vs Gates performance of RS, HS1, HS2 at g = 0, [Vasicek 2014]

The input of removed gate is connected to its output and its fitness is computed. Same process is done for the other gate. This process is repeated for all gates in the circuit. The circuit with best fitness is chosen as parent for next generation**.**

c) HS2

Gate removal mechanism for HS2 is similar to HS1. But in HS2, approximations of successively smaller sizes (m-1,m-2,m-3 …..) are created from the same fully functional circuit. By applying the gate removal process on iteratively. This is different from HS1 because HS1 is to the circuit obtained from optimization to create successively smaller seeds, while HS2 creates seeds of all sizes before starting optimization process.

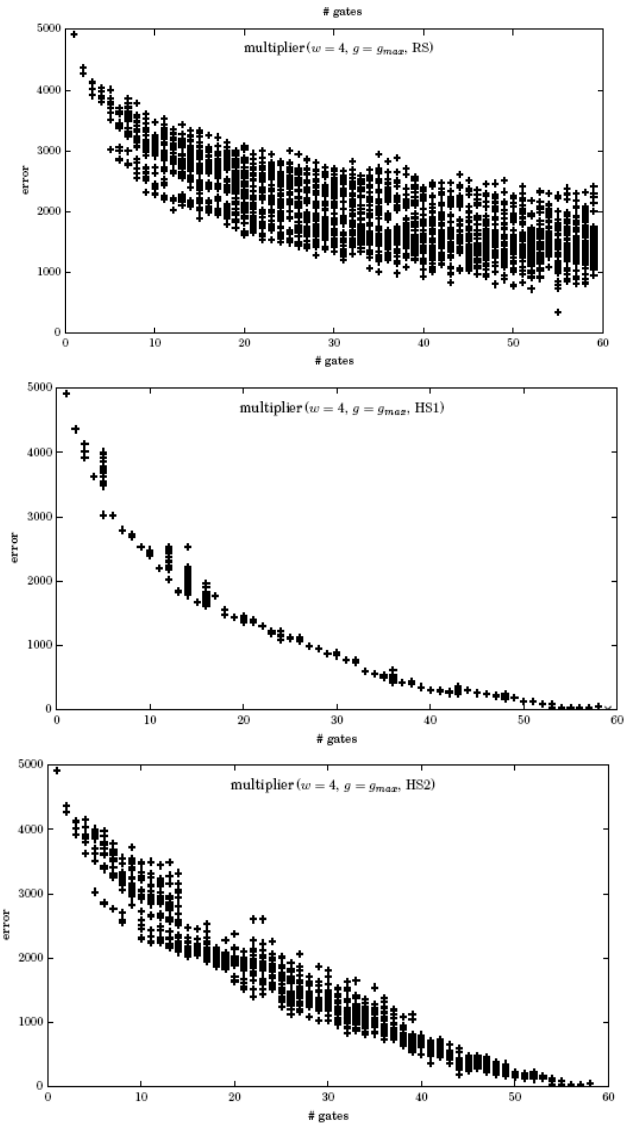


Figure 4: Error vs. Gates for RS, HS1, HS2 at g = g_{max} [Vasicek 2014]

1) Experimental Results

Author has tried to evolve multiplier circuits for 50 runs using three seeding mechanisms. Random seeding performed worst. Among HS1 outperformed HS2 as is clear from figure 5, the reason for superiority of HS1 is because smaller circuit is created from previously optimized circuit. Therefore, evolutionary search is not thrown off the cliff as algorithm moves to circuits of successively smaller sizes. While for HS2, seeds of all sizes are created before starting evolution, therefore search is interrupted as we circuit becomes smaller. This behavior is also clear by greater variance of error for HS2 as shown in figure 5.

2) Relationship between Chip area, Number of gates and Power Consumption

Since it was not feasible to calculate power consumption for entire population during evolution, the author assumed positive correlation between number of gates and power consumption. In the end, he verified this relationship for 50 best of run circuits for all possible number of gates. He has It is clear from the figure 7 that number of gates is proportional to chip area. Furthermore positive correlation between chip number of gates and power consumption is also clear form figure 6 for multiplier circuits.

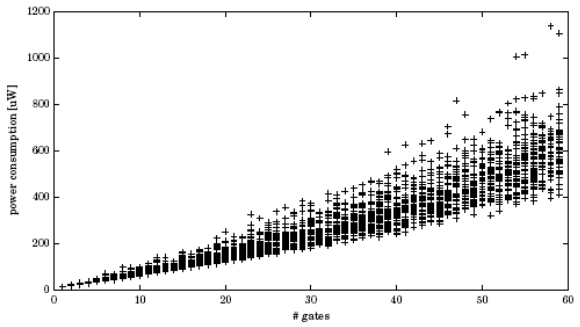


Figure 5: Number of gates vs. power consumption for 4 bit multiplier [Vasicek 2014]

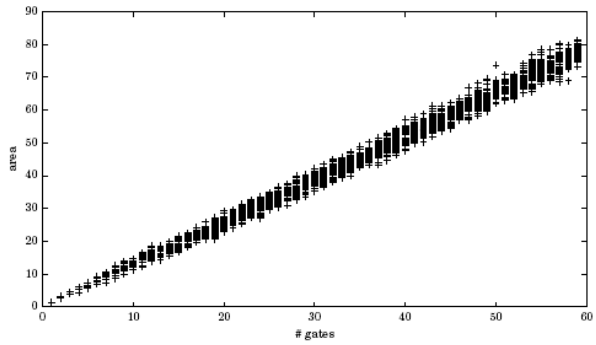


Figure 6: Number of gates vs. Area for 4-bit multiplier, [Vasicek 2014]

B. Comparison of Fitness Functions for Adders

In this work [11], Sekanina and Vasicek has evolved approximate adders for different number of gates. CGP has been used because its representation facilitates evolution of digital circuits.Author has used two fitness functions (Sum of Hamming distance-SHD, Sum of absolute difference-SAD) and studied their effects on final evolved circuit.

Author has also evolved approximations of four circuits (cm152, sym9, t481) from LGSynth93 suite. In order to find the best tradeoff between power and circuit size, whole spectrum of circuits of successively smaller size is evolved. Figures [11] and [12] show tradeoff between functionality and power for the evolved circuits.

adder 4,4

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(17)	(15)	(13)	(11)	(10)	(8)	(6)	(5)	(3)	(1)
func.	100%	95%	92%	90%	88%	85%	80%	77%	72%	65%
power	0%	-3%	17%	25%	43%	48%	62%	70%	80%	93%
saving	(191)	(197)	(156)	(143)	(107)	(99)	(71)	(56)	(37)	(12)
fSAD	100%	93%	92%	85%	84%	83%	83%	86%	86%	87%
avgad	0.0	12.1	8.7	11.7	11.0	9.3	7.8	6.4	6.1	4.7
maxad	0	16	20	20	20	20	16	12	14	10
errs	0%	16%	25%	39%	43%	57%	67%	67%	71%	87%

Figure 7: Efficiency of 4 bit adder for SAD fitness function [Vasicek 2012]

adder 4,4 (SAD)

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(17)	(15)	(13)	(11)	(10)	(8)	(6)	(5)	(3)	(1)
func.	100%	99%	99%	97%	97%	95%	95%	94%	92%	91%
power	0%	17%	31%	44%	13%	56%	64%	69%	93%	99%
saving	(201)	(166)	(137)	(112)	(175)	(86)	(71)	(61)	(13)	(1)
fSHD	100%	85%	85%	73%	74%	63%	62%	65%	55%	55%
avgad	0.0	1.0	1.0	1.6	1.7	2.0	1.8	3.0	2.8	2.9
maxad	0	1	1	3	3	7	4	7	8	8
errs	0%	25%	25%	43%	43%	69%	78%	57%	85%	88%

Figure 8: Efficiency of 4 bit adder for SHD fitness function [Vasicek 2012]

For evolution of multiple output circuits, author has evolved three 3 x 3 bit, 3 x 4 bit and 4x 4bit adders. For multiple outputs, two different fitness functions used were SAD and SHD. Although SAD slows down search process because of rugged fitness landscape, it evolved accurate circuits than SHD for same number of gates. Following figure compares functionality vs gates graphs for two fitness functions

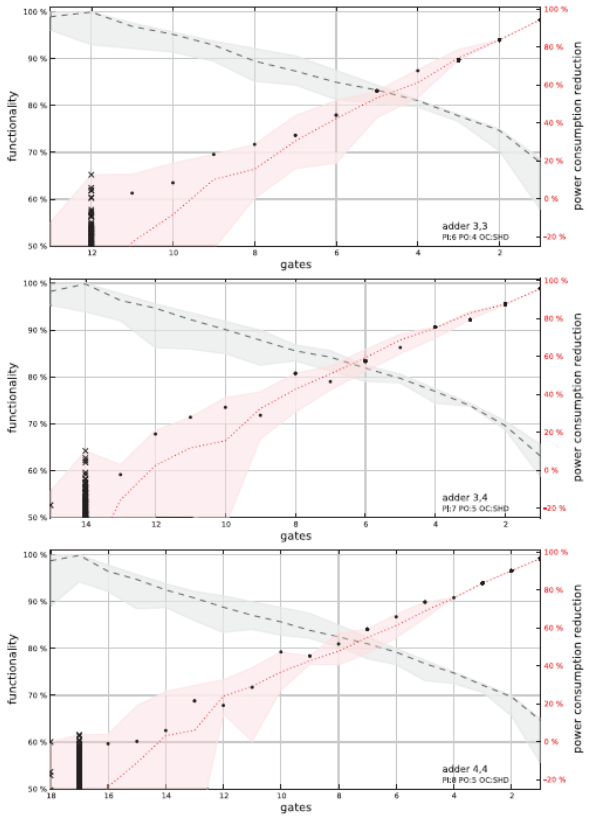
This work concludes that SAD based fitness evaluation is better than normally used SHD fitness function for adders, although care must be taken to extend this generalization

C. Equivalence Checking based fitness funtion

In this work, author has employed CGP to reduce number of gates in LGsynth93 circuits. For fitness evaluation, formal verification has been used. Both reference circuit and evolved circuits are represented in CNF form and compared for equivalence. If two circuits turn out to be equivalent, then circuit with less number of gates is considered fitter..

This approach as added advantage of not requiring exhaustive input output based fitness function. Since equivalence checking is NP-hard by nature, therefore computational complexity of SAT solvers also grows exponentially with circuit size. Complexity of SAT based equivalence can be improved by checking by keeping track of mutations in the circuit and evaluating only mutated portions of the circuit in next portion.

For parent selection, this work has compared two methods.



.Figure 9: Functionality and power consumption of adders for different number of gates using SHD. [Vasicek 2012]

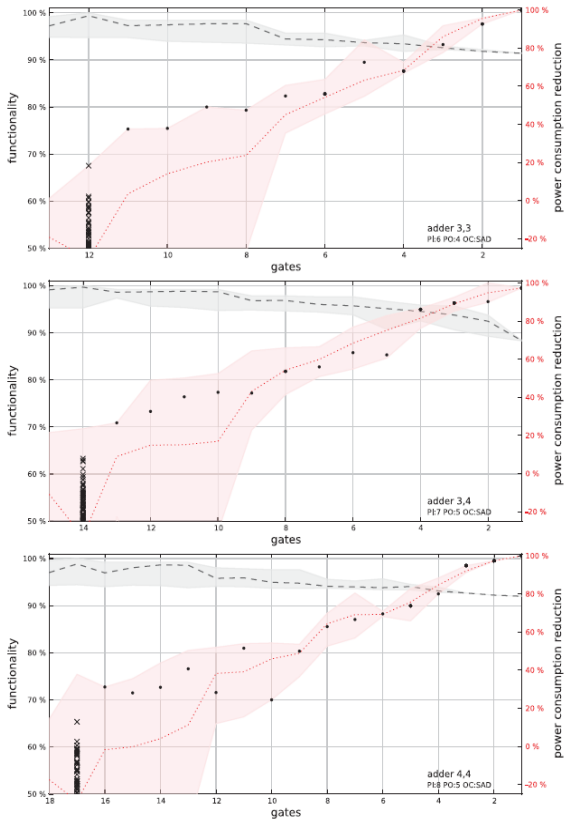


Figure 10: Functionality and power consumption of adders for different number of gate using SAD [Vasicek 2012]

Method A

This is standard CGP parent selection mechanism. The fittest individual from both current generation and previous generation is used as parent for next generation.

Method B

This is a slightly different form of elitism. In this strategy, parent (p) is stored separately (β) from current population of individuals

$$\beta' = \begin{cases} \beta & when \quad f_\beta \geq f_{x_i}, i = 1 \ldots \lambda, \\ x_j & otherwise, \end{cases}$$

β stores fittest individual and f_β is the fitness of fittest individual found so far. β is updated when fitter individual is found.

For selection, parent of previous generation (p) is allowed to compete against best individuals of offspring Parent selection works as follows

$$p' = \begin{cases} p & when \quad \forall i, i = 1 \ldots \lambda : f_{x_i} < b_{max} \\ x_j & otherwise, \end{cases}$$

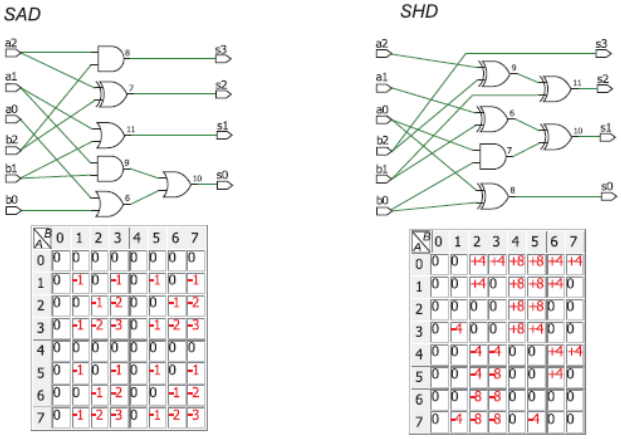


Figure 11: Circuits evolved using SHD and SAD fitness function at 50% area consumption [Vasicek 2012]

b_{max} is the fitness of best individual found so far. x_j denotes individual with higher fitness than β_{max} . If x_j contains more than one individual, then parent is chosen randomly from x_j.

Selection strategy based on method B outperformed method A in more than 50 % cases. It was because method B makes CGP more explorative as compared to method A. Strong exploration is always useful in rugged fitness landscapes (such as those of digital circuits).

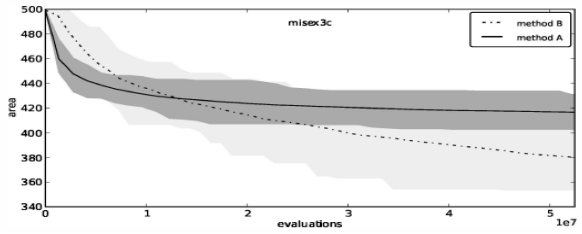


Figure 12: Comparison of Area efficiency of selection methods with functional Equivalence for misex3c [Sekanina 2013]

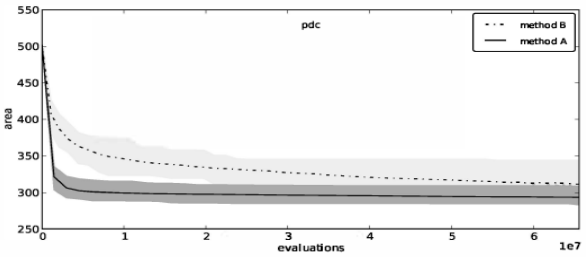


Figure 13: Comparison of Area efficiency of selection methods with functional Equivalence for pdc [Sekanina 2013]

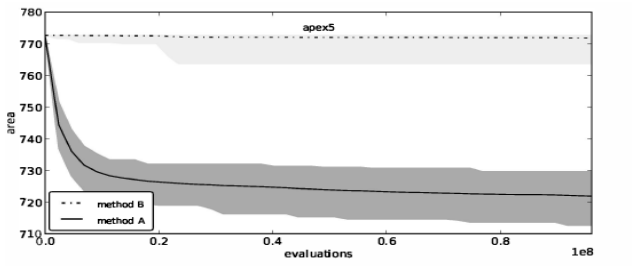


Figure 14: Comparison of Area efficiency of selection methods with functional Equivalence for apex5 [Sekanina 2013]

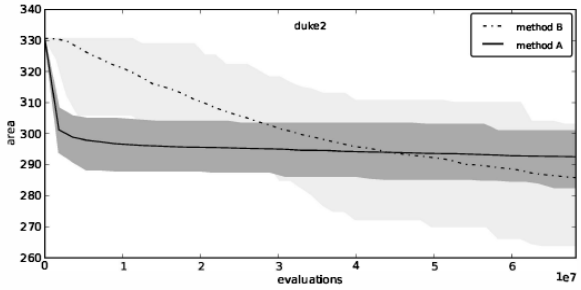


Figure 15: Comparison of Area efficiency of selection methods with functional Equivalence for duke2 [Sekanina 2013]

Although CGP was able to evolve compact circuits with significant area reduction, the evolved circuits had longer delays as compared to original circuits

D. Approximate Circuit Design using Multi Objective Evolutionary Algorithm

In his work [13], Sekanina has evolved approximate multiple constant multipliers (MCM) with the objective of minimizing number of components. MCM are used in implementing FIR filters. CGP has been used to evolve approximate and efficient implementation of MCM.

Multi-Objective optimization problem has the following form

$$\begin{array}{lll} \text{minimize/maximize} & f_m(x), & m = 1, 2, ..., M \\ \text{subject to} & g_j(x) \geq 0 & j = 1, 2, ..., J \\ & h_k(x) = 0 & k = 1, 2, ..., K \end{array}$$

For evolution of approximate MCM, the objective was to minimize four functions, f_1,f_2,f_3,f_{err} representing number of components, adders, delay and error respectively subject to the

$$\sum_{i=1}^{N} (y_i - y_i')^2 < E$$

constraints

Here E is error threshold specified by user. Here y_i is the target value of filter constants, y_i is the value searched by CGP.

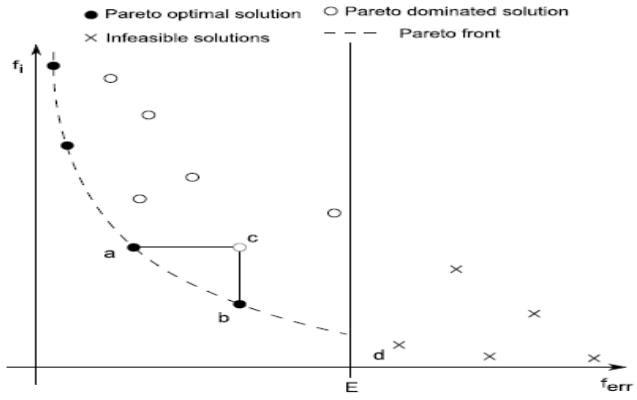
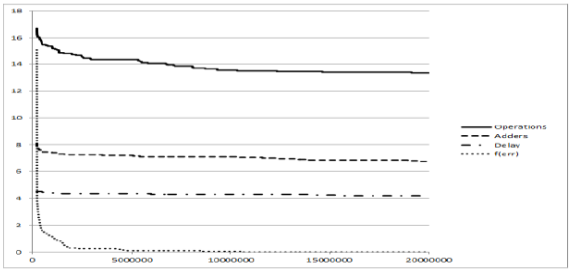


Figure 16: Pareto front for MCM circuit [Petrlik 2013]

Decision about the comparative fitness of competing individuals is based on 'Pareto dominance' or Pareto 'optimality' shown in figure [17].

In their experiments, author has given different weights to different components; the results for MOEA for different weights are shown in figure [18].



The progress of evolution for the MCM with 8 coefficients. Best results averaged from 20 independent runs are shown.

Figure 17: Evolution of MCM using MOEA [Petrlik 2013]

E. ESD immune Circuit Design Using CGP

ESD is the ability of circuits to withstand high instantaneous discharge at input or output pins. Conventionally ESD protection circuits induce undesirable properties in the circuits, such as delay, parasitic capacitance etc.

Evolution has two inherently characteristics that make it suitable for evolving circuits with enhanced ESD protection.

a) Diversity

b) Redundancy

In this work [12], author has used evolution to create ESD immune circuits. He has done so by exploiting the tendency of evolution to create redundant and degenerate circuits which results in better ESD performance of digital circuits. Such characteristics make circuits more robustness against external interferences such as ESD.

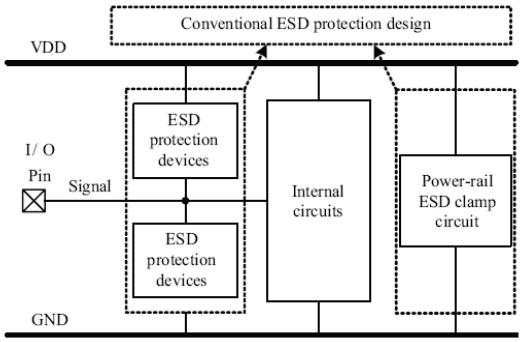


Figure 18: Modeling of ESD protection circuitry [Menghua 2013]

Algorithm used was same CGP as described earlier.
Functional correctness is measured as the percentage of correct outputs for all input combinations.

$$F = (1 - \frac{1}{2^n \cdot m} \cdot \sum_{i=1}^{2^n} \sum_{j=1}^{m} |d_{ij} - y_{ij}|) \cdot 100\%,$$

Here n is number of inputs, m is number of outputs. d_{ij} is output of evolved circuit, and y_{ij} is correct output.

Immunity is calculated as follows

$$I = \sum_{d \in D} \psi(d) \cdot F(d),$$

Here D is entire interference space. y (d) is probability of one interference instance to take place. In this work, author has assumed uniform probability of all interference. As a result, ESD immunity is average number of correct inputs under all possible interferences.

Figure [20] shows positive relationship between ESD immunity and circuit size.

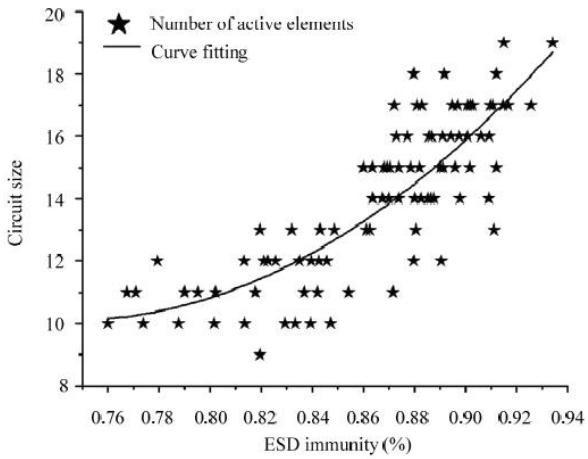


Figure 19: Relationship between ESD immunity and circuit size

The increased esd immunity is the result of redundancy and degeneracy. Figure [21] shows positive relationship between ESD immunity and redundancy, degeneracy for 2 bit adder.

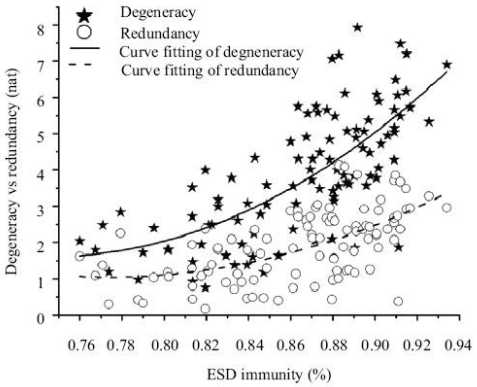


Figure 20: Relationship between ESD immunity and Degeneracy, Redundancy [Petrlik 2013]

V. COMPARISON

Although, all five approaches discussed so far use the similar platform for evolution of digital circuit, but they are all quite different because of distninct fitness functions. Based on fitness evaluation, we can divide them into three categories (i) simple input output based fitness evaluation, (ii) functional equivalence based fitness evaluation, (iii) multi objective fitness function. Each of these approaches holds certain advantages over other. For example, smaller circuits with large number of inputs, Functional equivalence based fitness evaluation will be faster, but as circuit size increases, functional equivalence becomes computationally expensive.

	Technique	Fitness Functions	Objective	Reported Results
[Vasicek 2014]	CGP with Seeding	Sum of Euclidean distance	Power Efficient multipliers and median filters	Graph between error and power
[Petrlik 2013]	*Multi-objective Optimization with CGP*	*Hamming distance*	*MCM with different constraints for different gates*	*Graph between component count generations*
[Menghua 2013]	ESD optimization with CGP	Sum of Hamming distance under Electrostatic Interference	ESD immune circuit	Graph between ESD immunity and Circuit size
[Sekanina 2013]	CGP	Functional equivalence	Power Efficient Adders	Upto 24% area improvement
[Vasicek 2012]	CGP with mixed selection	Sum of Hamming distance	Power Efficient Circuit	Graph between area and generations

Figure 21: Summary of five approaches covered

Multi objective optimization gives us better control on the properties of evolved circuit. This is especially beneficial for approximate circuits where we have competing objectives (accuracy, power, area, gates etc). In next section, we have discussed strengths and limitations of MOEA in greater detail.

VI. PROMISING APPROACH

In our opinion, multi objective evolutionary algorithm of Sekanina [13] is most promising approach for evolution of approximate circuits. This is because evolution of approximate circuit is inherently multi objective problem which requires maximization of competing (power efficiency and accuracy) objectives at the expense of each other. The goal is to find best tradeoff between power efficiency and accuracy. Since MOEA inherently optimizes several objectives simultaneously, that's why it is natural choice for evolution of approximate circuits.

A. Strengths

Most important strength of MOEA is the design of sophisticated fitness function from combination of simpler fitness criteria. For example, for evolution of approximate arithmetic circuits, in addition to error constraint, we can specify upper limits on sum of hamming distance and sum of absolute distance and get circuits and find best tradeoffs between these fitness function

Secondly**,** MOEA allows us to better control on the properties of evolved circuits by allowing us to specify separate constraints on individual components. For example, Sekanina [13] has assigned different weights to individual components during evolution of MCMs

Thirdly, it allows us to incorporate our prior knowledge/ experience in the form of additional constraints of multi objective fitness function.

B. Weakness

Although MOEA allows to design more sophisticated fitness functions by combining simpler fitness functions, but its fitness evaluation is computationally more expensive. This is especially a concern for complex circuits where evaluation of single I/O based fitness function is exponential in terms of number of inputs. Secondly, Author has tested MOEA on just one simple circuit. Its effectiveness needs to be verified to larger complex circuit.

VII. FUTURE WORK

In this section, we have discussed some challenges faced

A. Challenges in evolution of digital circuits

As complexity of digital circuits increases, evolutionary process becomes very slower, the effect is so strong that it becomes difficult to evolve moderately sized complex circuits such as multipliers.

Evolution works by searching for independent building blocks through mutation, and combining them through crossover. Building blocks are group of genes that give fitness boost together but do not contribute as much independently. Absence of cross over operators in CGP means that whole circuit is behaving like one giant building block that has to be searched through mutation. As the size of building blocks gets larger, their discovery takes longer thus slowing the speed of convergence. In short, because of absence of cross over operator, CGP takes very long to converge to optimal circuit

Secondly, CGP is very sensitive to changes in genotype. For example, a single change near the output of a perfect circuit can lead to drastic decrease in the performance of input output based fitness function. This results in bad fitness distance correlation co-efficient for CGP representation. And genetic algorithms are known to perform poorly [18] on problems whose fitness distance correlation is small.

Small changes in genotype (e.g near input/output nodes) result in large changes in phenotype behavior, As a result, algorithm has to wait for large number of generations before it can find good building blocks.

B. Spatially variable cross over operator

As discussed earlier, sensitivity of digital circuits to cross over operator increases as we move towards the last column in CGP representation (output end) of genotype. The problem of increased sensitivity to cross over is inherent to CGP due to nature of digital circuit. Instead of using uniform cross over/mutation operators (which generally results in performance degradation) anywhere in the genotype, we

propose applying low cross over/mutation operators in sensitive areas (near output end) of the circuit. This will give sufficient time for optimization of earlier parts of the circuit on which output depends. The underlying assumption is that performance of genes near output depends upon the performance of genes near inputs and middle part of genotype, therefore we need to explore more heavily near input and middle portion of circuit. This will give sufficient time for maturing the design in starting and middle areas of circuit on which output is dependent.

C. How will direction is more fruitful Direction

The advantage of this approach will be that it will explore more heavily in relatively non-sensitive areas resulting in formation of building blocks. Since we are using low values of mutation/cross over operator near the output will ensure that discovered blocks of circuit are not discarded prematurely due to changes near output end of circuit.

By combining variable cross over rate with mutation operator, search will be able to harness full potential of Genetic Algorithms on digital circuits. Purpose of mutation operator is to explore new areas of fitness landscape thus forming new building blocks. Introduction of cross over operator in CGP at controlled rate will allow combination of these building blocks resulting in faster convergence.

VIII. CONCLUSION

MOEA is the most promising approach for evolution of approximate circuits. Its main contribution lies in its ability to design sophisticated fitness function from combination of several simple fitness functions. It also allows us to easily incorporate our prior knowledge in the form of constraint specification. We have also discussed the reasons for absence of cross over operator in CGP. In the absence of cross over, entire circuit behaves like one giant building block that has to be discovered through mutation. To counter this problem, we have proposed idea of spatially variable cross over such that cross over rate is gets smaller and smaller as we move towards the output column of circuit. We expect that variable cross over rate would allow combination of building blocks without severely affecting the fitness thus allowing faster convergence by of building blocks.

REFERENCES

[1] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” in Proc. of USENIX ATC, 2010.

[2] P. C. Haddow and A. M. Tyrrell, “Challenges of evolvable hardware:past, present and the path to a promising future,” Genetic Programming and Evolvable Machines, vol. 12, no. 3, pp. 183–215, 2011.

[3] L. Sekanina, “Evolvable hardware,” in Handbook of Natural Computing.Springer Verlag, 2012, pp. 1657–1705.

[4] Miller J. F., Harding S. L. Tutorial on Cartesian Genetic Programming, Genetic and Evolutionary Computation Conference, Montreal, CA (2009)

[5] J. Clerk Vasicek, Z.; Sekanina, L. "Evolutionary Approach to Approximate Digital Circuits Design," IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, in-press, doi: 10.1109/TEVC.2014.2336175

[6] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *Proc. of the 18th IEEE European Test Symposium*. IEEE, 2013, pp. 1–6.

[7]] E. Le Sueur and Gernot Heiser. Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns. In Proceedings of the Workshop on Power Aware Computing and Systems, October 2010

[8] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” *IEEE Trans. On CAD of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2013.

[9] P. Kulkarni, P. Gupta, and M. D. Ercegovac, “Trading accuracy for power in a multiplier architecture,” J. Low Power Electronics, vol. 7, no. 4, pp. 490–501, 2011.

[10] Sekanina, L.; Vasicek, Z., "Approximate circuit design by means of evolvable hardware," Evolvable Systems (ICES), 2013 IEEE International Conference on , vol., no., pp.21,28, 16-19 April 2013, DOI= 10.1109/ICES.2013.6613278

[11] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan,“Salsa: systematic logic synthesis of approximate circuits,” in *The 49th Annual Design Automation Conference 2012, DAC ’12*. ACM, 2012, pp. 796–801.

[12] Menghua Mana, Shanghe Liua, Xiaolong Changa, Mai Lub, “The Biological Property of Synthetic Evolved Digital Circuits with ESD Immunity – Redundancy or Degeneracy?,” Journal of Bionic Engineering, pp. 396–403, 2013.

[13] Petrlik, J.; Sekanina, L., "Multiobjective evolution of approximate multiple constant multipliers," Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2013 IEEE 16th International Symposium on , vol., no., pp.116,119, 8-10 April 2013. DOI= 10.1109/DDECS.2013.6549800)

[14] Vasicek, Z.; Sekanina, L., "On area minimization of complex combinational circuits using cartesian genetic programming," Evolutionary Computation (CEC), 2012 IEEE Congress on , vol., no., pp.1,8, 10-15 June 2012. DOI= 10.1109/CEC.2012.6256649

[15] Venkataramani, S.; Sabne, A.; Kozhikkottu, V.; Roy, K.; Raghunathan, A., "SALSA: Systematic logic synthesis of approximate circuits," Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE , vol., no., pp.796,801, 3-7 June 2012

[16] A. Thompson, “Silicon Evolution,” In Proceedings of the First Annual Conference on Genetic Programming (GECCO '96). . 1996. MIT Press, Cambridge, MA, USA, 444-452.

[17] P. C. Haddow and A. M. Tyrrell, “Challenges of evolvable hardware: past, present and the path to a promising future,” *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 183–215, 2011.

[18] Terry Jones , Stephanie Forrest, Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms

Proceedings of the Sixth International Conference on Genetic Algorithms (1995)

[19] Terry Jones , Stephanie ForrestN. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *The 47th Annual Design Automation Conference, DAC '10*. ACM, 2010, pp. 859–867.