The Advantages of Landscape Neutrality in Digital Circuit Evolution

Vesselin K. Vassilev[1] and Julian F. Miller[2]

[1] School of Computing, Napier University
Edinburgh, EH14 1DJ, UK
v.vassilev@dcs.napier.ac.uk
[2] School of Computer Science, University of Birmingham
Birmingham, B15 2TT, UK
j.miller@cs.bham.ac.uk

Abstract. The paper studies the role of neutrality in the fitness landscapes associated with the evolutionary design of digital circuits and particularly the three-bit binary multiplier. For the purpose of the study, digital circuits are evolved extrinsically on an array of logic cells. To evolve on an array of cells, a genotype-phenotype mapping has been devised by which neutrality can be embedded in the resulting fitness landscape. It is argued that landscape neutrality is beneficial for digital circuit evolution.

1 Introduction

Digital circuit evolution is a process of evolving configurations of logic gates for some prespecified computational program. Often the aim is for a highly efficient electronic circuit to emerge in a population of instances of the program. Digital electronic circuits have been evolved intrinsically [1] and extrinsically [2–6]. The former is associated with an evolutionary process in which each evolved electronic circuit is built and tested on hardware, while the latter refers to circuit evolution implemented entirely in software using computer simulations.

A possible way to study the evolvability of digital circuits is to consider the evolutionary design as a search on a *fitness landscape* [7]. The metaphor comes from biology to represent adaptive evolution as a population flow on a mountainous surface in which the elevation of a point qualifies how well the corresponding organism is adapted to its environment [8]. In evolutionary computation the fitness landscapes are simply search spaces that originate from the combination of the following objects

1. A set of configurations that are often referred to as *genotypes*.
2. A cost function that evaluates the configurations, known in evolutionary computation as a *fitness function*.
3. A topological structure that allows relations within the set of configurations.

These define the structure of the fitness landscape. Recently it has been shown that the landscape structure affects the evolvability of a variety of complex

systems [9–15]. In evolutionary computation the notion of evolvability refers to the efficiency of evolutionary search.

The circuit evolution landscapes associated with the evolution of various arithmetic functions were studied in [7, 16, 17]. It was shown that these landscapes are products of three subspaces with different landscape characteristics. These are the functionality, internal connectivity, and output connectivity landscapes. In general they are characterised with neutral networks and sharply differentiated plateaus. A set of genotypes defines a *neutral network* if the set represents a connected subgraph of genotypes with equal fitnesses [18, 19]. This characteristic of fitness landscapes is referred to as *neutrality*.

The landscape neutrality in digital circuit evolution originates mainly from the genotype-phenotype mapping by which a digital circuit is encoded into a genotype. The mapping is defined in such a way that it allows neutrality. This affords a study of the important question of the role of landscape neutrality in the evolutionary design of digital circuits. For the purpose of this study, digital circuits are evolved extrinsically. This allows freedom to explore the methodology and thus to extract principles of the evolutionary design of circuitry in general [17].

Studies in evolutionary biology suggest that adaptive evolution is facilitated by a genetic variation that is due to neutral or nearly neutral mutations [20–23]. In [24] it was suggested that the role of landscape neutrality for adaptive evolution is to provide a "path" for crossing landscape regions with poor fitness. This implies a scenario of adaptive evolution in which a population evolves on a neutral network until another neutral network with a higher fitness is reached [25]. Similar conclusions were attained in a study of the technique of genotype-phenotype mapping that appeared to be suitable for solving constrained optimisation problems by genetic programming [26]. Evidence that fitness improvements may occur in a genetically converged population due to neutrality was given also in [27].

Does the evolutionary design of digital circuits benefit from the neutrality in the fitness landscapes? In this paper this question is answered in the affirmative. The paper is organised as follows. The next section introduces digital circuit evolution in greater details. Section 3 represents the evolution of a three-bit multiplier. The neutrality of the fitness landscapes is studied in section 4. Section 5 studies the benefit of landscape neutrality for the adaptive design of digital circuits. The paper closes with a summary and suggestions for future work.

2 Digital Circuit Evolution

The technique used in the evolutionary design of digital circuits in this paper is that adopted in the framework of Cartesian Genetic Programming [28, 29] and it uses an evolutionary algorithm with truncation selection and mutation. The latter is defined as a percentage of the genes in a single genotype which are to be randomly mutated. In this paper the percentage chosen results in 3

mutated genes per genotype. The algorithm deals with a population of digital feed-forward electronic circuits that are instances of a particular program. The population consists of $1 + \lambda$ genotypes where λ is usually about 4. Initially the elements of the population are chosen at random. The fitness value of each genotype is evaluated, by calculating the number of total correct outputs of the encoded electronic circuit in response to all appropriate input combinations. For convenience, in this paper the fitness values are scaled in the interval $[0, 1]$. To update the population, the mutation operator is applied to the fittest genotype, to generate offspring. These together with the parent constitute the new population. This mechanism of population update has some similarities with that employed in other evolutionary techniques such as $(1+\lambda)$ Evolution Strategy [30, 31] and the Breeder Genetic Algorithm [32].

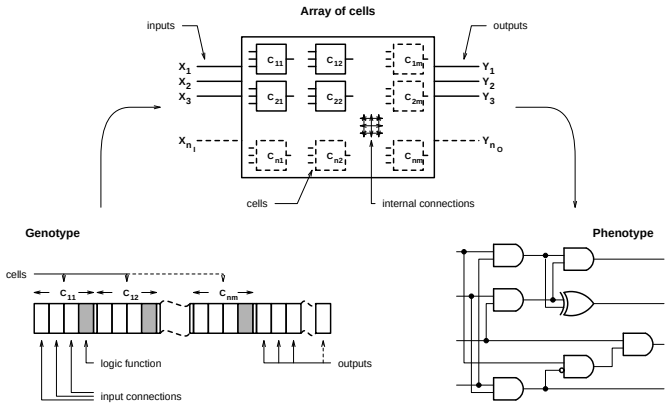

Fig. 1. The phenotype that is a digital circuit is encoded within a genotype by an array of logic cells.

To encode a digital electronic circuit into a genotype, a genotype-phenotype mapping is defined. This is done via rectangular array of cells each of which is an atomic two-input logic gate or a multiplexer. Thus the genotype is a linear string of integers and it consists of two different types of genes that are responsible for the functionality and the routing of the evolved array. The genotype is characterised by four parameters of the array of cells: the *number of allowed logic functions*, the *number of rows*, the *number of columns*, and *levels-back*. The first parameter defines the functionality of logic cells, while the latter three parameters determine the layout and routing of the array. Note that the number of inputs and outputs of the array are specified by the objective function. The genotype-phenotype mapping is defined using an array of $n \times m$ three-input cells with n_I inputs and n_O outputs, and together with the genotype representation is shown in Figure 1. The array is a composition of cells each of which can be any allowed two-input logic gate or alternatively a multiplexer.

The internal connectivity of the array is defined by the connections between the array cells. The inputs of each cell are only allowed to be inputs of the array or outputs of the cells with lower column numbers. The internal connectivity is also dependent on the levels-back parameter that defines the array inputs and cells to which a cell or an array output can be connected in the following manner. Consider that the levels-back parameter is equal to L. Then cells can be connected to cells from L preceding columns. If the number of preceding columns of a cell is less than L then the cell can also be connected to the inputs of the array. In this paper the array cells and outputs are maximally connectable since the number of rows is set to one and the levels-back is equal to the number of columns.

The gate array output connectivity is defined in a similar way. The output connections of the array are allowed to be outputs of cells or array inputs. Again, this is dependent on the neighbourhood defined by the levels-back parameter.

The genotype is a string of integers that encode either logic functions or connections. The logic functions are represented by letters associated with the allowed cell functionality. The connections are defined by indexes that are assigned to all inputs and cells of the array. Each array input X_k is labelled with $k-1$ for $1 \leq k \leq n_I$, and each cell c_{ij} is labelled with an integer given by $n_I + n(j-1) + i - 1$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. Thus the genotype consists of groups of four integers that encode the cells of the array, followed by a sequence of integers that represent the indexes of the cells connected to the outputs of the array. The first three values of each group are the indexes of the cells to which the inputs of the encoded cell are connected. If the cell represents a two-input logic function, then the third connection is redundant. This type of redundancy is referred to as *input redundancy*. The last integer of the group represents the logic function of the cell. Cells may also not have their outputs connected in the operating circuit. This is another form of redundancy called *cell redundancy*. The redundancy in the genotype related to the function of the array may also be *functional redundancy*. This is the case in which the number of cells of a digital circuit is higher than the optimal number needed to implement this circuit.

3 Evolution of a Three-bit Multiplier

To study the role of landscape neutrality in the evolutionary design of digital circuits, a three-bit multiplier is evolved using binary multiplexers. The three-bit multiplier is a good candidate to be used in this study for the following reasons: firstly, the circuit is difficult to evolve, and secondly, it is a fundamental building block of many digital devices. In addition it has been shown that the fitness landscapes associated with the evolution of the three-bit multiplier are similar to the landscapes of other arithmetic functions in terms of landscape neutrality [17]. The binary multiplexers are defined by the universal-logic function

$$f(a, b, c) = a \cdot \overline{c} + b \cdot c \tag{1}$$

taken four times with inputs a and b inverted in various ways (gates $16 - 19$ as labelled in [17]). The reason for using only multiplexers is to simplify the

evolutionary model by allowing only the existence of cell redundancy and functional redundancy. 100 evolutionary runs of 10 million generations were carried out. The array had 1×24 cells and the levels-back was set to 24. 27 perfect solutions were found, three of which were circuits that required 21 gates. For each evolutionary run the best fitness of the population and the corresponding genotype were recorded for the generations in which improvements of the fitness have been attained. In addition, the number of neutral changes between every two fitness improvements was evaluated. Thus the number of neutral mutations for each fitness improvement were calculated cumulatively. The aim is to attain understanding of the process of evolving digital circuits, particularly the three-bit multiplier. The results for a typical evolutionary run in which a three-bit multiplier of 21 gates was obtained are given in Figure 2. The figure represents (a) the best fitness, and (b) the cumulative number of neutral mutations in the (1) functionality, (2) internal connectivity, and (3) output connectivity configurations. The circuit was attained at generation $4,970,271$, and its schematic is given in Figure 3. The circuit is efficient in term of gate usage, since it consists of 21 logic gates that is 20% less than the number of gates of the best conventional design. The logic gates used in the figure are multiplexers (given with rectangles), `AND`, `OR`, and `XOR`. The two-input gates in the depicted multiplier came about because some multiplexers had two inputs connected together.

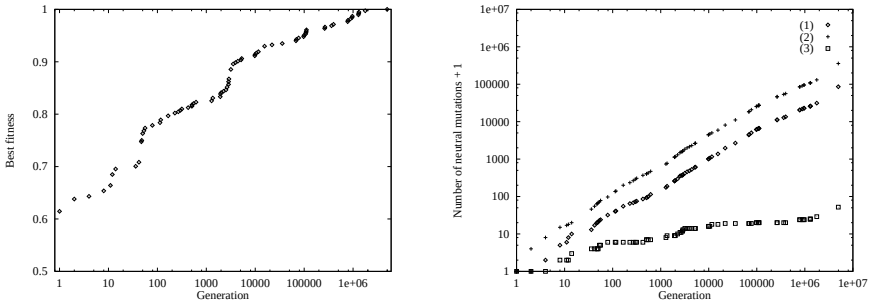

Fig. 2. The evolution of a three-bit multiplier: (a) the best fitness, and (b) the cumulative number of neutral mutations in the (1) functionality, (2) internal connectivity, and (3) output connectivity configurations, recorded at each fitness increase (74 in total).

Figure 2a shows that the best fitness is marked by periods of a sharp fitness increase followed by periods of stasis or a slight fitness increase. It is hypothesised that each period of stasis is a hidden process of search performed by neutral walks so that the population could traverse wide search space areas with lower or equal fitness. To investigate this phenomenon the number of all neutral mutations at each generation for functionality, input connectivity and output connectivity configurations are also evaluated cumulatively (Figures 2b). It is

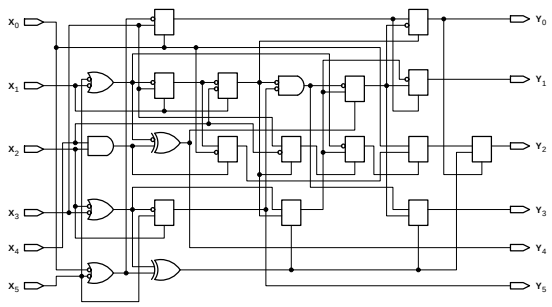

Fig. 3. The schematic of the evolved three-bit multiplier. The most significant bits are inputs X_0 and X_3, and output Y_0.

shown that the cumulative number of neutral mutations increases with a lower than linear rate at each generation, and the evolutionary process is constantly accompanied by neutral mutations. The plots also show that neutral changes are most likely to appear in the input connectivity configurations, less in the functionality configurations, and least in the output connectivity configurations.

4 Landscape Neutrality

The landscapes of the evolved three-bit multiplier result from the Cartesian product of three configuration spaces [7] defined on the functionality and connectivity alphabets that in this particular case have sizes $l_\alpha = 4$ and $l_\beta = 24$, respectively. The former alphabet is defined by the number of allowed logic functions while the latter is defined by the levels-back parameter. The structure of the three subspaces - functionality, internal connectivity, and output connectivity - did not differ significantly from the structure of the landscapes associated with other electronic circuits, such as two-bit and three-bit multipliers evolved for various values of the functionality and connectivity parameters [17]. They are characterised with neutrality that prevails over the landscape smoothness and ruggedness in the internal connectivity subspace. This was not valid for the functionality and output connectivity landscapes. It was also found that the neutrality of the output connectivity landscapes was more strongly dominated by the landscape ruggedness than that found in the functionality landscapes. These findings were revealed by studying the information characteristics [33] of time series obtained via random walks with respect the three subspaces.

An interesting issue related to the role of neutrality in the evolution of digital circuits is the relation between the size and the height of the landscape plateaus. It is believed that the neutral walks are longer at a lower altitude fitness level. Thus it can be surmised that the length of the neutral walks will decrease as the best fitness increases. The reason is that the genotype redundancy is expected to decrease in an efficient evolutionary search. This can be illustrated by measuring the length of the neutral walks that start from those genotypes recorded at

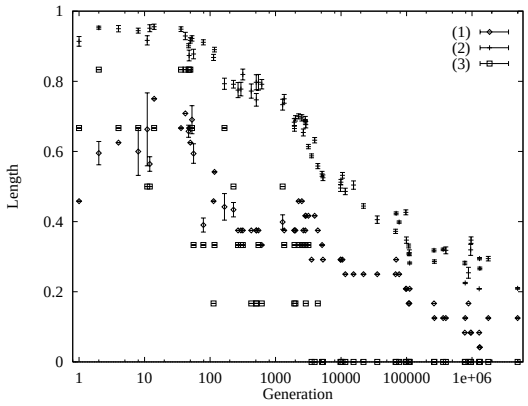

Fig. 4. The length of neutral walks on (1) functionality, (2) internal connectivity, and (3) output connectivity landscapes scaled in the interval $[0, 1]$.

every fitness increase in the evolutionary run shown in Figure 2. The algorithm of neutral walks as given in [18] is defined as follows: start from a configuration, generate all neighbours, select a neutral one at random that results in an increase in the distance from the starting point and continue moving until the distance cannot be further increased. $1,000$ neutral walks per configuration were performed. The means and the standard deviations of the lengths scaled in the interval $[0, 1]$ are given in Figure 4. The scaling was done by dividing the length of each neutral walk by the length of the corresponding configuration. Note that the functionality, internal connectivity, and output connectivity configurations consist of 24, 72, and 6 genes, respectively. The figure confirms the findings of the information analysis of these landscapes. It is also shown that the length of the neutral walks decrease during the evolution with a higher than linear rate. An interesting result related to the evolved three-bit multiplier is that the length of the neutral walks on internal connectivity subspaces that start from the obtained functionally correct digital circuit exceeds the expected length regarding the cell redundancy. For this genotype, the expected length of the internal connectivity configuration is 9 since the number of redundant cells is 3 (9 redundant genes). However, the measured lengths were about 15. Therefore, there exist functional redundancy in the evolved internal connectivity configuration. This implies that some of the multiplexers of the circuit might be replaced with two-input logic gates this is also revealed by Figure 3.

5 Neutral Mutations and Search

The results represented thus far showed that digital circuit evolution is accompanied by a random genetic drift caused by neutral mutations. These are more likely to appear during the search on the functionality and input connectivity

landscapes since these subspaces are characterised with neutral networks that originate from the cell redundancy. The amount of neutral changes in the output connectivity configuration is much lower than in the functionality and internal connectivity ones. Note that the neutrality of the output connectivity subspace is determined only by the functional redundancy. It was also revealed that the cumulative number of neutral changes in the functionality and internal connectivity configurations increases during the evolutionary run with a lower than linear rate. This is to be expected since the fitness increase during the evolutionary run reduces the redundancy in general. This process in itself affects the landscape neutrality so that the size of neutral areas decreases with a higher than linear rate. The decrease of the landscape neutrality was revealed by measuring the length of the neutral walks that start from those genotypes recorded at every fitness increase (section 4). The interesting question here is how the increase of the cumulative number of neutral changes relates to the decrease of the landscape neutrality during the evolution. This is answered by Figure 5 that shows the derivatives (absolute values) of the plots of (1) the cumulative number of neutral changes (Figure 2b), and (2) the length of neutral walks (Figure 4). The derivatives are calculated for each generation characterised with a fitness

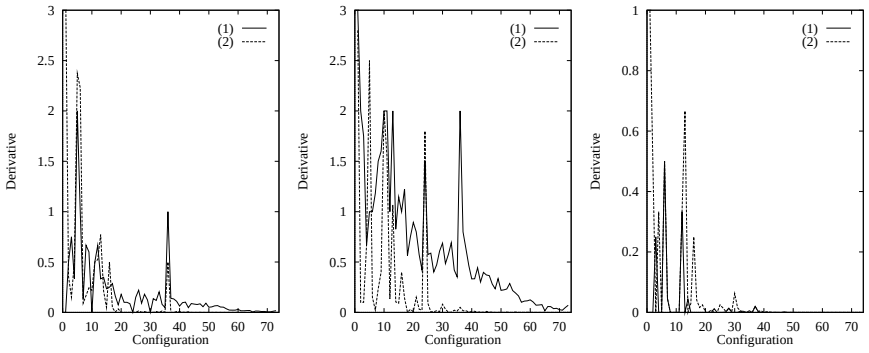

Fig. 5. Derivatives of the plots depicted in (1) Figure 2b and (2) Figure 4 calculated at each generation characterised with a fitness increase (74 in total): (left) functionality, (centre) internal connectivity, and (right) output connectivity configurations.

increase. The figure shows that the number of neutral changes decreases more slowly than the length of the neutral walks. This also holds for the output connectivity subspaces, although it is difficult to see this in the figure. The findings suggest that in the beginning of the run, neutral changes occur as a consequence of the high redundancy in the genotype. This however does not appear as a reason for the neutral changes at end of the run, since the redundancy becomes low. It appears that the selective mechanisms promote the neutral changes since this is the only feasible way for the population to explore the search space. This is also indicated in the plot in Figure 6. The plot shows the Hamming distance

of every two consecutive genotypes each of which resulted in a fitness increase for the evolutionary run studied in section 3. The Hamming distance increases with the length of the periods of stasis that is another indication of the genetic drift caused by neutral changes. For instance, the Hamming distance of the last two genotypes obtained at generations $1,771,234$ and $4,970,271$ is equal to 75, although the difference between the fitness values of these genotypes is approximately 0.0027 (this is exactly a difference of one bit of the corresponding truth tables). This is a significant difference when considering that the length of the genotype is 102. The drift was attained after $281,163$ neutral mutations. Hence neutral evolution is vitally important for the search especially when close to the global maximum where the likelihood of deleterious mutations to occur is high.

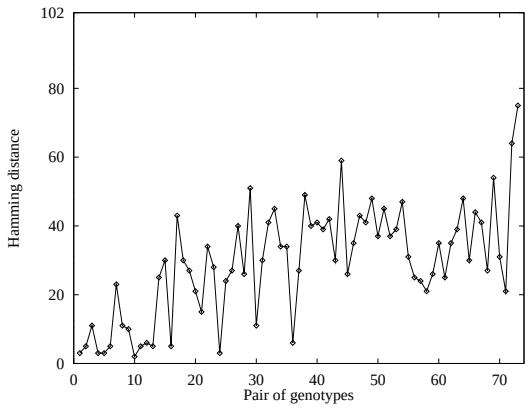

Fig. 6. The Hamming distance of two consecutive genotypes each of which represents a fitness increase in the evolutionary run shown in Figure 2.

The existence of neutrality helps the evolutionary design of digital circuits. Indeed if digital circuit evolution is implemented without neutral mutations, the result is not encouraging. This is illustrated by Figure 7. The figure shows the best fitness attained in the evolution of a three-bit multiplier in the experiment of 100 runs with "allowed" neutral mutations described in section 3, and the best fitness attained for 100 runs with "forbidden" neutral mutations. To allow neutral mutations, the algorithm was set up to choose a new parent even if the new fittest members of the population have fitness values that are equal to the fitness of the previous parent. Alternatively, to forbid neutral mutations, the algorithm was set up to change the parent only if a fitter member of the population occurs. The runs in which neutral mutations were allowed generated 27 perfect solutions: 5 with 24, 10 with 23, 9 with 22, and 3 with 21 logic gates. Although, the attained best fitness in the experiment with forbidden neutral mutations is fairly high, no perfect solution was evolved. This again supports the importance of landscape neutrality for the success of the evolutionary search.

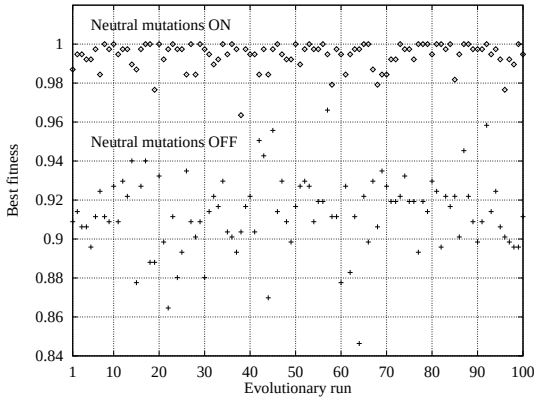

Fig. 7. The best fitness values of 2×100 evolutionary runs of 10 million generations with allowed (diamonds) and forbidden (crosses) neutral mutations.

The neutral evolution allows the population to avoid being trapped in a local optimum and thus to escape, and furthermore to cross wide landscape areas with lower fitness.

6 Summary

The importance of landscape neutrality for the evolution of digital circuits, particularly the three-bit multiplier, was revealed in a comparison between the amount of neutral changes and the size of the neutral areas during a successful evolutionary run. It was shown that the evolutionary process is accompanied by neutral mutations, the number of which, decreases with a lower rate when comparing with the decrease of the size of the neutral areas (Figure 5). Consequently, the neutral changes were employed in the evolutionary search (see also Figure 6). The landscape neutrality appeared to be vitally important for the evolutionary design of digital circuits in that it firstly prevents the evolved sub-circuit from deleterious mutations, and secondly, it allows the evolutionary search to avoid entrapment at local optima. This was empirically demonstrated in section 5 where it was shown that the search with allowed neutral changes is better.

Further research should be carried out to answer the following questions. How exactly does the circuit change in every period of stasis? How does evolution preserve the attained circuit modules? This remains for the future.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable and detailed comments on the earlier draft of the paper.

References

1. Kajitani, I., Hushino, T., Nishikawa, D., Yokoi, H., Nakaya, S., Yamauchi, T., Inuo, T., Kajihara, N., Iwata, M., Keymeulen, D., Higuchi, T.: A gate-level ehw chip: Implementing GA operations and reconfigurable hardware on a single LSI. In Sipper, M., Mange, D., Pérez-Uribe, A., eds.: Proceedings of the 2nd International Conference on Evolvable Systems: From Biology to Hardware. Heidelberg, Springer-Verlag (1998) 1–12.
2. Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H., Furuya, T.: Evolving hardware with genetic learning: A first step towards building a Darwin machine. In Meyer, J.A., Roitblat, H.L., Stewart, W., eds.: From Animals to Animats II: Proceedings of the 2nd International Conference on Simulation of Adaptive Behaviour. Cambridge, MA, MIT Press (1993) 417–424.
3. Hemmi, H., Hikage, T., Shimohara, K.: Adam: A hardware evolutionary system. In: Proceedings of the 1st International Conference on Evolutionary Computation. Piscataway, NJ, IEEE press (1994) 193–196.
4. Hemmi, H., Mizoguchi, J., Shimohara, K.: Development and evolution of hardware behaviours. In Brooks, R., Maes, P., eds.: Artificial Life IV: Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems. Cambridge, MA, MIT Press (1994) 371–376.
5. Miller, J.F., Thomson, P., Fogarty, T.: Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. In Quagliarella, D., Periaux, J., Poloni, C., Winter, G., eds.: Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Wiley, Chechester, UK (1997) 105–131.
6. Iba, H., Iwata, M., Higuchi, T.: Machine learning approach to gate-level evolvable hardware. In Higuchi, T., Iwata, M., eds.: Proceedings of the 1st International Conference on Evolvable Systems: From Biology to Hardware. Heidelberg, Springer-Verlag (1997) 327–343.
7. Vassilev, V.K., Miller, J.F., Fogarty, T.C.: Digital circuit evolution and fitness landscapes. In: Proceedings of the Congress on Evolutionary Computation. Volume 2., Piscataway, NJ, IEEE Press (1999) 1299–1306.
8. Wright, S.: The roles of mutation, inbreeding, crossbreeding and selection in evolution. In Jones, D.F., ed.: Proceedings of the 6th International Conference on Genetics. Volume 1. (1932) 356–366.
9. Kauffman, S.A.: Adaptation on rugged fitness landscapes. In Stein, D., ed.: Lectures in the Sciences of Complexity. SFI Studies in the Sciences of Complexity. Addison-Wesley, Reading, MA (1989) 527–618.
10. Manderick, B., de Weger, M., Spiessens, P.: The genetic algorithm and the structure of the fitness landscape. In Belew, R.K., Booker, L.B., eds.: Proceedings of the 4th International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann (1991) 143–150.
11. Mitchell, M., Forrest, S., Holland, J.: The royal road for genetic algorithms: Fitness landscapes and ga performance. In Varela, J., Bourgine, P., eds.: Proceedings of the 1st European Conference on Artificial Life. Cambridge, MA, MIT Press (1991) 245–254.
12. Palmer, R.: Optimization on rugged landscapes. In Perelson, A., Kauffman, S., eds.: Molecular Evolution on Rugged Landscapes. Volume IX of SFI Studies in the Sciences of Complexity. Addison-Wesley, Reading, MA (1991) 3–25.
13. Wagner, G.P., Altenberg, L.: Complex adaptations and the evolution of evolvability. Evolution **50** (1995) 967–976.

14. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82.
15. Stadler, P.F., Seitz, R., Wagner, G.P.: Evolvability of complex characters: Dependent fourier decomposition of fitness landscapes over recombination spaces. Technical Report 99-01-001, Santa Fe Institute (1999).
16. Vassilev, V.K., Fogarty, T.C., Miller, J.F.: Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In Ghosh, A., Tsutsui, S., eds.: Theory and Application of Evolutionary Computation: Recent Trends. Springer-Verlag, London (2000) In press.
17. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the evolutionary design of digital circuits. Journal of Genetic Programming and Evolvable Machines **1** (2000).
18. Reidys, C.M., Stadler, P.F.: Neutrality in fitness landscapes. Technical Report 98-10-089, Santa Fe Institute (1998).
19. Stadler, P.F.: Spectral landscape theory. In Crutchfield, J.P., Schuster, P., eds.: Evolutionary Dynamics — Exploring the Interplay of Selection, Neutrality, Accident and Function. Oxford University Press, New York (1999).
20. Kimura, M.: Evolutionary rate at the molecular level. Nature **217** (1968) 624–626.
21. King, J.L., Jukes, T.H.: Non-darwinian evolution. Science **164** (1969) 788–798.
22. Ohta, T.: Slightly deleterious mutant substitutions in evolution. Nature **246** (1973) 96–97.
23. Ohta, T.: The nearly neutral theory of molecular evolution. Annual Review of Ecology and Systematics **23** (1992) 263–286.
24. Huynen, M.A., Stadler, P.F., Fontana, W.: Smoothness within ruggedness: The role of neutrality in adaptation. Proceedings of the National Academy of Science U.S.A. **93** (1996) 397–401.
25. Huynen, M.A.: Exploring phenotype space through neutral evolution. Journal of Molecular Evolution **43** (1996) 165–169.
26. Banzhaf, W.: Genotype-phenotype-mapping and neutral variation — a case study in genetic programming. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature III. Berlin, Springer-Verlag (1994) 322–332.
27. Harvey, I., Thompson, A.: Through the labyrinth evolution finds a way: A silicon ridge. In Higuchi, T., Iwata, M., Liu, W., eds.: Proceedings of the 1st International Conference on Evolvable Systems. Berlin, Springer-Verlag (1996) 406–422.
28. Miller, J.F.: An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the 1st Genetic and Evolutionary Computation Conference. Volume 2., San Francisco, CA, Morgan Kaufmann (1999) 1135–1142.
29. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Proceedings of the 3rd European Conference on Genetic Programming. Berlin, Springer-Verlag (2000).
30. Schwefel, H.P.: Numerical Optimization of Computer Models. John Wiley & Sons, Chichester, UK (1981).
31. Bäck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolutionary strategies. In Belew, R., Booker, L., eds.: Proceedings of the 4th International Conference on Genetic Algorithms. San Francisco, CA, Morgan Kaufmann (1991) 2–9.
32. Mühlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation **1** (1993) 335–360.
33. Vassilev, V.K., Fogarty, T.C., Miller, J.F.: Information characteristics and the structure of landscapes. Evolutionary Computation **8** (2000) 31–60.