The Input Pattern Order Problem: Evolution of Combinatorial and Sequential Circuits in Hardware

Martin Trefzer, Tüze Kuyucu, Andy Greensted, Julian Miller and Andy Tyrrell

Department of Electronics[†], University of York, UK

Abstract. Evolution is particularly good at finding specific solutions, which are only valid for exactly the input and environment that are presented during evolution. In most evolution experiments the *input pattern order problem* is not considered, even though the ability to provide a correct result for any input pattern is a prerequisite for valid circuits. Therefore, the importance of including randomness in the input pattern applied during evolution is addressed in this paper. This is shown to be mandatory—particularly in the case of unconstrained intrinsic evolution of digital circuits—in order to find valid solutions. The different ways in which unconstrained evolution and constrained evolution exploit resources of a hardware substrate are compared. It is also shown that evolution benefits from versatile input configurations. Furthermore, hierarchical fitness functions, previously introduced to improve the evolution of combinatorial circuits, are applied to the evolution of sequential circuits.

1 Introduction

In evolvable hardware the correct operation of the evolved circuits is crucial. It is important that the evolved circuits provide a fully functional device, which meets the design specifications. Evolution is particularly good at finding specific solutions, unfortunately these are likely to be only valid for exactly the set of inputs and the environment that are presented during evolution. Even when certain parameters are varied, e.g. the location on the substrate where the candidates are tested, or the order of the input vectors, evolution is likely to produce circuits that only meet these minimal requirements. In the worst case, an evolved circuit can be just a pattern generator that always generates the desired output irrespective of the applied inputs. As a consequence, the resulting circuits are not fully functional. Although this is particularly an issue in the case of circuits that are evolved on systems that feature sequential components and delays, it might also occur in the case of combinatorial circuits.

There are only a few examples where input pattern problems are discussed, e.g. [1]. Usually the work done is either related to validation and built-in-self-testing (BIST) [2, 3] or sets the focus on the fitness function [4] rather than the input pattern.

In this paper we address the input pattern order problem (IPOP) using unconstrained evolution on the reconfigurable integrated system array (RISA) evolvable hardware

[†] This work is part of a project that is funded by EPSRC - EP/E028381/1.

http://www.bioinspired.com - {mt540,tk519,ajg112,jfm7,amt}@ohm.york.ac.uk

platform [5], which features both combinatorial and registered logic as well as sequential feedback loops. Hence, RISA allows for asynchronous loops to occur and this makes it effectively impossible to perform the presented experiments in simulation. In this respect, a hardware substrate yields a richer variety of physical effects to be exploited by evolution.

Since the structure and behaviour of an evolving circuit are unknown, hence evolutionary algorithms (EAs) are working on a black box [1], it is necessary to include randomness in both inputs and environment of the evolving circuit. Tone discriminators and 4 bit parity circuits are evolved and tested for their ability to cope with input patterns that are random and therefore previously unknown to the circuits evolved. The paper also presents an investigation and comparison of the behaviour of circuits found by means of unconstrained evolution to those obtained through constrained evolution. In addition, the concept of hierarchical fitness functions, introduced in [6] to improve the evolution of combinatorial circuits, is now applied to the evolution of sequential circuits. It is investigated whether this kind of fitness function improves the validity of the found solutions and to what extent versatile input configurations improve evolution.

2 RISA Hardware Evolution Platform

The reconfigurable integrated system array (RISA) is a reconfigurable digital device, which was designed as a platform for intrinsic hardware evolution and development at the Department of Electronics, University of York. One RISA chip provides both a programmable microcontroller and configurable logic, which are inspired by the main constituents of biological cells, namely the nucleus and the cell body, as shown in figure 1.

The custom designed microcontroller on RISA is called a simple networked application processor (SNAP), and the configurable logic is designed in a similar fashion to field programmable gate arrays (FPGAs). In this paper, FPGA will refer to RISA's FPGA fabric unless indicated otherwise. Like a biological cell's body, the FPGA fabric carries out the tasks of the respective RISA module. Additionally, configurable logic of different RISA modules can be directly interconnected, in order to build larger circuits or, in terms of biology, larger organisms.

In addition, the FPGA offers features that make it particularly suitable for evolution experiments: first, it is designed in a way that it cannot be destroyed by random bit strings. As a consequence, as concluded in [7], unconstrained evolution can take place. The latter feature is not generally present in current commercial FPGAs: the synthesis tools of the manufacturers either constrain the access to the bit-string, in order to protect the device, or it is actually possible to destroy it. Second, the configuration of clusters can be changed independently from each other, hence, the logic offers partial reconfiguration. This can considerably accelerate hardware evolution [8], since only those parts of the bit-string, which have actually been changed by the EA, need to be reloaded into the device, instead of reconfiguring the entire device. A detailed description of RISA can be found in [5, 9].

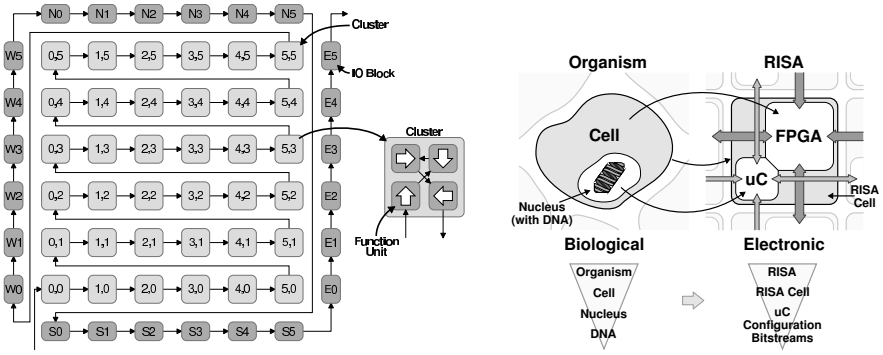

Fig. 1. *Left:* the FPGA substrate of RISA consists of an array of 6×6 functional clusters surrounded by input/output (IO) blocks. Each cluster and IO block can be configured individually, providing partial reconfiguration. Each cluster provides four functional units that can either be configured as 16 bit look-up table (LUT), shift register or random access memory (RAM). Thus, RISA offers a rich variety of configuration options: 152 bits are required to configure the logic and 320 bits are required to configure the routing of one cluster, resulting in a total of 16992 bits for the whole configuration bit-string. *Right:* the structure of the RISA cell is inspired by biological cells. The microcontroller operates as a centre for cell operations, controlling the cell functionality implemented in the FPGA fabric.

3 Evolution of Valid Circuits in Hardware

This paper presents investigations into a number of fundamental issues that have a significant effect on the success of intrinsic evolvable hardware systems:

Randomness of the Input Pattern: the set of input patterns consists of the finite number of entries of the truth table, which all have to be included into the test pattern in order to entirely define the desired digital circuit. It is not possible to divide it into partitions and apply only a fraction during evolution. Unfortunately, there is yet no algorithm that is able to automatically generate a suitable set of test vectors, which are then guaranteed to cover all relevant test cases to fully assess a found solution [10]. As a consequence, it is even harder to assess an evolving, not yet converged circuit in an evolutionary experiment.

Moreover, in the case of sequential circuits or a hardware evolution substrate that features feedback loops and delays, it is not sufficient to measure the whole truth table only once, due to the fact that one test vector can alter the state of the candidate circuit and therefore change the result of subsequent test vectors. Thus, it will be likely that evolution finds circuits that produce the desired output, which might not be correlated to the input and will therefore generally fail for random test patterns.

Versatile Input Patterns: a setup where the test-pattern is applied to multiple different inputs of the evolution substrate, in order to provide richer input to EA is referred to as *versatile input*. Particularly in the case of intrinsic hardware evolution, this makes it easier for the EA to cope with predefined fixed routing. Furthermore, in the case of un-

constrained evolution, it becomes less likely that the input is completely disconnected from the circuit by disabling one connection.

The concept of *versatile input* could easily be extended to *versatile output*. In the latter case, multiple outputs of the circuit would be monitored and evolution might produce the desired pattern at one of these outputs, rather than being forced to use one predefined output.

Fitness Measuring Methods: a further question is whether the fitness measuring methods—particularly hierarchical fitness evaluation methods that are introduced in [6]—have an effect on the ability of an evolved circuit to cope with random, unknown input test vectors. When *random input patterns* are applied during evolution, a desired property of the fitness function is not to immediately dismiss candidate circuits that obtain low fitness in only one case. At the same time, it is not supposed to promote bad solutions when they obtain a high fitness by chance.

Constrained vs Unconstrained Intrinsic Evolution: one of the key questions when using intrinsic hardware evolution is whether unconstrained evolution, evolution at the bit level, yields unforeseen behaviour by exploiting physical properties of a given substrate. In the case of the experiments presented in this paper the additional question arises whether unconstrained evolution or constrained evolution is more likely to find valid circuits. In both cases, sequential and combinatorial logic is available to the EA. However, in the case of constrained evolution, the routing is simplified as described in section 5.

4 Validating Evolved Circuits

Three different tests are carried out in order to assess the evolved circuits: first, the success rate for random test patterns is measured. Second, candidate circuits are measured at different locations on the chip. Third, different sampling frequencies are tested and the frequency discrimination range of the tone discriminator is determined.

Success Rate for Random Test Patterns: the most effective test to assess whether a resulting circuit is sufficiently fit to cope with previously unknown input vectors, is to measure its output multiple times applying random test patterns. The success rate of the latter measurement provides a measure for the validity of the evolved circuit.

In order to investigate the effect of including randomness in the input pattern, three different methods of organising the input pattern are used for the experiments in this paper: first, the *static ordered input pattern*, where the input pattern is fixed during evolution and the samples are ordered according to the truth table of the logic function. Second, the *static random input pattern*, where the input pattern contains all entries of the truth table of a logic function in random order. Third, *random input patterns*, which are newly created for each generation in order to prevent evolution to exploit regularities in the input pattern. Therefore, evolution is driven to find more general solutions and is kept away from not optimal solutions caused by static input patterns. However, using *random input patterns* makes it also harder to find solutions.

Measuring at Different Locations on the Chip: a valid circuit is expected to be independent of the particular location of the chip it has been evolved on. Hence, the presented evolved circuits are tested at different locations of the chip, whenever the architecture allowed a move to another location. This mechanism could also be exploited to produce even more robust circuits by evaluating candidate solutions at different locations (or on different substrates) during the course of evolution. There are currently only a few examples where this has been done [11, 12].

Testing at Different (Unknown) Frequencies: there are three places where timing matters in hardware: first, when the logic of the evolution substrate requires a settling-time, which is dependent on the current configuration. Second, when the sampling of input and output can be done at different frequencies and third, when the input pattern itself contains temporal information (different frequency components). In the first case, it is usually sufficient to measure the output slow enough, in order to guarantee that a worst case settling-time has elapsed. The second and third cases represent the same problem; for example, having the frequency components of the input pattern slower is equivalent to measuring at a higher sampling rate. The sampling rate should be chosen between twice the speed of the fastest change in the input pattern and the actual operating frequency of the substrate in order to detect all changes in the output signal.

The evolved circuits presented in this paper are tested to work at different sampling frequencies, and the frequency range for which tone discriminators still work is determined.

5 Experimental Setup

A total of 10 clusters of the RISA chip is used for the evolution of both tone discriminators and 4 bit parity. The remaining 26 clusters are configured in a way that they pass incoming signals unchanged to their opposite side. This ensures that the circuit's output reaches the IO blocks and can be measured from outside the chip. Inputs are applied to the west side of the chip and the output is measured at the east side, as depicted in figure 2.

RISA is operated at a frequency of 4 MHz and the inputs and outputs are sampled with a frequency of 0.5 MHz, in order to account for the delay of the input and output buffers of the IO blocks and the expected delay of the candidate circuits. The measuring and the EA are carried out with a Spartan3 FPGA. The input and output patterns consist of 512 samples for each measuring cycle. Thus, there is a fair amount of space for a redundancy of $32\times$(the full truth table) of 4 bit parity and about 100 frequency samples for the tone discriminator. The three different kinds of input patterns, described in section 3, are used for the evolution of the circuits: *static ordered input pattern*, *static random input pattern* and *random input pattern*. A $2+5$ evolutionary strategy is used, with a fitness proportional mutation rate of 1%..10%. 20 randomly initialised evolution runs have been carried out for all experiments and the generation limit is 5000.

Two different genotypes are used for the experiments: first, in the case of unconstrained evolution, the genotype is represented by the full configuration bit string that

is necessary to configure the RISA FPGA (4720 bits for 10 clusters). Second, in the case of constrained evolution, the genotype contains a reduced set of 34 routing bits, which are mapped to simplified, predefined routing patterns. The bits that configure the LUTs, however, are still present, however the routing within the function units is also simplified. Local and inter-cluster feedback loops as well as delays are still possible for the EA. The constrained genome consists of a total of 58 bits per cluster.

6 Evolution of a Tone Discriminator

Tone discriminators are both non-trivial examples for sequential circuits and widely known in the field of evolvable hardware and evolutionary computation [13]. They are useful for testing the ability of an EA to generalise, particularly in the case of intrinsic hardware evolution, because feedback and delays are required to find a solution. The task is to distinguish between a 31.25 kHz and a 250 kHz tone. Evolved solutions are evaluated with random input test patterns and at different locations on the chip. Furthermore, it is tested whether solutions found are able to distinguish frequencies, other than the ones they are evolved for.

A series of six experiments is carried out for the constrained evolution of tone discriminators. The experiments cover the three different ways of generating and applying the input, and it is investigated whether evolution benefits from a more versatile input setup as described in section 3. Additionally, hierarchical fitness functions are applied to the evolution of sequential circuits and the performance of bitwise fitness calculation is compared to hierarchical bit-string sampling (HBS) fitness, introduced in [6]. In the case of unconstrained evolution, four experiments are performed leaving out those with the static input patterns.

6.1 Results

As can be seen from table 1, it is mandatory to use random input patterns in order to evolve valid sequential circuits. In the case of the *static ordered input patterns*—which is considered to be the easiest case—perfect solutions for tone discriminators were found. However, not surprisingly, these solutions are specific to the input pattern used and therefore fail any test with random input patterns and no longer work when the input used during evolution is only slightly changed. Premature convergence and evolution not being able to exploit memory as function generators, are the reasons why no solution is found in the constrained *static random* case.

The runs where the versatile input configuration is used, as opposed to the case where only one input is present indicate that, in the case of the tone discriminators, the EA benefits from input signals that are more easily accessible from different locations. This confirms also the observation, which has been made throughout the course of all experiments and which is possibly true for most intrinsic hardware evolution experiments: the task of routing and distributing signals on a given substrate is much more difficult than solving the computational task itself.

In all experiments, the HBS fitness evaluation performs worse than the bitwise fitness calculation. It is shown in section 7.1 that the reason for this is rather the nature

Table 1. The results for the tone discriminators and 4 bit parity are shown. The number of runs where a perfect solution was found during evolution is given in column *solution found*. The size of the subset of the latter circuits, which are successfully reloaded to the chip and tested with multiple (20) random input patterns is given in column *input pattern*. Solutions, which also pass the test when measured at a different location on the chip are given in column *different location*.

genome	fitness	input pattern	tone discriminators solution found	tone discriminators random pattern	tone discriminators different location	4 bit parity solution found	4 bit parity random pattern	4 bit parity different location
constr.	bitwise	static ordered	8	0	0	10	0	0
constr.	bitwise	static random	0	0	0	9	5	5
constr.	bitwise	random versatile	9	9	8	-	-	-
constr.	bitwise	random	6	4	4	7	6	6
constr.	HBS	static random	-	-	-	10	9	9
constr.	HBS	random versatile	6	6	6	-	-	-
constr.	HBS	random	2	0	0	6	6	6
unconstr.	bitwise	static random	-	-	-	1	0	0
unconstr.	bitwise	random versatile	10	7	6	-	-	-
unconstr.	bitwise	random	7	0	0	2	1	1
unconstr.	HBS	static random	-	-	-	8	2	2
unconstr.	HBS	random versatile	2	0	0	-	-	-
unconstr.	HBS	random	0	0	0	9	3	3
unconstr. seq.	bitwise	random	-	-	-	4	4	3
unconstr. seq.	HBS	random	-	-	-	11	10	10

of the task than the unsuitability of the hierarchical fitness evaluation for sequential circuits.

None of the solutions found work at different sampling frequencies of the IO signals. This is not surprising, due to the fact that tone discriminators are inherently strongly depending on timing and, moreover, applying the input at a different sampling rate is equivalent to applying other input frequencies. Furthermore, the ability of the resulting circuits to distinguish other pairs of frequencies than those demanded during evolution has been tested. Indeed the evolved circuits are able to correctly distinguish different frequencies: the typical range for the lower frequency is $0..62.5\,\mathrm{kHz}$. The range for the higher frequency is $125..250kHz$.

6.2 Resource Consumption of Constrained and Unconstrained Evolution

The clusters, which are actually relevant for the operation of a solution found, are identified by successively clamping configuration bits to zero until the fitness gets worse. Analysing the resource consumption of typical solutions found with constrained and unconstrained evolution, reveals the different behaviour of the two approaches: in the case of unconstrained evolution almost all features of the chip and all provided clusters are exploited, in order to find solutions, whereas in the constrained case the resource

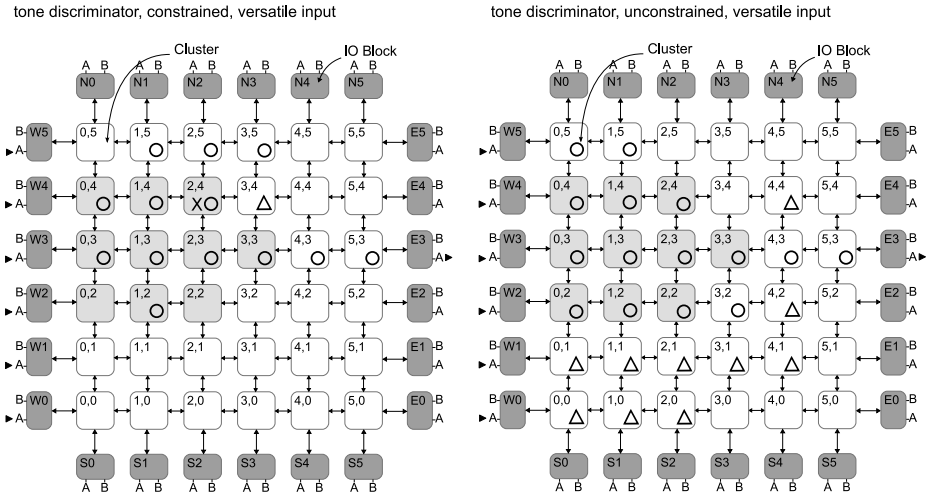

Fig. 2. A graph of the RISA clusters that are relevant for the operation of typical solutions found with constrained and unconstrained evolution respectively. Clusters that were relevant for all tested random input patterns are marked with a circle. Those, which became relevant in some cases are marked with a triangle and those which are no longer necessary when the ones with the triangles become relevant are marked with a cross.

consumption is much lower. In the latter case, the reduced design space is already taken into account.

It is interesting to see that unconstrained evolution even exploits clusters, which are not explicitly subject to evolution and configured with the default, unchanged pass through configuration. This effect is also present in the case of constrained evolution, however, it is stronger in the unconstrained case. Another surprising observation is related to testing with random input patterns: in order to provide the correct output for certain input patterns, additional clusters became relevant, as can be seen from figure 2.

7 Evolution of 4 Bit Parity

The evolution of 4 bit parity is chosen as an example for a combinatorial circuit. It is investigated whether random input patterns are also necessary to find valid solutions for combinatorial circuits. For both constrained and unconstrained evolution the results obtained with *random input patterns* are compared to those with *static random input patterns*. As with the tone discriminator experiments, the validity is tested with random input patterns. Additionally, the HBS fitness function is compared to the bitwise fitness calculation. A series of four experiments is carried out for constrained and unconstrained evolution respectively. In the case of constrained evolution, an additional series of experiments is carried out using a *static ordered input pattern*.

Owing to the question that arose in the results for the tone discriminators (6.1) whether the performance of HBS is problem specific or does not work well for sequential circuits, two more series of experiments have been undertaken, where the 4 bit

parity task is set up in a sequential fashion. A stream of bits is thereby applied to only one input of the circuit and the output shall always deliver the even parity result for window of the last 4 bits.

7.1 Results

The results in table 1 show that in the case of a combinatorial circuit, valid solutions are found in both cases using *static random input patterns* and *random input patterns*. Among the perfect solutions found, there were always solutions that passed the tests with new random inputs and at different locations on the chip. The success rate is thereby higher in the case of constrained evolution than for unconstrained evolution.

The HBS fitness method works equally well as the bitwise fitness calculation in the case of constrained evolution and it features a significantly better performance in the case of unconstrained evolution. When the 4bit parity task is implemented in a sequential way, the HBS fitness method works equally well as the bitwise fitness calculation. This suggests that hierarchical fitness evaluation is not limited to combinatorial circuits but can also be applied to sequential problems. The fact that HBS performs worse than the bitwise fitness in the case of the tone discriminators indicates that its performance is task dependent, rather than implementation dependent.

Furthermore, it is satisfying to observe that, in the case of a combinatorial circuit, typical solutions are working at different sampling frequencies between 2.5 kHz..2 MHz. The upper limit is due to implementation issues, when the input pattern is applied faster than the chip is actually clocked.

8 Conclusion & Future Work

In this paper the importance of the way in which inputs are applied during the evolution of digital circuits has been discussed. Furthermore, it was shown that the concept of hierarchical fitness functions, introduced in [6], is equally suitable to improve the evolution of sequential circuits as it is for combinatorial ones. The results presented are obtained with intrinsic constrained and unconstrained evolution on RISA, a unique hardware platform designed for evolutionary computation. The concepts presented are successfully applied to the evolution of tone discriminators and 4 bit parity, which are well known examples for a sequential and combinatorial circuit respectively.

In most evolution experiments the IPOP is not considered, although the ability to provide a correct result for any input pattern is the prerequisite for any valid circuit. The results show that it is mandatory to include randomness in the input patterns applied during the course of evolution, in order to obtain solutions, which are able to cope with any order or sequence of inputs. During the course of the experiments it has been observed that randomising the input also helps to keep evolution out of local optima, hence, to increase the yield of successful runs.

Furthermore, it is shown that; valid solutions work at different locations on the chip, tone discriminators can work on a wider range of frequencies than those they are evolved for, and solutions for the 4 bit parity work at different input sampling frequencies. These results suggest that the methods presented are able to find general and

reliable solutions for combinatorial and sequential circuits. Particularly in the case of unconstrained intrinsic hardware evolution, the results for the tone discriminator suggest that the evolution process has been further improved, when versatile input was provided. The different behaviour of constrained and unconstrained evolution was shown by resource consumpion analysis. Hierarchical fitness functions are successfully applied to sequential circuits for the first time.

Future work will apply these methods to the intrinsic unconstrained evolution of circuits with a large number of inputs and outputs. The aim will be to investigate the evolution process when increasing the randomness of the inputs and the versatility of the environment. It would be interesting to be able to provide information about which kinds of problem domains will unconstrained provide advantages as opposed to constrained approaches.

References

1. Imamura, K., Foster, J.A., Krings, A.W.: The test vector problem and limitations to evolving digital circuits. In: EH '00: Proceedings of the 2nd NASA/DoD workshop on Evolvable Hardware, Washington, DC, USA, IEEE Computer Society (2000) 75
2. Skobtsov, Y.A., Ivanov, D.E., Skobtsov, V.Y., Ubar, R.: Evolutionary approach to the functional test generation for digital circuits. In: In Proc. of 9th Biennial Baltic Electronics Conf., BEC 2004, Tallinn Univ. of Techn. (October 2004) 229–232
3. Corno, F., Prinetto, P., Reorda, M.: A genetic algorithm for automatic generation of test logic for digital circuits. In: Proceedings of the IEEE International Conference On Tools with Artificial Intelligence, Toulouse, France (November 1996)
4. Torresen, J.: A dynamic fitness function applied to improve the generalisation when evolving a signal processing hardware architecture. In: Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002, London, UK, Springer-Verlag (2002) 267–279
5. Greensted, A., Tyrrell, A.: Extrinsic evolvable hardware on the RISA architecture. In: Proceedings of 2007 International Conference on Evolvable Systems. (September 2007)
6. Kuyucu, T., Trefzer, M., Greensted, A., Miller, J., Tyrrell, A.: Fitness functions for the unconstrained evolution of digital circuits. In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong (June 2008)
7. Hollingworth, G., Smith, S., Tyrrell, A.: The safe intrinsic evolution of virtex devices. In: 2nd NASA/DoD Workshop on Evolvable Hardware, Silicon Valley, USA (July 2000)
8. Hollingworth, G., Smith, S., Tyrrell, A.: The intrinsic evolution of virtex devices through internet reconfigurable logic. In: 3rd International Conference on Evolvable Systems: from Biology to Hardware, Edinburgh,, Springer-Verlag (April 2000) 72–79
9. Greensted, A., Tyrrell, A.: RISA: A hardware platform for evolutionary design. In: Proceedings of 2007 IEEE Workshop on Evolvable and Adaptive Hardware. (April 2007)
10. Kwan-Ting Cheng, V.D.A.: Unified methods for vlsi simulation and test generation, Kluwer Academic Publishers (1989)
11. Langeheine, J.: Intrinsic Hardware Evolution on the Transistor Level. PhD thesis, Rupertus Carola University of Heidelberg, Seminarstrasse 2, 69120 Heidelberg (July 2005)
12. Stoica, A., Zebulum, R.S., Keymeulen, D.: Mixtrinsic Evolution. In: Proc. ICES 2000, LNCS 1801, Edinburgh, UK, Springer Verlag (April 2001) 208–217
13. Thompson, A., Layzell, P., Zebulum, R.S.: Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. IEEE Trans. on Evolutionary Computation **3** (September 1999) 167–196