

Proceedings of the 30th IEEE International Symposium on Multiple Valued Logic, Portland, Oregon, USA, 2000, pp.253-258

Evolutionary Multi-Level Network Synthesis in Given Design Style*

T. Łuba

Warsaw University of Technology, POLAND,
luba@tele.pw.edu.pl

C. Moraga

Dortmund University, GERMANY,
moraga@cs.uni-dortmund.de

S. Yanushkevich

M. Opoka

V. Shmerko

Technical University, Szczecin, POLAND, yanushkevich@wi.ps.pl

Abstract

This paper extends the technique of evolutionary network design. We study an evolutionary network design strategy from the position of design style. A hypothesis under investigation is that the uncertainty of a total search space (the space of all possible network solutions) through evolutionary network design is removed faster if this space is partitioned into subspaces. This idea has been realized through a parallel window-based scanning of these subspaces. Such a window is determined by the parameters of a multi-level network architecture in a given design style. Our approach allows to synthesize networks with more than two hundred quaternary gates. Moreover, we show that information theoretical interpretation of the evolutionary process is useful, in particular, in partitioning of network space and measuring of fitness function. The experimental data with 6-input quaternary and 11-inputs binary benchmarks demonstrate the efficiency of our program, EvoDesign, and an improvement against the recently obtained results.

* Supported in part by visiting grant Poland-Belarus-Germany from the Committee of Scientific Researches (POLAND)

1 Introduction

There are different scenarios of network design in Computer Aided Design (CAD) systems, with respect to functions of CAD tools, levels of machine learning support, human-machine interface or, in general, regarding designer's role through the network design process. We solve the problem: *generate multi-level logic networks automatically in different design styles*. In contrast to recently developed algorithms, we study the properties that characterize flexibility of network design based on evolutionary paradigm.

The idea of evolutionary network design is very captivating and attractive, because the designer needs no special software to implement a given architecture and design style. These are provided by evolutionary (adaptively) search for the network solution, i.e. the designer becomes an expert. Unfortunately, this idea has been demonstrated on small networks only (2-6 bit multiplexers, 2-5 bit adders and multipliers, 2-bit decoder, 4-8-state machine, 3-bit counter) because of many difficulties [1], [6], [9], [11], [12]. A generalization of some aspects toward multiple-valued logic (MVL) networks is known too, for example [7]. More details on current work and trends in evolutionary circuit design can be found in

[14]. In this paper, we synthesize 6-input quaternary and 11-inputs binary networks in a given design style, and, for the first time, manipulate with more than 200 quaternary gates (in previous studies, the concept of style has not been taken in attention).

There are many obstacles for network evolutionary design: the runtime for the evolving process is too long (sometimes a simple network requires dozens hours to be synthesized) and there are many invalid networks generated during the process. So, any of recent results cannot be discussed in the context of industrial requirements yet.

Based on the analysis of various evolution strategies for network design, we substantially suggest an extension of the existing technique and the ways to improve the recently obtained results as follows:

(i) Partitioning of a network search space into subspaces via decomposition of initial logic function; below we show that this approach extremely improves results even for simplest evolutionary strategy and simplest decomposition.

(ii) Massive parallel window-based scanning of subspaces of possible network solutions; this innovation, as one of the main distinctive features of our approach, follows directly from the partitioning of network space into subspaces and gives us an opportunity to scan subspaces independently; one can consider this as most powerful resource (together with the known parallel genetic algorithms (GAs) [3], [15] for reduction of executing time.

(iii) The information theory paradigm, recent results on its applications to CAD problem and stochastic nature of GA, - all of these factors are attractive for interpretation and estimation of the evolutionary process in notation of quantity of information; in this paper we continue our investigation in this direction.

2 Technique of evolutionary network design

In contrast to the rule-based evolutionary technique that deals with forms of representation of logic functions (LITERAL/MIN/SUM, AND/EXOR and so on) and explores algebraic rules, the idea of the evolutionary network design is that network structure is adaptively searched in the space of possible network solutions. In this paper we consider the evolutionary network design under a given scenario, i.e. multi-level network architecture in a *target design style* (TDS) that is defined over a fixed library of gates \mathcal{L} .

Verification of the evolved network consists in testing whether the evolved network performs the given truth table of a function and comparison of the achieved architecture with the desired parameters.

To the best of our knowledge, in recent results on evolutionary network design, they were not concentrated on design style. The crucial idea of the previously proposed

approaches can be defined as *free* choosing (with some limitation) a gate from a cell library \mathcal{L} and creating a network based on evolutionary paradigm.

The focus of our study is a window-based scanning process of the network search subspaces, each in different TDS. This is one of the main components of our evolutionary design scenario. We consider a K -valued logic networks defined over a fixed library \mathcal{L} .

Definition 1 A TDS of a multi-level network specifies types of logic cells from the library \mathcal{L} of cells, architecture of network (permissible interconnections between cells, levels, inputs and outputs of the network), the maximal number M of levels, the maximal number N of gates and types of gates in each level of the evolved network.

TDS is closely related to a set of characters to be implemented by a chromosome to evolve network, i.e. alphabet.

In this paper we consider a *normal* networks only, i.e. network that corresponds a $\mathcal{G}_{M \times N}$ scanning window over a fixed library of cells \mathcal{L} if none of gate outputs in a given level is connected to an input of another gate from this level.

Suppose that the network space is partitioned into R parts. Let us evolve a network for each part in the same TDS by *scanning window* $\mathcal{G}_{M \times N}$. For example, for a switching function f , $R = 2^d$, where $d < n$ is the number of decomposed variables within n variables of the function f .

The TDS is a tool that allows us 'to scan' the network search space in order to implement a desired logic function by a network. The scanning process has to be optimized.

It is worthy to mention that realization of the TDS is closely related to the well-known in CAD concept of permissible functions and perturbations in multi-level network optimization (see, for example, [8]).

3 Simple experiment

We start with simple experiments as motivation to use TDS and to demonstrate the efficiency of our idea to use window-based scanning with given TDS.

AND-OR-EXOR 3-level network is one of the simplest 3-level architecture. Figure 1 illustrates two network solutions obtained with respect to different design strategies. The network in Fig.1(a,b) has been synthesized based on *Sasao-Debnath* formal method [5]. Note that complemented variables in this style are available as inputs, and each gate has unlimited fan-in and fan-out. The networks in Figure 1 (b) was evolved by a simple GA in AND-OR-EXOR TDS (in 9947 iteration (about one hour) based on a 5-level $\mathcal{G}_{5 \times 5}$ scanning window. This is because the size of a scanning window is dynamically changed through evolutionary process in accordance with

the equation $\mathcal{G}_{M_i \times N_j} \subseteq \mathcal{G}_{M \times N}$ for integer positive numbers $M_i \leq M, N_j \leq N$ ($M, N \geq 2$). The network given in Figure 1(c) was created in AND-OR-EXOR TDS over the two-input gates. If we do not use TDS, we obtain network (d).

Observation 1 *The $\mathcal{G}_{M \times N}$ scanning window in AND-OR-EXOR TDS allows us to flexibly manipulate types of gates and topology of the created 3-level network. Only limited knowledge about the problem is necessary to achieve satisfactory solution.*

4 Information estimations as fitness function

In contrast to the recently developed approaches, we consider all phases of evolving network process from information theoretical viewpoint, i.e. scanning of a network subspaces, information content of library of cells and scanning window *etc.* In this paper we demonstrate our results on partitioning the space of possible network solutions and estimation of fitness function. The justification of our approach is grounded on the fact that we use a unified measure (entropy) for evolving process that is of stochastic nature as well. Moreover, it is well known that entropy of a logic network is sensitive to the types of gates, number of inputs, outputs and connections [4].

4.1 Technique of information measure

Let us $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n\}$ with the probabilities p_1, \dots, p_n be a complete set of events. The entropy of the set be $H(\mathcal{A}) = -\sum_{j=1}^n p(\mathcal{A}_j) \cdot \log p(\mathcal{A}_j)$, where $p(\mathcal{A}_j)$ is the probability of the event \mathcal{A}_j in the partition and logarithm is of base 2. Note that $\log p_k \leq 0$ and so $H(\mathcal{A}) \geq 0$, i.e. the entropy never be negative. The entropy is zero if and only if there is complete certainty. For finite fields of events \mathcal{A} and \mathcal{B} with probabilities $\{p_j\}, j = 1, \dots, n$, and $\{q_k\}, k = 1, \dots, m$, respectively, the conditional entropy of field \mathcal{A} with respect to \mathcal{B} is defined by $H(\mathcal{A}|\mathcal{B}) = -\sum_{j=1}^n \sum_{k=1}^m q_k \cdot p_{kj} \cdot \log p_{kj}$. The mutual information of two finite fields \mathcal{A} and \mathcal{B} is defined by $I(\mathcal{A}; \mathcal{B}) = H(\mathcal{A}) - H(\mathcal{A}|\mathcal{B})$.

Example 1 *The entropy of a switching variable x is the function of its signal probability, $H(x) = -p \cdot \log p - (1-p) \cdot \log(1-p)$.*

Example 2 *For the 4-valued gate MIN, the input and output entropy be $H(X) = -16 \cdot \frac{1}{16} \cdot \log_4 \frac{1}{16} = 2$ bit and $H(f) = -\frac{7}{16} \cdot \log_4 \frac{7}{16} - \frac{5}{16} \cdot \log_4 \frac{5}{16} - \frac{3}{16} \cdot \log_4 \frac{3}{16} - \frac{1}{16} \cdot \log_4 \frac{1}{16} = 0.87$ bit accordingly. The information theoretic measure of a gate is equal to $I_{gate} = H(X) - H(f) = 1.13$ bit. The conditional entropy is $H(f|x_0) = H(f|x_1) = -\frac{4}{16} \cdot \log_4 \frac{4}{4} - \frac{7}{16} \cdot \frac{1}{16} \cdot \log_4 \frac{1}{4} - \frac{3}{16} \cdot \log_4 \frac{3}{4} - \frac{2}{16} \cdot \log_4 \frac{2}{4} = 0.54$ bit.*

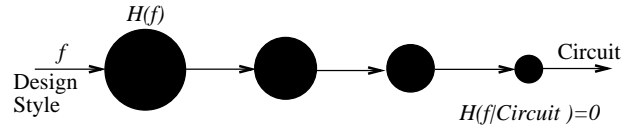


Figure 2. The uncertainty of a network search space is removed by window scanning

4.2 Entropy based fitness function

Of particular interest in our study is entropy based estimation of fitness function (the function to be optimized).

Consider the evolving process for a logic function f in terms of uncertainty.¹ The start state of the process, given the truth table of the function, can be characterized by the entropy $H(f)$ of the function f . Next we will obtain the current solutions Net_t for the output of the network. Let us consider a conditional entropy $H(f|Net_1), \dots, H(f|Net_u), H(f|Net)$, i.e. entropy of the function f with respect to the current solution, as an information measure for the network design process. Figure 2 interprets this process as $H(f|Net_1) = H(f) > H(f|Net_2) > H(f|Net) = 0$, where $H(f|Net) = 0$ is the estimation of a satisfactory solution.

Mutual information $I(f; Net)$ between functions f and Net can be interpret as an amount of the logical work required to design a network. We can conclude that the mutual information $I(f; Net)$ and conditional entropy $H(f|Net)$ are related to the entropy $H(f)$ as $I(f; Net) = H(f) - H(f|Net)$.

We interpret the fitness function in the GA as conditional entropy of a target function f given a current function Net

$$H(f|Net) = -\sum_{j=0}^{K-1} \sum_{i=0}^{K-1} p_{ij} \log \frac{p_{ij}}{p_j}, \quad (4.1)$$

where f is the given function and Net is the evolved function; p_{ij} is the probability that functions f and g takes values i and j correspondingly, and p_j means the probability that function Net takes value j . Since $p_{ij} = k_{ij}/k$ and $p_j = k_j/k$, where k_{ij} represents the number of combinations (i, j) of values of the functions f and Net ; k_j represents the number of values j in the Net ; and finally k is the total number of combinations. Formula (4.1) follows from the classical definition of conditional entropy. Note that $H(f|Net) = 0$ for α -order cyclic inversion $\widehat{Net}^\alpha = Net + \alpha(mod K)$.

¹ For the first time we reported these results for switching functions at the 8th Int. Workshop on Post-Binary Ultra-Large Scale Integration - ULSI'99, Germany, Freiburg, 1999.

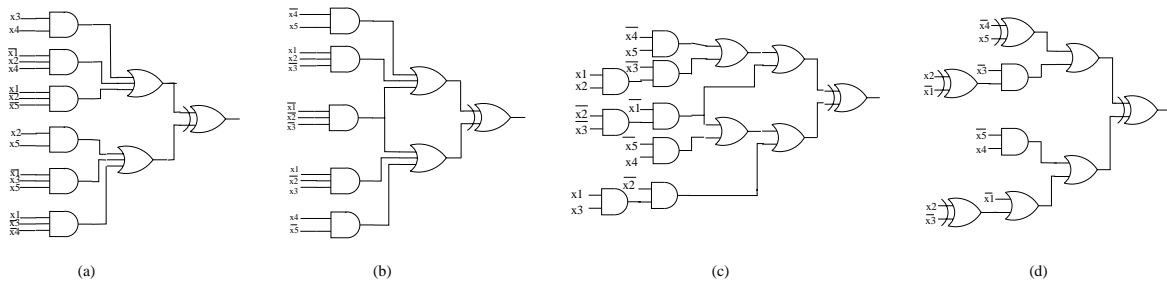


Figure 1. Synthesis of AND-OR-EXOR networks via regular and evolutionary method

Example 3 Function s (Table 1) is evaluated through evolutionary process as follows. Started with $s = Net_0 = 2$, the fitness function, in accordance with (4.1), be $H(f|Net_0) = -\sum_{i=0}^2 \sum_{j=0}^2 k_{ij} / k \log^{k_{ij}} / k_j = -(^6/_{18} \log^6 /_{18} + ^6/_{18} \log^6 /_{18} + ^6/_{18} \log^6 /_{18}) = 1.585$ bit. For instance, in 380th generation, we have obtained the truth vector $s = Net_{380} = [012020202200001012]$, so that $H(f|Net_{380}) = -(^6/_{18} \log^6 /_9 + ^1/_{18} \log^1 /_9 + ^3/_{18} \log^3 /_3 + ^2/_{18} \log^2 /_6 + ^1/_{18} \log^1 /_9 + ^4/_{18} \log^4 /_6) = 0.853$ bit. At last, when a correct network solution obtained, we get $H(f|Net_{12610}) = -(^6/_{18} \log^6 /_6 + ^6/_{18} \log^6 /_6 + ^6/_{18} \log^6 /_6) = 0$ bit.

Table 1. Ternary adder with carry

c_i	x	y	c_{i+1}	s	c_i	x	y	c_{i+1}	s
0	0	0	0	0	1	0	0	0	1
0	0	1	0	1	1	0	1	0	2
0	0	2	0	2	1	0	2	1	0
0	1	0	0	1	1	1	0	0	2
0	1	1	0	2	1	1	1	1	0
0	1	2	1	0	1	1	2	1	1
0	2	0	0	2	1	2	0	1	0
0	2	1	1	0	1	2	1	1	1
0	2	2	1	1	1	2	2	1	2

5 Partitioning of network search space

We use decomposition of the initial logic function in order to make the network search space partitioned into subspaces. In our study we explore the property of conditional entropy, to choose a decomposed variable with respect to maximal information (minimal entropy) and entropy based Shannon (S) decomposition. In logic design, S-decomposition for a switching function is defined as $f = \bar{x}f|_{x=0} \vee xf|_{x=1}$. Entropy based S-decomposition of a switching function f given variable x , is expressed by the relation below.

Definition 2 The entropy of a switching function f given variable x , $H^S(f|x)$, consists of entropies of subfunctions

given $x = 0$ as well as $x = 1$, $H^S(f|x) = p_{|x=0}H(f|_{x=0}) + p_{|x=1}H(f|_{x=1})$.

Information measure of S-expansion is equal to the conditional entropy $H(f|x)$, i.e. $H^S(f|x) = H(f|x)$.

Example 4 The conditional entropy of the function c_{i+1} (Table 1), with respect to variable c_i is $H(c_{i+1}|c_i) = -^6/_{18} \log^2 /_3 - ^3/_{18} \log^1 /_3 - ^3/_{18} \log^1 /_3 - ^6/_{18} \log^2 /_3 = 0.92$ bit. By analogy, $H(c_{i+1}|x) = -2 \cdot ^5/_{18} \log^5 /_6 - 2 \cdot ^1/_{18} \log^1 /_6 - 2 \cdot ^3/_{18} \log^3 /_6 = 1.12$ bit, and $H(c_{i+1}|y) = -2 \cdot ^5/_{18} \log^5 /_6 - 2 \cdot ^1/_{18} \log^1 /_6 - 2 \cdot ^3/_{18} \log^3 /_6 = 1.12$ bit. Variable c_i carries more information than variables x and y , therefore, in the first step of S-decomposition, we choose c_i . The conditional entropy of the function s with respect to variables c_i , x , y are equal: $H(s|c_i) = H(s|x) = H(s|y) = 6 \cdot ^3/_{18} \log^1 /_3 = 1.53$ bit.

We use the generalization of S-expansion for MVL functions in the Galois field [13] and its information theory notation [16].

6 Algorithm and experiments

Our attempts to evolve ternary and quaternary 2-digit adders and multipliers based on recently developed techniques failed.

That is why we have combined the S-decomposition of the initial function (partition of the network search space) and GA in our program *EvoDesign*. Our program have been written in C++ and performed on Pentium II 300MHz processor. The experiments include two phases. At the first phase, the search network space was partitioned into subspaces via decomposition of the initial function. At the second phase, a GA was applied.

In our GA, a population (space of network solutions) is produced, tested, scored, and selected based on survival of the fittest paradigm for reproduction and creation of the next generation. A single chromosome represents a network. During one search cycle (generation) the members of the population are ranked accordingly to a fitness function, and

those with minimal entropies are probabilistically selected to become candidates to the next generation.

6.1 Ternary and quaternary multipliers

In these series of the experiments we show that one can manipulate design style and parallel processing, i.e. use different TDSs for independent searching in each network subspace. We demonstrate in Table 2 the results of evolving 2-digit (4-inputs) ternary multiplier without carry, where $P[\%]$ is a percentage of correct solutions in 50 runs. Let us denote inputs of the multiplier as $A = \{a, b\}$, $B = \{c, d\}$ and output as Y . The space of network solutions C was partitioned by the simplest S-decomposition with respect to one variable into seven subspaces of network solutions $C_0, C_{00}, C_{01}, C_{02}, C_{10}, C_{11}$ and C_{12} . The largest subspace C_0 has been created as a result of decomposition of the initial function with respect to the first variable, $a = 0$. The residual subspaces were obtained via decomposition of the initial function with respect to two variables. For instance, the subspace C_{10} has been produced via decomposition with respect to $a = 1, b = 0$.

Table 2. Parallel synthesis of ternary and quaternary multipliers

Ternary multiplier								
	C_0	C_{10}	C_{11}	C_{12}	C_{20}	C_{21}	C_{22}	Total
P[%]	8	100	44	72	64	52	76	
Gate	15	3	13	9	12	12	9	73+3
t	249	0.03	57	45	48	56	49	249

Quaternary multiplier					
	C_{0i}	C_{1i}	C_{2i}	C_{3i}	Total
P[%]	96	41	25	50	
Gate	35	61	76	48	220+5
t	45	148	137	112	148

Parameter setting: CPU-time (in min); $P_{popul} = 60$; maximal number of generations is $2 * 10^4$, $P_{cross} = 0.7$; for the $\mathcal{G}_{8 \times 6}$ window, $P_{mutat} = 0.008$ and $L_{back} = 4$; for the $\mathcal{G}_{6(7 \times 4)}$ and $\mathcal{G}_{4 \times 4}$ windows, $P_{mutat} = 0.13$, $L_{back} = 3$.

The window-based scanning of the subspace C_0 with 8-level $\mathcal{G}_{8 \times 6}$ window allowed us to synthesize subnetwork with 15 gates of the given types. Then, we scanned the residual subspaces with 7-level $\mathcal{G}_{7 \times 4}$ window (including different limitations of types of gate from the library of cells \mathcal{L}) and found network solutions with 3, 13, 9, 12, and 9 gates of given types. So, the final network includes 76 gates (73 gates plus 3 ternary multiplexers). Its structure is shown in Figure 3. We observe, that when parallel and independent processing of network subspaces, the different TDSs can be used for each subspace.

The results of evolving 2-digit (4-inputs) quaternary multiplier without carry are presented in Table 2. The space of network solutions C was partitioned into 16 subspaces C_{0i}, C_{1i}, C_{2i} and C_{3i} , $i = 0, 1, 2, 3$. The resulting network

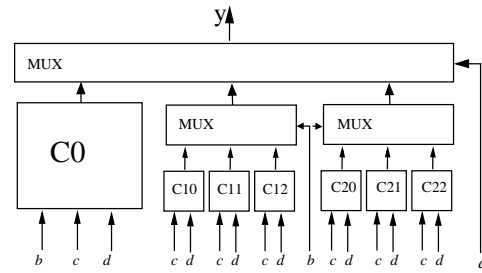


Figure 3. The structure of ternary multiplier

includes 220 gates and 5 multiplexers. We conclude that ternary and quaternary multipliers can be evolved in independent and parallel way by the window-based scanning.

6.2 Quaternary networks

In the final series of experiments we synthesized MVL networks for a given 4-valued incompletely specified functions (Table 3). First, we used $\mathcal{G}_{8 \times 6}$ window over a fixed library of cells, without partitioning of the network space. Then, we used entropy based S-decomposition with respect to one variable and $\mathcal{G}_{5 \times 4}$ window over the same library of cells \mathcal{L} , for each part of the partitioned network space.

Consider, for example, the results for benchmark monks1tr. Here, 6-gates network has been created in three hours. Then, the network space was partitioned into three subspaces C_0, C_1 and C_2 , and the resulting network with 7 gates and one multiplexer for 0.3 hour has been obtained. Because of parallel scanning, runtime was reduced in 10 times. Notice, the attempt to synthesize network monks3tr without decomposition has failed.

Table 3. Parallel synthesis of quaternary networks

Test	$\mathcal{G}_{8 \times 6}$ window		Subspaces			Gate/t
	P[%]	Gate/t	C_0	C_1	C_2	
monks1tr	98	6/3	2/0.1	2/0.3	3/0.2	7/0.3
monks1te	100	5/4	2/0.2	2/1.6	3/1.5	7/1.6
monks2tr	14.3	21/8	8/4.0	9/3.9	10/4.0	27/4.0
monks2te	100	5/3	2/0.1	2/1	3/1.5	7/1.5
monks3tr	-	-	6/6	4/4	8/4	18/6.1
monks3te	84	7/60	7/26	5/24	5/24	17/26

Note: CPU-time (t) in hours; all benchmarks are with 6 inputs and 1 output; P[%] is a percentage of correct solution in 50 runs; in all cases was used one MUX.

Table 4. The comparison of three strategies for synthesis of binary networks

Test	I/O	Entropy based S-decomposition				B-decomposition [2]			SIS [10]		
		$\mathcal{G}_{R(M \times N)}$	Gate+MUX	Level	$t(\text{min})$	$\mathcal{G}_{R^*(M \times N)}$	Gate	$t(\text{min})$	Gate	Level	$t(\text{min})$
rd53	5/3	4(4 × 4)	21+3	6	8	2(5 × 5)	16	57	34	6	0.06
con1	7/2	16(5 × 5)	13+15	9	0.25	4(5 × 5)	22	122	30	8	0.11
inc	7/9	16(3 × 6)	10+15	7	0.3	9(5 × 5)	82	220	171	10	0.3
5xp1	7/10	16(4 × 8)	121+15	8	152	10(5 × 5)	61	84	94	10	0.15
rd84	8/4	32(5 × 5)	191+31	10	40	6(5 × 5)	32	202	330	17	0.48
misex1	8/7	16(5 × 5)	1+15	8	0.5	9(5 × 5)	61	114	49	7	0.06
sao2	10/4	64(5 × 4)	120+63	11	209	20(5 × 5)	165	120	203	16	0.15
misex46	11/1	128(4 × 5)	24+127	11	0.00	6(4 × 5)	36	22	52	10	0.1
misex47	11/1	128(4 × 5)	12+127	11	0.00	7(4 × 5)	26	27	34	9	0.1

7 Concluding remarks

The main contribution of this paper is an extension of the multi-level network synthesis due to a parallel window-based scanning of the network space. We showed that evolutionary design with TDS allows to achieve flexible parallel scanning of the subspaces of possible network solutions. The proposed type of parallelism, combining with parallelization of GA [3], [15], allows to achieve massive parallelism. These two levels of the parallelization naturally lend themselves to different parallel hardware, for example, MIMD and SIMD machines. This is especially desirable in the CAD systems where parallel machines and distributed workstations are widely used.

The next innovation is using of the information measure and information-theoretical interpretation of the evolutionary process. We showed the usefulness of these measures through different phases of the evolutionary process.

8 Current and future work

We focus in our current work on different practical styles for binary and MVL networks including AND/EXOR, Kronecker and ESOP expressions. Unfortunately, the problem of optimization of a TDS through evolving process, even in most simple formulation, has not been solved yet.

However, the current results are very promising. For justification of our optimism, we give in Table 4 the fragment of our experiments for evolving binary networks. First of all, let us compare the evolved networks (in gates, levels L and time t) for two strategies of partitioning the network space, i.e. entropy based S-decomposition and so called *balanced* decomposition [2] of the initial switching function. We observe that in the most of cases, the number of gates has been considerably reduced. Note that the balanced decomposition does not require multiplexers for building a final network. For example, function **rd53** was partitioned by S-decomposition into 4 subfunction and final

4-level network with 21 gates plus 3 multiplexers *MUX* was realized by 4-parallel processing in 8 *min.* After balanced decomposition, we evolved a 5-level network with 16 gates via parallel processing of two subspaces for 57 *min.*

Observation 2 *In most cases, the partition of the network search space via balanced decomposition gives better results, in the number of gates and levels of the network realization, than S-decomposition and SIS.*

References

- [1] T. Aoki, N. Homma, and T. Higuchi. Evolutionary design of arithmetic circuit. *IEICE Transactions on Fundamentals*, E-82-A(5):798–806, 1999.
- [2] J. A. Brzozowski and T. Łuba. Decomposition of Boolean functions specified by cubes, Part 1: Theory of serial decompositions using blankets. Part 2 (with M. Nowicka): Practical results. In *Research Report CS-97-01, Dept. of Comp. Sci., University of Waterloo, Canada*, 1997, Revised in October 1998.
- [3] E. Cantú-Paz. Analysis, design and implementation of parallel genetic and evolutionary algorithms. In *Proc. Genetic and Evolutionary Computation Conf. Tutorial*, pages 421–444, 1999.
- [4] R. W. Cook and M. J. Flynn. Logical network cost and entropy. *IEEE Trans. on Computers*, C-22(9):823–826, 1973.
- [5] D. Debnath and T. Sasao. Exclusive-OR of two sum-of-products expressions: Simplification and upper bound on the number of products. In *Proc. 3rd Int. Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 45–60. Oxford, UK, 1997.
- [6] T. Higuchi, M. Iwata and W. Liu (Eds.), *Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science, vol. 1259*. Springer, 1996.
- [7] C. Moraga and W. Wang. Evolutionary methods in the design of quaternary digital circuits. In *Proc. IEEE 28th Int. Symposium on Multiple-Valued Logic*, pages 89–94, 1998.
- [8] S. Muroga, Y. Kambayashi, H. C. Lai, and J. Cullinley. The transduction method-design of logic networks based on permissible function. *IEEE Trans. on Computer Aided Design*, 38(10):1404–1424, 1989.

- [9] D. Quagliarella, J. Périaux, C. Poloni and G. Winter (Eds.). *Genetic Algorithm and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*. John Wiley and Sons Ltd, 1998.
- [10] S. Release 1.2. In *UC Berkeley Soft. Distr.*, July, 1994.
- [11] E. Sanchez, and M. Tomassini (Eds.). *Towards Evolvable Hardware, Lecture Notes in Computer Science, vol. 1062*. Springer, 1996.
- [12] M. Siper, D. Mange, and A. Pérez-Urbe (Eds.). *Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science, vol. 1478*. Springer, 1998.
- [13] R. Stanković. Functional decision diagrams for multiple-valued functions. In *Proc. IEEE Int. Symp. on Multiple-Valued Logic*, pages 284–289, 1995.
- [14] A. Stoica. Evolvable hardware: From on-chip circuit synthesis to evolvable space systems. In *this Proceedings*.
- [15] K. S. Tang, K. Man, and S. Kwong. Parallel genetic algorithms: Implementable hardware solutions. *Circuit and Systems*, 10(2):3,8–11, 1999.
- [16] S. Yanushkevich, D. Popel, V. Shmerko, V. Cheushev, and R. Stanković. Information theory approach for minimization of polynomial expressions over GF(4). In *this Proceedings*.