

Chapter 1

AN APPROACH TO FLEXIBLE MULTI-LEVEL NETWORK DESIGN

V. Cheushev, J. Kolodziejczyk, T. Luba,
C. Moraga, P. Sapiecha, V. Shmerko and S. Yanushkevich

Abstract The paper is focused on integrating the flexibility in the design of logic networks. We achieve the flexibility by several ways: evolving the design and applying the decomposition technique. To perform the circuit evolution, we use one of more powerful techniques borrowed from information theory and decision making, to evaluate the design. This technique can be used along with any multi-level networks design optimization strategy over a fixed library of cells. As an example, we use the evolutionary network synthesis and our previously introduced concept of a target design style. Also, in order to alleviate the problem of large-size network search space, we use the partition of the space by the state-of-the-art decomposition technique.

Keywords: multi-level synthesis, evolutionary design, parallelism, information theory approach

1. Introduction

Our study is relevant to the problem of design networks based on FPGA. We utilize the paradigm of flexible network synthesis [10] in order to generate networks over a given design style. The methodology we use is referred to as artificial evolution and machine learning methods of the circuitry design. Our final goal is to improve recently obtained results in evolutionary network synthesis based on information theory [9] and state-of-the-art methods of digital circuit design such as balanced decomposition [2]. Evolutionary network design approach operates with very small circuits only (2 - 6 bit multiplexers, 2 - 5 bit adders, 2-bit decoder, 4 - 8-state machine, 3-bit counter) and it is not yet of practical implementation (sometimes run time for the evolved simple network requires dozens of hours) [1], [4], [11], [5], [12].

In this paper we justify that based on concepts of scanning window and target design style in the evolutionary network synthesis, we are able to apply decomposition technique and extremely improve recently obtained results (including functions where the traditional evolutionary algorithms failed). This is because the network search space is partitionable and, therefore, parallel processing of subspaces is possible. Moreover, because genetic algorithms [15] are very good candidate for parallelization, we can achieve a massive parallel search. We get benefit from utilization of new computing resources for other purposes. In this paper we use these resources for manipulation with design styles (any limitations such as type of gates, possible connections and so on, extremely increase the time of the searching process).

2. Our strategy of the evolutionary network design

Adaptive search in the space of possible network solutions is the basic idea of the evolutionary network design. We interpret the search process as a scanning of a network space by a window and consider more complex problem, namely, the evolutionary multi-level network design under the given design style and scenario, and continue our study reported first in [7], [10], [9]. The window is a tool that allows us 'to scan' the network search space in order to implement a desired logic function by a multi-level network. Scanning window is codified by a stream of integer numbers that is a chromosome. Each gene of the chromosome corresponds to a primitive gate assigned with an integer number. So, first, one has to choose gates and codify them, along with interconnections between gates, to produce the gene encoding.

We denote a scanning window as $\mathcal{G}_{M \times N}$ that specifies the maximal number of levels M and maximal number of gates N in each level of the network realization over a fixed cell library \mathcal{L} . We consider the networks where the gate outputs in a given level are not connected to an input of another gate from this level.

The size of a scanning window $\mathcal{G}_{M \times N}$ is dynamically changed through the evolutionary process in accordance with the equation $\mathcal{G}_{M_i \times N_j} \subseteq \mathcal{G}_{M \times N}$ for integer positive numbers $M_i \leq M, N_j \leq N$ ($M, N \geq 2$).

Definition 1 *The level back $L_{back}(\omega)$ with respect to $\omega \in \{IN, OUT\}$ specifies one of three scenarios: (i) inputs IN can be connected to the first $a < M$ levels of the created circuit, $L_{back}(IN) = a$, or (ii) outputs OUT can be connected to the last $b < M$ levels of the circuit, $L_{back}(OUT) = b$, or (iii) connections of the depth a , $L_{back}(IN) = a$, and b , $L_{back}(OUT) = b$, with respect to inputs IN and outputs OUT accordingly are allowed.*

Example 1 *The simplest case, $L_{back}(IN) = 1$ and $L_{back}(OUT) = 1$, means that connections of inputs IN and outputs OUT with the first and the last levels of the created network accordingly are allowed. If $L_{back}(IN) = 2$ and*

$L_{back}(OUT) = 3$, then the depth of the connections with respect to IN and OUT is two and three accordingly, i.e. inputs IN can be connected with two first levels, and outputs OUT can be connected with three last levels of the network.

In accordance to our design style, we specify the types of logic cells from the library \mathcal{L} , architecture of network (permissible interconnections between cells, levels and inputs and outputs of the network), the maximal number M of levels, the maximal number N of gates in every level, and types of gates in every level of the evolved network.

Example 2 $\mathcal{G}_{5 \times 4}$ window over the library of cells $\mathcal{L} = \{AND, OR\}$ can be considered as "a guide" to design M -level, $M \in \{2, 3, 4, 5\}$, networks under different scenarios. For the $\mathcal{G}_{5 \times 4}$ window, GA can create different networks (from 2-level to 5-level architecture) via dynamical reconfiguration of its size, i.e. $\mathcal{G}_{M \times N} \subseteq \mathcal{G}_{5 \times 4}$ for $M \leq 5, N \leq 4$.

In this respect, effectiveness of the window based scanning process in a given Target Design Style (TDS) and evolutionary design strategy, in general, can be estimated as *an ability to utilize the freedom of the choice* that is given by the network space corresponded to completely and incompletely specified initial function.

The upper bound of the number of gates in 2-level, n input single-output network created by a scanning window $\mathcal{G}_{2 \times N}$ in $AND - OR$ TDS is $2^{n-1} + 1$ gates. This follows from the well-known fact that, in order to realize an arbitrary function of n variables in an $AND - OR$ 2-level network, one needs not more than $2^{n-1} + 1$ AND and OR gates. Note that we consider correct (not optimal) networks only. Utilizing the scanning window $\mathcal{G}_{3 \times N}$ in $OR - AND - OR$ TDS, provides creation of a 3-level n -input m -output network with at most $m^{1/2} \cdot 2^{(n/2)+1} + m$ gates.

Lemma 1 *The number of different 2-level, n -input, single-output $AND - OR$ networks with $V = m^{1/2} \cdot 2^{(n/2)+1} \cdot (1 - \xi)$ or fewer gates created by $\mathcal{G}_{2 \times N}$, is at most 3^{nV} .*

3. Information theory approach for estimation of evolving network process

Here we offer information estimations to measure the efficiency of the evolution search [10], [9]. Justification of our approach is grounded on the fact that information measuring matches, more than any other one, the stochastic nature of evolutionary process.

We consider the evolutionary network design as a process along with the conditional entropy of the function f with respect to the current solution Net_t ,

$H(f|Net_t)$ is reduced from $H(f)$ to entropy of the correct network solution estimated by $H(f|Net) = 0$, i.e.

$$H(f) \geq H(f|Net_0) \geq H(f, Net_0|Net_1) \geq \dots \quad (1.1)$$

$$\geq H(f, Net_0, Net_1, \dots, Net_{m-2}|Net_{m-1}). \quad (1.2)$$

This equation follows from the well-known results on information theory if one assumes that f and Net_t are independent, then $H(f) \geq H(f|Net_t)$. A generalization of this equation toward f and Net_t , $t = 0, \dots, m - 1$, implies (1.1). Equation (1.1) is true in more simple form $H(f) \geq H(f|Net_1) \geq \dots \geq H(f|Net_{m-1})$ too.

Definition 2 *The mutual information $I(f; Net)$ and the conditional entropy $H(f|Net)$ are related to the entropy $H(f)$ as follows: $I(f; Net) = H(f) - H(f|Net)$. This is an amount of the logical work required to design the network Net .*

Verification of the evolved network consists in testing whether this network achieves the target (desired) design style and the functionality of the given switching function. One can evaluate the efficiency of the searching process via measuring of the fitness function, for example, as the percentage of the correct networks (with desired functionality and style) evolved by the GA. In our study we use the mentioned traditional measure, as well as and information theory measure, in particular, entropy based fitness function.

Definition 3 *An entropy based fitness function is the conditional entropy of a target function f given current function Net :*

$$H(f|g) = - \sum_{i=0}^1 \sum_{j=0}^1 \frac{k(i, j)}{k} \log \frac{k(i, j)}{k(j)} \quad (1.3)$$

where p_{ij} is the probability that f and Net take values i and j accordingly, and p_j means the probability that the output of Net takes value j . Note, that formula (1.3) follows from the definition of conditional entropy.

Example 3 *Consider the computation of the entropy-based fitness function for 2-bit multiplier $(x_1x_0y_1y_0)$ that returns a 4-bit number. Here is how the most significant bit $f = [0\dots01]$ has been changes during computation. The fitness function is calculated as follows: $H(f|Net_0) = -^{15}/_{16} \log ^{15}/_{16} - ^1/_{16} \log ^1/_{16} = 0.337$ bit for $Net_0 = [1\dots1]$, $H(f|Net_{16}) = -^{12}/_{16} \log ^{12}/_{12} - ^3/_{16} \log ^3/4 + ^1/_{16} \log ^1/4 = 0.203$ bit for $Net_{16} = [0000010100000101]$, $H(f|Net_{2503}) = -^{14}/_{16} \log ^{14}/_{14} - ^1/_{16} \log ^1/2 - ^1/_{16} \log ^1/2 = 0.125$ bit for $Net_{16} = [0\dots0101]$, $H(f|Net_{2506}) = -(^{15}/_{16} \log ^{15}/_{15} - ^1/_{16} \log ^1/1) = 0$ bit. It follows from the above that Net_0 , Net_{16} and Net_{2503} are invalid networks.*

4. Partition of a network search space

Most CAD problems are hard optimization problems. To handle the complexity, a technique of partitioning a problem into a number of sub-problems is widely used in CAD. We realize our idea of parallel scanning of a network search space based on its partition into a number of subspaces.

We consider any space C of possible network solutions can be partitioned into $R \geq 2$ subspaces.

In our assumption we state that a network space corresponds to the given switching function that can be decomposed into subfunctions, each of these subfunctions corresponds to a subnetwork space.

Each subspace R of the network solutions can be searched independently and simultaneously with a different windows $\mathcal{G}_{M_i \times N_j}$ and TDSs. Let us denote this evolutionary process as $\mathcal{G}_{R(M_i \times N_j)}$.

There are two different ways to combine obtained subnetworks $SubNet_0, \dots, SubNet_{R-1}$. The first approach is based on implementation of multiplexers MUX_s , i.e. resulting network created as $Net = \{SubNet_t, MUX\}$, $t = 0, \dots, R - 1$. The second approach does not require additional gates to create the final network.

A set of partitions of a network search space is generated through decomposition of an initial function. We explore the principle of decomposition: a problem of large dimension is decomposed into several problems of smaller dimensions, which can be solved separately and with smaller effort. In our study we have used two types of partition of the network search space: Shannon (S) decomposition and its information theoretical notation - entropy based S-decomposition, as well as so-called *balanced* decomposition [2].

In our study we explore the property of conditional entropy to choose a variable with respect to maximal information (minimal entropy) and entropy based S-decomposition. In logic design, S-decomposition of a switching function is defined as $f = \bar{x}f|_{x=0} \vee xf|_{x=1}$. Entropy based S-decomposition of a function f with respect of variable x , is expressed by the relation below [14].

Definition 4 *The entropy of a function f given variable x , $H^S(f|x)$, consists of entropies of subfunctions given $x = 0$ as well as $x = 1$: $H^S(f|x) = p_{|x=0} \cdot H(f|_{x=0}) + p_{|x=1} \cdot H(f|_{x=1})$.*

Information measure of an S-expansion is equal to the conditional entropy $H(f|x)$, i.e. $H^S(f|x) = H(f|x)$. The idea to use information measures is very attractive in decomposition technique, see, for example, [6].

Example 4 *The conditional entropy of the function $f = [10111110]$ with respect to variable x_1 is $H(f|x_1) = -1/8 \log^{1/4} -1/8 \log^{1/4} -3/8 \log^{3/4} -3/8 \log^{3/4} = 0.8113$ bit. By analogy, $H(f|x_2) = -1/8 \log^{1/4} -1/8 \log^{1/4} -3/8 \log^{3/4} -3/8 \log^{3/4} = 0.8113$ bit, and $H(f|x_3) = -0 \cdot \log 0 -1/4 \log^{1/2}$*

$-1/2 \log 1 - 1/4 \log 1/2 = 0.5$ bit. So, variable x_3 conveys more information than variables x_1 and x_2 .

We showed in [8] that the final network *Net* created from subnetworks *SubNet_t* obtained via S-decomposition, includes $R = 2^d - 1$ multiplexers where d is the number of decomposed variables.

S-decomposition has some obstacles that do not allow, in some cases, to realize partitioning of a network space efficiently. We use also another method to solve the partitioning problem, B-decomposition, that relies on decomposition of a switching function with either parallel or serial decomposition applied at each phase of the synthesis process [2].

In the parallel decomposition, the set of output variables Y of a multi-output function f is partitioned into subsets, Y_0 and Y_1 , and the corresponding functions, f_0 and f_1 , are derived so that, for either of these two functions, the input support contains fewer variables than the set of input variables X of the original function f . An objective of the parallel decomposition is to minimize the input support of f_0 and f_1 . In the serial decomposition, the set of input variables X is partitioned into subsets, U and V , and functions, A and B are derived so that the set of input variables of A is $V \cup W$, where W is a subset of U , the set of input variables of B is $U \cup Z$, where Z is the set of output variables of A , and B has fewer input variables than the original function f , i.e. $f = B(U, A(V, W))$. The B-decomposition is an iterative process, at each step, either parallel or serial decomposition of a selected component is performed. The process is carried out until all resulting subfunctions are small enough to fit blocks with a given number of input variables.

The idea of intertwining parallel and serial decomposition has been effectively exploited in the system for logic synthesis, *DEMAIN*, developed at the Institute of Telecommunications, Warsaw University of Technology.

5. Experiments

We have implemented the algorithm as *EvoDesign* program in C++ for OS LINUX on computer with Pentium II 300MHz processor. Our experiments have included two phases. At the first phase, the search network space was divided into subspaces via a decomposition of an initial function. At the second phase, a GA was applied, given population size 60 – 70, crossover $P_{cros} = 0.7$, the number of experiments evaluated to 50 and maximal generation number 10^4 .

Here we verify our approach on parallel evolving of networks using the entropy based S-decomposition and B-decomposition for several benchmarks. In our program *EvoDesign*, we combined GA, the modules to control TDS, the tools for partitioning a network search space, including program *DEMAIN* for B-decomposition and package *Info* [14] for entropy based S-decomposition. We introduce our results in Table 1.1, where $\langle * \rangle$ $\mathcal{G}_{1(2 \times 3)}$, $\mathcal{G}_{2(3 \times 4)}$, $\mathcal{G}_{6(4 \times 5)}$;

$\langle ** \rangle \mathcal{G}_{2(2 \times 3)}, \mathcal{G}_{1(3 \times 4)}, \mathcal{G}_{5(4 \times 5)}$; $\langle *** \rangle \mathcal{G}_{1(3 \times 4)}, \mathcal{G}_{4(4 \times 5)}$. The parameters for S-decomposition were $L_{back} = 3$, $T_{select} = 2$, $T_{discrim} = 90\%$, $P_{mutat} = 0.014$ for 5xp1 was 1.2%, for con1, rd84, alu2, sao; $P_{mut} = 0.02$ for inc, and $P_{mut} = 0.022$ for rd53, rd73, mixex1, 9sym; the number of runs was 50. For B-decomposition, $L_{back} = 4$, $T_{select} = 2$, $T_{discrim} = 90\%$, $P_{mutat} = 0.014$, $P_{cross} = 0.7$ and the number of runs was 100.

For example, for rd53, the network space was divided into four subspaces based on S-decomposition with respect to two variables, and the solutions were found for each of the subspaces in 4-level $\mathcal{G}_{4 \times 4}$ TDS. In particular, the following result was obtained: 21 gates and three multiplexers *MUX*. It is a 4-parallel and independent search process. Let us compare these results with the networks evolved via B-decomposition of the initial function. Note, that the number of subnetworks R^* was calculated by the program *DEMAIN* (meanwhile, $R = 2^d$ for S-decomposition). Two subspaces only were exploited. For 5-level $\mathcal{G}_{5 \times 5}$ TDS window, we obtained a network with 16 gates (without multiplexers). One can observe that both types of decomposition give us extremely better results in gates and levels against the SIS system [13], a state-of-the art synthesis tool: a 6-level network with 34 gates, but in 0.1 *sec*.

We compared our algorithm *EvoDesign* with an exact algorithm for finding a minimal *NAND* networks (pruning technique) [3] and SIS [13] using script *rugged*. In Table 1.2 we give experimental results. As can be seen, the evolutionary strategy still requires on the average more gates and time than Pruning; but at the same time, in most test the evolutionary strategy leads to circuits with less gates than with SIS. Moreover, there are cases (decom,l,m and decom,p,q) where *EvoDesign* obtains sub-optimal solutions, but more than ten times faster than by Pruning. Even though the sample is relatively small, it supports the evidence that the evolutionary circuit design is reaching a level of competitiveness with respect to well established methods used in industry. Further improvements are presently under development.

6. Concluding remarks

We showed that the idea of evolutionary network design can be more captivating and attractive, if the concept of a given design style is realized. In this case, the designer really becomes an expert, i.e. the designer needs no special software to implement a design style, the style is provided by the evolutionary search for the best solution.

The main contribution of this paper is an extension and improvement of the state-of-the-art of gate-level evolutionary networks synthesis, namely, concept of a TDS allows us to achieve flexible scanning of the subspaces of possible network solutions. We have found that, in general, B-decomposition of the network search space is more preferable, because it does not require any multi-

Table 1.1. Efficiency of the network search space partition via entropy based S- and B-decomposition

<i>Test</i>	<i>I/O</i>	<i>Entropy based S-decomposition</i> ¹				<i>B-decomposition</i> ² [2]		
		$\mathcal{G}_{R(M \times N)}$	<i>G+MUX</i>	<i>L</i>	<i>t</i>	$\mathcal{G}_{R^*(M \times N)}$	<i>G</i>	<i>t</i>
rd53	5/3	4(4 × 4)	21+3	6	8	2(5 × 5)	16	57
con1	7/2	16(5 × 5)	13+15	9	0.25	4(5 × 5)	22	61
rd73	7/3	16(4 × 4)	83+15	8	28	5(5 × 5)	28	40
inc	7/9	16(3 × 6)	10+15	7	0.3	9(5 × 5)	82	220
5xp1	7/10	16(4 × 8)	121+15	8	152	10(5 × 5)	61	84
rd84	8/4	32(5 × 5)	191+31	10	40	6(5 × 5)	32	92
misex1	8/7	16(4 × 4)	1+15	8	0.5	9(5 × 5)	61	49
life	9/1	32(4 × 5)	169+31	9	24	5(4 × 5)	34	41
9sym	9/1	32(4 × 4)	144+31	9	376	6(5 × 5)	37	39
misex20	10/1	64(4 × 5)	18+63	10	0	< * >	45	38
newtpla2	10/4	64(4 × 4)	217+63	10	7	< ** >	26	5
sao2	10/4	64(5 × 4)	120+63	11	209	20(5 × 5)	165	120
alu2	10/6	64(4 × 6)	510+63	10	4448	12(5 × 5)	71	105
misex46	11/1	128(4 × 5)	24+127	11	0	6(4 × 5)	36	22
misex47	11/1	128(4 × 5)	12+127	11	0	< *** >	26	11
Total	129/57		1654+677	137	5292		742	1049

<i>Test</i>	<i>I/O</i>	<i>SIS</i> [13]		
		<i>G</i>	<i>L</i>	<i>t</i>
rd53	5/3	34	6	0.1
con1	7/2	30	8	0.1
rd73	7/3	118	11	0.2
inc	7/9	171	10	0.3
5xp1	7/10	94	10	0.3
rd84	8/4	182	13	0.48
misex1	8/7	155	7	0.4
life	9/1	138	16	0.2
9sym	9/1	132	14	0.3
misex20	10/1	56	11	0.1
newtpla2	10/4	74	8	0.2
sao2	10/4	203	16	0.15
alu2	10/6	604	19	2.9
misex46	11/1	52	10	0.1
misex47	11/1	34	9	0.1
Total	129/57	2317	182	2.8

plexers to merge a network from subnetworks. In any case, the decomposition gives additional and unique possibilities for massive parallel processing. In our current work we focus on different practically preferable styles including AND/EXOR, Kronecker, ESOP expressions and others.

Table 1.2. The comparison of SIS approach, pruning technique and evolutionary strategy

Test	SIS		Pruning		EvoDesign	
	I/O	Gate	Gate	$t(sec)$	Gate	$t(sec)$
c17	5/2	9	6	0.03	15	528.58
cm42a.e,f	4/2	14	9	19.74	11	36.79
cm42a.g,h	4/2	13	8	9.43	10	50.87
cm42a.i,j	4/2	13	8	9.47	10	43.47
cm42a.k,l	4/2	12	7	1.39	10	40.39
cm42a.m,n	4/2	13	8	9.33	10	40.26
decod.f,g	5/2	17	10	68.61	17	1369.43
decod.h,i	5/2	18	11	1603.07	17	1769.8
decod.j,k	5/2	18	11	1592.00	17	1377.88
decod.l,m	5/2	19	12	27818.76	16	1246.88
decod.n,o	5/2	18	11	1620.73	17	1397.72
decod.p,q	5/2	19	12	26921.29	17	1205.56
decod.r,s	5/2	19	12	26445.97	17	2246.47
majority	5/1	14	9	37.43	12	475.38
x2.k,l	3/2	11	8	8.11	10	26.64
x2.m,o	4/2	11	10	119.66	11	254.54
z4ml.27	3/1	10	8	0.6	14	1359.7

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] R. Drechsler and W. Gunther. Exact circuit synthesis. In *International Workshop on Logic Synthesis, Lake Tahoe*, 1998.
- [4] D. Quagliarella *et. al.* (Eds.). *Genetic Algorithm and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*. John Wiley and Sons Ltd, 1998.
- [5] M. Siper *et. al.* (Eds.). *Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science*, vol.

1478. Springer, 1998.
- [6] L. Jóźwiak and A. Chojnacki. Functional decomposition based on information relationship measures extremely effective for symmetric functions. In *Proc. EUROMICRO Conf.*, pages 150–159, 1999.
 - [7] T. Łuba, C. Moraga, S. Yanushkevich, M. Opoka, and V. Shmerko. Evolutionary multi-level network synthesis in given design style. In *Proc. IEEE Int. Symposium on Multiple-Valued Logic*, 2000.
 - [8] T. Łuba, C. Moraga, S. Yanushkevich, V. Shmerko, and J. Kolodziejczyk. Application of Design Style in Evolutionary Multi-Level Networks Synthesis, In *Proc. IEEE Int. Symposium on Digital System Design (DSD'2000)*, Netherlands, pages 156–163, 2000.
 - [9] V. Cheushev, S. Yanushkevich, C. Moraga, V. Shmerko, and J. Kolodziejczyk. Remarks on circuit verification through the evolutionary circuit design, In *Proc. IEEE 31-th Int. Symp. on Multiple-Valued Logic*, pages 201–206, 2001.
 - [10] V. Cheushev, S. Yanushkevich, C. Moraga, V. Shmerko. Research Report, *Flexibility in Logic Design. An Approach Based on Information Theory Methods*, Forschungsbericht 741, University of Dortmund, Germany.
 - [11] 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.
 - [12] Proc. NASA/DoD Workshop on *Evolvable Hardware*, Pasadena, California. 1999.
 - [13] SIS Release 1.2. In *UC Berkeley Soft. Distr.*, July, 1994.
 - [14] V. P. Shmerko, D. V. Popel, R. S. Stanković, V. A. Cheushev, and S. N. Yanushkevich. Information theoretical approach to minimization of AND/EXOR expressions of switching functions. In *Proc. IEEE Int. Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, pages 444–451, 1999.
 - [15] K. S. Tang, K.F. Man, and S. Kwong. Parallel genetic algorithms: Implementable hardware solutions. *Circuit and Systems*, 10(2):3,8–11, 1999.