AN EVOLUTIONARY FPAA RECONFIGURATION ALGORITHM BASED ON THE OPEN TRAVELING SALESMAN PROBLEM

Claudia FARAGÓ Paul FARAGÓ Sorin HINTEA Gabriel OLTEAN
Technical University of Cluj-Napoca, Romania
Str. Baritiu 26-28, 400027 Cluj-Napoca, Phone/Fax +40264591340, Claudia.Farago@bel.utcluj.ro

Abstract: This work proposes an FPAA reconfiguration algorithm formulated on a modified version of the open traveling salesman problem (OTSP), which targets the achievement of a figure of merit (FOM). Genetic algorithms (GA) were chosen to optimize the modified OTSP. Several modifications were made to the genetic operators to guarantee population validity along the evolutionary process. The efficiency of the modified GA is demonstrated over the classical GA for small-sized FPAA mapping, while the classical GA is used for larger FPAAs. Simulation results illustrate the applicability of the proposed evolutionary FPAA mapping algorithm based on the modified OTSP routine.

Keywords: *Field programmable analog array, analog filter, open traveling salesman problem, genetic algorithms.*

I. PAGE FORMAT

The simultaneous deployment of digital and analog processing functions is the driver for a variety of application fields, including automotive, communications, digital lifestyle, energy, environmental control, healthcare, security and entertainment [1, 2]. Thus, recent development trends in mixed-signal systems-on-a-chip (SoC) answer the continuously increasing demands for high processing capabilities within low-cost integrated circuits.

Although signal processing is mostly digital, there are several analog-specific processing functions, such as biasing, signal conditioning and conversion, interfacing, intermediate-frequency filtering, etc. Moreover, the analog implementation of several signal processing functions provides several benefits over digital, such as reduced power consumption, high speed and high-frequency operation [3].

The increasing demand in analog processing functions has attracted new paradigms in the design of analog integrated circuits. While analog processing was traditionally available in the shape of application-specific integrated circuits (ASIC), optimized for performance and power consumption, modern analog integrated circuits target the implementation of certain parameter programmability and topological reconfigurability [4].

Field programmable analog arrays (FPAA) are the analog counterpart of field programmable gate arrays (FPGA), which were proven to be a versatile solution for rapid prototyping and testing of digital processing systems. Indeed, FPAAs have gained considerable attention and have been a subject of research for the past two decades, to offer an attractive alternative to ASICs.

FPAAs consist of configurable analog blocks (CAB), which implement basic analog operations such as gain and integration, and a programmable interconnection network, which implements signal routing among the CABs in order to implement the desired transfer function. The CAB deploys the FPAA with the parameter programmability feature, with limited configurability to select among the basic analog operations. The programmable interconnection network implements the reconfigurability feature on the FPAA. Then, FPAA reconfiguration accounts for the routing of the programmable interconnection network, in order to implement the desired analog transfer function.

Traditionally, FPAA reconfiguration employs lookup tables (LUT) which store the binary control words for mapping several analog processing functions. LUT-based reconfiguration is straightforward and is mainly used in FPAAs which favor performance rather than generality, e.g. [5]. LUT-based reconfiguration however is deterministic and doesn't account for post-silicon performance parameters, such as propagation delays, effects of random process drifts and device mismatch, effects of process, voltage and temperature (PVT) variations due to environmental changes, etc. [6]. All of these factors affect the implemented transfer function and must be accounted for during the reconfiguration process.

Alternatively, software-driven approaches for FPAA reconfiguration maintain the LUT-based programming for intra-CAB reconfiguration, but apply computational intelligence techniques to optimize the routing of the interconnection network. For example, Becker et al. has applied genetic algorithms for the mapping of analog filters [7]. Similarly, Stoica et al. has applied genetic algorithms for the synthesis of novel amplifier structures on a programmable transistor array [8]. In either of the reported examples, the GA searches among random routes in the FPAA, and thus the evolutionary process is non-transparent with respect to the implemented analog circuit.

In this work, we have proposed a novel FPAA reconfiguration algorithm for the implementation of analog filters, which treats the reconfiguration problem in similarity with the open traveling salesman problem (OTSP). For this purpose we have defined a new variation of the OTSP, which targets the determination of a route realizing an

Manuscript received February 16, 2015; revised February 24, 2015

imposed figure of merit (FOM), rather than minimum length.

We have solved the proposed variation of the OTSP using genetic algorithms (GA). To do so, we have proposed a variable length representation and a modification to the genetic operators in the classical AG routine to guarantee population validity along the evolutionary process.

This article is organized as follows. Section 2 presents a description of the high-level FPAA model developed for the generalization of programmable analog arrays. Next, Section 3 presents the proposed modification to the OTSP which targets the achievement of a required FOM on a non-Hamiltonian walk. Also, Section 3 describes the GA to solve the FOM-OTSP and proposes some modifications brought to the classical GA in order to gain some algorithm efficiency. Section 4 proposes the proposed FPAA reconfiguration algorithm for the mapping of analog filters, which employs of the evolutionary FOM-OTSP routine. Section 5 presents some conclusive simulation results which demonstrate the efficiency of the proposed modified genetic operators over the classical GA scheme for small-sized reconfiguration problems. FPAA reconfiguration is demonstrated for both small-sized and larger FPAAs, for the mapping of a Tow-Thomas cascade, and two leap-frog filters with arbitrary transmission zeros. Finally some conclusions are drawn and some research perspectives are formulated.

II. THE FPAA HIGH-LEVEL MODEL

The evolutionary FPAA reconfiguration algorithm was developed to generalize the FPAA reconfiguration process for the implementation of analog filters. Assuming the multiple loop feedback (MLF) filter topology, the high-level FPAA model must be able to accommodate [9]:

- the filter backbone (BB) – consisting of a cascade of lossless integrators, which gives the filter order by generating origin poles,
- the feedback (FB) loops – which place the poles in the complex plane,
- the feedforward (FF) paths – which generates and places arbitrary transmission zeros in the complex plane.

We have proposed a generalized high-level FPAA model to accommodate the FPAA reconfiguration algorithm for mapping of analog filters. The proposed high-level FPAA model is illustrated in Figure 1.

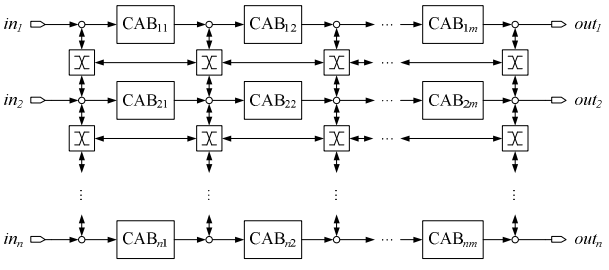

Figure 1. The high-level FPAA model.

The FPAA model is built around a grid architecture resembling an [n x m] matrix, exhibiting n input and n output ports respectively, thus being able to implement n parallel signal processing paths.

The CABs in the high-level FPAA model from Figure 1 were designed to generalize the analog signal processing operations in terms of gain and filtering. The proposed CAB model is illustrated in Figure 2, and consists of three parallel paths: the direct path (DP), the feed-back (FB) path and the feed-forward (FF) path respectively.

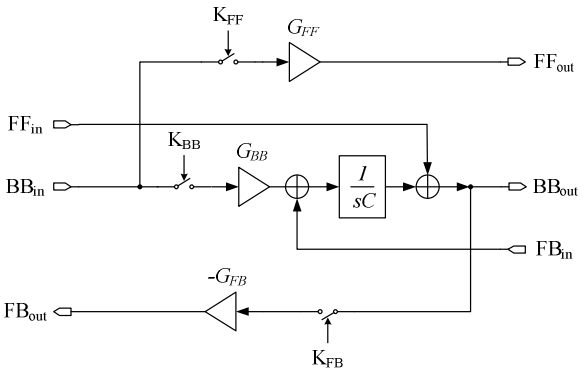

Figure 2. The high-level model of the CAB.

The DP consists of an amplifier, a lossless integrator which introduces an origin pole, two source nodes for a FB loop and a FF path respectively, and two summation nodes to terminate a FB loop and a FF path respectively. With the proposed high-level CAB model, the filter BB, and consequently the filter order, is built by the DP from successive CABs.

Further on, the CAB contains two gain elements G_{FB} and G_{FF} on the FB and FF paths respectively. Assuming that the FB path produces a loop containing k lossless integrators on the filter BB, a k^{th} order pole is defined, introducing a transfer function expressed according to Mason's rule as:

$$T^{(pole)}(s) = \frac{\prod_{i=1}^{n} G_{BB_i}}{s^n \prod_{i=1}^{n} C_i + s^{k-l} F_{FB_k} \prod_{i=l+1}^{k} G_{BB_i} \prod_{i=1}^{l} C_i \prod_{i=k+l}^{n} C_i} \quad (1)$$

Similarly, assuming that the FF path bypasses k lossless integrators on the filter BB, a k^{th} order arbitrary transmission zero is defined, introducing a transfer function expressed according to Mason's rule as:

$$H_0 \cdot \frac{b_{k-1}s^{k-1} + \ldots + b_1 s + b_0}{a_n s^n + \ldots + b_1 s + b_0} \quad (2)$$

Switches K_{BB}, K_{FB} and K_{FF} state the enclosure of the corresponding CAB paths to the overall analog processing chain. The LUT for CAB programming is listed in Table 1, along with the implemented transfer function.

To be noted is that, the signal buses implementing the CAB local interconnections in the FPAA from Figure 1 are actually implemented with three parallel interconnections for the CAB DP, FB and FF paths respectively. Global interconnectivity is achieved within the switching matrices by employing the *fat-tree* interconnection network [10] illustrated in Figure 3. Besides the fact that the fat-tree interconnection network provides the FPAA with full reconfiguration capabilities, the binary nature of the network makes signal routing straightforward.

Table 1. LUT for CAB reconfiguration.

Switch state			**Implemented function**	
K_{BB}	**K_{FB}**	**K_{FF}**	**High-level component**	**Transfer function**
off	off	**ON**	Direct path gain	G_{FF}
ON	off	Off	Lossless integrator	$\frac{G_{BB}}{sC}$
ON	**ON**	Off	Lossy integrator (FB path source and destination nodes in the same CAB)	$\frac{G_{BB}}{sC+G_{FB}}$
ON	**ON**	off	k^{th} order pole (FB path source and destination nodes in the different CABs k and l, $k>l$)	eq. (1)
ON	off	**ON**	Arbitrary transmission zero (FF path source and destination nodes in the different CABs k and l, $k>l$)	eq. (2)

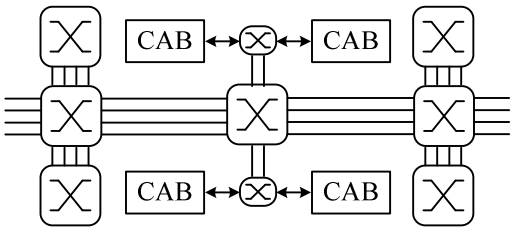

Figure 3. Section from the fat-tree interconnection network.

For the mathematical representation of the FPAA using graph theory we have employed two adjacency matrices which are presented as follows. For resemblance with graph theory, the switching matrices from Figure 3 will further be referred to as switching nodes.

Transfer-function adjacency matrix A^{FPAA}, expressed in (3), encodes the transfer function implemented between any two FPAA nodes. The route passing through a CAB, e.g. $CAB_{i,j}$, contributes its transfer function $H_{i,j}(s)$ to the global FPAA transfer function. Any other route passes successive switching points implementing unity transfer function, and brings no contribution to the global transfer function. Accordingly, A^{FPAA} is used to determine the transfer function implemented on the FPAA. To be noticed is that, although matrix A^{FPAA} gives specific information regarding the implemented transfer function, it does not give any information concerning the hardware resources involved, e.g. number of switching points, etc..

The distance adjacency matrix D^{FPAA}, expressed in (4), encodes the number of FPAA blocks, namely switching points and CABs, between any two nodes, and is used to determine the length of a walk, or in terms of graph theory the cost of a walk. Thus, matrix D^{FPAA} gives specific information regarding the HW resources involved in FPAA reconfiguration.

$$A^{FPAA}=\begin{bmatrix} 0 & H_{11}(s) & 1 & \cdots & 1 \\ 1 & 0 & H_{12}(s) & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{bmatrix} \quad (3)$$

$$D^{FPAA}=\begin{bmatrix} 0 & 1 & 3 & \cdots & m+1 \\ 2 & 0 & 1 & \cdots & m \\ 3 & 2 & 0 & \cdots & m-1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ m+1 & m & m-1 & \cdots & 0 \end{bmatrix} \quad (4)$$

Further adjacency matrices can be defined to encode other design specifications, such as sensitivity vs. components, signal delays, etc.

III. THE FOM-OTSP PROBLEM

FPAA reconfiguration accounts for the determination of the optimal route between the required input and output nodes which implements the required transfer function. Thus, the FPAA reconfiguration problem resembles strong similarity with the open traveling salesman problem.

The open traveling salesman problem [11] (OTSP) is defined on a map with n cities and m roads binding the cities. The problem definition states the requirement to determine a minimum-length walk, which visits all the cities exactly once.

In the mathematical formulation of the OTSP using graph theory, the map is represented with an oriented graph G:

$$G=(V,E)=\{c_{i,j}\mid i,j\in V;[i,j]\in E,i\neq j\} \quad (5)$$

where $V = [1\ldots n]$ is the set of vertices representing the cities, E is the set of edges representing the roads binding the cities, and $c_{i,j}$ is the cost of edge $[i, j]$ representing the length of the road binding cities i and j.

In graph theory, the OTSP translates to the requirement to determine the minimum-cost Hamiltonian walk w^* in graph G:

The main difference between the OTSP and the FPAA reconfiguration problem is that, while the former searches for a walk which visits all the vertices, the latter doesn't necessarily visit all nodes. Therefore, we have relaxes the constraint of a Hamiltonian walk and have formulated a novel variation of the OTSP which states the requirement to determine a minimum-cost walk w^* with imposed departure and destination nodes, which achieves a required figure of merit FOM^*:

$$w^*\subset W\mid \cos t(w^*)\leq \cos t(w_j),\forall j\in[1,card(W)] \quad and \quad FOM(w^*)=FOM^* \quad (6)$$

where W is the set of walks in graph G, defined as:

$$W = \{(v_1, v_2, ..., v_k) \mid k \le n; (v_i, v_{i+1}) \in E, \forall i \in [1, k-1]\} \quad (7)$$

To be noted is that the proposed FOM-OTSP formulation relaxes the constraint of a Hamiltonian walk. Rather than visiting all the nodes as was the case in the classical OTSP formulations, the FOM-OTSP targets the accomplishment of a FOM.

The traveling salesman problem, in either variation, is NP-complete [12], and thus the only method to solve the problem is exhaustive enumeration and evaluation [13]. This approach however results in unacceptably long execution time for large values of n.

Alternative approaches to solve the OTSP problem make use of search heuristics, e.g. Greedy search, Minimum Spanning tree, k-nearest neighbors, etc., which aim to improve the search procedure, yet result in exponential runtime and increased computational burden. Among other optimization heuristics, population-based optimization techniques, e.g. genetic algorithms, simulated annealing, colony optimization, artificial immune systems, etc., have been reported to find a near-optimal route in finite runtime, thus providing a satisfactory solution of the OTSP problem [12].

In this paper we have employed GAs to optimize the solution to the proposed FOM-OTSP problem. Genetic algorithms are a search heuristic, which operates on a population of solution candidates, rather than on individual solutions, and mimics the process of natural evolution towards iteratively optimizing an objective function [4, 7, 12].

The block diagram of the GA-based algorithm to solve the FOM-OTSP is illustrated in Figure 4 and is explained as follows. The input argument to the GA is the map consisting of the city locations, the distance between cities and the FOM. Based on this map, a number of individuals, representing walks in the oriented graph, are randomly generated to form the initial population. Next, the population is evaluated and a fitness value measuring the cost and the FOM of the walk is attached to each of the individuals in the population.

Should the optimization criteria be fulfilled, the best individual from the population is provided as optimization result. Otherwise the evolutionary loop is entered and genetic operators of biological inspiration, namely selection, cross-over and mutation, are applied to optimize the population. Additionally, we have employed a survival operator to have the promising individuals survive from one generation to the next along the evolutionary process.

For the GA stopping criteria, we have considered the achievement of either of two individual stopping criteria: a maximum number of generations, or a maximum number of consecutive generations exhibiting no significant improvement of the objective function.

We have made several modifications to the classical GA routine as follows, in order to solve the FOM-OTSP problem. In accordance with the proposed variation of the OTSP, which searches for a minimum-length walk that doesn't necessarily visit all nodes, we have defined a variable length representation (VLR). Accordingly, the GA chromosome to represent a walk in graph G is a structure *chr* expressed as

$$chr = \{L, (v_1, v_2, ..., v_L)\} \quad (8)$$

where L is the number of vertices in the walk and v_i, i=1…L are the vertices of the oriented graph from the encoded walk. In order to have imposed starting and ending nodes respectively, each chromosome must satisfy:

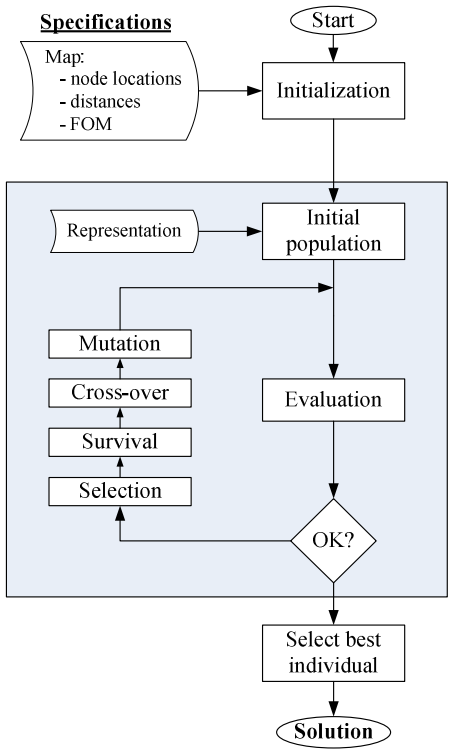

Figure 4. The GA-based algorithm to solve the FOM-OTSP problem.

$$\forall i = 2...L \Rightarrow [v_{i-1}, v_i] \in E, [snode, v_1] \in E, \quad [v_L, enode] \in E \quad (9)$$

where *snode* and *enode* are the starting and ending nodes of the walk.

The evaluation function must consider the fact that the proposed FOM-OTSP formulation defines two distinct objectives which have to be obtained simultaneously: 1) the desired FOM should be obtained on a 2) minimum-length walk. Accordingly, we have defined the validity of a walk in the graph in terms of achievement of the required FOM.

While the length of the walk is subject to optimization, the FOM is an "*all or nothing*" specification. Then, the objective function of individual i, i.e. walk w_i, is expressed as:

$$OF(w_i) = \begin{cases} length(w_i), if \quad FOM(w_i) = FOM^* \\ p, if \quad FOM(w_i) \neq FOM^* \end{cases} \quad (10)$$

Objective function (10) states that, if the FOM specification is met the fitness function equals the length of the walk encoded by the individual. Otherwise, the individual is penalized with penalty factor p, which must be

sufficiently large to dominate the maximum length achievable by a walk in the graph.

Considering the random nature of the population generation function and the genetic operators, it is sensible to assume that not all individuals from the population will represent valid walks. We have extensively simulated the population creation function and the genetic operators and have analyzed the simulation data. Indeed, we found that for random population creation 70-80% of the population consists of invalid individuals. Similarly, after applying the cross-over and mutation operators 50-60% of the instances result in invalid individuals. Such individuals are penalized in the evaluation stage according to objective function (10), and do not contribute to the optimization process. The result is a considerably longer runtime of the evolutionary process, which increases even more with larger population sizes, making the classical GA operators inefficient for the proposed problem.

To compensate for this drawback, we have proposed a modification to the population creation function and the genetic operators to generate inherently valid individuals while still maintaining their random nature. The modified population creation function is implemented with a repetitive loop which repeats the random walk generation until a valid walk is determined. Similarly, the cross-over operator is implemented with a repetitive loop which repeats the random generation of the cross-over points until the offspring represents valid individuals. For the mutation operator, the random generation of the mutation loci until the mutation result is a valid individual.

IV. THE PROPOSED FPAA MAPPING ALGORITHM

The objective of the FPAA reconfiguration algorithm is to determine a route in the FPAA, which passes through a minimum number of switching nodes and implements the required filter transfer function.

Since the FPAA topology with the available degrees of freedom is known prior to the reconfiguration process, a LUT-based FPAA mapping would seem straightforward. However, LUT-based reconfiguration cannot take into account post-silicon FPAA parameters such as transmission delays, parasitic effects of random process drifts, device mismatch etc., as well as effects of process, voltage and temperature (PVT) variations due to environmental changes.

In this section we propose the employment of the proposed evolutionary FOM-OTSP routine to solve the FPAA reconfiguration problem in a software-driven reconfiguration approach.

The block diagram of the proposed FPAA reconfiguration algorithm is illustrated in Figure 5 and is explained as follows. Unlike the FOM-OTSP routine from Figure 4 which evaluates walks in the graph, the proposed FPAA reconfiguration algorithm evaluates analog filters mapped on the FPAA. For generality, the proposed reconfiguration algorithm handles the mapping of each FB and FF connection individually within a dedicated FOM-OTSP, and therefore the proposed reconfiguration algorithm exhibits a nesting of three GAs.

The top-level GA loop operates towards mapping the filter BB on the FPAA, between the desired input and output nodes. In similarity to the FOM-OTSP routine, BB mapping operates towards implementing a minimum-length route on the array with the implementation of the desired transfer function.

The cost of the walk was defined to include the number of switching points as well as further post-silicon performance measures. Therefore the distance matrix for the FOM-OTSP routine is expressed as

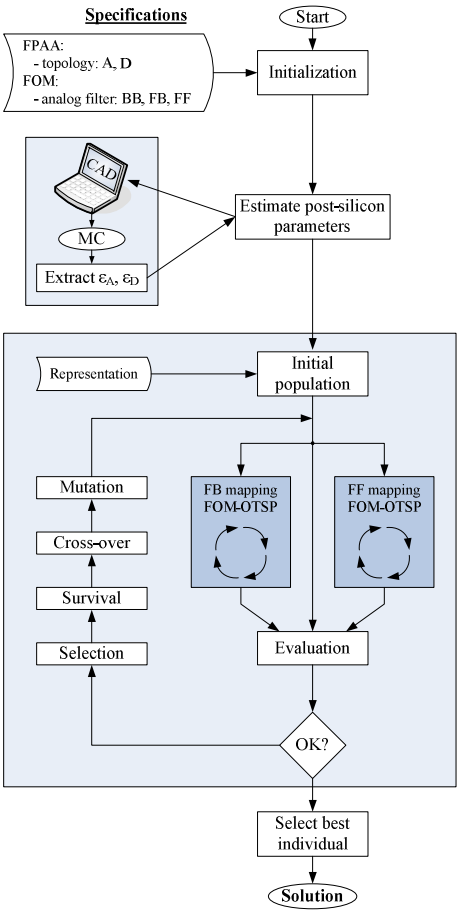

Figure 5. The proposed FPAA reconfiguration algorithm.

$$D = D^{FPAA} + \varepsilon_D \tag{11}$$

where D^{FPAA} is the FPAA distance adjacency matrix expressed in (4), and ε_D is an [nxm] matrix consisting of post-silicon transmission delays, which are due to random process drifts and PVT variation.

The FOM on the other hand was defined as the implemented transfer function, thus the transfer function matrix for the FOM-OTSP routine is expressed as:

$$A = A^{FPAA} + \varepsilon_A \tag{12}$$

where A^{FPAA} is the FPAA transfer function adjacency matrix expressed in (3), and ε_A is an [nxm] matrix consisting of the post-silicon variations in the transfer function, which are due to random process drifts and PVT variation.

To be noticed is that, while the cost of the walk is subject to optimization, FOM is a "all or nothing" specification, the satisfaction of which is mandatory.

Although the determination of matrices ε_D and ε_A requires measurements on the fabricated FPAA, in this work we have employed a simulation-based methodology for the

estimation of post-silicon performance parameters, as illustrated in Figure 6.

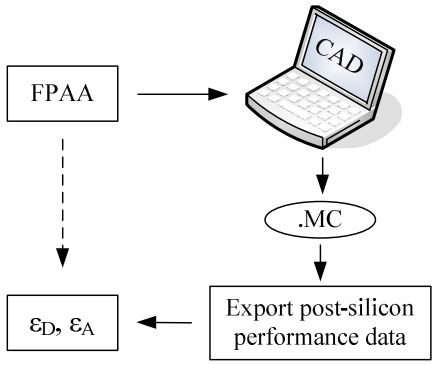

Figure 6. Simulation-based methodology to estimate post-silicon performance parameters.

The proposed FPAA model was implemented in Mentor Graphics computer aided design (CAD) environment using first-order ideal models for the gain and integration elements. Device tolerances available from statistical deviation information formerly recorded in [14] have been applied to the ideal elements. One Monte Carlo (MC) simulation run was performed to emulate specific information regarding post-silicon performance for one FPAA instance. The MC simulation output data was used to fill in matrices ε_D and ε_A.in (11) and (12).

After having a population of filter BBs in the top-level GA, the next stage in the proposed FPAA reconfiguration algorithm is to perform the mapping of the required FB and FF loops around the BB integrators in dedicated GA loops.

The feedback and feedforward routes between a fixed starting and ending nodes must also satisfy the criteria of minimum length and required FOM. To handle FB mapping with the FOM-OTSP routine, feedback adjacency matrices A_{fb} and D_{fb} are expressed as follows:

$$\begin{cases} A_{fb} = A' \\ D_{fb} = D' \end{cases} \tag{13}$$

The transpose operation in (13) expresses the fact that the feedback loops have an opposed orientation than the filter backbone. Consequently, pole order is given by reverse-order visiting of consecutive nodes.

Similarly, to handle FF mapping with the FOM-OTSP routine, feedforward adjacency matrices A_{ff} and D_{ff} have been expressed as:

$$\begin{cases} A_{ff} = A \\ D_{ff} = D + \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \end{cases} \tag{14}$$

The addition of the unity matrix to D expresses the fact that a FF route between consecutive nodes in the oriented graph actually bypasses the CAB, and thus passes through an additional switching point.

The evaluation functions within the FB and FF mapping procedures evaluate the length and the FOM of the FB and FF connections respectively. Considering the nesting of three GA loops, the evaluation function within the top-level GA will estimate the length and the FOM of the analog filter as a whole and not of the BB, FB and FF routes individually.

The GA stopping criteria are expressed similarly to the FOM-OTSP routine in terms of maximum number of generations, and a maximum number of consecutive generations with no significant improvement of the objective function.

After evaluation, should the stopping criteria be satisfied, the evolutionary process stops and the best individual in provided as optimization result. Otherwise, the evolutionary loop is entered and genetic operators are applied to generate the next generations.

V. SIMULATION RESULTS

The GA-based algorithm to solve the FOM-OTSP problem for FPAA reconfiguration was implemented in Matlab. The FOM-OTSP routine was extensively simulated to investigate the effects of the GA parameters on the optimization performance, and to compare the efficiency of the classical GA implementation with the proposed modifications to the GA operators. The tests were carried out on two randomly generated maps with 10 and 20 cities respectively.

The evolution of the mean objective function value vs. generations for the 10 city map is illustrated in Figure 7, with dashed line for the classical GA operators and with solid line for the modified GA operators. The objective function value along the evolutionary process is higher for the classical GA implementation, illustrating the fact that the population is less fit to solve the optimization problem. This is due to the penalty factors applied to invalid individuals in the population. The modified GA operators on the other hand guarantee population validity, thus exhibiting a lower objective function value along the evolutionary process and shortening the evolution length. Also, the GA runtime with the modified genetic operators is considerably shorter, namely 27 minute in comparison to 48 minutes for the classical GA implementation.

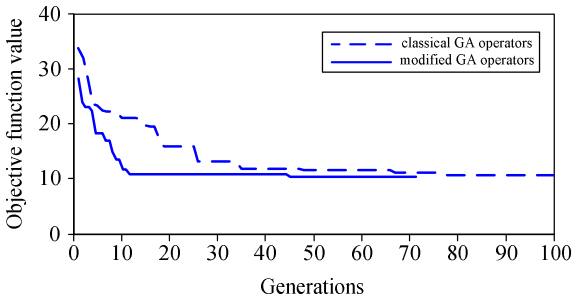

Figure 7. Evolution of the mean objective function value vs. generations for the 10 city map.

The evolution of the mean objective function value vs. generations for the 20 city map is illustrated in Figure 8, with dashed line for the classical GA operators and with

solid line for the modified GA operators. Again, the modified GA operators accounted for a shorter evolution length in terms of number of generations. However, given the much larger search space in the 20 city map, the modified GA operators accounted for a longer runtime due to the repetitive loops to generate valid individuals.

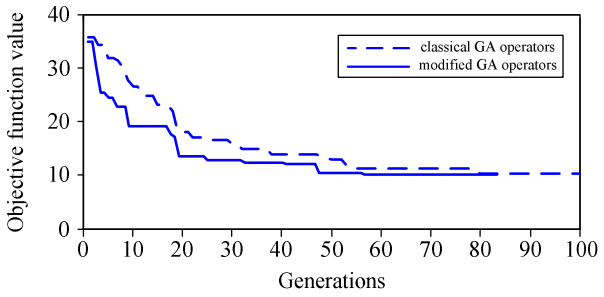

Figure 8. Evolution of the mean objective function value vs. generations for the 20 city map.

Based on this set of tests, we have concluded that for the reconfiguration of small-sized FPAAs, i.e. up to 15 switching nodes, the modified operators are suitable to implement the top-level GA. For large FPAAs, i.e. 20 or more switching nodes, the classical GA implementation will be used. In either case, the FB and FF mapping GAs will be implemented with the modified GA operators.

Accordingly, the parameters of the GA routines within the FPAA reconfiguration algorithm are set as follows. The top-level GA was implemented with a 50 individual population size and a survival rate of 5 individuals. The stopping criteria account for an evolution of maximum 100 generations or 20 consecutive generations with no improvement to the objective function. The fulfillment of either of the two criteria leads to the stopping of the evolutionary process. The FB and FF mapping GA was implemented with a 20 individual population size and a survival rate of 2 individuals. The stopping criteria account for an evolution of maximum 50 generations or 10 consecutive generations with no improvement to the objective function.

The proposed FPAA reconfiguration algorithm was extensively simulated in order to prove its applicability for the implementation of analog filters. Monte Carlo simulation of the FPAA model within the reconfiguration routine was performed with Eldo from Mentor Graphics.

The first design scenario illustrates the routing of a 5th order low-pass filter implemented with a Tow-Thomas (TT) biquad cascade topology, on a [3x3] FPAA. Considering the size of the FPAA, the modified genetic operators are employed in the top-level GA. The FPAA routing of the 5th order TT cascade is illustrated in Figure 9.

The evolution of the mean objective function value vs. generations for the top-level FOM-OTSP routine is illustrated in Figure 10, with dashed line for the classical genetic operators and solid line for the modified genetic operators. The comparison of the two illustrates that the modified GA operators indeed operate towards shortening evolution length and optimization runtime, demonstrating their efficiency over the classical GA operators for the small-sized FPAA reconfiguration problem.

The second design example illustrates the routing of two 8th order leap-frog (LF) low-pass filters on a [5x5] FPAA. To illustrate the mapping of FF paths, the two filters exhibit a first-order arbitrary transmission zero. Considering the size of the FPAA, namely 24 switching nodes, the top-level GA was implemented with the classical genetic operators. The FPAA routing of the two 8th order LF filters is illustrated in Figure 11.

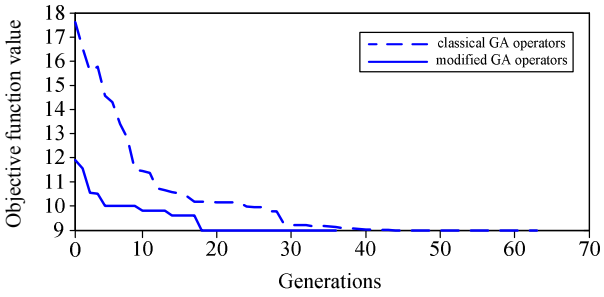

Figure 10. Evolution of the mean objective function value vs. generations for the top-level FOM-OTSP routine.

VI. CONCLUSIONS

The GA-based algorithm to solve the FOM-OTSP problem Recent development trends in mixed-signal systems-on-a-chip answer the increasing demands for high-speed processing within low-cost integrated circuits. While most of the processing is digital, there are several analog-specific processing function. Field programmable analog arrays answer the requirement for a versatile hardware platform to host programmable analog processing functions.

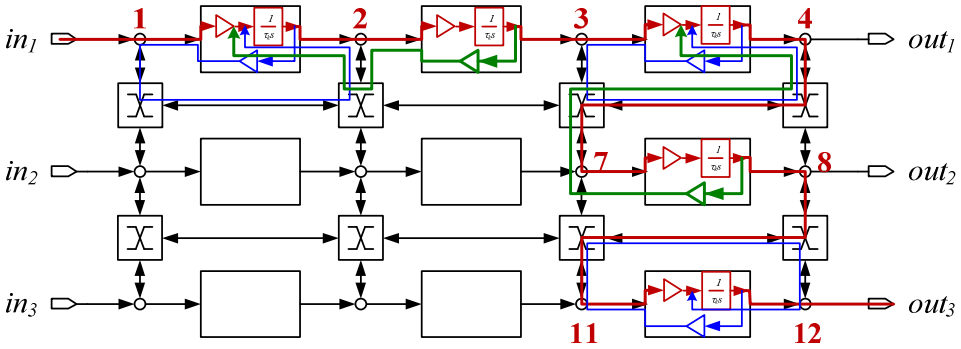

Figure 9. FPAA routing of the 5th order TT cascade.

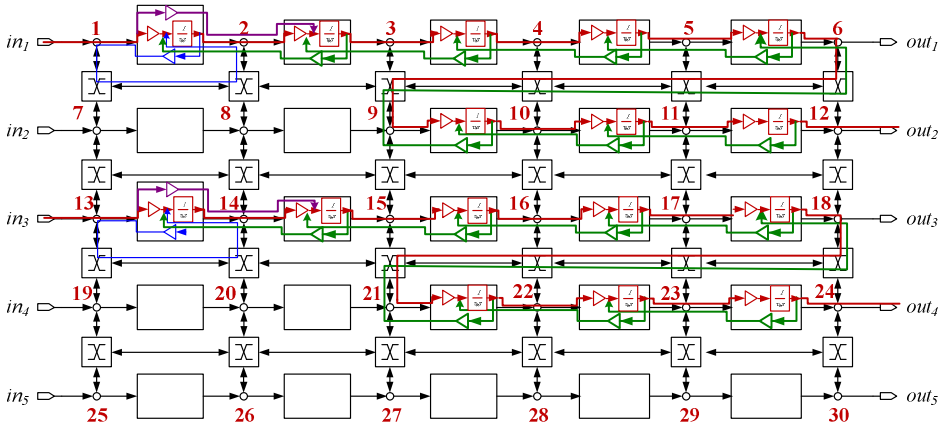

Figure 11. FPAA routing of two 8th order LF filters with arbitrary transmission zeros.

This work proposes an FPAA reconfiguration algorithm based on the open traveling salesman problem (OTSP) for the mapping analog filters. For this purpose, a high-level model of an FPAA was defined, which is built around first-order CABs to implement the singularities of analog filter transfer function.

A modified version of the OTSP proposed to resemble the FPAA reconfiguration problem. The modified OTSP relaxes the requirement of visiting all cities from the map, and rather targets the achievement of an imposed figure of merit with a minimum-cost route. Genetic algorithms were chosen as optimization heuristics to solve the modified version of the OTSP. To gain algorithm efficiency and reduce runtime, several modifications were made to the genetic operators, in order to guarantee population validity along the evolutionary process.

The efficiency of the modified GA is demonstrated over the classical GA for small-sized FPAA mapping, while the classical GA is used for larger FPAAs. Simulation results illustrate the applicability of the proposed evolutionary FPAA mapping algorithm based on the modified OTSP routine for the implementation of analog filters on FPAAs.

For future research, we target the integration of the proposed FPAA mapping algorithm along with a diagnosis routine into an auto-adaptive circuit, e.g. [15], in order to implement self-healing and self-repair features.

ACKNOWLEDGEMENTS

This paper is supported by the Sectoral Operational Programme Human Resources Development POSDRU/159/1.5/S/137516 financed from the European Social Fund and by the Romanian Government.

REFERENCES

[1] W. Arden, M. Brillouët, P. Cogez, M. Graef, B. Huizing, R. Mahnkopf, J. Pelka, J.-U. Pfeiffer, A. Rouzaud, M. Tartagni, C. Van Hoof, J. Wagner, "Towards a "More-than-Moore" roadmap", *Report from the CATRENE Scientific Committee*, November 8, 2011. [Online]. Available: http://www2.imec.be/content/user/File/MtM%20WG%20report.pdf [Accessed: 13 February 2015].

[2] European Nanoelectronics Initiative Advisory Council (ENIAC), *Draft of the Annual Work Programme 2012*, Version 9, 2011. [Online]. Available: http://www.aeneas-office.eu/web/downloads/strategic-docs/awp_2012_v9.pdf [Accessed: 13 February 2015].

[3] T. Ndjountche, *CMOS Analog Integrated Circuits: High-Speed and Power-Efficient Design*, CRC Press, 2011.

[4] S. Hintea, G Csipkes, D. Csipkes, L. Festila, R. Groza, P. Farago, M.Cirlugea, *Reconfigurable Analog Circuits for Mobile Communications - Variable topology filters and design automation*, Editura Casa cărtii de ştiinţa, 2011.

[5] D. Csipkes, G. Csipkes, S. Hintea, H. Fernandez-Canque, "An analog array approach to variable topology filters for multi-mode receivers", *Electronics and Electrical Engineering*, no. 9, pp. 43 – 48, 2010.

[6] P. Farago, D. Bogăţeanu, E. Ceuca, C. Moisă, S. Hintea, "Device mismatch analysis and effect compensation in fundamental analog cells", *2013 36th International Spring Seminar on Electronics Technology (ISSE)*, pp. 280 – 285, 2013.

[7] J. Becker, Y. Manoli, "Synthesis of Analog Filters on a Continuous-Time FPAA Using a Genetic Algorithm", *International Conference on Field Programmable Logic and Applications*, pp. 1-4, 2006.

[8] A. Stoica, R. Zebulum., D. Keymeulen, R. Tawel, T. Daud, A. Thakoor, "Reconfigurable VLSI Architecture for Evolvable Hardware: From Experimental Field Programable Transistor Array to Evolution-Oriented Chips", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 227-232, 2001.

[9] P. Farago, L. Feştilă, P. Söser, S. Hintea, "*Automatic Filter Synthesis Based on Tree Generation and Evolutionary Optimization*", Knowledge-Based And Intelligent Information And Engineering Systems, Lecture Notes in Computer Science, vol. 6884, pp. 455-464, 2011

[10] R.D. Kanphade, *Programmable and Reconfigurable Analog Signal Processing Using OTA1 Based Functionalities*, PhD Thesys, Sant Gadge Baba Amravati University, 2009.

[11] Z. Čičková, I. Brezina, J. Pekár, "Open Traveling Salesman Problem with Time Windows", 1st Logistics International Conference Belgrade, Serbia, 28 - 30 November, pp. 40-43, 2013.

[12] M. R. Bonyadi, M. R. Azghadi, H. Shah-Hosseini, "Population-Based Optimization Algorithms for Solving the Travelling Salesman Problem", in Ed. F. Greco, *Travelling Salesman Problem*, ISBN 978-953-7619-10-7, I-Tech, Vienna, Austria, September 2008.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[14] P. Farago, D. Bogăţeanu, E. Ceuca, C. Moisă, S. Hintea, "Device mismatch analysis and effect compensation in fundamental analog cells", *2013 36th International Spring Seminar on Electronics Technology (ISSE)*, pp. 280 – 285, 2013.

[15] P. Faragó, G. Csipkes, D. Csipkes, C. Faragó, S. Hintea, "An FPAA Approach to Adaptive Filter Design with Evolutionary Software-driven Reconfiguration", *Elektronika ir Elektrotechnika*, vol 20, no 5, pp. 89-96 , 2014.