An FPGA-based Approach to Multi-Objective Evolutionary Algorithm for Multi-Disciplinary Design Optimisation

Conference Paper · October 2011

CITATION
1

READS
306

4 authors:

Jon Kok
Australian Institute of Marine Science
11 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Luis Felipe Gonzalez
Queensland University of Technology
141 PUBLICATIONS **900** CITATIONS
SEE PROFILE

Neil Kelson
Queensland University of Technology
61 PUBLICATIONS **452** CITATIONS
SEE PROFILE

Jacques Periaux
CIMNE International Center for Numerical Methods in Engineering
424 PUBLICATIONS **5,377** CITATIONS
SEE PROFILE

EVOLUTIONARY AND DETERMINISTIC METHODS FOR DESIGN, OPTIMIZATION AND CONTROL
C. Poloni, D. Quagliarella, J. Périaux, N. Gauger and K. Giannakoglou (Eds.)

AN FPGA-BASED APPROACH TO MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM FOR MULTI-DISCIPLINARY DESIGN OPTIMISATION

Jonathan Kok, Felipe Gonzalez
Australian Research Centre for Aerospace Automation (ARCAA)
Queensland University of Technology (QUT)
Eagle Farm, 4009 Queensland, Australia
Email: jonathan.kok@student.qut.edu.au; felipe.gonzalez@qut.edu.au
Web page: www.arcaa.aero

Neil Kelson
High Performance Computing and Research Support Group, Division of TILS
Queensland University of Technology (QUT)
Brisbane, 4000 Queensland, Australia
Email: n.kelson@qut.edu.au
Web page: www.itservices.qut.edu.au/hpc

Jacques Periaux*
International Center for Numerical Methods in Engineering (CIMNE)/UPC
Universidad Politecnica de Cataluna
Campus Norte UPC, 08034 Barcelona, Spain
Email: jperiaux@gmail.com
Web page: www.cimne.upc.edu

Abstract. This paper investigates the field programmable gate array (FPGA) approach for multi-objective and multi-disciplinary design optimisation (MDO) problems. One class of optimisation method that has been well-studied and established for large and complex problems, such as those inherited in MDO, is multi-objective evolutionary algorithms (MOEAs). The MOEA, nondominated sorting genetic algorithm II (NSGA-II), is hardware implemented on an FPGA chip. The NSGA-II on FPGA application to multi-objective test problem suites has verified the designed implementation effectiveness. Results show that NSGA-II on FPGA is three orders of magnitude better than the PC based counterpart.

Key words: Field programmable gate array (FPGA), multi-disciplinary design optimisation (MDO), multi-objective evolutionary algorithm (MOEA)

1 INTRODUCTION

Multi-disciplinary design optimisation (MDO) has been and is actively applied for solving design problems in aerospace, mechanical and electrical engineering. The aim of MDO is to generate superior designs by the simultaneous exploitation of incorporated interactions between disciplines[1]. However, the inclusion of interacting subsystems increases the problem complexity and resource requirements, where the search space is larger and the associated objective functions are computationally and memory intensive. Therefore, robust and efficient optimisation methods that are also practical in terms of computational run-time are indispensable in the field of MDO.

One class of optimisation method that has been well-studied and established for large and complex problems, such as those inherited in MDO, is Multi-Objective Evolutionary Algorithms (MOEAs)[2,3,4]. MOEAs belong to a class of generic population-based metaheuristic optimisation methods built from the principles of biological evolution. MOEA simultaneously optimises a set of candidate design solutions through genetic operations that explore and exploit interesting regions of the search space without *a priori* knowledge, thus offering exceptional search adaptability for finding Pareto fronts on large and complex unknown problem domains. MOEAs have been suggested primarily because of their ability to emphasise the search towards the true Pareto optimal region and obtain a set of Pareto optimal design solutions in one simulation run. The nature of MOEAs being population-based metaheuristics makes them well suited for solving MDO problems[2], as the fundamental evolutionary process deals simultaneously with a population of candidate design solutions which are iteratively improved after each generation. Hence, constantly maintaining and producing a population of optimised design solutions that relates to a particular Pareto front. MOEA techniques differ in three implementation details — namely, fitness assignment, diversity preservation, and elitism.

One approach to speed up the runtime of MOEAs and MDO methods is to implement its behaviour on an FPGA device, where true parallel execution and hardware dedication is possible. An FPGA device is made up a finite number of programmable logic components to be configured for performing complex combinational functions. FPGA technology benefits from faster response times and customised functionality to accurately meet application requirements, which is contributed by the capability of controlling the design from the hardware level. With this, MDO methods that take several hours to run could be executed in factors of seconds, impacting significantly on the development cycle and cost of a project. Furthermore, the nature of MOEAs being population-based in which individuals optimises independently, makes them well suited for the adoption of true hardware parallelism on FPGA. Theory on FPGA programming and features can be found in Chu[5].

This paper describes the extension of previous work by Kok *et al.*[6] on coupling FPGA to MDO search algorithms. Our proposed hardware design adopts main features from the popular nondominated sorting genetic algorithm II (NSGA-II)[7], such as crowding distance assignment[8] and domination-based Pareto front ranking[9].

The rest of the paper is organised as follows: Section 2 presents the methodology for the NSGA-II and its FPGA mapping aspects. Section 3 shows validation and comparison of the NSGA-II and NSGA-II on FPGA by solving multi-objective mathematical test. Finally, Section 4 presents the conclusions.

2 METHODOLOGY

In multi-objective optimisation, all relevant disciplines are associated with one or several optimisation objectives, there is no single design solution that is uniquely optimum as compared to every other possible design solution with respect to all objectives, given that its optimality is subjected to the compromise arising from other conflicting objectives. Thereby, the goal of MOEA is to obtain a set of design solutions in which no other solutions are superior to those in its set when all objectives are considered. This non-dominated set of solutions, also known as the Pareto front, provides decision makers with the baseline for making an educated compromised choice from amongst the many.

In the following subsection, the evolutionary optimisation methods, the NSGA-II and NSGA-II implemented on FPGA are presented.

2.1 NSGA-II

In the instance of the NSGA-II, it incorporates fast nondominated sorting, crowding distance assignment, and elitism selection process[7].

The NSGA-II algorithm (see Figure 1) is a well known algorithm[7] but its description is repeated here as to provide background for the NSGA-II on FPGA. It starts by initialising the parent population, P_0, randomly, upon which each of the M objectives value, $V_m, m \in \{1, \ldots, M\}$, is evaluated as accordingly. The population is made up of N individuals that are represented by a vector consisting of a candidate solution, $\mathbf{x}$, its objective values, $\{V_1, \ldots, V_M\}$, the overall fitness value, F, and neighbourhood diversity value, D. Beginning of each generation, t, an offspring population, Q_t, which has the same chromosome structure as P_t, is selected for the evolutionary process with the selection pressure focusing on the elites which have better F and D values. Q_t then undergoes genetic operations involving simulated binary crossover (SBX) and polynomial mutation. The permutated Q_t is then evaluated for each of the M objectives value, $V_m, m \in \{1, \ldots, M\}$. Next, the concatenated $R_t = P_t \bigcup Q_t$ is fast nondominated sorted, where F of each chromosome is assigned a rank value according to the nondominated front it lies on with respect to every other chromosome in R_t. A generation cycle is completed with the assignment of D for each individual in R_t, which is a diversity value according to the crowding distance it constitutes with respect to the adjacent chromosomes on its nondominated front rank. The NSGA-II algorithm executes iteratively until a specified stopping criteria such as maximum number of generations, t_{max}, has been met.

```
NSGA-II()
Initialise population()
WHILE (Termination Criteria NOT MET)
      Elitism selection technique()
      Genetic operations()
      Objectives evaluation()
      Fast nondominated sorting()
      Crowding distance assignment()
END
```

Figure 1: Pseudocode for NSGA-II.

2.2 NSGA-II on FPGA

The theoretical foundations of the evolutionary algorithms rely on a binary coded representation, where genetic operators produce the best outcome due to the binary-chained nature by which biological evolution is handled[10]. This binary coded aspect is complimentary when mapping MOEA on FPGA, where hardware circuits operate on a binary logic level. Using modern FPGA design softwares, such as Xilinx ISE design suite, circuits can be easily designed and implemented for rapid prototyping on hardware, thereby avoiding the long fabrication processing of application specific

integrated circuit (ASIC) design. After configuring a predefined circuit on an FPGA, system developers are allowed to make design changes or functional enhancements as necessary without requiring the time and cost involved in ASIC redesign, hence offering viable long-term maintainability. In operational mode, the parallelism and pipelining design capabilities of FPGAs contribute to the significant speedup over instruction stream processors.

2.2.1 Overview

The proposed algorithmic architecture of the NSGA-II on FPGA, which exploits parallelism on an iteration level, is depicted below in Figure 2. The algorithm incorporates the key features, fixed-point representation, random number generator, crossover, mutation, Pareto front ranking, crowding distance assignment, selection, and evaluation, which are needed for securing diverse Pareto-optimal fronts. The population, consisting of candidate design solutions, is stored on the dedicated block RAM onboard the FPGA. Tournament selection is randomly carried out across the population, determining better candidate design solutions to be genetically altered, which produces the preceding offspring population. After the offspring have been crossovered, mutated and evaluated, they are concatenated with the parent population to undergo Pareto front ranking and crowding distance assignment. The higher ranking and wider spread solutions are updated back into the population block RAM. It should be noted that the data flow arrow in Figure 2 denotes parallel processing.

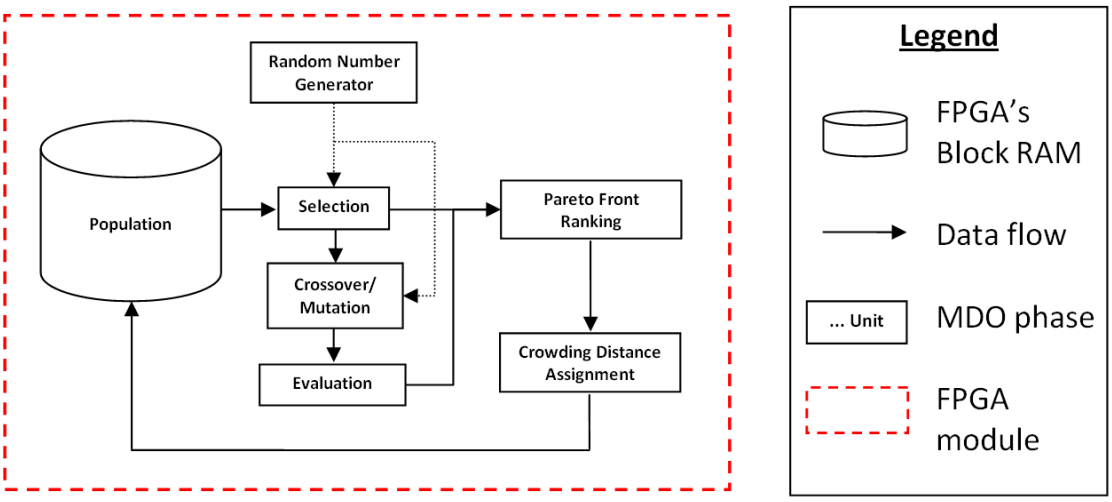

Figure 2: Architecture of the FPGA-based NSGA-II algorithm.

2.2.2 Representation

The representation of candidate design solutions can be coded in real-coded binary, floating-point or fixed-point numbers. Michalewicz[10] experimented and concluded that floating-point and fixed-point representation is faster, more consistent and higher in precision than real-coded representation. Since the complexity of floating-point arithmetic consumes a larger logic footprint and is not as efficient

as fixed-point arithmetic, representation of the candidate parameters is therefore encoded in fixed-point format.

2.2.3 Random number generator

Randomness and periodicity are two main factors to consider when implementing a logic level random number generator (RNG) for an FPGA. Matsumoto and Nishimura[11] proposed a pseudo RNG called Mersenne Twister, which was argued to be as fast and random as the standard ANSI-C "rand()". The Mersenne Twister is essentially a uniformly distributed pseudo RNG based on a matrix linear recurrence over a large finite binary field. Another advantage of FPGA implementation of a Mersenne Twister is its low resource consumption and its ability to generate new random sequences at every clock cycle.

2.2.4 Population module

The Population module, which is implemented as a block RAM, contains the population matrix comprising n chromosomes of candidate solution, $\mathbf{x}$, objective values, $\mathbf{V}$, fitness value, F, and diversity value, D, (see Figure 3). The Population module is responsible for broadcasting selected chromosomes for the evolutionary process. At the end of each generation, it receives the evaluated offspring and updates them into the Population module.

Chromosome index	Candidate solution			Objective values			Fitness value	Diversity value
1	$x_{1,1}$	...	$x_{1,N}$	$V_{1,N+1}$	...	$V_{1,N+M}$	$F_{1,N+M+1}$	$D_{1,N+M+2}$
2	$x_{2,1}$	...	$x_{2,N}$	$V_{2,N+1}$	...	$V_{2,N+M}$	$F_{2,N+M+1}$	$D_{2,N+M+2}$
⋮	⋮		⋮	⋮		⋮	⋮	⋮
n	$x_{n,1}$	...	$x_{n,N}$	$V_{n,N+1}$	...	$V_{n,N+M}$	$F_{n,N+M+1}$	$D_{n,N+M+2}$

Figure 3: Design of Population module.

2.2.5 Crossover/mutation

An SBX operation and a polynomial mutation proposed by Deb and Agarwal[12] are implemented. The intention of a crossover operation is to exchange useful information between two candidate solutions, whereas a mutation operation is aimed to slightly alter a candidate solution. Thus, crossover and mutation can be seen as exploitation and exploration respectively. A balance between them is applied to guide a search algorithm.

2.2.6 Evaluation

The evaluation module consists of the analysis functions to be optimised for the specific MDO application.

2.2.7 Pareto front ranking

Pareto front ranking is based on the non-dominance feature of a candidate solution[9]. Design solution A is said to dominate design solution B if all of design solution A's

fitness is better than design solution B's, else design solution B is non-dominated. Non-dominated design solutions are allocated higher rank than dominated ones, hence ensuring the preservation of Pareto optimal design solutions.

2.2.8 Crowding distance assignment

A technique proposed by Deb[9] known as crowding distance assignment is implemented to maintain the diversity of the Pareto fronts. The advantage of this technique lies in the nature by which it operates, whereby it does not require any performance dependent parameter.

2.2.9 Selection

Tournament selection based on the Pareto front rank and crowding distance is used as a competition winning criteria for the next generation of offspring. Higher ranking solutions wins over lower ranking solutions. If two solutions are of the same rank, the solution with higher crowding distance wins. This method ensures the survival of the fittest.

2.3 NSGA-II on FPGA Implementation

The NSGA-II features described above are translated into very-high-speed integrated circuits hardware description language (VHDL), which is a hardware description language used to describe logic circuitry for FPGAs.

3 EXPERIMENTS AND RESULTS

3.1 Test Problems

The general formulation of a multi-objective optimisation problem can be represented in the following form:

$$\left.\begin{array}{lll} \text{Minimise/Maximise} & f_m(\mathbf{x}), & m = 1, 2, \ldots, M; \\ \text{subjected to} & g_j(\mathbf{x}), & j = 1, 2, \ldots, J; \\ & x_n^{(L)} \leq x_n \leq x_n^{(U)}, & n = 1, 2, \ldots, N. \end{array}\right\} \quad (1)$$

where $f_m, m \in \{1, \ldots, M\}$, are the objective functions to be minimised or maximised, $\mathbf{x} = (x_n, \ldots, x_N), n \in \{1, \ldots, N\}$, is the "optimisation vector" of N design variables that are individually restricted by lower $x_n^{(L)}$ and upper $x_n^{(U)}$ bounds, and $g_j, j \in \{1, \ldots, J\}$, are constraint functions.

Four multi-objective mathematical test problems are used to validate the performance of NSGA-II on FPGA. The first two test problems are SCH[13] and FON[14] where the true Pareto fronts are convex and concave, respectively. The subsequent two test problems are POL[15] and KUR[16] where both true Pareto fronts are difficult, discontinuous and concave. These test problems are described in Table 1. Note that concave problems pose difficulties for classical weighted sum approaches[9].

3.2 Algorithm Parameters

The software version of the NSGA-II is implemented on an Intel(R) Core(TM)2 Duo CPU E8600 @ 3.33GHz, 3.49 GB of RAM, whereas the proposed NSGA-II on FPGA was implemented and simulated on a Xilinx Virtex 4 (xc4vlx200-11ff1513).

The parameters used for both algorithms and all the test problems are set according to Deb[9] recommendations:

- population size = 100,
- number of generations = 250,
- crossover rate = 90%,
- mutation rate = 10%.

Test problem	Objective functions	Pareto optimal front geometry
SCH	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	Convex
FON	$f_1(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}}))$ $f_2(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}}))$	Concave
POL	$f_1(x) = [1 + (A_1 - B_1)^2 + (A_2 - B_2)^2]$ $f_2(x) = [(x_1+3)^2 + (x_2+1)^2]$ $A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2$ $A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2$ $B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2$ $B_2 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 1.5\cos x_2$	Discontinuous, concave
KUR	$f_1(x) = \sum_{i}^{2}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$ $f_2(x) = \sum_{i=1}^{3}\lvert x_i\rvert^{0.8} + 5\sin x_i^3$	Discontinuous, concave

Table 1: Test problems used in this study.

3.3 Experimental Results

Five simulation runs are experimented for each SCH, FON, POL, and KUR test problems to test the effectiveness of NSGA-II on FPGA. Figure 4 and Figure 5 show one of the results comparing the Pareto front obtained by the NSGA-II and NSGA-II on FPGA for the test problems. It can be seen that good solution quality was achieved for each test problem. The computational runtime results are shown in Table 2.

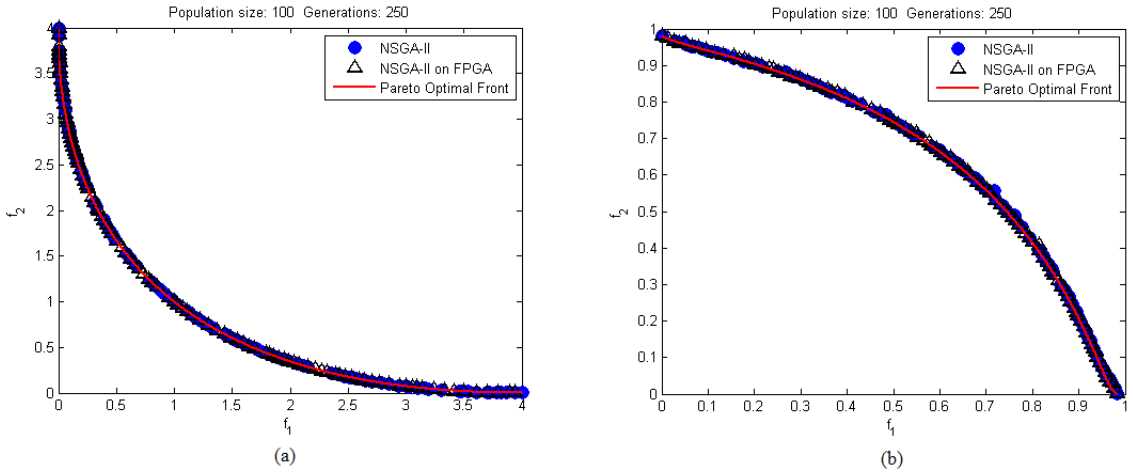

Figure 4: Comparison of Pareto front obtained by the NSGA-II and NSGA-II on FPGA for (a) SCH and (b) FON.

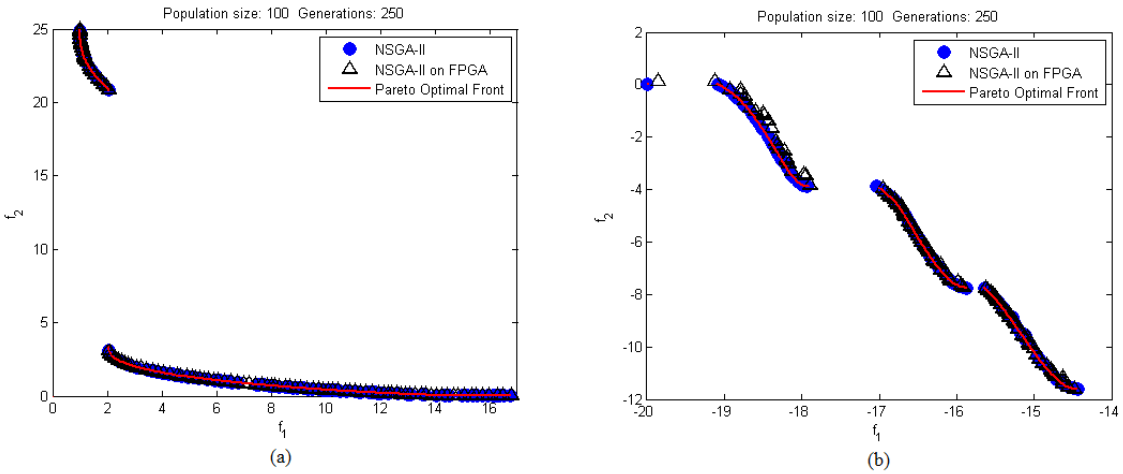

Figure 5: Comparison of Pareto front obtained by the NSGA-II and NSGA-II on FPGA for (a) POL and (b) KUR.

Test problem	Computational run-time (ms)		Speed improvement
	NSGA-II	NSGA-II on FPGA	
SCH	6 902.4	6.25	× 1 104
FON	7 981.2	6.42	× 1 243
POL	4 341.0	7.14	× 608
KUR	9 535.8	6.70	× 1 423

Table 2: Computational run-time results averaged over five simulation runs.

Size and execution speed analysis of the designed NSGA-II on FPGA is as follows. There is a total 178 000 configurable logic blocks utilised on the FPGA, which is equivalent to 5% of the overall resources available on a Xilinx Virtex 4 FPGA chip. The maximum working frequency is 32 MHz. The selection module takes two clock cycles to compare two random candidate design solutions. The crossover and mutation modules each takes one clock cycle to perform genetic operation on each candidate design solution. The Pareto front ranking and crowding distance assignment modules each takes 100 clock cycles to compare the entire population. Since the objective is to verify the NSGA-II on FPGA effectiveness, the evaluation module is implemented as lookup tables, in which takes one clock cycle to assign the appropriate objective values.

4 CONCLUSION

The hardware implementation of the NSGA-II algorithm for MDO problems is proposed in this paper. The NSGA-II is effective without *a priori* problem knowledge and capable of simultaneously optimising the design problem in a single simulation run. The proposed NSGA-II on FPGA performance is demonstrated from four examples and comparisons. The simulation results indicate that the solutions obtained from the NSGA-II on FPGA are comparable to its software counterpart. The hardware NSGA-II implementation achieved a speed up of approximately 1 300 times over the software implementation, which is attractive for practical multi-objective and MDO applications.

ACKNOWLEDGEMENTS

Computational resources and services used in this work were provided by the Australian Research Centre for Aerospace Automation (ARCAA) and the High Performance Computing and Research Group of the Queensland University of Technology (QUT), Brisbane, Australia.

REFERENCES

[1] Sobieszczanski-Sobieski, J. and Haftka, R. T. Multi-disciplinary aerospace design optimization: Survey of recent developments. *Structural and Multidisciplinary Optimization* **14**, 1–23 (1997).

[2] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. *Evolutionary algorithms for solving multi-objective problems*. Genetic and Evolutionary Computation Series. Springer, (2007).

[3] Lee, D. S., Gonzalez, L. F., Periaux, J., and Srinivas, K. Robust design optimisation using multi-objective evolutionary algorithms. *Computers and Fluids* **37**(5), 565–583 June (2008).

[4] Zitzler, E. Two decades of evolutionary multi-criterion optimization: A glance back and a look ahead. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, 318 (, Honolulu, Hawaii, 2007).

[5] Chu, P. P. *FPGA prototyping by VHDL examples: Xilinx Spartan-3 version*. Wiley-Interscience, (2008).

[6] Kok, J., Gonzalez, F., Kelson, N., and Gurnett, T. A hardware-based multidisciplinary design optimisation method for aeronautical application. In IV International Conference on Computational Methods for Coupled Problems in Science and Engineering, M. Papadrakakis, E. O. and Schrefler, B., editors, (2011).

[7] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197 April (2002).

[8] DeJong, K. A. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, (1975).

[9] Deb, K. *Multi-objective optimization using evolutionary algorithms*. Wiley-Interscience series in systems and optimization. John Wiley and Sons, 1st edition, (2001).

[10] Michalewicz, Z. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, 3rd rev. and extended edition, (1996).

[11] Matsumoto, M. and Nishimura, T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Transactions on Modeling and Computer Simulation* **8**(1), 3–30 January (1998).

[12] Deb, K. and Agrawal, R. B. Simulated binary crossover for continuous search space. *Complex Systems* **9**(2), 115–148 (1995).

[13] Schaffer, J. D. *Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition)*. PhD thesis, Vanderbilt University, Nashville, TN, USA, (1984).

[14] Fonseca, C. M. and Fleming, P. J. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation* **3**(1), 1–16 (1995).

[15] Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering* **186**(2-4), 403–420 June (2000).

[16] Kursawe, F. A variant of evolution strategies for vector optimization. In Parallel Problem Solving from Nature, Schwefel, H.-P. and Manner, R., editors, volume 496 of *Lecture Notes in Computer Science*, 193–197. Springer Berlin / Heidelberg (1991).