Benchmarking in Digital Circuit Design Automation

LECH JÓŹWIAK, DOMINIK GAWŁOWSKI, ALEKSANDER ŚLUSARCZYK
Faculty of Electrical Engineering
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
THE NETHERLANDS
L.Jozwiak@tue.nl
http://www.ics.ele.tue.nl/~ljozwiak/

Abstract: - This paper focuses on benchmarking, which is the main experimental approach to the design method and EDA-tool analysis, characterization and evaluation. We discuss the importance and difficulties of benchmarking, as well as the recent research effort related to it. To resolve several serious problems related to quality of benchmarking and use of practical industrial benchmarks, we proposed an adequate benchmarking methodology based on the statistical experimental design approach, and developed corresponding digital circuit benchmark generators. These benchmark generators enable research, evaluation and fine-tuning of circuit synthesis methods and EDA-tools largely independent of the actual industrial benchmarks, and much better than having only some industrial benchmarks. Using the results of extensive experiments that involved large sets of diverse benchmarks generated with our FSM benchmark generator, we discuss several crucial problems of benchmarking and demonstrate how to resolve them.

Key-Words: - Microelectronic Circuit Design, Digital Circuits, EDA-tools, Benchmarking, Statistical Experimental Design, Benchmark Set Quality

1 Introduction

The recent spectacular progress in microelectronics enabled implementation of a complex system on a single chip, autonomous and mobile computing, wire-less communication, global networking, and facilitated a fast progress in these areas. A big stimulus has been created towards development of various kinds of embedded and high-performance systems. On the other hand however, the spectacular advances introduced unusual silicon and system complexity, and created many new difficult to solve issues. The opportunities created by modern microelectronics cannot effectively be exploited without adequate methods and electronic design automation (EDA) tools for designing of high-quality circuits and systems. Therefore, the quality of the EDA methods and tools, as well as of the methods and means for their quality analysis is of crucial importance.

Two basic approaches are used to study performance of objects (e.g. methods or tools): analytical and experimental. The analytical approach is based on (mathematical) models and use of the logical analysis and proof methods. The experimental approach is based on performing sets of tests, collecting and processing data from the tests, and drawing logical conclusions from the collected and processed data.

This paper is related to experimental performance analysis of the digital circuit design methods used in EDA-tools. It focuses on benchmarking that is one of the main and most difficult issues of the design method and EDA-tool analysis, characterization and evaluation. A benchmark is a design problem that constitutes a point of reference against which methods or tools can be measured, analyzed, characterized or compared in order to assess their (relative) performance. Benchmarking is the process in which, using benchmarks, various aspects of a methods or tools are measured, analyzed, characterized, evaluated or compared. In this paper, we discuss several crucial problems of benchmarking, specifically related to the adequacy of the benchmark set and evaluation of benchmarking results, and demonstrate how to resolve them.

While some benchmarking problems related to combinational circuit design were addressed by researchers in recent years, and both an approach and several particular solutions have been proposed (see e.g. [6][22][33][18][29][30]), the benchmarking in sequential design is a much less recognized territory. The paper [26] authored by the first author of this paper and [23] belong to unique works in this field. Therefore, in this paper we focus on benchmarking in sequential design. Despite this fact, a large part of the discussion of the paper and its

more general conclusions are not limited to the sequential design, but can be extended through analogies to the whole field of digital circuit design, and some of them even to a larger area of electronic design automation.

The main aim of the paper is to discuss several crucial benchmarking problems, such as the complex problem of the benchmark set quality or the lack of satisfactory academic and industrial benchmark sets, and to propose and discuss their adequate solutions. The sequential synthesis methods implemented in the contemporary EDA-tools are used here rather in the role of examples. However, the realization of the main aim includes demonstration that an adequately performed benchmarking is in state to reveal strengths and weaknesses of the benchmarked methods. Therefore, we will also briefly comment on the quality of the sequential circuit synthesis methods implemented in the contemporary EDA-tools, and show a big room for their improvement.

In recent years a growing fraction of the electronic systems is implemented using the FPGA-technology. Several reasons of this trend are explained [28]. We therefore performed our experimental research using as test bed methods and EDA-tools for FPGA-targeted circuit synthesis. To resolve serious problems related to the use of practical industrial benchmarks and ensuring the adequate quality of the benchmark set, we developed and implemented sequential and combinational benchmark generators. These benchmark generators make it possible for us to efficiently construct well-characterized circuit specifications with different characteristics, including specifications representative to various typical industrial application areas. The generators greatly reduce the necessity of having the actual industrial benchmarks and enable research, comparison, evaluation and fine-tuning of circuit synthesis methods and tools largely independent of the industry, and much better than having only some industrial benchmarks. In this paper, the discussion of several benchmarking issues and their solutions is performed and illustrated using the results of our experimental research that involved several sequential circuit synthesis methods implemented in the contemporary academic and commercial EDA-tools, and large sets of diverse FSM benchmarks generated with our Finite State Machine (FSM) Benchmark Generator ***BenGen*** [32].

The contributions of the research reported in this paper include amongst others:

- discussion of the importance and difficulties of benchmarking in the EDA field, as well as of the recent research effort related to it;
- proposal of an adequate benchmarking methodology based on the statistical experimental design approach, as well as an appropriate generation and application of the high-quality representative benchmark sets;
- the first published finite state machine (FSM) benchmark generator that enables an efficient construction of various well-characterized sequential circuit specifications and representative FSM benchmark sets;
- experimental analysis of several FSM state assignment methods using large sets of benchmarks from this generator that demonstrated several important benchmarking issues, mainly related to the benchmark set representativeness and statistical processing of the benchmarking results;
- demonstration that using only a single statistic on the whole benchmark set the information obtained and conclusions are very limited, while considering more statistics and/or narrower benchmark classes gives more, more precise and more useful information and conclusions;
- demonstration of the trade-off between the benchmarking time, the amount and preciseness of information obtained from a benchmarking experiment, and the confidence level of the statistical conclusions;
- demonstration that an inadequately performed benchmarking, and specifically performed when using a non-representative benchmark set, can easily result in faulty conclusions and this way mask important quality issues, while an adequately performed benchmarking is able to reveal significant quality concerns;
- demonstration that the pragmatic FSM state assignment approaches commonly applied in today's commercial circuit synthesis tools for FPGAs are effective in only some special cases.

2 Benchmarking issues in EDA

In this section, we will briefly discuss some of the major benchmarking issues in the EDA field, when focusing on benchmarking of digital circuit design methods used in the EDA-tools. The main problems in digital circuit design are computationally complex (NP-hard [16]), including the FSM state assignment problem discussed further in this paper [26][27]. In general, finding a strictly optimal solution to such a problem requires a solution time that exponentially grows with the problem dimensions. Therefore, heuristic solution methods are used for such problems. The heuristic

methods, although not guaranteeing the strict optimality, are often able to find some high-quality solutions in an acceptably short time and using an acceptable quantity of other resources. In consequence, in most of the practical cases, the strictly optimal solutions remain unknown. Even if one of them is actually obtained, it is usually unknown that the obtained solution is optimal. Analytical performance study of the heuristic methods for these complex problems is also complex itself, and consequently, the experimental approach is typically used. This paper is therefore devoted to the experimental performance analysis of digital circuit design methods and tools. The experimental performance analysis uses some test cases (benchmarks) for performing the sets of tests.

In general, a **benchmark** is a point of reference against which objects can be measured, analyzed, characterized or compared in order to assess their (relative) performance. **Benchmarking** is the process in which, using benchmarks, various aspects characteristics of a particular object are measured, analyzed, characterized, evaluated or compared in relation to values of the same characteristics of other similar objects serving the same or similar purposes, or in relation to the best (actual or predicted) values of those aspects, to their other characteristic values or their values' distributions.

In the design area, a benchmark corresponds to a design problem against which design methods, processes, techniques, algorithms, tools, etc. can be measured, analyzed, characterized or compared. In the field of digital circuit design, a benchmark corresponds to a circuit specification in a particular form (e.g. HDL, tabular, diagram, graph, net-list etc.) to which circuit design methods, processes, techniques, tools, algorithms etc. can be applied, and against which they can be measured, analyzed, characterized or compared in order to assess their (relative) effectiveness and/or efficiency. The effectiveness and efficiency are typically expressed through the result quality (e.g. the circuit's area, speed, power dissipation, testability, manufacturability,…), robustness (i.e. the ability to steadily deliver some high-quality solutions in the whole range of possible input data), and use of resources (e.g. computation resource use, as CPU-time or memory use, or human resource use).

Benchmarking is one of the main issues of the design method and EDA-tool development and quality assurance. By providing a feedback on the method or tool qualities, benchmarking is a major source of information used to control the direction of the method or tool development and improvement, and has a decisive influence on their final quality. Adequately performed benchmarking may reveal unknown inefficiencies or new issues and trigger work towards innovations enabling their elimination or resolution. Moreover, benchmarking enables the design method and tool users to analyse and compare alternative methods or tools and to decide which of the available alternatives best suit their particular needs. It is thus of primary importance for both the design method and tool researchers and developers, and the method and tool users.

Summing up, the circuit design methods and tools can be thoroughly analyzed, characterized, compared, and evaluated by applying them to some sets of (practical or hypothetical) designs (benchmarks), and analyzing the results produced by them. However, the value of the experimental research performed when using benchmarks, as well as the value of the conclusions drawn based on the results from such research, heavily depend on the quality of the benchmark set used for the experiments, as well as the way the experiments are performed and their results analyzed.

Unfortunately, the benchmarking experiments in the EDA area are typically performed using a relatively small number of isolated, unrelated, unsystematically collected and not characterised benchmarks. Processing of the results from these experiments is usually also extremely simplistic. In most cases it is limited to presenting some raw data on an unsystematic, not characterised and relatively small benchmark sets, and computing some totals or simple averages for such benchmark sets. If the benchmarks' characteristics and their distributions are unknown, those totals or averages are in most cases meaningless. Moreover, it is impossible to reliably draw any more general conclusions from the raw data on an unsystematic, not characterised and/or small benchmark set. For example, in [2] the algorithms were tested on just a single instance of one particular problem, and in [41] on only four instances of a single problem differing in size, but without accounting for any other features or factors of the problem. This sort of experimentation may only be used as an example further explaining the algorithms, but it does not allow for any generalizations and any general conclusions regarding the algorithms' qualities. In [15] and [35], the methods proposed are tested on just a few "standard" benchmarks. This can only give a first impression on how the methods behave for a few typical problem cases, but it does not allow for any reasonably reliable generalizations. It even does not allow for a reliable simple average computation, because the number of cases is not statistically significant and the different problem features are not

well covered by the experimentation. In [40] the benchmarking experiments were performed somewhat better. In parallel to a few "standard" benchmarks ten randomly generated benchmarks with different parameters were used. Results on these generated benchmarks show somewhat more on how the methods perform for different cases and may already allow for a computation of a reasonably reliable simple average. The number and diversity of benchmarks are however still much too low to make any more precise characterisation of the methods. Unfortunately, the cases briefly discussed above are not exceptional. They represent the typical benchmarking practice in the electronic design, EDA and related areas, which is far from being of an acceptable quality.

Furthermore, the method (tool) evaluations made on isolated results obtained for isolated benchmarks do not provide much information on their actual quality, even for these isolated cases. Since in most cases the strictly optimal solutions for a typical computationally complex circuit design problem remain unknown, there is no reference point to which the isolated results could be compared, and even if the result of one method or tool is better than from another, it is still unknown how bad (or good) both these results are. In 1990 the first author of this paper proposed a technique that overcomes the above problems of benchmarking result analysis and evaluation. In particular, he demonstrated that distribution of solutions for the FSM state assignment problem can be well estimated by the normal (Gaussian) distribution and showed that the normal distribution can be used to well model many other complex circuit design problems having analogous properties [26]. He proposed to evaluate the performance of the design methods and tools for solving such problems through comparison of the result from a method on a particular benchmark to the normal distribution of results for this benchmark[26][27]. This result evaluation technique is briefly explained further in this paper. Although this solved in a way the problem of the result evaluation, the questions related to the benchmark set quality remained open, and we started to work on them.

3 Experimental design approach to benchmarking

Although this paper is devoted to benchmarking in EDA, and particularly in digital circuit design automation, the problem of an adequate benchmarking is very important to many areas. In particular, in parallel to our works in the EDA benchmarking, several researchers in the area of heuristic methods and algorithms (for different than EDA applications) also tried to find a satisfactory solution for the benchmarking in their area. We will briefly discuss the works in the area of heuristic method and algorithm benchmarking, because both the area and the research results are related to the area and results of our research presented in this paper. The researchers of heuristic methods and algorithms proposed to adapt to the benchmarking in their area the statistical experimental design (or design of experiments) paradigm pioneered by R. A. Fisher [14] and used in many areas, including physical sciences, medicine, agriculture and other life sciences [4][9]. The statistical experimental design methodology consists in an adequate planning and performing of an experiment which ensures that appropriate data is collected and analyzed by statistical methods to obtain meaningful and satisfactorily objective conclusions. In particular, in his paper from 1994 [19], J. N. Hooker argued that an empirical science of algorithms is needed to alleviate several benchmarking problems, in the sense of investigating "how algorithmic performance depends on problem characteristics". He wrote: "Rather than agonize over whether a problem is representative of practice, one picks problems that vary along one or more parameters", and suggested usage of generated benchmarks and adoption of the experimental design paradigm. Although we agree with many ideas presented in his paper, we do not support his suggestion that "whether a problem is representative of practice" is not so important. In our opinion, the practical representativeness of problems and varying of problems along various parameters are both important, and both should be taken care of when preparing and performing the benchmarking experiments. We will more extensively comment on this further in this paper. Our opinion is shared by several other researchers and practitioners, including R. S. Barr and his collaborators who wrote: "Real-world problems reflect the ultimate purpose of heuristic methods, and are important for assessing the effectiveness of a given approach. Of particular value are instances that are representative, in some manner, of those encountered in a given problem domain" [3]. Also Hooker himself partly changed his opinion on this aspect, and in his next paper [20] wrote: "…the well-known pitfalls of this approach (random generation of test problems), the most obvious of which is that random problems generally do not resemble real problems."

To overcome some of the benchmark set

representativeness and result processing problems, he proposes to adopt a type of experimental design known as factorial design. An analogous proposal is included in the paper by R. S. Barr at al [3], who stressed that: "A variety of different types of problems should be generated to reflect the diversity of factors that could be encountered. Generated problem instances should be representative of problems likely to be found in the field or designed to explore key hypotheses. In general, the more test problems evaluated, the more informative the study". In general, we agree with the above statements, but with the last one only upon the condition that the numerous test problems are divergent and their characteristic features (factor values) are known and taken into account by the statistical result analysis.

The main ideas of factorial design are as follows. First a set of factors is found, such that each of them is an independent controllable variable that may substantially affect the experiment results (e.g. performance measures) being dependent variables. The factors can e.g. represent the problem size, its various functional or structural features, etc. For each factor, its several value subsets or ranges are distinguished (e.g. size levels, function types or structure types). For each possible combination of the factors' value ranges a randomized problem (benchmark) set of statistical size is generated. For each such problem set meaningful average values of the experiment results can be computed and statistical analysis can be performed to reveal the influence of particular factors on the experiment result or interactions among the factors. The statistical factorial design of experiments is based on basic principles of test replication and randomization to offset unaccounted factors and enable a meaningful statistical processing.

Our benchmarking methods and related digital circuit benchmark generators discussed further in this paper overcome the problems of the benchmark set representativeness and result processing bias, because they efficiently combine the best features of the two different experimental analysis techniques: factorial design and testing on practical designs. They are based on the factorial design, but they do not generate random benchmarks. They generate benchmarks whose structure and other features very closely resemble those of the actual practical designs. Moreover, our benchmarking methods are able to use as some of the benchmarks just the available practical benchmarks.

In [21] H. Hoos and T. Stützle criticized the common simplistic practice of the benchmarking results processing: "Usually, the final performance measure is obtained by averaging over all instances from the test set. This last step, however, is potentially extremely problematic. …a fundamental problem with averaging … is the mixing of two different sources of randomness in the evaluation of algorithms: the nondeterministic nature of the algorithm itself, and the random selection of problem instances." Further in their paper, they focused on the first problem being the nondeterministic nature of algorithms. In this paper, we focus on the second problem being an adequate selection of problem instances to ensure meaningful averages and other results of statistical analysis. In the scope of the research presented in this paper the first problem was inactive, as the methods of the FSM state assignment considered are deterministic.

In the field of digital circuit design and its automation, a benchmark typically represents a digital circuit specification at a certain stage of the circuit design. Such a circuit (benchmark) has its own specific functional, structural and parametric characteristics. The experimental analysis (benchmarking) of the circuit design methods and tools is based on running them on sets of benchmarks, collecting and processing data from the tests, and drawing logical conclusions from the collected and processed data. The circuit (benchmark) characteristics relevant to a specific experimental performance analysis represent the factors of the statistical factorial design. According to different values of particular characteristics (factors) or characteristics combinations different benchmark classes can be distinguished.

In parallel to our works on benchmarking in EDA, F. Brglez and his collaborators N. Kapur, D. Gosh and R. Drechsler performed research on scientific experimentation targeted to EDA benchmarking, and specifically, combinational circuit benchmarking, from which results were published in the late 1990s [7][33]. Their combinational circuit benchmarking method through analysis of equivalence class mutant circuit distributions is in fact a specific kind of the statistical factorial design. In brief, they proposed to consider a number of known combinational circuit benchmarks and for each such benchmark construct a large sample of mutant circuits that belong to the same equivalence class of signature-invariant circuits. The reference circuit represents a net-list of a known design, and the signature-invariance guarantees that the reference net-list and all of its mutants have the same number of I/Os and the same number of nodes distributed across the same number of levels [33. This corresponds to the selection of the number of I/Os and the number of nodes distributed

across the levels as the factors in factorial design, and setting the factor values to the same values as in the reference circuit. The equivalence class of signature-invariant circuits is in fact a randomized problem (benchmark) set of statistical size generated in a specific way for a given combination of the factors' values defined by the reference circuit. This shows that their proposal is indeed a specific kind of the statistical factorial design. Moreover, Brglez and his collaborators stressed that: "Random graphs, a potentially unlimited source of benchmarks, have not been accepted as realistic circuit benchmarks. Mutants, as defined here, are not random net-lists".

Although their proposal was a step in a good direction, it did not ultimately solve the problems of benchmark set representativeness and of unbiased result processing for sets of signature invariant classes. In particular, it did not address the problems of high and uniform coverage of different possible combinations of the factor values by the reference circuits, necessary to guarantee a high benchmark set representativeness and unbiased result analysis and evaluation. Moreover, their results are limited to combinational circuits and they did not provide any hints that could suggest if or how their results could be used for sequential circuits. Our factorial design based benchmarking method and FSM benchmark generation tool presented in this paper resolve these problems.

In the area of heuristic methods and algorithms, the proposal to adapt the statistical design of experiments approach for the algorithm benchmarking and analysis was relatively well received. In the successive years this resulted in several papers which further researched and explained this approach or reported its use. Since this paper is devoted to benchmarking in digital circuit design automation, and we will not make any extensive discussion of the algorithm experimentation bibliography, but we would like to recommend several relatively general papers from the area of algorithm experimentation supplementing the material presented in this paper. McGeoch [36][37], Johnson [25] and Moret [38] give some general advice and tips related to the experimentation with algorithms and/or reporting the experimental results. Coffin at al. [11] discuss some statistical concepts relevant to analysis of tests related to algorithms and heuristics. General information on design of experiments in other areas can be found in [4][9].

Unfortunately, in the EDA field not many researchers and practitioners followed the proposal to adapt the statistical experimental design approach to the benchmarking. Some of the few follow-up developments were the following. In the late 1990s, we implemented our combinational circuit specification generator based on the factorial design approach, and used it for benchmarking related to our new information-driven approach to digital circuit synthesis [29][30]. Since the principles on which this combinational circuit specification generator is based constitute a subset of the principles used to develop the FSM benchmark generator reported in this paper, we will not further comment on it in this paper. S. Davidson and J. Harlow edited a Special Issue on Benchmarking for Design and Test of the IEEE Design and Test of Computers [13], in which amongst others, a short paper by Brglez was included that informally discusses the statistical design of experiments approach to CAD benchmarking [5]. Brglez and his collaborators used this approach later to some other EDA related problems, as e.g. design of experiments and evaluation of BDD ordering heuristics [18] and SAT algorithms [7][8] . In their research related to BDD ordering heuristics [18] they focused on logically equivalent isomorphism classes consisting of circuits that are logically identical and graph isomorphic instances of a reference circuit. To illustrate some major drawbacks of the traditional benchmarking approach in EDA that uses some relatively small collections of isolated, unrelated, not well characterized benchmarks, as e.g. the well known ISCAS or MCNC benchmark sets [17], they selected some specific isolated instances from the isomorphic classes and demonstrated that: "specific, isolated instances randomly selected from an isomorphic class cannot characterize the performance of an algorithm". They concluded that: "Without examining statistically meaningful samples of such populations, any reported results on the benchmark sets are subject to large, unknown random variability." This means that many differences in results and related "improvements" reported using the traditional benchmarking approach may be due to a chance related to benchmark selection and/or result processing, and not due to any actual differences in the methods, algorithms or tools tested. They also wrote: "… the principles of experimental design … are generally applicable over a wide range of CAD tasks, but the particulars of the experiments will vary, depending on the problem domain" and "… sequential circuit applications are different, and will require appropriate treatments … and experimental designs which will differ from those presented here".

Similarly to the works of Brglez and his collaborators, our research presented in this paper is devoted to benchmarking in digital circuit design

automation based on the design of experiments approach. However, our work in its general part is focused on the problems of a high benchmark set representativeness and unbiased result processing, and in its specific part on sequential circuit benchmarking. Our research and its results are thus different and complementary to those of Brglez and his collaborators.

4 Motivation of our research and development of an FSM benchmark generator for experimental design based benchmarking

Several recent papers reported that the available digital circuit synthesis tools are not effective for many important classes of circuits (see a. o. [12][30][32]). Moreover, due to the recent rapid progress in the microelectronic CMOS technologies, both the importance relationships among various circuit characteristics changed (e.g. the interconnects being of secondary importance in the past have now a dominating influence on the important circuit characteristics, the static power being negligible in the past technologies is becoming dominating in the modern nano-dimension CMOS technologies), and the multi-objective circuit optimization and tradeoff exploitation became much more important. Due to those changes and several other issues, the available circuit synthesis tools do not well address the needs of circuit synthesis for the modern nano CMOS technologies. Consequently, a new generation of more adequate circuit synthesis methods and tools is necessary.

Unfortunately, the proliferation and absorption of the statistical experimental design approach in the EDA field is weak. The industry and most of the researchers and developers in EDA field still typically apply the traditional benchmarking approach, using relatively small sets of isolated, unrelated, unsystematically collected and not characterized benchmarks, and inadequately processing the benchmarking results. In consequence, many quality issues and inefficiencies of the circuit synthesis methods and tools remain not revealed. This masks the actual problems and has a restraining effect on innovations. It makes an adequate evaluation of innovations impossible, and disables or delays their actual industrial take-up and exploitation. It also makes it difficult for the design method and tool users to decide which of the available alternative methods and tools best suit their particular needs. In consequence, it disables or delays progress. The critical importance of an adequate benchmarking for the progress in the EDA methods and tools, and in the digital system area, confronted with the weak proliferation and absorption in the EDA field of the more adequate statistical experimental design approach to benchmarking, motivated us to publish this paper.

In addition to the weak proliferation and absorption of the statistical experimental design approach in the EDA field, several other important factors motivated us to perform research in the experimental design based benchmarking and to develop and implement the FSM benchmark generator ***BenGen***, including the following:

- actual need to use this approach in our own research related to the analysis and evaluation of our new information-driven methodology to digital circuit synthesis,\
- inadequacy of the standard MCNC FSM benchmark set [17],
- problems with the actual industrial FSM benchmarks, and
- lack of any FSM benchmark generator,

According to our knowledge *BenGen* is the first and the only FSM benchmark generator on which any information has been published. It is described in one of the successive sections of this paper.

Similarly to the above discussed design of experiments methods in EDA of Brglez and his collaborators, our FSM benchmark generator is based on the principles of the statistical factorial design. However, the actual implementation of the statistical factorial design approach in our FSM benchmark generator very much differs from the methods of Brglez and his collaborators. First of all, the benchmarks from our generator represent the sequential circuit specifications and not the combinational circuit specifications. Differently than in their methods, the benchmarks from our FSM benchmark generator that belong to a particular benchmark equivalence class are not limited to only some mutants of a known practical design or logically identical circuits. In our method a benchmark class is defined by a particular combination of the factors' value subsets. Benchmarks of a certain class are basically generated by our generator through deciding the value subsets for each particular factor (representing e.g. a certain size level, function type or structure type), and subsequent random generation of a benchmark set of a statistical size for the selected combination of the factors' value subsets. This includes, but only as special cases, the possibility to generate benchmarks as mutants of a given practical FSM specification or sets of logically identical FSM specifications. The realistic character of the

generated FSM benchmarks is basically ensured not through a direct mutation of particular practical benchmarks, but through the covering of the function types, structure types, size levels and other characteristics of the benchmarks encountered in practice through the characteristics of the generated benchmarks. Our FSM benchmark generator enables an efficient, flexible generation of a different number of benchmark equivalence classes and different number of benchmarks in each class, dependent on the precision requirements of a particular experiment, as well as on the computation resource and experiment time limitations.

Our FSM benchmark generator also much differs from the sequential circuit generator by M. Hutton and his collaborators [23][24]. Their sequential circuit generator has been constructed through an extension of their earlier combinational circuit generator [22] that generates structural combinational net-lists with a specific size defined by the number of edges and I/Os, a specific shape, and given edge length, fan-out and re-convergence distributions. Their sequential circuit generator generates structural net-lists of binary sequential circuits, where a sequential circuit is defined by them as "a hierarchy of two or more combinational circuits connected by "flip-flop-edges (FF-edges) and "back-edges"". Consequently, their generator produces specifications of binary sequential circuit structures as are obtained after the sequential and combinational synthesis of FSMs. On the contrary, our FSM benchmark generator *BenGen* generates just the original symbolic FSM specifications, as constructed by human designers or behavioral synthesis tools, and being input to the successive sequential and combinational logic synthesis. Their binary sequential benchmarks are unable to directly serve several purposes to which our symbolic FSM benchmarks can serve, and are unable to accommodate several important features that our FSM benchmarks can accommodate. For instance, their binary benchmarks cannot directly be used as input to all the "symbolic" tasks of sequential synthesis, as e.g. FSM decomposition, state assignment, state minimization, etc., while our symbolic FSM benchmarks represent the natural input for these tasks. After binary state encoding, our FSM benchmarks can be directly used as an input to combinational synthesis, while the Hutton's binary circuit structures have to be first reverse-engineered before their corresponding next-state and output functions could be used as an input to combinational synthesis. Finally, the binary circuit structures resulting from sequential and combinational synthesis of our FSM benchmarks can directly be used for the same purposes as their binary net-lists of sequential circuits. While their sequential circuit benchmarks represent just particular binary circuit structures corresponding to completely specified binary next-state and output functions of sequential circuits with exclusively 2^k states (where k is a number of flip-flops), our FSM benchmark generator enables construction of completely, incompletely and weakly specified FSMs with any given number of states. These are of course only some of the major differences. More information on the specifics of our FSM benchmark generator can be found in one of the successive sections of this paper devoted to it.

5 Benchmark set quality

As discussed in previous sections, the value of the experiments with benchmarks and conclusions drawn heavily depend on:

- the quality of the benchmark set used;
- the adequacy of the experimentation process; and
- the adequacy of the experimental result analysis.

Since these are three main aspects of one whole being an adequate benchmarking, they are interrelated. In this section, we focus on the first problem, but in combination with the third one, i.e. on an adequate construction of a benchmark set to ensure high-value conclusions, and specifically meaningful averages and other results of statistical analysis.

To enable true and reliable conclusions, the ***benchmark set must be representative***. However, many different aspects and their complex interrelationships are hidden under this notion of being ***representative***, including the following:

- the benchmark set must include design specifications identical or similar to the typical specifications of practical designs for all substantially different functions and their various relevant characteristics, from all diverse application fields for which the use of the benchmarked method or tool is predicted;
- the values of all important characteristics of all benchmarks of the benchmark set must be known;
- the benchmark set should include characteristic design specifications for all various benchmark classes defined by all possible combinations of the benchmarks' important characteristics;
- each benchmark class defined by a certain value combination of the benchmarks' important characteristics should contain a statistically meaningful number of benchmarks, and should

be constructed using an unbiased random generation or selection of benchmarks;
- the distribution of the benchmarks from the benchmark set among the various benchmark classes defined by all possible combinations of the benchmarks' important characteristics must be (close to) uniform or if not (close to) uniform then known and taken into account by the statistical processing of the benchmarking results, to enable any useful and reliable conclusions from the statistical processing.

Satisfaction of the first three of the above conditions is necessary to ensure an adequate representativeness of the benchmark set to practice, so that it well accounts for the real-world designs, and consequently, the benchmarking conclusions will be relevant for practice. Satisfaction of the second condition and of the two final conditions is required to ensure an adequate structure of the benchmark set for the meaningful statistical processing of the experimental data. Assuming the usage of the statistical factorial design approach, the values of all important primary benchmark characteristics (factors) must be known. To offset unaccounted (secondary) factors and enable a meaningful statistical processing of benchmarking results within each benchmark class, the principles of test replication and randomization have to be used. To enable unbiased averaging over results from different classes, the distribution of the benchmarks among the various benchmark classes must be (close to) uniform or if not (close to) uniform then known and taken into account by the statistical processing.

In benchmarking experiments particular results for particular benchmark instances are directly measured and expressed through some measures related to selected effectiveness and efficiency aspects. In the field of electronic design and its automation, we are however interested not only in the fact how a particular method or tool behaved on a particular design, but we are interested as well in how it behaves in general or for a particular class of designs, or we are trying to predict how it will behave for some future designs. To answer questions of this kind, we often compute averages or other statistics.

However, averaging of particular results over a subset of benchmark instances or computing of any other statistics corresponds to computation of a certain function defined by the statistics. Since any function not being a one-to-one function involves abstraction, i.e. loss of information compared to information contained in its input data, averaging or computation of any other statistics over a nontrivial sub-set of benchmark instances always involves abstraction, i.e. loss of information. This means that an average and other statistics are from their nature unable to faithfully reflect the behaviour of a method or tool on individual benchmark instances. Each statistic only represents a particular abstraction from a combination of behaviours on individual instances. From this it should be clear that value of an average or other statistics very much depends on the selection of the individual instances over which the average or other statistics is computed. Thus, to obtain meaningful and useful information from a statistic, both the statistic itself has to be adequately selected to well reflect the actual aim of its use, and the benchmark sub-sets on which the statistic is computed have to be properly constructed.

In parallel to the representativeness of the benchmark set to practice, the ***proper benchmark sub-set construction*** includes such aspects as:
- ways of grouping benchmarks into classes,
- class granularity (generality/specifics),
- class size, and
- construction of particular classes and proportions between the classes to avoid bias towards specific groups of benchmarks.

These aspects decide, correspondingly, which sort of conclusions, as well as, how precise (general/specific), how statistically correct, and how unbiased conclusions can be drawn from the benchmarking experiments using a given benchmark set.

In consequence, it is not easy to construct a representative benchmark set that is strongly related to practice, actually addresses the issues of interest and enables true and reliable conclusions. Also, "representative" does not designate just a single point, but a degree. Different benchmark sets are representative to various degrees, and the quality and reliability of the conclusions obtained from the experimental research using the benchmarks increase with the degree.

We had an opportunity to work with several industrial circuit synthesis benchmark sets from 8 different system houses and EDA companies, involving from tenths to more than a thousand of circuits. Unfortunately, none of them was well representative. Most of them represented some ad hoc collections of non-characterized benchmarks with very irregular coverage of different circuit sub-classes. Also the standard circuit synthesis benchmarks commonly used by researchers are not representative enough. Moreover, there are several ***serious problems related to the use of practical industrial benchmarks***:
- the industry does not want to make their benchmarks accessible, motivating that these are

their or their clients' important designs, and thus they cannot be disclosed;
- having even a large set of the actual industrial circuit specifications it is difficult to see how large and divergent parts of the actual circuit space do they cover: are the circuits very different or almost the same; do they cover the circuit space regularly or are just cumulated in some particular regions; what are their characteristics?
- to be representative the industrial benchmarks have to involve very different functions typical to diverse application fields; they have thus to be collected from many industrial partners, but this results in a multitude of their formats, standards and/or representations.

It should be stressed that the use of a non-characterized or non-representative set of (industrial) benchmarks to research, compare or evaluate the synthesis methods or tools can very easily lead to completely wrong conclusions. In particular, even a large set of non-characterized benchmarks used may only include very similar or trivial benchmarks. Consequently, from the fact that a method or tool worked excellently for all of them one cannot conclude that it will work reasonably for another, but different and non-trivial, benchmark. In the last part of the paper, various cases of non-representative benchmark sets will be discussed and illustrated with experimental results.

To resolve the serious problems related to the use of practical industrial benchmarks in research, comparison and evaluation of the electronic design automation methods and tools, we proposed to use the representative sets of generated benchmarks. The term *representative benchmark set* is used here in the sense discussed above in this section, and covers both being representative to practice and adequate for the statistical benchmarking result processing. Of course, some mixed benchmark sets involving both some practical industrial benchmarks (if available) and generated benchmarks may also be used. In this case however, the available industrial benchmarks have first to be characterized, and knowing their characteristics, the remaining benchmarks have to be carefully generated to complete the (in most cases not representative) industrial benchmark sub-set into a representative benchmark set. Additionally, to satisfy the requirements of the unbiased random construction of the benchmark subsets representing particular benchmark classes and uniform benchmark distribution among the classes, some of the industrial benchmarks may have to be removed, if they are identical or very similar to some other benchmarks.

Benchmarks are particular instances of objects considered to be processed by a particular design method or tool. To characterize benchmarks or to generate benchmarks with particular characteristics, it is first necessary to find a set of all the relevant characteristic features of the objects considered to be processed, i.e. all such characteristic object attributes that changes in their values may substantially influence the processing effectiveness or efficiency of the methods or tools to be analysed. Each particular object (benchmark) can then be characterized by a particular vector of its attribute values. For instance, in the case of a circuit, the number of inputs, the number of outputs, the type of function to be implemented etc. may represent such characteristic attributes. Different ranges of the attribute values and their various possible combinations define the object sub-classes. To be representative, a benchmark set has to include a large enough unbiased selection of benchmark instances in each particular class, and uniformly cover all the sub-classes. If the benchmark distribution among the sub-classes is not (close to) uniform then it must be known and taken into account by the statistical processing of the benchmarking results to enable drawing of any useful and reliable conclusions from the statistical processing. It is of course possible to distinguish the ranges of the attribute values with lower or higher precision, which results in a smaller or larger number of the corresponding object sub-classes, and in a lower or higher benchmarking precision, correspondingly.

Since the number of different object sub-classes defined by the number of various attribute ranges combinations may be very high, and each of the sub-classes must be covered by a sufficient number of benchmarks, the representative sets of benchmarks may easily involve hundreds or thousands of benchmarks of various sizes and with various features. Moreover, to efficiently perform sensitivity analysis of EDA methods and tools to changes in the benchmark characteristics, it must be possible to quickly perform the corresponding changes in benchmarks. To perform the complex benchmark preparation efficiently, the benchmark generation process has to be automated.

To resolve the serious problems related to the benchmark set quality and use of practical industrial benchmarks in research, comparison and evaluation of circuit synthesis methods, we developed and implemented sequential and combinational circuit benchmark generators that make it possible for us to efficiently construct well-characterized circuit specifications with various characteristics, including

circuit specifications representative to various typical industrial application areas. These benchmark generators greatly reduce the necessity of having the actual industrial benchmarks, and enable research, comparison, evaluation and fine-tuning of circuit synthesis methods and tools largely independent of the industry, and much better than having only some industrial benchmarks. One of them is the Final State Machine (FSM) Benchmark Generator *BenGen*. In this paper, we use some results of our experimental research that involved large sets of diverse benchmarks generated with *BenGen* and a number of sequential circuit synthesis methods implemented in the contemporary EDA-tools to illustrate several crucial problems of benchmarking and demonstrate how to solve them.

6 FSM benchmark generator *BenGen*

Several factors explained in the previous sections motivated us to perform research in the experimental design based benchmarking and to develop and implement the FSM benchmark generator ***BenGen***.

FSMs are commonly used for specification and development of sequential circuits and control-dominated systems.

A ***sequential machine/finite state machine (FSM)*** $M = (I, S, O, \delta, \lambda)$ is an algebraic system,
with:
I - a finite nonempty set of inputs,
S - a finite nonempty set of internal states,

O - a finite set of outputs,
δ - the next-state function: $\delta: S \times I \rightarrow S$,
λ - the output function:
$\lambda: S \times I \rightarrow O$ (a Mealy machine), or
$\lambda: S \rightarrow O$ (a Moore machine).

An FSM can be represented in the form of a state transition table or state transition graph, where the transitions between states correspond to the computations defined by the next-state function δ.

Our FSM benchmark generator *BenGen* constructs the sequential circuit specifications originally in the form of state transition graphs. The state transition graphs are directly translated into the corresponding tabular form in KISS format used for the MCNC sequential benchmark set [17] and the corresponding Verilog representation. No restructuring is performed during this translation, and there is a one-to-one correspondence among these three FSM benchmark representations.

Benchmark set generation with *BenGen* is based on the principles of the statistical factorial design. The first author of this paper, which has more than thirty years of experience in digital circuit and system design, and EDA, analyzed several thousands of FSMs from various application areas and serving various purposes, like e.g. controllers, protocol machines, or sequential data-path circuits. He observed that all the analysed FSMs involve various combinations of several basic state-transition patterns, and their next-state and output functions can be quite well reconstructed using various combinations of several reasonably simple basic function generation rules. He generalized and parameterized the basic state-transition patterns, function generation rules, and several FSM characteristics (e.g. the number of inputs, outputs, states, transitions). In this way, he constructed a set of factors that represent several main FSM characteristics related to e.g. a function type, structure type, size characteristics, proportions, etc., so that a particular combination of their corresponding values almost completely (but in most cases not completely) defines a particular FSM. This means in fact that a particular combination of the factor values defines a quite narrow FSM class, in which all FSMs have the same controlled characteristics. The remaining FSM construction freedom can be exploited by random generation. Based on these principles the first author of this paper designed the FSM benchmark generator *BenGen* that for a particular combination of the controlled factor values constructs a corresponding class of FSMs through instantiating a corresponding combination of the FSM basic elements (i.e. the sets of inputs, states, outputs and transitions of the required dimensions), basic state-transition patterns (each involving a corresponding number of states and branches), basic next-state and output function generation rules, etc., and exploiting the remaining FSM construction freedom by random generation.

The benchmark instances form *BenGen* are thus not random, but have a clear structure and are only randomized within the framework of this structure. They represent some test cases that resemble the actual real-world cases, but are randomized within each specific (small) class. This way the FSM benchmarks can be generated that have the same or similar values of the controlled factors as the practical industrial FSMs, and consequently, quite closely resemble the practical FSMs. Moreover, the generated FSMs can be easily modified to even more closely mimic practical FSMs or more precisely research particular features.

BenGen was subsequently implemented by the authors of this paper and used by them to generate large benchmark sets exploited in numerous experiments related to the FSM design method and tool analysis, testing and benchmarking. Some of the benchmark sets from *BenGen* are also used by other research groups in several countries.

From the above it can be concluded that building a satisfactory benchmark generator that enables a high representativeness of the generated benchmarks is a difficult task that involves an extensive analysis of a large number of divergent industrial benchmarks by an experienced designer, generalization of the analysis results, as well as an actual development of a generator based on the generalized and parameterized basic patterns, generation rules and structural characteristics. Such a generator however can then serve many researchers and developers, as well as various experimental analysis tasks.

BenGen allows us to efficiently construct FSMs with various characteristics. This includes FSMs having:

- different number of states and various transition patterns between the states (e.g. chains of states with forward and/or backward transitions, loops, conditional "case" structures etc. and their combinations);
- different numbers of inputs and outputs, and different proportions between the next-state and output logic (state-dominated, balanced or output-dominated), as well as between the primary-input and state-input (input-dominated, balanced, state-dominated), and their mixtures;
- various dependences of particular transitions and output variables on the number of inputs and input conditions;
- completely, incompletely and weakly specified next-state and/or output functions.

The possibility to generate FSMs covers the FSMs representative to various typical industrial application areas, for instance having typical structure of controllers or protocol machines from various application areas, or representing various sequential data-path circuits. *BenGen* gives us an efficient FSM construction, but also modification of the constructed or industrial FSMs, and very precise "fine-tuning". This last feature is very useful in the sensitivity analysis of EDA methods and tools to the changes in their input data, i.e. changes in the FSM characteristics.

FSM benchmarks from BenGen enable us to do the following:

- analysis of a wide spectrum of design cases and problems related to the FSM architecture and logic synthesis, in the space spanned by various FSM functions, structures and different implementation options, and in the light of various optimization constraints and objectives;
- thorough and very precise testing of EDA methods and tools;
- extensive research and precise characterization of EDA methods and tools regarding their performance and robustness in relation to FSMs with various known characteristics;
- very precise sensitivity analysis of EDA methods and tools to changes in the FSM characteristics;
- reliable comparison and quality evaluation of different EDA methods and tools in relation to various FSM classes

From the above it can be concluded that *BenGen* allows us to effectively and efficiently perform many benchmarking tasks that would be very difficult to be well performed without a benchmark generator or having only a set of practical industrial benchmarks.

Although we took a lot of care to ensure that the generated benchmarks have characteristics very similar to practical industrial benchmarks from many application areas, we are unable to guarantee that the generated benchmark sets cover each specific FSM type. Therefore, the industrial benchmarks can still be useful to ensure that the FSM types represented by them are actually covered by the generated benchmarks, and to perform some final confirmation checks. Another possibility is a mixed use of the generated and practical industrial benchmarks. As described in the previous section, a particular benchmark class is not required to contain only generated benchmarks, but some available and characterized industrial benchmarks can be included into the class instead of the corresponding generated benchmarks. The generated benchmarks are thus not expected to completely eliminate the need of using of some industrial benchmarks. It is however also not expected that practical industrial benchmarks can eliminate the need of using generated benchmarks. Even if one would be able to form a large collection of divergent industrial benchmarks (which is a difficult task) then the probability is very low that the benchmarks would to a sufficiently high degree and uniformly cover all the different circuit classes. Consequently, after the characterisation and selection of a subset of suitable industrial benchmarks, adequate generated benchmarks would be required to supplement the industrial benchmarks. Moreover, the generated benchmarks would be extremely difficult to replace in the tasks of thorough and precise testing, research,

characterization and sensitivity analysis of EDA methods and tools.

BenGen has ***two work modes***: batch and interactive mode. In the ***batch mode***, the parameters of the FSM to generate are supplied in a script file. These parameters include:

- the number of the FSM's branches;
- the branch characteristics, such as the branch start and end state, the number of states in the branch, the percentage of backward transitions and the percentage of loops in the branch;
- the number of inputs and outputs of the FSM;
- the number of inputs active for transitions from a given state or in a given branch;
- the output type (Mealy or Moore);
- the number of outputs active in a given state, for transitions from a given state or in a given branch;
- the percentage of "don't cares" in the next-state and/or output function;
- the percentage of "0s" to "1s" in the outputs.

Most of these parameters can be specified both as concrete values and in the form of a probability distribution to randomize their values in the specific instances of the generated FSMs. As a result, in the batch mode BenGen can be used to easily generate large sets of FSM benchmarks with similar characteristics using the same script file. The script file can be easily modified to generate a next batch of different FSMs. The ***interactive mode*** of *BenGen* provides more control over the generation process. The user can interactively enter any of the parameters available in the batch mode. Additionally, operations allowing modifications of single branches and transitions are provided. The interactive mode *BenGen* is especially useful for fine-tuning of the generated or industrial FSMs to possess some very specific characteristics, required for instance to analyze the behaviour or sensitivity of a method or tool in relation to a certain aspect.

In both modes, *BenGen* organizes the benchmark construction process and takes away the burden of tedious specification of single transitions, or checking the consistency of the constructed FSMs. Instead, the user can focus on specifying the required high-level FSM's characteristics. In this specification process, the user is guided by *BenGen*, being asked several questions, whose answers combined by *BenGen* result in the required high-level FSM specification. Having the specification of the required controlled FSM's characteristics *BenGen* generates the corresponding state chains, with the requested number of states, and appropriate backward transitions between and self-loops in the states, if needed. Given the state-transition behaviour defined in abstract terms, *BenGen* generates the input conditions for particular transitions, etc. In this process not only does it consider user's requirements concerning the active inputs for transitions from a given state or for a given branch, but also ascertains that the machine is consistent, i.e. distinct transitions have disjoint input conditions and all possible input conditions are specified (for completely specified FSMs). The generator also determines the output values, taking into account the outputs active for a given state, for transitions from a given state or for a given branch, as well as the percentage of "don't cares" and percentage of "1s" to "0s". With the above characteristics, BenGen allows a very efficient generation of large sets of various sorts of well-characterized FSM benchmarks, with a minimized effort of the user. On the other hand, it also enables fine-grained control over the generation process and editing of the generated or industrial FSMs. The generation time on a PC is negligible for small FSMs and is in the order of tenths of minutes for complex FSMs with tenths of inputs and outputs and thousands of transitions. This is however a nonrecurring cost, related moreover to the time of a computer and not of a human designer.

Currently, *BenGen* is extensively used in our research related to the analysis of design problems and use of various EDA methods and tools for the multi-objective optimal circuit synthesis targeted to modern FPGAs and re-configurable SoC platforms. In particular, we used large sets of diverse benchmarks generated with *BenGen* to perform an analysis of problems related to the quality of the contemporary commercial and academic methods and tools for the FPGA-targeted FSM state assignment. A part of the experimental results from this research is used in the following sections of this paper to discuss and illustrate several crucial problems of benchmarking and demonstrate how to resolve them. More information on *BenGen* can be found in [32].

7 Sequential circuit synthesis

With the contemporary synthesis flows for FPGAs, the sequential circuit synthesis process from the RTL-level FSM specification to its actual circuit implementation is very simple, and consists of the two following main sub-processes:

- assignment of a binary representation for the FSM's symbolic internal states, and
- synthesis of the FSM's combinational binary-logic component that results from the state assignment.

As shown it in the successive sections of this paper, the FSM state assignment methods used in the contemporary commercial FPGA-targeted circuit synthesis tools are very simplistic what is a reason of inferior results. In the sequel, we will use the benchmarking of the FSM state assignment techniques to illustrate and discuss several major benchmarking issues and their solutions, but the quality of the assignment results will be estimated by performing combinational component synthesis and getting the corresponding area, delay and power related data after the circuit placement and routing (P&R).

Hardware implementation of an FSM requires two sorts of components: combinational logic (implementing the binary representation of the next-state function δ and output function λ), and binary memory elements (implementing the state memory). As opposed to inputs and outputs of an FSM, which represent some external-world signals that are usually binary already in the initial FSM specification (i.e. pre-assigned), the internal states are initially in a symbolic form in most cases. To construct a binary logic circuit implementing an FSM having symbolic states, every symbolic state has to be implemented with a corresponding combination of values of the binary memory elements (e.g. flip-flops). The choice of binary codes for the FSM's symbolic states finally decides the binary-level next-state function, output function and state memory of the FSM circuit implementation, and in consequence greatly affects all the main characteristics (e.g. area, speed, power dissipation) of the circuit implementation. An optimal **FSM state assignment** consists in choosing an appropriate binary representation for the FSM's symbolic internal states, so that the resulting binary logic is optimal for specific objectives. Finding an optimal or close to optimal assignment often creates a significant advantage, but is computationally complex (NP-hard) [16][26][27]. In a strict sense, it has never been solved, except for exhaustive search that is impractical or impossible for larger machines. Therefore, approximate heuristic assignment approaches have to be used that are able to find high quality assignments using reasonable computation resources. One of the basic requirements for the use of heuristic methods in practical applications of circuit synthesis is their ***robustness***, in the sense of *steadily delivering high-quality solutions in the whole range of possible input data*.

The ***heuristic state assignment approaches*** can be subdivided into three main categories:

- ***structural (constructive) approaches***, that construct (near-)optimal assignments using some knowledge on the internal structure of an FSM (the representatives of this category are SECODE [31][39], Jedi[34] and MAXAD[27]);
- ***statistical (generative) approaches*** that generate and check the assignments, but do not use any information about the internal structure of a sequential machine (e.g. genetic algorithms [1][10] and the best-random approach that consists in choosing the best from n randomly generated assignments); and
- ***pragmatic approach***, commonly used in today's commercial tools for FPGAs, that consists in applying an assignment that is "known" as possessing some "nice" properties; most often used encoding methods in this category are: one-hot, Gray and natural binary encoding.

8 Benchmarking result evaluation

The quality of benchmarking results from a particular method or tool can be analyzed, characterized, evaluated or compared in relation to the results from some other methods or tools for the same aim, or in relation to the optimal results. In sections 3 and 4 we explained how to construct a benchmark set that will ensure the adequacy of the statistical experimental design analysis, and particularly, correct and meaningful averages and other statistical parameters. Such a benchmark set enables correct and meaningful comparison of methods or tools to some other methods or tools. Let us now focus on benchmark result comparison to the optimal results.

Due to the computational complexity of many electronic design automation problems, the strictly optimal solutions remain usually unknown. In particular this is the case of the state assignment for the most practical FSMs. Consequently, there are no reference points to which the results from a particular method or tool could be compared. Even if the result of one method or tool is better than from another, it is still unknown how bad (or good) both these results are. To resolve this problem, the first author of this paper proposed to compare the FSM state assignment result obtained for a particular benchmark from a particular method or tool to the assignment result distribution for this benchmark obtained by random assignment generation [26]. This solved in a way the problem of the benchmarking result analysis and evaluation. An extensive analysis of the state assignment solution space was presented by us in [26] and [27], and therefore only the most important considerations, definitions and conclusions are recalled here.

In [26] and [27], the first author of this paper demonstrated that the distribution of solutions for state assignment can be well estimated by a normal (Gaussian) distribution. For non-trivial FSMs, and especially for large ones, many factors influence the state assignment quality, but none of them is a deciding factor in separation. In this situation, that is typical to all natural or technical processes that show a normal distribution, there are many more constellations of factors that lead to medium value assignments than constellations that lead to very good or very bad assignments (arrangements of only the best or only the worst factors, correspondingly). Of course, for larger machines, on average, more factors decide an assignment's quality and there are more possible constellations of these factors, while each of them in isolation can decide less about the quality of the assignment. Therefore, the estimation of the distribution of solutions by the normal distribution is on average better for larger machines than for smaller ones. Very small machines may show substantial deviations from the normal distribution. Numerous examples of typical empirical distributions are given in [26][27]. It is important to notice that the benchmarking result analysis technique presented in this section is not limited to the FSM state assignment problem and methods of its solution. It can be used to any problem having a solution space that can be well modelled with a normal distribution, and to solution methods of such problems.

A ***normal distribution*** *N (m, σ)* is defined as a distribution with the probability density function:

$$f(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) * e^{-(x-m)^2/(2\sigma^2)}$$

where: m - expected value; σ - mean square deviation.

The cumulative distribution function F(x)= P(t ≤ x) for a normal distribution is given by:

$$F(x) = \begin{cases} 0{,}5 + \Phi\left(\frac{x-m}{\sigma}\right) \text{ for } x > m \\ 0{,}5 - \Phi\left(\frac{x-m}{\sigma}\right) \text{ for } x \le m \end{cases}$$

where: $\Phi(x)$ is an integral from the probability density function of the normalized Gaussian distribution *N(0,1)* in the integration range *<0,x>*. The values of $\Phi(x)$ are provided in tabular form in mathematical guides (also on the Internet).

We must however remember that the random variable *X* describing the quality of a solution is a discrete variable. So, we must calculate $P(X=X_i)$ as an integral from the probability density function $f(x)$ in the integration range $<X_i-0.5, X_i+0.5>$:

$$P(X = X_i) = \int_{X_i-0,5}^{X_i+0,5} f(x)dx =$$

$$= F(X_i + 0{,}5) - F(X_i - 0{,}5)$$

and $P(X \le X_i) = F(X+0{,}5)$

To calculate the probability of a number of successes ***k*** in ***n*** random choices, the ***binominal distribution*** can be used:

$$P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k} \text{ and}$$

$$P(S_n > 0) = 1 - P\binom{n}{k} p(S_n = 0) = 1 - (1-p)^n$$

where: p – the probability of success for one random choice.

For large instances of computationally complex problems (e.g. state assignment for large FSMs) a strictly optimal solution is relatively unknown. Consequently, it is virtually impossible to define the terms "good" or "suboptimal" in relation to the optimal solutions. Therefore, we proposed some probabilistic quality measures both for particular solutions and for solution methods that are based on the estimation of the solution distribution in the solution space with a normal distribution. With these measures, we compare the quality of a result for a particular benchmark (for instance of an assignment obtained for a particular FSM from a particular method) to the normal distribution of results (assignments) for this benchmark. In contrast to knowing or finding of a strictly optimal solution for a particular benchmark, one can always easily find the normal distribution of solutions for the benchmark. It can be simply realized through random generation of a statistically relevant number of solutions (e.g. 20) for a given benchmark, evaluation of their quality, and finding the parameters m and σ of the corresponding normal distribution from the experimental results or just drawing the graphical representation of the distribution from the experiment. The quality of a solution for a particular benchmark, and consequently, the quality of a particular method that constructed the solution, can be compared this way to the quality of solutions that can be obtained for the benchmark from random generation. The ideas behind this comparison are the following. If the

quality of a given solution is worse than the quality that can be on average obtained through the random solution generation (i.e. with the high probability of *0.5*) than this solution is bad, and consequently, the method which in most cases constructs such solutions is also bad (i.e. worse than the random generation). If, in contrast, the method consistently construct the solutions that are very difficult to be generated on random (i.e. the probability of their random generation is very low) than the method is actually a very good method. An extension, formal formulation and further discussion of these ideas are presented below.

In the digital circuit design field the result quality is often measured through such attributes as the circuit area, delay or power consumption, and the result is better in relation to a particular attribute if the value of the attribute is lower. Therefore, in the definitions and discussions below we assumed that a lower attribute value means higher quality. Of course, in the opposite case all the definitions and discussions remains valid – only the inequality sign by the result quality comparison has to change its direction.

Definition 1: A *solution is good* if, and only if, its quality X_G is no worse than the average quality: $X_G \leq m$ *(i.e.* $P(X \leq X_G) \leq 0.5$*).*

In the opposite case (i.e. if X is worse than the average quality: $X > m$), a *solution is bad*. Thus, *good* means just *no worse than the average one that can be obtained through random generation*, or *good* means just *not bad*. When X_G is close to m then the corresponding assignment, although still good, is not much better than the best of the bad assignments. Due to the normal distribution of solutions and the characteristic features of its probability density function that has its maximum for $X= m$ and is symmetrically decreasing in both directions with distance from m, the probability of randomly generating a *good* solution is high (*0.5*) and equal to the probability of randomly generating a *bad* solution. Thus, it is equally easy to randomly generate good and bad solutions, but the randomly generated solutions will be concentrated around the solution of the average quality *m*. Observe that it is difficult to randomly generate *good* or *bad* solutions whose quality is far from *m*. The probability of their random generation is very low, and decreases with the distance from *m*. However, our aim is to find some "as good as possible" solutions, and thus, some good solutions whose quality is as far as possible from *m*. Our aim is to find the solutions that can be considered as near-optimal or even of uniquely good quality, whose random generation probability is very or extremely low, correspondingly. The above observations motivated the following definitions.

Definition 2: A *solution is near-optimal* if, and only if, the probability of randomly generating a solution with a quality X no worse than X_N of the near-optimal solution is not higher than *0.05* ($P(X \leq X_N) \leq 0.05$).

Definition 3: A *solution is unique* if, and only if, the probability of randomly generating a solution with a quality X no worse than X_U of the unique solution, is not higher than *0.005* ($P(X \leq X_U) \leq 0.005$).

To consistently find such good solutions that their quality is far from the average quality *m*, and consequently, their probability by random generation is very low, and particularly to consistently find the *near-optimal* or *unique* solutions, we cannot rely on random generation, but we must have a systematic method. The method will be of higher quality, if it will be able to consistently find better solutions, and thus less probable solutions.

Definition 4: A *method is good (near-optimal, unique)* if, and only if, it computes good (near-optimal, unique) solutions in most cases.

Since in the practical circuit and system design situations our aim is to find high-quality solutions that are (close to) the best possible solutions, only the unique or near-optimal methods are of practical value. For instance, the trivial random generation "method" is on the border between good and bad, is unable to find near-optimal or unique solutions in most cases, and therefore it is difficult to qualify it as an actual method. Also, it is difficult to qualify any solution finding activity as an actual method of a practical value, if it is only able to find good solutions, but is unable to find near-optimal or unique solutions in most cases, i.e. works in a comparable fashion to the random generation.

It is important to notice that the probabilistic quality measures defined above are not limited to the FSM state assignment problem and methods of its solution. They can be used to any problem having a solution space that can be well modeled with a normal distribution, and to solution methods of such problems.

Observe that due to the normal distribution of solutions, the solutions generated by the generative statistical assignments methods will be concentrated

around the average quality assignments. For instance, it follows directly from the normal distribution that with ten random choices, the probability of obtaining a solution that is no worse than the average, is *0,999*, but the probability of obtaining a near-optimal solution X_N is *0,4012* (less than *0,5*!) and the probability of obtaining a unique solution X_U is only *0.0489*. Thus, better than average assignments can be generated with high probability by statistical methods, but the probability of generating very good assignments is low, while generating unique assignments is very unlikely. In contrast, structural methods that construct assignments using relevant information on a given FSM, are able to construct near-optimal or even unique assignments with good probability [26][27][32][34][39]. Moreover, to find a very good solution, the generative algorithms usually need to evaluate tens or hundreds of solution generations, each composed of tens of solutions, through their time-consuming combinational synthesis. In consequence, the generative algorithms are orders of magnitude slower than the structural methods in delivering of comparable quality solutions. Therefore, they will not be considered further in this paper.

9 Experimental research

In our experiments reported in this paper, more than 500 FSMs were used that exhibit various characteristics typical to FSMs encountered in different real-life applications. As more precisely described in Section 6, we identified a number of typical basic schemes of sequential behaviour present in a large number of real-life FSMs from various application areas, and generalized and parameterized the basic schemes and some other FSM characteristics. This way a set of factors has been constructed that represent several main FSM features. A particular combination of the factor values defines a quite narrow FSM class, in which all FSMs have the same controlled characteristics, but may differ on the remaining FSM construction freedom that is exploited by random generation. The corresponding FSM benchmark generation process implemented in *BenGen* has been used to generate benchmarks for the experiments. Since the number of different possible combinations of the controlled factor values is unlimited, because some of the factors (e.g. the number of states, inputs or outputs) are theoretically unlimited, we had to limit the benchmark generation process to the values that most often occur in practice. For example, in the experiments reported here, we did not consider FSMs having more than 200 states, because the FSMs specified by human designers or constructed by the behavioral synthesis tools have typically not so many states. Larger controllers and other sequential sub-systems are typically specified as a composition of smaller collaborating FSMs which are separately processed by EDA-tools. Observe please, that we are dealing here with the initial specifications of particular FSMs, and not with the final synthesized circuit net-lists of large systems after flattening of their originally modular and hierarchical representations. Even after limiting the factor values to those that most often occur in practice, the number of different possible combinations of the controlled factor values was too large to efficiently process a statistical number of FSM benchmarks for each such combination with all the involved circuit synthesis tools. To perform our experiments reasonably efficiently, when at the same time ensuring their acceptable quality, we first performed a coarse sensitivity analysis of the FSM synthesis results to changes in particular factors. The factors to which the results were less sensitive could be quantized more coarsely. This enabled us to replace the benchmark generation for all different possible combinations of the controlled factor values with the benchmark generation for a large number of different particular combinations of the controlled factor values, corresponding to a large number of the narrow FSM classes. For each such narrow FSM class, we initially generated on random more than 40 benchmarks (and totally more than 1000 benchmarks), what was a satisfactory statistical number of benchmarks to have a very high confidence level for the statistical tests. However, it turned out that the experiments will still require a prohibitively long time taking into account the efficiency of the circuit synthesis tools used for the experiments. Therefore, we finally used in the experiments approximately 20 benchmarks for each narrow FSM class (and totally somewhat more than 500 benchmarks), what is actually close to the lowest value of the statistical number of benchmarks that still ensures an acceptable confidence level, but enabled us to perform the experiments in a reasonable time.

In the experiments, we researched several major benchmarking issues and effectiveness of several most representative industrial and academic methods for the FPGA-targeted sequential circuit synthesis. The ***experiments involved***: ***1-hot, Gray*** and ***natural binary encoding***, that represent the ***pragmatic industrial assignment approaches*** prevalent in the contemporary commercial tools for FPGAs, and our

own FPGA-targeted FSM state assignment tool SECODE, in comparison to Jedi. ***JEDI*** and ***SECODE*** belong to the most advanced academic tools representing the ***constructive structural assignment approaches*** [29][34][39].

We performed the combinational synthesis of the encoded FSMs, as well as the placement and routing (P&R) of the synthesized circuits in the actual FPGA device performed with the tool of the FPGA vendor, and analyzed and compared the FSM encoding results after the synthesis and P&R. We analyzed and compared the results from each particular encoding method both in relation to the results from the other encoding methods (using JEDI as a reference method) and to the normal distributions of the encoding results for the particular benchmark FSMs. The main focus of this paper is on the benchmark set quality issues, and we can clearly demonstrate several of these issues when presenting the results from particular encoding methods in comparison to the results from JEDI. Moreover, the issues of analysis and comparison of the encoding results in relation to the normal distributions of the results for the particular benchmarks have been discussed in our previous papers [26][27][32]. Therefore, in this paper we included only some limited information on this useful approach mainly to enhance its proliferation, and to create a broader context and better understanding of the benchmarking related to the state assignment problem and similar complex problems in digital circuit design.

10 Experimental results

To compute the relative quality of a particular method on a particular benchmark in relation to JEDI, the values of each considered quality metric (area, delay or power-dissipation) were computed for each encoding by the commercial tool used for the experiments. Subsequently, for each benchmark, the percentage of difference in the value of each particular quality metrics for a particular encoding method *M* and JEDI (*J*) in relation to JEDI was computed according to the equation:

$$QD(QM, M) = ((QM(M) - QM(J)) / QM_{min}(M, J)) * 100\%$$

where:

QD(QM, M) – the relative quality difference in % between a particular encoding method *M* and JEDI on a particular quality metrics *QM*, *QM*: area, delay or power-dissipation;

QM(M) – the quality metrics value for the circuit resulting for a particular benchmark from a particular encoding method *M*, *M*: JEDI, SECODE, 1-hot, Gray and natural binary encoding;

QM(J) – the quality metrics value for the circuit resulting for a particular benchmark from JEDI encoding;

$QM_{min}(M, J) = min\{QM(M), QM(J)\}$ – the lower of the two *QM* values for the two methods *M* and *J*, used instead of *QM(J)* or *QM(M)* to avoid bias to any of the two compared methods *M* or *J*.

We only included the results related to the circuit area in this short paper, because they are fully sufficient to illustrate several crucial problems of benchmarking. Moreover, instead of including the very large tables with complete raw results for all benchmarks, we show the results of the statistical processing of the raw results, i.e. the relative average area results from different methods in comparison to JEDI for various FSM classes, and distribution of the relative results from each method in relation to the average area for the method. We categorized the FSMs according to following three criteria: the size, the proportion of the number of primary input bits to the number of state bits, and the proportion the number of primary output bits to the number of state bits. The ***size*** criterion divides FSMs into ***small*** (max. 8 states), ***medium*** (9 to 32 states) and ***large*** (more than 32 states). The ***proportion of the number of primary input/output bits to state bits*** categorizes the FSMs as ***input/output dominated*** if the number of input/output bits is 50% larger than the number of state bits, ***state dominated*** if the number of state bits is larger than the number of input/output bits and ***balanced*** otherwise.

An important data not shown in the figures below is that our tool SECODE consistently produced results that were on average somewhat more than 20% better than the results from JEDI for all circuit classes (for the corresponding result tables and their discussion see [31][39]). The results from SECODE are not included in the figures below, because this will not introduce any extra information (it will result in a point on the level of somewhat lower than –20% in each of the figures), but will substantially increase the area of the figures.

In Figure 1, the relative results for area are presented that were produced by each particular encoding method: 1-hot (**H**), Gray (**G**) and natural binary encoding (**B**) for each particular class of benchmarks distinguished according to: size (**Size**), proportion of the number of primary input bits to state bits (**Input**) and proportion of the number of primary output bits to state bits (**Output**). In Figure 2, the distribution of the relative area results from the particular FSM encoding methods is shown.

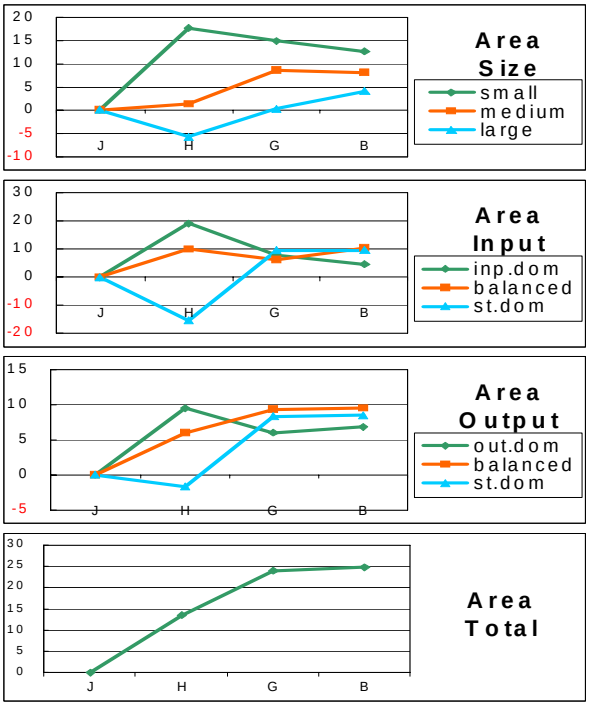

Fig. 1 Relative area results from the particular FSM encoding methods

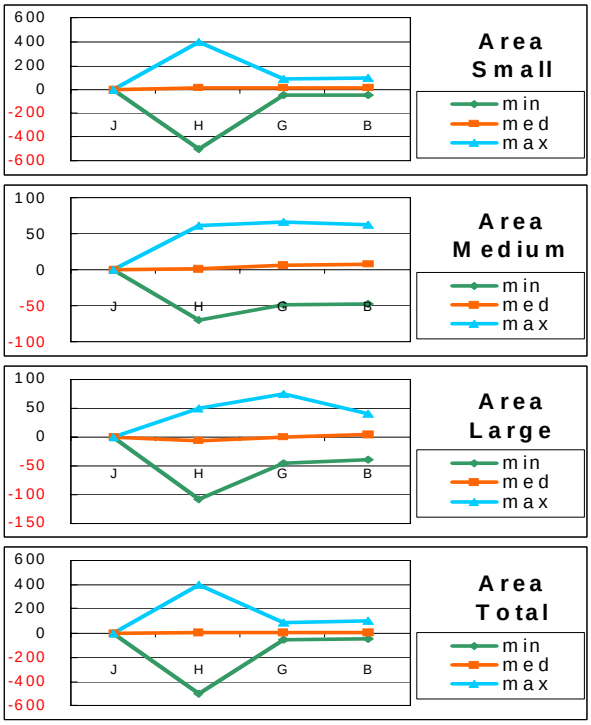

Fig. 2 Distribution of the relative area results from the particular FSM encoding methods

Many interesting conclusions can be drawn from the results of the experiments performed. It follows among others that:

- the constructive FSM encoding methods represented in the experiments by JEDI and SECODE produce the best results on average;
- the natural binary and Grey encodings are the worst encodings of all the considered;
- 1-hot is able to produce very good results for the medium and large state-dominated FSMs; however, it is not robust (in Fig. 2 the difference between max and min is largest for 1-hot) - it often produces very bad results.

Moreover, it turned out that only the constructive structural state assignment approaches, represented by SECODE and JEDI are unique and robust, with JEDI being close to the border between near-optimal and unique. In general, the pragmatic FSM state assignment approaches commonly applied in today's commercial circuit synthesis tools for FPGAs, represented by 1-hot, Gray and binary encodings in the experiments, are only good or near-optimal, and not robust in general. However, 1-hot encoding is unique and robust for the medium and large state-dominated FSMs. The pragmatic approaches are thus effective in only some special cases.

The experimental results clearly show a big room for improvement regarding the state assignment methods applied in the contemporary commercial EDA tools targeted to FPGA synthesis, and in this way demonstrate that an adequately performed benchmarking is able to reveal important quality issues. In the contemporary commercial tools, only the medium and large state-dominated FSMs are quite well covered by the 1-hot encoding, but there is a large room for improvement regarding all the other FSM classes for which all the three pragmatic approaches (1-hot, Grey and natural binary encoding) produce inferior circuits comparing to the structural approaches represented by JEDI and SECODE. In consequence, the most effective published approach of the FSM state encoding for the FPGA-implemented systems seems to be our new information-driven encoding method implemented in SECODE [31][39], supplemented with 1-hot encoding applied to some state-dominated FSMs.

Now let us focus back on the benchmarking issues and their demonstration through the experimental results from our research. Please observe that if only one statistic would be used on the whole benchmark set, e.g. the average relative area (see Fig. 1, Area Total), then the conclusions from the experiments, although obtained for the representative benchmark set and correct, would be very limited. One could only conclude that one-hot encoding is on average approximately 13% worse than JEDI and 12% better than Grey and natural binary encoding. Having additional information on the distribution of the relative area results (the raw

distribution from the experiment as in Fig. 2 or a statistically processed one in the form of a standard deviation), one can immediately conclude much more, namely: although on average 1-hot is only approximately 13% worse than JEDI, for particular benchmark instances it can produce as much as 400% worse results than JEDI, but also as much as 500% better results. Thus, 1-hot is not robust, can produce both very bad and very good results, but it is able to produce excellent results in some cases.

Without considering some smaller benchmark classes, as we did in the presented experiments, and computing the statistics over the smaller classes, one could however never know for which kinds of FSMs 1-hot actually produces these excellent results, and for which produces very bad results. Considering narrower classes, defined by more factor variables and narrower sub-ranges for the variables, gives more precise information. Including more benchmarks in the benchmark set for each such class enhances the confidence level of the statistical conclusions. On the other hand however, both increase the total number of benchmarks to process, and with a certain large number of narrow classes and large number of benchmarks in each class, the total number of benchmarks can become too large to be processed in the required practical time. Thus, there is a trade-off between the benchmarking time (total number of benchmarks used), the amount and preciseness of information obtained from the benchmarking experiment, and the confidence level of the statistical conclusions.

Often one would like to have an as small as possible benchmark set that still enables to take the conclusions of interest, within a limited error margin and with high enough confidence level of the statistical result processing. Obtaining of such a reduced representative benchmark set is possible, and is illustrated below. The above results in Figure 1 are obtained for the complete representative benchmark set containing somewhat more than 500 FSMs. Having this large set, we constructed a four times smaller reduced representative benchmark set consisting of about 160 FSMs, for which the area results are presented in Fig. 3. The results on the reduced set are almost identical to the results on the original large set (the difference is within 2%). Obtaining such a well-representative, but reduced, benchmark set was possible through a careful selection of the most characteristic benchmarks from each class of the original large representative set (e.g. the benchmarks that result in the best and the worst results in a class or have certain class specific features that strongly influence the result quality). This was virtually impossible without first

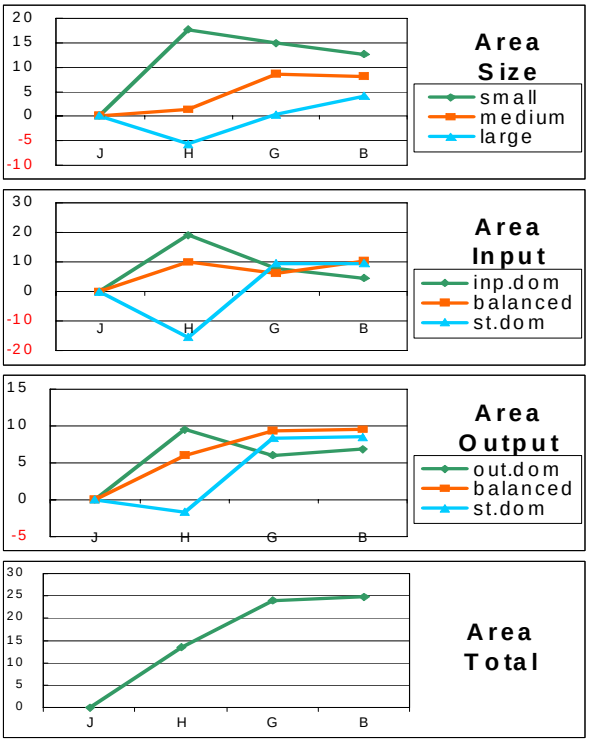

Fig. 3 Relative area results for the reduced representative benchmark set

constructing the original large representative benchmark set. Also, having a large representative benchmark set, an automatic construction of a smaller representative set is possible through controlled removal of the least characteristic benchmarks driven by the objective of change minimization in the relevant statistics for each class of the reduced set in comparison to the original large set.

Now let us observe what happens when the benchmark selection does not satisfy one or more conditions of the representative benchmark set. In Fig. 4, we can observe a result of a non-uniform benchmark distribution among the classes. In this set, there are less state-dominated benchmarks on the input side with a simple transition structure between the states compared to the number of benchmarks in the remaining benchmark classes.

Observe that the results and conclusions are now different than for the representative benchmark set. In particular, the relative quality of 1-hot encoding computed for the benchmark set with the non-uniform benchmark distribution among the classes is much lower (the relative area is much higher) than for the representative benchmark set. In Fig. 5, the results are presented for a benchmark set in which some benchmarks are repeated twice (namely some state-dominated benchmarks on the input side with a simple transition structure between the states). The same effect can be obtained when having many

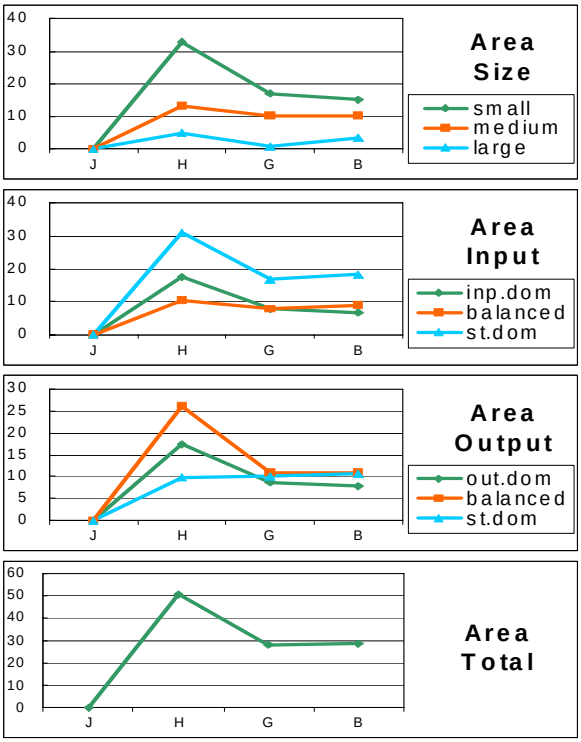

Fig. 4 Relative area results for the non-uniform benchmark distribution

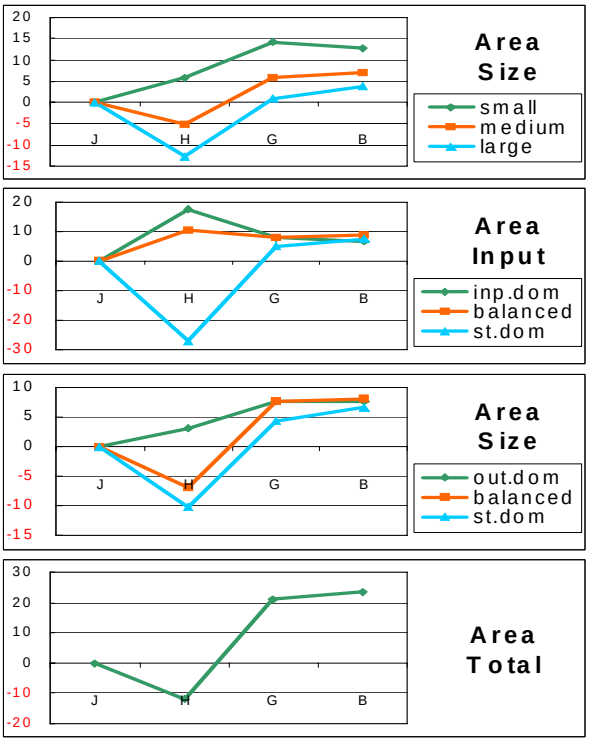

Fig. 5 Relative area for the case of repeated benchmarks

similar benchmarks in the benchmark set. Also in this case the results and conclusions are different and incorrect. In particular, the relative quality of 1-hot encoding computed for the benchmark set with some repeated benchmarks is now much higher (the relative area is much lower) than for the representative benchmark set. This demonstrates that both a biased selection of benchmarks within a class and non-uniform distribution of benchmarks among the classes introduce substantial changes in the benchmarking results being the input data for the statistical processing, and in consequence result in faulty conclusions.

11 Conclusion

This paper reported several results and conclusions from our research of benchmarking issues in the digital circuit design automation. In particular, we briefly discussed the recent research effort related to benchmarking in the EDA field and related field of heuristic methods and algorithms, considered the importance and difficulties of benchmarking, analysed and illustrated with examples the problems related to the construction and use of a representative benchmark set, explained how to adopt the statistical factorial design methodology to resolve some major benchmarking problems in electronic design and EDA, and discussed the FSM benchmark generator developed and implemented by us to resolve serious problems related to the usage of practical industrial benchmarks and so called “standard benchmark sets” commonly used by researchers.

Our FSM benchmark generator is based on the principles of the statistical factorial design also used in the EDA-related design of experiments methods of Brglez at al. However, its aim is different and the actual implementation of the statistical factorial design approach in *BenGen* very much differs from the methods of Brglez at al. Our FSM generator also very much differs from the generator of sequential circuit structures proposed by M. Hutton at al, and has several advantages comparing to Hutton’s generator. *BenGen* allows us to efficiently construct well-characterized sequential circuit specifications with various characteristics and representative FSM benchmark sets.

Using large sets of benchmarks from *BenGen* that included two representative benchmark sets and two non-representative benchmark sets, we experimentally demonstrated several important benchmarking issues, mainly related to the benchmark set representativeness and statistical processing of the benchmarking results. In particular, we showed that using only a single statistic on the whole benchmark set (e.g. averaging over all the benchmarking results) the information obtained is very little and the conclusions very

limited. Considering more statistics and/or narrower benchmark classes gives more information and more precise information, enabling more precise and more useful conclusions. We also demonstrated that there is a trade-off between the benchmarking time, the amount and preciseness of information obtained from a benchmarking experiment, and the confidence level of the statistical conclusions. The experiments also showed that the pragmatic FSM state assignment approaches commonly applied in today's commercial circuit synthesis tools for FPGAs are effective only in some special cases. Through showing a big room for improvement regarding the state assignment methods applied in the contemporary commercial EDA tools targeted to FPGA synthesis, we demonstrated that an adequately performed benchmarking is able to reveal serious quality concerns. We also demonstrated that an inadequately performed benchmarking, and specifically performed when using a non-representative benchmark set, can easily result in faulty conclusions and this way mask important quality issues.

Although this paper delivered much information on the quality of several FSM state assignment techniques in their application to the FPGA-targeted circuit synthesis, the FSM synthesis problem and related techniques mainly served here as an example. The main aim of the paper was to demonstrate the importance and difficulties of benchmarking in digital circuit design automation, and to propose and discuss effective solutions to several benchmarking problems. A large part of the discussion of the paper is thus not limited to the digital circuit design, but pertains to the larger area of electronic design and its automation.

The discussions and experimental results of this paper clearly demonstrate the inadequacy of the traditional benchmarking approach in EDA, advantages of the design of experiments approach, and a big room for improvement in the sequential circuit design methods and tools.

Trough its substantial introductory and survey part, as well as explanations and examples presented, the paper has also a popularizing character. We hope therefore that this paper will be an important step towards a broader acceptance of the statistical design of experiments approach for benchmarking in the EDA field, and will trigger some works towards more adequate digital circuit design methods and tools.

Acknowledgements – The concept and initial design of the FSM Benchmark Generator BenGen have been developed by Lech Jóźwiak in his private capacity. The implementation of *BenGen* and subsequent experimental research have been performed at the Faculty of Electrical Engineering of the Eindhoven University of Technology, and were partly supported by the Technology Foundation STW, applied science division of NWO, and the Technology Program of the Ministry of Economic Affairs.

References:

[1] A.E.A. Almaini, et al.: State Assignment of Finite State Machines using a Genetic Algorithm, *IEE Proc. on Computers and Digital Techniques*, 1995, pp.279-286.

[2] M. Azlinah, Y. Marina, A. M. Itaza, M. Sofianita, A. R.Shuzlina: Constraint Satisfaction Problem Using Modified Branch and Bound Algorithm, *WSEAS Transactions on Computers*, Vol. 7, No 1, Jan. 2008, pp. 1-7.

[3] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart: Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics*, Vol. 1, No. 1, 1995, pp. 9-32.

[4] G.E.P. Box, W.G. Hunter, J.S. Hunter: *Statistics for experimenters: an introduction to design, data analysis, and model building*, Wiley, 1978.

[5] F. Brglez: The Scientific Method and Design and Test, *IEEE Design Test Computers*, Vol. 17, No 3, 2000, pp. 142-144.

[6] F. Brglez, R. Drechsler: Design of Experiments in CAD: Context and New Data Sets for ISCAS'99, *Proc. ISCASS'99*, May 30-June 2, 1999.

[7] F. Brglez, M.F. Stallmann, and Xiao Yu Li: SATbed: A Configurable Environment for Reliable Performance Experiments with SAT Instance Classes and Algorithms, *Proc. 6th Int. Conf. on Theory and Applications of Satisfiability Testing*, S. Margherita Ligure – Portofino, Italy, May 5-8, 2003.

[8] F. Brglez, Xiao Yu Li and M.F. Stallmann: On SAT instance classes and a method for reliable performance experiments with SAT solvers, *Annals of Mathematics and Artificial Intelligence*, Kluwer, 2004.

[9] K.A. Brownlee: Statistical theory and methodology in science and engineering, Krieger, 1984 (Reprinted with revisions from second edition, 1965).

[10] S. Chattopadhyay, P. Pal Chaudhuri: Genetic Algorithm Based Approach for Integrated State

Assignment and Flipflop Selection in Finite State Machine Synthesis, Proc. of Int. Conf. on VLSI Design, 1997, pp. 522-527.

[11] M. Coffin and M. J. Saltzman: Statistical Analysis of Computational Tests of Algorithms and Heuristics, *INFORMS Journal on Computing*, Vol. 12, No. 1, 2000, pp. 24-44.

[12] J. Cong and K. Minkovich: Optimality Study of Logic Synthesis for LUT-Based FPGAs, *Proc. FPGA'06*, February 22-24, 2006, Monterey, California, USA, ACM, pp. 33-40.

[13] S. Davidson and J. Harlow: Guest Editorial Introduction: Benchmarking for Design and Test, *Proc. IEEE Design and Test of Computers*, IEEE Computer Society Press, 17(3), July-September 2000, pp. 12-14.

[14] R.A. Fisher: *Statistical methods, experimental design, and scientific inference*, Oxford University, 1993.

[15] M. D. Galanis, G. Dimitroulakos, C. E. Goutis: Performance Gains from Executing Critical Software Segments to Coarse-Grain Reconfigurable Hardware, *WSEAS Transactions on Circuits and Systems*, Vol. 5, No 7, July 2006, pp. 1111-1118.

[16] M. R. Garey, D. S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co, 1979.

[17] J.E Harlow: Overview of Popular Benchmark Sets, *IEEE Design Test Computers*, vol. 17, No3, 2000, pp. 15–17.

[18] J.E. Harlow and F. Brglez: Design of Experiments and Evaluation of BDD Ordering Heuristics, *Int. Journal STTT*, 2001, No. 3, pp. 193-206.

[19] J. N. Hooker: Needed: An Empirical Science of Algorithms, *Operations Research*, No 42, 1994, pp.201-212.

[20] J. N. Hooker: Testing Heuristics: We Have It All Wrong, *Journal of Heuristics*, No 1, 1995, pp. 33-42.

[21] H. Hoos, T. Stützle: Evaluating Las Vegas algorithms – pitfalls and remedies, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufman, 1998, pp. 238–245.

[22] M. Hutton, J.P. Grossman. J. Rose and D. Corneil. Characterization and Parametrized Random Generation of Digital Circuits, *Proc. of 33rd ACM/IEEE DAC*, 1996.

[23] M. Hutton, J.P. Grossman, J. Rose and D. Corneil: Generation of Synthetic Sequential Benchmark Circuits, *Proc. ACM Symposium on FPGAs*, 1997, pp. 149-155.

[24] M. Hutton, J. Rose and D. Corneil: Automatic Generation of Synthetic Sequential Benchmark Circuits, *IEEE Trans. CAD*, Vol.21, No 8, 2002, pp. 928-940.

[25] D. S. Johnson: A theoretician's guide to the experimental analysis of algorithms, in *Fifth and Sixth DIMACS Implementation Challenges* (M.H. Goldwasser at al, eds.), Am. Math. Society, 2001, pp. 215-250.

[26] L. Jóźwiak: Efficient Suboptimal State Assignment for Large Sequential Machines, *Proc. EDAC-European Design Automation Conference*, 1990, pp. 536 - 541.

[27] L. Jóźwiak: An Efficient Heuristic Method for State Assignment of Large Sequential Machines, *Journal of Circuits, Systems and Computers*, Vol. 2, No.1, 1992.

[28] L. Jóźwiak: Life-inspired Systems and Their Quality-driven Design, *Proc. ARCS'06: 19th International Conference on Architecture of Computing Systems - System Aspects in Organic Computing*, Frankfurt/Main, Germany, March 13 - 16, 2006 (Keynote Paper), pp. 1-16.

[29] L. Józwiak, A. Chojnacki: Functional Decomposition Based on Information Relationship Measures Extremely Effective for Symmetric Functions, *Proc. 25th EUROMICRO Conference*, 1999, pp. 150-160.

[30] L. Jóźwiak, A. Chojnacki: Effective and Efficient FPGA Synthesis through General Functional Decomposition, *Journal of Systems Architecture*, Elsevier Science, Amsterdam, The Netherlands, Vol. 49, No 4-6, September 2003, pp. 247-265.

[31] L. Jóźwiak, A. Ślusarczyk, A. Chojnacki: Fast and Compact Sequential Circuits for the FPGA-based Reconfigurable Systems, *Journal of Systems Architecture*, Vol. 49, No 4-6, September 2003, pp. 227- 246.

[32] L. Jóźwiak, D. Gawlowski and A. Slusarczyk: An Effective Solution of Benchmarking Problem - FSM Benchmark Generator and Its Application to Analysis of State Assignment Methods, *Proc. DSD'2004 - Euromicro Symposium on Digital System Design*, 2004, pp. 160-167.

[33] N. Kapur, D. Ghosh, F. Brglez: Towards a New Benchmarking Paradigm in EDA: Analysis of Equivalence Class Mutant Circuit Distributions. *Proc. ACM Int. Symp. on Physical Design*, 1997, pp. 136-143.

[34] B. Lin, A.R. Newton: Synthesis of Multiple Level Logic from Symbolic High-Level Description Languages, *Proc. IFIP Int. Conf. on VLSI*, 1989, pp.187-196.

[35] A. Mahdoum, N. Badache, H. Bessalah: Low-

Power Scheduling Tool for System On a Chip Designs, *WSEAS Transactions on Circuits and Systems*, Vol. 6, No 12, December 2008, pp. 608-624.

[36] C. C. McGeoch: Experimental analysis of optimization algorithms, *Handbook of Combinatorial Optimization* (P. M. Pardalos and M. G. C. Resende, eds.), Oxford University Press, 2000.

[37] C. C. McGeoch: Experimental analysis of algorithms. In Handbook of Global Optimization, Volume 2: Heurstic Approaches (P. Pardalos and E. Romeijn, eds.), Kluwer, 2001.

[38] B. M. E. Moret: Toward a discipline of experimental algorithmics, in *Fifth and Sixth DIMACS Implementation Challenges* (M.H. Goldwasser at al, eds.), Am. Math. Society, 2001.

[39] A. Ślusarczyk: Decomposition and Encoding of Finite State Machines for FPGA Implementation, Ph.D. dissertation, Eindhoven University of Technology, The Netherlands, 2004.

[40] Z. Wenbiao, Y. Zhang, Z. Mao: Link-load Balance Aware Mapping and Routing for NoC, *WSEAS Transactions on Circuits and Systems*, Vol. 6, No 11, Nov. 2007, pp. 583-591.

[41] M. Yoshikawa, K. Otsuka, H. Terai: Dedicated hardware for inheritance-oriented crossover operation, *WSEAS Transactions on Circuits and Systems*, Vol. 7, No 3, March 2008, pp. 109-117.