

DEVELOPMENT OF CONSISTENT EQUIVALENT MODELS BY MIXED-MODEL SEARCH

Xin Guo (*), Adrian Stoica, Ricardo Zebulum and Didier Keymeulen

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Correspondence to: adrian.stoica@jpl.nasa.gov

(*) Chromatech, Alameda CA 94501

Abstract

This paper introduces a new approach to the development of equivalent models. Models of various accuracy and simulation speed may be needed in different contexts of design and analysis, or within different simulators. The models may be of similar or different nature, and could be for example structural or behavioral. Traditional model development and tuning is manual, and proceeds by finding one of the models, which is then used to derive equivalent model(s), e.g. a simpler behavioral model. It is not guaranteed that this simpler equivalent model is consistent with the thing it models in the first place (although it may be an approximation of the more complex model). This paper offers a means to automate modeling and derive the two or more equivalent models simultaneously and consistent with each other. The approach presented here relies on search algorithms to automatically explore the space of possible solutions in different model search spaces, alternating the evaluation of models of different type and resulting in models that have consistent behavior. This mixed-model search (MMS) approach is demonstrated with an example in which an evolutionary algorithm used as the search method automatically determines consistent equivalent models for a problem in which a software model and a hardware model would be otherwise inconsistent.

KEY WORDS

Evolvable Hardware, Mixtrinsic Evolution, Reconfigurable Circuits.

1. Introduction

The purpose of this paper is to introduce a new approach to the development of consistent equivalent models that may be used to characterize a system in

various contexts, such as within different simulators. These consistent equivalent models are referred in the following as forming a *model family*, whose individual members could be each of different nature, or, of same nature, i.e. of same structure/topology with differences for example in the levels of resolution or accuracy of their descriptive power. The choice of model determines the speed with which the simulation and search can occur and whether the simulation can converge to a solution within a practical time frame. These models differ in the modeling detail (e.g. modeling the switches with back-to-back transistors or with resistors, faster to simulate yet less accurate). There is often a tradeoff between speed of simulation/convergence and model accuracy.

In this paper we take as example the case of development of electronic circuit models. One should make the remark here that the problem of automatic model determination is analogous to the problem of automated circuit design or synthesis problem. In both cases one starts from some desired behavior and one seeks means to reproduce it. A variety of models, such as macro models and behavioral models, are used in electronics. The examples that will be used for illustration in this paper refer to models of circuits that will be mapped on configurable devices, and for which the reference will be to software models of various resolution, as well as a hardware model – which is a programmed configuration on a reconfigurable device.

In creating and tuning an alternative (often simplified) model the responses of candidate models are compared to a reference, which may be the actual system to be modeled or a trusted model of the system. For example a SPICE model is first created. Then a model order reduction (MOR) takes place by a technique such as PRIMA [1] and MMM [2]. These MOR approaches serve two functions: 1) reduce the circuit size so that efficient

time-domain analysis can be performed; 2) accurately extract frequency-domain characteristics by identifying dominant pole information. Speeds-up could be significant; for example MMM on a 19177 nodes circuit took 10-25 seconds while SPICE couldn't finish the circuit, one quarter of the circuit taking 6 days on SPICE [2].

It is essential that the simplified models are consistent with the models that generate them *and with the system to be modeled*. Usually one finds a first model that matches the real world and then one derives a simplified model which approximates the first. This simplifications are usually through repetitive manual trials and iterations and may result in simplified models that are approximations of the first model but not necessarily consistent with the system to be modeled. The technique proposed here automates this process and also results in consistently equivalent models. It performs a simultaneous search for models of the mode family. In our case target data is the considered obtained from and equivalent to the "real" system.

The paper is structured as follows: Section 2 details the proposed mixed-mode search. Section 3 demonstrates the technique applied to the automatic determination of circuit models (an automated circuit synthesis problem) in which evolutionary algorithms perform the search, and which illustrates how consistent models can be obtained while without applying the technique this was not the case.

2. Mixed-model search

In the following the reference will be to models of circuits, however the methodology is general for any kind of model. The automated scheme discussed here is that of design (modeling) by simulation, in which various candidate designs (models) are simulated during an automated search process. The proposed method of obtaining consistent models uses a heterogeneous mix of models of various types, e.g. of both high and low levels of resolution. In one embodiment, every candidate solution is modeled with many or all models corresponding to many or all possible levels of resolutions, and for each candidate circuit (model family), the fitness functions of the various models of that circuit are combined in evaluating the candidate circuit. In another embodiment, each candidate circuit is first modeled with a single model, different candidates being assigned models of different resolution levels; with each iteration of the simulation, each candidate circuit is assigned a different resolution level model, so that after a number of iterations, each candidate circuit has been modeled with all levels of resolution - alternatively, the reassignment to different resolution level models is performed randomly so that not all candidate circuits are

assigned to a different resolution level model at each iteration of the simulation.

One example of how high and low resolution models can differ is in the modeling of the switches S1, S2, S3, etc., of the configurable circuit of a programmable chip (e.g. Figure 1) [5]. In a low-resolution model, each switch can be modeled as a simple ON/OFF device having a very low resistance (e.g., 1 Ohm) in the ON state and a very high resistance (e.g., 1 Giga Ohm) in the OFF state. In a high resolution model, each switch is modeled as it is actually implemented, namely as a pair of parallel complementary MOSFETS using, for example, standard SPICE models for the PMOS and NMOS FET's. This latter model is more complex but exhibits a simulated behavior that more closely resembles the behavior of the actual switch.

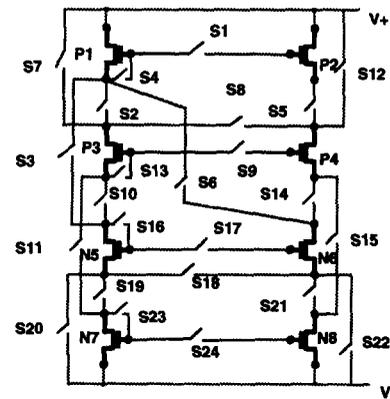


Figure 1: Reconfigurable circuit with 24 switches (See [5] for details).

A set of candidate circuits C1 through CN (representing, for example, different configurations of the reconfigurable circuit of Figure 1) is defined by a set of "chromosomes" that are fed to a high-resolution model to produce N high-resolution models M1 through MN. A simulator simulates the physical behavior of each of the models M1 through MN in response to a predetermined stimulus. The responses of each model are compared to a desired response and a fitness function is produced for each model, namely the fitness functions F1 through FN. A standard search process determines the next search points based on this information.

In the proposed MMS each one of the candidate circuit C1 through CN is modeled by both a high-resolution model and by one (or more) low-resolution models. Thus, for the N candidate circuits C1 through CN, there are N pairs of models M1, m1 through MN, mn. There are N high-resolution models M and N low-resolution models m. The pair of high/low resolution models (e.g., M2 and m2) (family model) representing a particular candidate circuit (e.g., C2) produces a pair of fitness functions (e.g., F2 and f2). A combiner combines

each pair of fitness functions to produce a combined score for the corresponding candidate circuit. For example, the combiner may compute the average of the two fitness functions as the combined fitness function or score. The combined score for each candidate circuit is provided to a search process that controls the simulator. The average may be a weighted average in which, for example, the fitness function of a higher resolution model is given more (or less) weight than that of a lower resolution model. Alternatively, the average may be unweighted.

A computational savings may be realized by employing only one model for each candidate circuit during any single iteration of the evolution process. With each iteration of the evolution process, a different resolution level model is assigned to each (or at least many) of the candidate circuits. As a result, after a number of iterations, each candidate circuit has been modeled with all levels of resolution. Such assignments may be carried out in a random fashion. For example a model could be considered for each candidate circuit, different candidate circuits being modeled with a model of a different resolution level. The first two candidate circuits C1 and C2 could be modeled with a high-resolution model (M1, M2 respectively) while the third candidate circuit C3 is modeled with a low-resolution model (m3). The simulator produces data that would lead to a fitness function from each model (F1, F2, f3, etc.) which is provided to the search decision mechanism.

The assignment of a particular candidate circuit to a model of a particular resolution level preferably, but not necessarily, is performed randomly so that the different resolution levels are distributed among all candidate circuit. Likewise, the transition at the end of each iteration of various candidate circuits to models of different resolution levels preferably, but not necessarily, is carried out in a random manner. Such random processes may be carried out in accordance with instructions furnished to the evolution process.

3. Mixtrinsic evolution experiments

3.1. Evolvable hardware

Evolvable hardware (EHW) refers to automated synthesis/optimization of HW (e.g. electronic circuits) using evolutionary algorithms. *Extrinsic EHW* refers to evolution using software (SW) simulations of HW models, while *intrinsic EHW*, refers to evolution with HW in the loop, evaluating directly the behavior/response of HW. For several reasons (including mismatches between models and physical HW, limitations of the simulator and testing system, etc.) circuits evolved in SW may not perform the same way when implemented in HW, and vice-versa circuits evolved in HW may not produce the desired response when simulated. This *portability*

problem limits the applicability of SW evolved solutions, and on the other hand prevents the analysis (in SW) of solutions evolved in HW. Mixtrinsic EHW (MEHW) that we have introduced is a third approach to EHW and a particular case of MMS. In MEHW evolution takes place with hybrid populations in which some individuals are evaluated intrinsically and some extrinsically, within the same generation or in consecutive ones. A set of experiments using Field Programmable Transistor Array (FPTA) architecture is presented to illustrate the portability problem, and to demonstrate the efficiency of mixtrinsic EHW in solving this problem

In *extrinsic* evolution (EEHW), schematically illustrated in Figure 2, the candidate solutions are evaluated as SW models (of HW) and evaluations are done using a simulator. The population is homogeneous, and consists of SW models (e.g. SPICE netlists) that describe an electronic circuit to a certain degree of accuracy. In *intrinsic* evolution (IEHW), the candidate solutions are in the form of physical HW configurations on programmable devices/architectures, which are evaluated using some test/evaluation equipment; IEHW is illustrated in Figure 3.

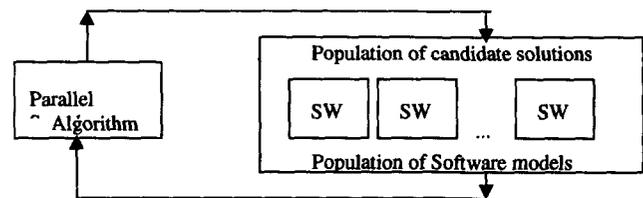


Figure 2 Extrinsic EHW: evaluations of software solutions

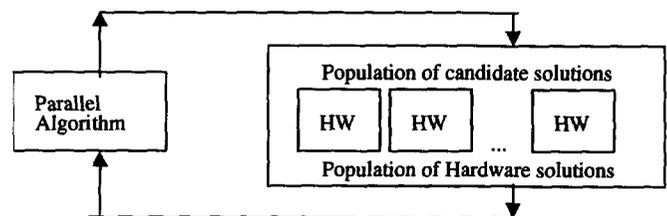


Figure 3 Intrinsic EHW: evaluations of hardware solutions.

3.2. Illustrating the portability problem in EHW

Early experiments in EHW made apparent that the solutions obtained by evolutionary design might suffer a portability problem. For example, it was observed that some circuits obtained through evolutionary design on one HW platform had a different behavior when tested on a second platform, although the two were of similar type/construction. Thus, a circuit evolved on a corner of

an FPGA did not reproduce the same behavior when it was implemented on a different part of the same FPGA [3]. Another situation is related to porting to HW a circuit evolved in SW (or vice-versa validating in SW a solution evolved in HW) as reported in [4]. Some of the circuits resulting as solutions from extrinsic evolution do not produce the same correct response when implemented/ported into HW. Vice-versa, many the circuit topologies resulting from intrinsic evolution do not produce a good response (as obtained in the real HW) when they are simulated in SW.

In these examples the target data is the considered obtained from the "real" system. In the first case the software model is determined and it approximates well the behavior of the system. The hardware model mapped from it does not behave consistently with software model, not fitting the data. A similar situation is if a hardware model is obtained first and the software model obtained from it (by the same circuit, yet with different physical properties).

3.3. Results of mixtrinsic evolution

Mixtrinsic evolution relates to applying MMS by evolving on mixed/heterogeneous populations, composed partly of models and partly of real HW [4]. This would constrain evolution to a solution that jointly simulates well in SW, and performs well in HW, i.e. a solution that exploits only the HW characteristics included in the SW model for producing the desired behavior (see Figure 4). Solutions based on HW properties outside the SW model are eliminated by evolution. In ME the population of candidate solutions is robust, more likely to be in agreement with common design rules, and, if novel, more likely to be patentable (i.e. to have generality and not depend on a fabrication process). The greatest advantage of the resulting solution is that can both operate in HW and can be analyzed in SW to explore its behavior outside the domain within which it was evolved – this is the only way to have insights and confidence in the evolved HW solution. Also, the resulting circuit is more likely to be portable to other HW platforms.

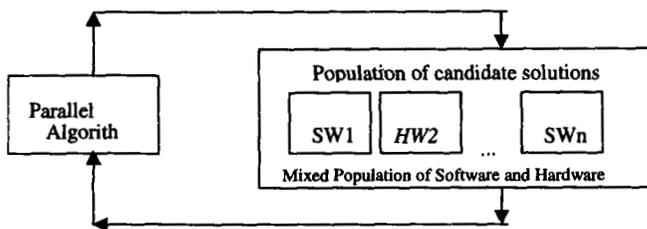


Figure 4. Mixtrinsic EHW: evaluations of mixed populations comprised of both hardware and software.

Two types of ME are further detailed: *complementary and combined ME*. In *complementary MEHW*, candidate

solutions are evaluated after being alternatively reassigned to either a HW or a SW platform (subject to random or deterministic choice). For example, an individual in a generation would have probability P to be evaluated in HW and probability $1-P$ to be evaluated in SW. Assuming HW evaluates faster than SW one can speed-up evaluations by having a high value of P , which will cause a larger population to be evaluated in HW. The probability P , and related to it the ratio of individuals evaluated in HW over the total population, could also be variable parameters, adjustable during evolution.

In what we refer here as *combined MEHW*, each individual is evaluated both in HW and SW, and a combined fitness function is calculated. In the simplest case this can be a simple average of the two components or may involve adjustable weights etc.

We refer to the above description as a *matching ME*, for which the emphasis was on reinforcing the matching of similar characteristics of the SW models and the HW it describes. An opposite idea would be reinforce dissimilarities and reinforce HW (or SW) distinctive characteristics, i.e. mismatches, and we will refer to this as *mismatching MEHW*. This paper gives an example of MEHW using a Field Programmable Transistor Array (FPTA) as evolutionary testbed.

The FPTA was proposed as a flexible, versatile platform for EHW experiments. The cell is largely a "sea of transistors" interconnected by other transistors that act as signal passing devices (gray-level switches). Details of the FPTA, its HW implementation and evolutionary experiments on FPTA can be found in [5]. The FPTA was exercised on a testbed that supports HW and SW evaluations (intrinsic/extrinsic). The SW subsystem makes use of the Caltech 256-processor HP Exemplar parallel computer to run multiple copies of SPICE. The HW subsystem is built around National Instruments LabView, associated data acquisition boards, signal generators, and other equipment, see [6] for more details.

The following exemplifies the portability problem and demonstrates the ME's ability to solve this problem. The following experiments are shown: a) Extrinsic evolution, with the resulting solution valid in SW but invalid when tested in HW, b) Intrinsic evolution, with the resulting solution valid in HW but invalid when tested in SW, c) Mixtrinsic evolution, with the resulting solution valid both in SW and HW.

The experiments show evolutionary synthesis of an AND gate and use one FPTA cell. The input signals follow all input combinations of logic levels 1s and 0s. The level 'high' input signals corresponding to logical '1' were controlled to keep their level for 5 ms, which corresponds to 20 samples on LabView graphs of acquired signal from HW. All experiments (about 20 runs for each case in a) and b) and 5 each for c1) and c2) below) used 30 individuals for 30 generations.

a) Two best individuals in the last generation are the solutions presented here for discussion. One of the solutions could be validated in HW. However, the circuit shown in Figure 5, which is in fact the solution with the highest fitness, does not give satisfactory response when downloaded into HW. Thus, two direct observations can be made: a) one solution is validated in HW while the other is not, b) in this particular case, the “better “ (in the sense of the fitness function that rewarded for higher value of the ‘1’ level) solution in SW performs worse in HW. It appears that the solutions obtained through extrinsic evolution may not work in HW. Moreover, in many cases, there is no way to know for sure if it works without validating in actual HW.* (* We believe this reflects the current state-of-the-art, but admittedly we are strongly biased by our own experience with a certain model and HW. We believe that increasingly higher confidence in a solution would come from minimizing the negative effects of the factors discussed in Section 3. We also refer mainly to effects in analog circuits, and

especially to those NOT relying on well understood building blocks, such as Op. Amps etc).

b) Intrinsic evolution. A circuit obtained intrinsically (best individual after a 30 individuals for 30 generations run) and its response in HW and SW are shown in Figure 6. The conclusion is that the solutions obtained through intrinsic evolution may not work in SW.

c1) Combined Mixtrinsic Evolution (Matching). Each individual was evaluated both in HW and SW. The combined fitness was a simple average. The SW and HW responses are similar. The resulting solution is shown in Figure 7.

c2) Complementary Mixtrinsic Evolution (Matching). Each individual was allocated either to HW and SW evaluation with a 50% probability. The response of the resulting solution is identical to that illustrated in Figure 7 (although the circuit is slightly different) and is omitted for space reasons.

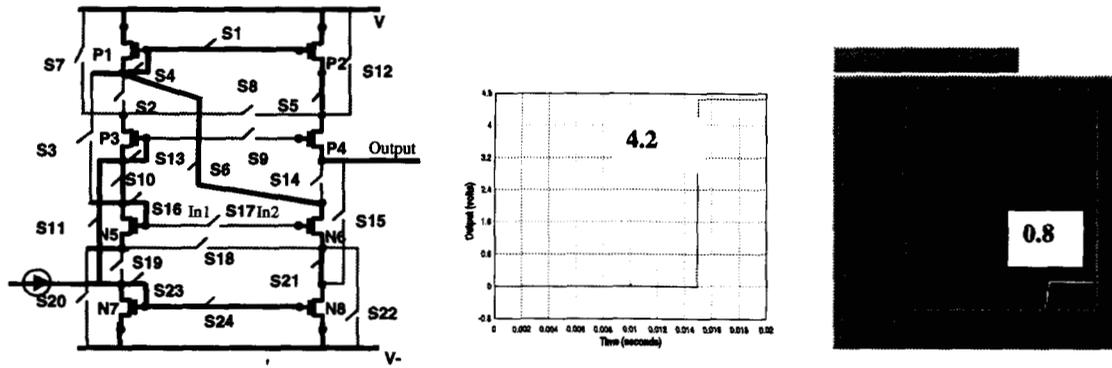


Figure 5. Extrinsically evolved circuit, its response in SW (middle) and invalid response in HW (right).

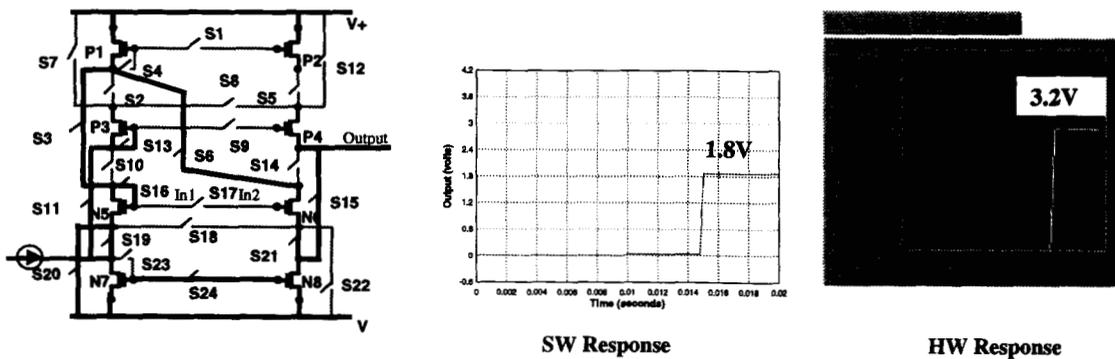


Figure 6. Intrinsically evolved circuit, its response in HW (right) and its invalid response in SW (middle).

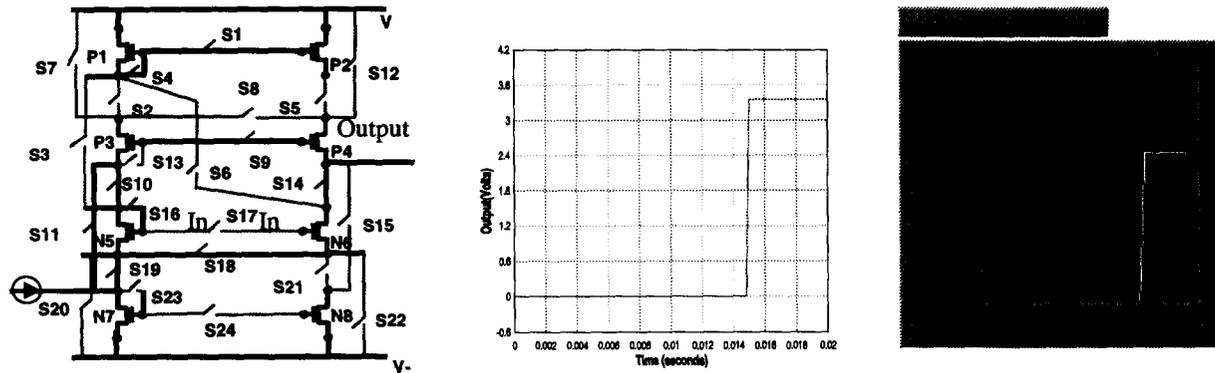


Figure 7. Circuit obtained by mixtrinsic evolution, its valid responses in SW and in HW

In all experiments the best 6 individuals of the last generation were tested both in HW and SW and displayed similar response. Although this is only empirical evidence, there is a good reason to believe that selection pressure would indeed favor solutions that display similar response in HW and SW.

d) Divergent ME: exploiting the distinctive characteristics of HW (or SW): Once accounted with the likelihood of obtaining mismatched responses between HW and SW, it appears straightforward to accept that selection pressure can force things in this direction (of mismatches). We are currently performing experiments in which we use combined evolution (each individual is evaluated twice, once in HW and once in SW). The combined fitness functions are either ratios of fitness of HW over fitness of SW, or derivations of it such as sum of HW fitness and inverse of SW fitness. Preliminary experiments illustrate that indeed resulting circuits produce the expected result in HW, while not being able to give a good response in SW.

4. Conclusion

The mixed-model search was introduced as a general method to obtain consistent equivalent models. The search seeks simultaneously with model families, including models of different resolution or type. In the context of evolutionary design and evolvable hardware the technique is particularized as *mixtrinsic* evolution and uses heterogeneous populations of individuals some of which are evaluated extrinsic and some intrinsic. Used to alleviate the portability problem, *convergent mixtrinsic* evolution reinforces similarities between SW and HW behavior.

Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration.

References

- [1] PRIMA A. Obadasioglu, M. Celik, and L. T. Pillage, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm" IEEE Trans on Computer-Aided Design, Vol CAD-17, No 8 pp 645-654, Aug 1998
- [2] MMM Hui Zheng and Lawrence T. Peleggi, Robust and Passive Model Order Reduction for Circuits Containing Susceptance Elements. ICCAD Int. Conference on Computer Aided Design, San Jose, CA, Nov 10-14, 2002, p761-774
- [3] Thompson, A. Silicon Evolution. In: Proceedings of Genetic Programming 1996 (GP96), J.R. Koza et al. (Eds), pages 444-452, MIT Press 1996
- [4] A. Stoica, R. Zebulum and D. Keymeulen. "Mixtrinsic Evolution". In T. Fogarty, J. Miller, A. Thompson and P. Thompson, (eds.), In Proc. of the Third International Conference on Evolvable Systems (ICES2000). April, 2000, Edinburgh, UK. New York, USA, Springer Verlag. (pg.208-217)
- [5] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, Reconfigurable VLSI Architectures for Evolvable Hardware: from Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips. In IEEE Trans. on VLSI Systems, Special Issue on Reconfigurable and Adaptive VLSI Systems, vol. 9, No. 1, Feb. 2001. (pp.227-232).
- [6] D. Keymeulen, Gerhard Klimeck, R. Zebulum, Adrian Stoica, Yili Jin, Carlos-Salazar Lazaro. "EHWPack: an Evolvable Hardware Environment using the Spice Simulator and the Field Programmable Transistor Array". In the Proc. of ANNIE'2000 (Smart Engineering System Design), St. Louis, MO, pp. 233-338, ASME Press November 5-8, 2000.