An Efficient and Effective Performance Estimation Method for DSE

Chen Lin, Xueliang Du, Xinwei Jiang, Donglin Wang
Institute of Automation, Chinese Academy of Sciences
No.95, Zhongguancun East Rd, HaiDian Dist, Beijing, China
{chen.lin, xueliang.du, xinwei.jiang, donglin.wang}@ia.ac.cn

***Abstract*—Design Space Exploration (DSE) is a critical step in the chip design. The tradeoffs and interactions among parameters are traditionally evaluated by simulating or synthesizing a variety of designs which is intractable. The predictive modeling techniques have been applied to predict the design performance for DSE. For the system-on-a-chip (SoC) DSE cases, however, it is difficult to achieve high accuracy with previous methods due to their limitations. In this paper, we proposed a new estimation method based on Regression Random Forests (RRF) to build accurate and reliable prediction models. Our method can significantly improve the prediction accuracy and accelerate the procedure. In addition, due to the comprehensible tree model, RRF could guide the SoC design by ranking the parameters' importance. The experimental results show that RRF can reduce relative errors by about 60%, which demonstrate the effectiveness of our method.**

I. Introduction

The advancement of semi-conductor processing technology has made it feasible to fabricate a very large scale integrated (VLSI) circuit on the chip called system-on-a-chip (SoC). The requirements of multi-function and high-performance computing make SoC design a great challenge. Modern SoCs consist of multiple processing elements (PEs) and high-speed interconnection. Although the IP-based method [1] reduces the design difficulty and increases productivity by reusing pre-designed PEs and interconnections, Design Space Exploration (DSE) is still a critical and difficult step which determines the appropriate SoC configuration.

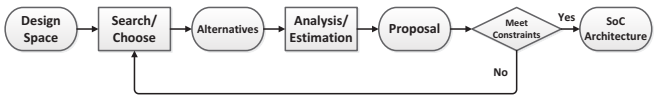

Fig. 1. A framework of DSE

As illustrated in Figure 1, DSE is a largely iterative trial-and-error procedure and composed of two key steps. One is searching for the optimum from a large design space. The other is evaluating alternatives, i.e., how to analyze or estimate design configurations. Traditionally, at every step, architects need to choose from a vast set of architectural techniques and parameters according to their own experience and intuition. Because of the large design space and time-consuming simulation or synthesis, it is intractable to explore the entire design space. To speed up the simulation, several fast performance analysis approaches were proposed. One of them is based on statistical analysis after scheduling and mapping tasks [2, 3]. It is suitable for data-flow application tasks, e.g., audio/video encoding and decoding. Another method uses CPU trace logs to speed up the simulation [4], which is mainly used for CPU DSE cases. In order to reduce the overall simulation costs, the predictive modeling which reduces the number of simulated configurations was also proposed [5–7]. However, the mentioned predictive approaches focus on DSE for microprocessors rather than SoCs.

In this paper, we proposed an estimation method based on the Regression Random Forests (RRF) to accelerate the step of estimating alternatives for DSE. The results of the experiments demonstrate that this method can provide with high accurate predictions in less time, and offer architects useful guide information due to its feature-ranking attribute.

The rest of this paper is organized as follows. Section II introduces the related work. Section III presents our RRF approach. Section IV reports the experiments. Section V concludes this paper.

II. Related Work

A. Design Space Exploration

The regression modeling technique reduces the number of simulated/synthesized design configurations by learning the relationships between design parameters and responses. Following the supervised learning framework, Joseph *et al.* [5] used clock-accurate simulator to build a linear regression model which shows the relationship between 26 importance parameters and CPI (Cycles Per Instruction). Ipek *et al.* [6] utilized ANNs to describe a non-linear mapping from CPU inner parameters to IPC (Instructions Per Cycle). Chen *et al.* [7] exploited unlabeled design configurations to improve accuracy of the M5P tree model, which is also used for microprocessor DSE. As so far, these approaches mainly focus on DSE for microprocessors. And there are a few studies about exploiting regression models to illustrate the relationships between SoC interconnection parameters and its metrics.

B. Random Forests

Beriman developed an algorithm for inducing a random forest [8], and used ” Random Forests” as the trademark. Random Forest randomly samples not only in the training examples, but also in the features, which can ensure the independence of all the decision trees and predict without bias by voting. Criminisi *et al.* proposed a decision forests

framework to extend the existing forest-based techniques and to unify classification, regression, manifold learning and semi-supervised learning. In recent years, Random Forests model is widely used in machine learning, computer vision, and medical image analysis [9].

III. Regression Random Forests for SoC Design Space Exploration

The random forests model is an ensemble learning method for classification, regression and other tasks, which constructs a multitude of decision trees at training time and outputting the class (classification) or mean prediction (regression) of the individual trees. Our proposed method is based on regression random forests.

In general, for regression, there are a set of labeled training examples $\{x_i, y_i\}_{i=1}^{N}$, where $x_i \in \mathcal{X} = \mathbb{R}^M$, $y_i \in \mathcal{Y} = \mathbb{R}^K$, N is the total number of the training examples. Regression Random Forests model (RRF) typically describes a non-linear mapping $\mathcal{M} : \mathbb{R}^M \rightarrow \mathbb{R}^K$, where an example x is mapped to a target prediction y. This mapping is learned by an ensemble of binary decision trees $\{\mathcal{T}_t\}_{t=1}^{T}$, where T is the number of these trees.

In the RRF-based estimation framework for SoC DSE, x_i is the i-th SoC design with M interested configuration parameters, e.g., width or frequency of interconnection, the depth of cross-clock domain FIFO. And y_i is the corresponding SoC response, e.g., performance metrics like the benchmark simulation time, power, area. Here we take benchmark simulation time as an example. The N design configurations randomly sampled from SoC design space are simulated by benchmarks to label the execution time. These labeled examples are used for RRF supervised-learning. The RRF supervised-learning procedure is as follows:

1) Construct a training set from labeled examples for one decision tree using the bootstrap sample method.
2) Each node of $\mathcal{T}_t$ randomly samples a set of splitting functions $\phi(x)$ to split training data into two subsets. To minimize the uncertainty of the target variables, the best splitting function $\phi^*(x)$ on every node is measured by the criterion of maximum information gain.
3) The $\mathcal{T}_t$ recursively splits until the stopping conditions are met, e.g., a maximum tree depth D or a minimum number of samples left in the splitting node. Once one of these criteria is fulfilled, the splitting node becomes the leaf node, whose density model is calculated by all samples falling in it.
4) Repeat the above steps to create T trees to form the RRF model.

The definition of T and D will be discussed in details in the part D of Section IV.

Compared with the most powerful learning models, such as ANNs and ensembles, RRF is a more comprehensible model. Because of the basis of decision tree model, RRF can evaluate the importance of different SoC parameters. The relative rank (i.e. depth) of a feature can be used to assess the feature's relative importance with respect to the predictability. Features that are closer to the top (i.e. root) contribute more to the final prediction.

IV. Experiments

A. SoC Design Space Definition

Figure 2 shows the SoC architecture in our experiment. CPU is the central processing unit whose responsibility is task scheduling. PE is the processing unit which is responsible for the large scale data computing. Bridge is the bus interconnection between a master and a slave with any combination of different data port widths, different clock frequencies, and different endianness. HSM is the high-speed memory, e.g., DDR3 DRAM. In our experiment, the instruction set architecture (ISA) of the CPU is ARMv7. An innovative algebraic processing engine is utilized as PE. In order to illustrate the university of RRF, we adopt three SoC topologies in the experiments, such as 4 PEs and 1 HSM (*4p1m*), 4 PEs and 2 HSMs (*4p2m*), 8 PEs and 2 HSMs (*8p2m*).

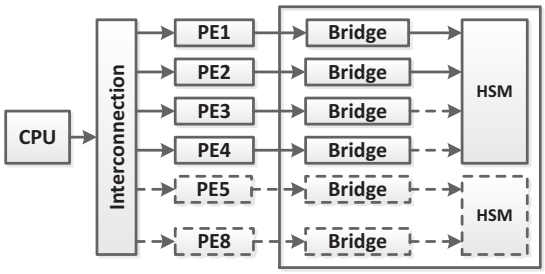

Fig. 2. The architecture of SoC.

In general, according to the requirements of SoC applications and the constraints of specifications, architects should define the SoC fixed and variable parameters at the beginning. In our experiment, the SoC fixed and variable parameters are respectively shown in Table I.

TABLE I
SoC fixed and variable parameters.

The fixed parameters		
Name	**Description**	
CPU	800MHz, data width = 128bit	
PE	1GHz, data width = 128bit	
Interconnection	AXI3.0, the max burst length = 16	
The variable parameters		
Name	**Description**	**Value**
HSM_width	high-speed memory width	32,64,128,256,512
w_x2x_depth	write channel FIFO depth	4,6,8,10,12,14,16
r_x2x_depth	read channel FIFO depth	4,6,8,10,12,14,16
MP_depth	master port synchronizer	2,3
SP_depth	slave port synchronizer	2,3
freq_PE_clk	PE interface frequency	400,500,600,700,800
freq_HSM_clk	HSM interface frequency	400,500,600,700,800
Total	7	24500

B. Benchmarks for SoC Evaluation

The most common way to evaluate the SoC performance is to measure the execution time of a specific application. We employed power flow calculation which is a power system

vocabulary as the benchmark. It is used to calculate all parameters of a stable power system based on its structure, the parameters of generators and loads. There are two well-known algorithms for power flow calculation, i.e., the Newton method and the PQ decomposition method. The Newton method consists of four key steps, i.e., Y matrix calculation (*ym*), polar to rectangular coordinate transformation (*p2r*), Jacobi matrix calculation (*jm*), gaussian iteration and revise solution to equation (*gr*).

In our experiments, we randomly generated 200 of 24500 design configurations without assuming the superiority of any specific one. To obtain the real performance value, we simulated the configurations with verilog HDL in Register Transfer Level (RTL) which can provide high accuracy. Simulating the configurations for 4 benchmarks of 3 different PEs and HSMs topologies consumed about 150 hours on a cluster with 16 Intel Xeon E5-2620 CPUs. Among the 200 simulated configurations, 180 design configurations were considered as the training set and the rest were the test set.

C. Performance Evaluation of RRF

To demonstrate the effectiveness of RRF on the performance of estimation for SoC DSE, we compared RRF with Ridge Regression model, Artificial Neural Network and Adaboost. Ridge regression model utilizes the regularized least squares method which can overcome the over fitting. ANN is the state-of-the-art predictive model [6]. Adaboost is the representative model ensemble method which is closely related to our method.

We used Scikit-learn toolbox [10] to implement Ridge Regression model, Adaboost and Regression Random Forest, and used pybrain toolbox [11] to implement ANN. Ridge Regression model adopted the optimal regularization parameter α of 0.5. ANN adopted one 16-unit hidden layer, a learning rate of 0.6, the max training epoch of 500. Adaboost adopted the decision tree model as the weak learner whose number is 20. RRF adopted T of 20 and D of 20.

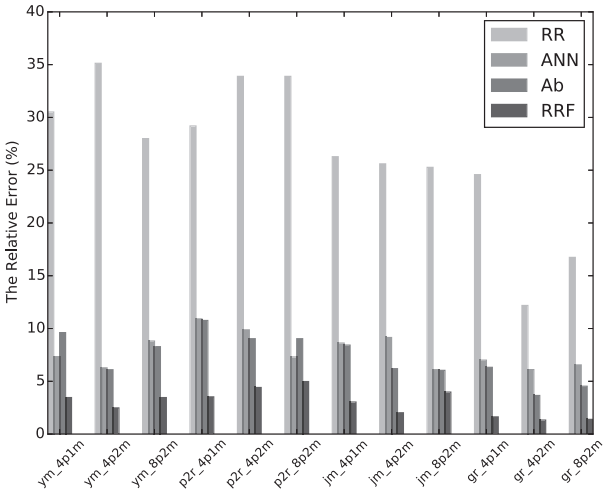

Fig. 3. The quantitative comparison of the different regression models.

Figure 3 presents the detailed comparisons of prediction results on test data, where Ridge Regression model (RR), ANN-based model (ANN), Adaboost model (Ab) and Regression Random Forests model (RRF). We can clearly see that RRF significantly outperforms other approaches over all 12 situations formed by combinations of 3 SoC topologies and 4 benchmarks. The average Relative Error (RE) of RRF is 2.93%, which is much smaller than the average RE of RR (26.77%), ANN (7.84%) and Adaboost (7.33%). Most notably, on the situation *gr_4p1m*, RRF reduces RE by 93.4% of RR, by 76.8% of ANN, by 74.3% of Adaboost. Even on the situation with the least RE reduction (*jm_8p2m*), RRF reduces RE by 84.3% of RR, by 34.8% of ANN, by 34.1% of Adaboost. On the easy-to-predict situation *gr_4p2m*, the average RE of these four models is 23% and RRF reduces RE by 89.1% of RR, by 78.3% of ANN, by 63.7% of Adaboost. On the hard-to-predict situation *p2r_4p2m*, the average RE of these four models is 57% and RRF reduces RE by 87.0% of RR, by 55.3% of ANN, by 51.2% of Adaboost.

TABLE II
THE AVERAGE TRAINING AND TEST TIME ON ALL SITUATIONS.

	RR	**ANN**	**Adaboost**	**RRF**
time(s)	0.0014	52.0423	0.0230	0.0227

On the other hand, from the comparisons of the average learning and test time on all situations shown in Table II, RR is the fastest. The speed of Adaboost and RRF is similar. RRF is slightly faster than Adaboost. ANN is the slowest because of the large number of iterations. Hence, we can conclude that RRF is much more practical than other three regression modeling techniques due to its high prediction accuracy and fast computing speed.

D. The Effect of RRF Parameters

We further evaluated the impact of the number of decision tree T and the maximum of tree depth D on the prediction accuracy of RRF.

Figure 4 offers an illustrative example about the relationship between the prediction accuracy of RRF and its parameters, i.e. T and D. When T is fixed, the prediction accuracy of RRF clearly increases as D increases when $D < 10$. And the accuracy tends to be a stable value without obvious overfitting when D is greater than 10. Meanwhile, as T increases, the curves which illustrate the relationship between prediction accuracy and D become smoother. Therefore, we find that although RRF is a parametric model, the definition of its parameters are not as difficult as ANN. According to the Figure 4, we believe that the RRF model with $T = 20$ and $D = 20$ is good and stable enough to predict reliable results.

E. Feature Ranking for SoC DSE

RRF can not only predict SoC performance, but also rank the importance of SoC parameters. Taking *ym_8p2m* as an example, Table III shows that RRF ranks the importance of SoC parameters regarding the execution time. In this SoC

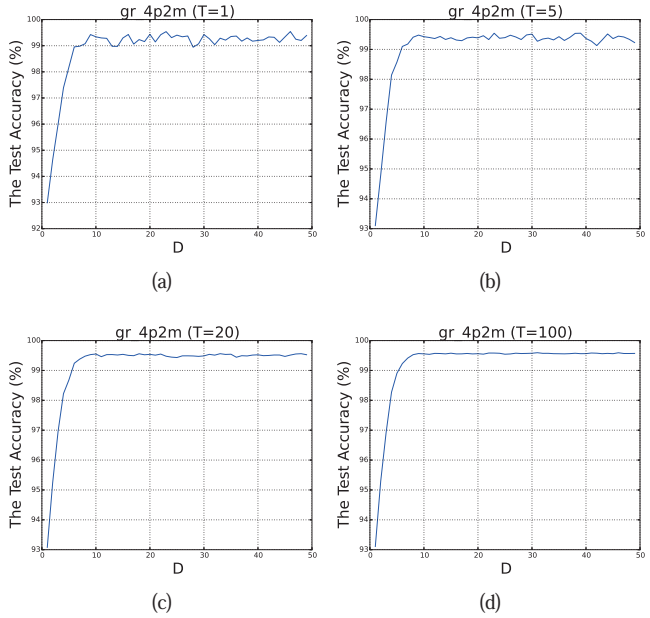

Fig. 4. The test accuracy of different T and D.

topology, the data width of HSM is the most effective factor of performance. The least two effective factors are the the number of synchronizers in bus bridge. These information will help architects efficiently improve SoC design. For example, since the effect of synchronizer number is subtle, this parameter can be fixeqd on the next exploration step to prone the SoC design space. And if the first target is SoC's performance, the top priority is the bandwidth of HSM. Because the SoC performance is sensitive to the frequency and width of HSM. Furthermore, we can find that the importance of write channel FIFO depth in bus bridge is greater than that of read channel. So architects can configure different write/read channel FIFO depths to meet the performance requirement and to reduce the area.

TABLE III
THE SOC PARAMETER RANKING.

1.	HSM_width	(0.728715)
2.	freq_HSM_aclk	(0.238023)
3.	freq_PE_aclk	(0.019272)
4.	w_x2x_depth	(0.011598)
5.	r_x2x_depth	(0.001610)
6.	MP_depth	(0.000504)
7.	SP_depth	(0.000278)

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed an estimation method based on RRF models to speed up SoC evaluation for DSE. RRF is a supervised-learning regression model which can explore the relationship between SoC parameters and performance. Compared with other regression models, the RRF approach predicts faster and more accurately. Furthermore, it presents the features/parameters ranking to guide the design, and thus, the optimal SoC configurations can be acquired more efficiently.

In the future, we plan to leverage unlabeled configurations and semi-supervised learning to improve the accuracy of RRF. We also aim to propose an innovative exploration method combined with RRF models to complete the DSE flow.

ACKNOWLEDGMENT

This work was supported by the Strategic Priority Research Program (Category A) of the CAS under Grant X-DA06011000.

REFERENCES

[1] P. Bricaud, "Ip reuse creation for system-on-a-chip design," in *Custom Integrated Circuits, 1999. Proceedings of the IEEE 1999*, 1999, pp. 395–401.

[2] S. Kim *et al.*, "Schedule-aware performance estimation of communication architecture for efficient design space exploration," in *Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference on*, Oct 2003, pp. 195–200.

[3] S. Sombatsiri *et al.*, "An amba hierarchical shared bus architecture design space exploration method considering pipeline, burst and split transaction," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*, May 2013, pp. 1–6.

[4] C. Lee *et al.*, "A systematic design space exploration of mpsoc based on synchronous data flow specification," *Journal of Signal Processing Systems*, vol. 58, no. 2, pp. 193–213, 2010.

[5] P. Joseph *et al.*, "Construction and use of linear regression models for processor performance analysis," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, Feb 2006, pp. 99–108.

[6] E. Ïpek *et al.*, "Efficiently exploring architectural design spaces via predictive modeling," *SIGPLAN Not.*, vol. 41, no. 11, pp. 195–206, Oct. 2006.

[7] T. Chen *et al.*, "Effective and efficient microprocessor design space exploration using unlabeled design configurations," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 20:1–20:18, Jan. 2014.

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] A. Criminisi *et al.*, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision: Vol. 7: No 2-3, pp 81-227*, 2012.

[10] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] T. Schaul *et al.*, "PyBrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.