SPIRIT: Spectral-Aware Pareto Iterative Refinement Optimization for Supervised High-Level Synthesis

Sotirios Xydis, Gianluca Palermo, Vittorio Zaccaria, and Cristina Silvano

I. Introduction

High level synthesis (HLS) has been recognized as a key enabler for automated coprocessor synthesis within shortened design cycles. Due to the large number of explorable HLS parameters exposed by modern tools (such as [1]–[3]), designers have faced the prob-lem of investigating on different trade-offs regarding the generated architectural configurations. Efficient design space exploration (DSE) strategies that supervise the invocation of HLS tools are of great importance for fast evaluation of the solution space [4].

Iterative exploration during HLS is common practice [5]–[8] used to evaluate the quality of the design solutions in a presynthesis step. In this way, only a reduced set of design configurations of the Pareto-front found during the high-level exploration is propagated down to gate-level and physical synthesis tools for further refinement. The delay of iteratively using even only the core HLS engine imposes long and in many cases unaffordable exploration runtime. While a lazy approach would suggest to limit the design space (e.g., excluding from exploration compiler parameters). References [8] and [9] have proven that this type of exploration simplifications inherently reduce the effectiveness of the DSE, thus the optimality of results.

In this context, machine-learning techniques [10]–[12] have been envisioned as an effective way to hide the complexity and to improve exploration's time. Xydis *et al.* [10] proposed an response sur-face model (RSM)-based Pareto iterative refinement HLS exploration methodology for customizing the compiler/architectural parameters of hardware coprocessors. Zuluaga *et al.* [12] proposed the incorporation of Gaussian process models that apply to the overall design space and use the estimated uncertainty to iteratively classify the points as either non or Pareto optimal. However, the evaluation of the technique has been performed on a limited size solution space. Although, Gaussian models enable to control the uncertainty bounds, they suffer from high runtime when the size of solution space scales up. Recently, Liu and Carloni [11] proposed an iterative refinement framework for supervised HLS that tries to *a-priori* minimize the uncertainty by utilizing random forests (RFs) and uses transductive experimental design (TED) for selecting representative configurations to train the predictive model. As noticed in [11], TED presents a poor scaling for extreme size design spaces. Randomized TED (RTED) is used to alleviate this problem, which, however, degenerates to an almost random sampling for large solution spaces.

In this paper, we address the problem of supervised HLS for application-specific co-processor synthesis by proposing for the first time the combination of spectral techniques with RSMs to efficiently approximate Pareto optimal design solutions. We show: 1) how the multiobjective HLS solution space can be modeled as time-series, thus being suitable for signal processing techniques and 2) how the typical structure of Pareto iterative refinement exploration strategies should be extended to efficiently integrate and manage spectral analysis to extract valuable information of the structure of the design space. Specifically, we introduce spectral analysis techniques to characterize the randomness of different regions of the solution space, and then we exploit the extracted knowledge to guide the iterative RSM refinement. Several new techniques have been proposed in this paper: 1) identification of hard-to-predict partitions in the solution space; 2) adaptive sampling on both the Pareto; and 3) the spectrum distribution, extending the structure of existing DSE frameworks.

While existing approaches [10]–[13] adopt refinement strategies focusing only on the optimization potential of the configurations, we extend this concept by incorporating configurations from both Pareto and high-variance regions of the design space, managing to refine the employed RSM toward higher quality solutions, improving its prediction accuracy. Considering very-large solution spaces—orders of magnitude larger than those used (see [11], [12]), we show the efficiency of the proposed exploration strategy with respect to state-of-art HLS exploration techniques, not only in terms of average accuracy (given the same exploration runtime latency), but also in terms of robustness, by guaranteeing a fast convergence toward optimal solutions.

II. Spectral Analysis in RSM-Based Exploration

We focus on the fundamental architectural synthesis problem of discovering design solutions that derive Pareto optimal delay-area trade-offs. Specifically, given a design space $\boldsymbol{D}$, the target problem can be formulated as the following multiobjective optimization problem:

$$\min_{x \in \boldsymbol{D}} \left[Delay(\boldsymbol{x}),\ Area(\boldsymbol{x}) \right] \in R^2$$

Manuscript received March 19, 2014; revised June 4, 2014 and August 7, 2014; accepted September 10, 2014. Date of publication October 20, 2014; date of current version December 17, 2014. This paper was recommended by Associate Editor S.-C. Chang.

S. Xydis is with the Institute of Communication and Computer Systems, National Technical University of Athens, Athens 15731, Greece (e-mail: sxydis@microlab.ntua.gr).

G. Palermo, V. Zaccaria, and C. Silvano are with the Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milan 20133, Italy.

Color versions of one or more of the figures in this paper are available online.

subject to the following constraints:

$$\big[Delay(\boldsymbol{x}), Area(\boldsymbol{x})\big] \leq \big[Max\_Delay,\ Max\_Area\big].$$

The variable $\boldsymbol{x} \in \boldsymbol{D}$ is a vector that refers to an instance of the HLS design space composed of the design parameters that are exposed to the designer. In our case, vector $\boldsymbol{x}$ includes both compiler and architectural parameters, as defined in [8].

A. Signal Representation of the HLS Solution Space

In this section, we show that the targeted HLS solution space can be modeled as a discrete-time signal with finite duration, i.e., a time-series defined only at discrete points in time but not in magnitude. This will enable the application of well-known signal analysis techniques to be incorporated during exploration. Without loss of generality, we define the discrete-time signal S, with $N_D = 2^k$ data points, as the signal representing a HLS solution space. Let $s[x]$ be the magnitude of the signal representing a HLS solution after applying configuration x. We define $s[x]$ as 1-D signal considering its area-delay product, that is

$$s[x] = Area(x) \times Delay(x), x \in D_o$$

where D_o is an ordering of the design space D, which corresponds to the traversal order in case of a full-search evaluation of the solution space is applied.

We focus on spectral analysis for the HLS signal to estimate its frequency content and the related power spectrum. The power spectrum of a signal represents a characterization metric of its randomness. The more correlated or predictable is a signal, the more concentrated is the power spectrum. Conversely the more random or unpredictable is a signal, the more spread is the power spectrum. Thus, while for periodic signals, the power is concentrated in narrow bands of frequencies, indicating the existence of structure and the predictable character of the signal, for a purely random signal the signal's power is spread equally in the frequency domain, indicating the lack of structure in the signal [14]. For the discrete-time signal $s[x]$ of length N_D samples, the total energy is finite, $\sum_{-\infty}^{+\infty} |s[x]|^2 < +\infty$. Thus, the discrete Fourier transform is defined as N uniformly spaced spectral samples

$$S[f] = \sum_{x=0}^{N_D-1} s[x] \times exp(-j2\pi fm).$$

The power spectrum analysis is concerned with the distribution of the signal power in the frequency domain. For the nonparametric HLS signal, S, the power-spectral density is

$$|P[f]|^2 = \frac{1}{N_D} \left| \sum_{x=0}^{N_D-1} s[x] \times exp(-j2\pi fm) \right|^2 = \frac{1}{N_D} |S[f]|^2.$$

Our intuition was that spectral analysis can be used to provide more localized information of the HLS signal. As in typical signal processing, the overall HLS signal can be partitioned in smaller windows, each one referring to a specific region of the HLS solution space. The power spectrum of each window characterizes the randomness of the region. Based on this observation, we propose a new exploration strategy that utilizes spectral analysis to drive the iterative refinement.

B. Proposed Exploration Strategy

The proposed exploration strategy exploits two major concepts.

1) The usage of RSM techniques to analytically capture the relations between the design parameters and the area-delay response variables, thus speeding up the process associated with a HLS synthesis evaluation.
2) The introduction of spectral analysis techniques for guiding iterative refinement with configurations found in "hard to predict" regions, to improve the knowledge on the design space.

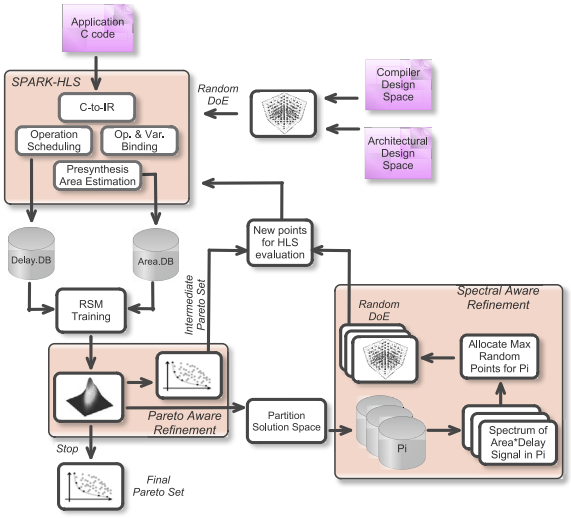

Fig. 1. Proposed exploration framework for coprocessor synthesis.

Fig. 1 shows the proposed HLS exploration framework. Given a behavioral specification (C code), the phases of intermediate representation (IR) generation, operation scheduling, operation and variable binding are performed for generating the coprocessor's RTL description. Then, during the presynthesis area estimation phase, the generated RTL description together with a gate-level precharacterized resource library are used as inputs for estimating the area complexity of the architectural solution. The design configurations passed to the HLS phases have been selected by: 1) a design of experiments (DoE) module and 2) a cooperative RSM-based Pareto- and spectral-aware refinement technique. This combined approach, representing the feedback loop of the framework, is the core of the proposed exploration strategy. The analytical approximation of the solution space is obtained by a RSM trained by using area and delay results of previously explored solutions. The RSM-based Pareto-aware refinement enables to selectively refine the solution set with points predicted as Pareto out of the entire design space, thus moving toward higher quality solutions. The spectral-aware refinement is also applied to the approximated solution space. However, the design space is incrementally partitioned at each iteration of the exploration strategy to identify high-variance regions. Then, design points located in high-variance regions of the design space are added to the set of points to be evaluated by the HLS tool to improve the prediction accuracy of the approximated solution space on the next iteration.

C. SPIRIT Algorithm

SPIRIT approach is described in detail in Algorithm 1. The product $it_{\max} \times nsynth_{\max}$ defines the budget constraint on HLS evaluations. First, the overall design space, **D**, is sampled with a uniform random distribution (line 2). A set of initial design configurations, **R**, is extracted from **D**. Each configuration is synthesized through HLS and the actual performance and area metrics are derived, providing an initial coarse view of the target design space (line 3). **F** represents the database containing all the information (configurations, area/delay metrics) about the configurations synthesized through HLS. Using **F** as training vectors, the RSM is trained by generating a prediction

Algorithm 1: Pareto and Spectral Aware Iterative Refinement Exploration Strategy

Input: Application source code: A
Input: Design-space definition: $\mathbf{D}$
Input: Number of iterations: $it_{\max}$
Input: Number of HLS evaluations per iteration: $nsynth_{\max}$
Input: Initial partition window size: $w_{\text{size}_{\max}}$
Input: Response-surface model: RSM
Output: Approximate Pareto set: $\mathbf{L}$

1 $nsynth \leftarrow N$
2 $\mathbf{R} \leftarrow$ RandomDoE($\mathbf{D}$, $nsynth$)
3 $\mathbf{F} \leftarrow$ HLS(A, $\mathbf{R}$)
4 $\mathbf{M} \leftarrow$ Train(RSM, $\mathbf{F}$)
5 $w_{\text{size}} \leftarrow w_{\text{size}_{\max}}$
6 $\{\mathbf{M}_i\} \leftarrow$ Windowing($\mathbf{M}$, w_{size}), $i \in \{0, \ldots, \frac{|\mathbf{M}|}{w_{\text{size}}} - 1\}$
7 **for** $it = 1$ **to** $it_{\max}$ **do**
8 $\mathbf{L} \leftarrow$ ParetoFilter($\mathbf{M}$)
9 $\mathbf{F}_L \leftarrow$ HLS($\mathbf{L}$)
10 $\mathbf{F} \leftarrow \mathbf{F} \cup \mathbf{F}_L$
11 $nsynth_{\text{pareto}} \leftarrow |\mathbf{L}|$
12 $nsynth_{\text{spectral}} \leftarrow nsynth_{\max}$ - $nsynth_{\text{pareto}}$
13 **foreach** $\boldsymbol{M}_i$, $i \in \{0, \ldots, \frac{|\boldsymbol{M}|}{w_{\text{size}}} - 1\}$ **do**
14 $\{P_k\} \leftarrow$ CalculatePowerSpectrum($\mathbf{M}_i$), $k \in 0 \ldots \frac{|\mathbf{M}_i|}{2}$
15 $Spectrum_i^{\text{pow}} \leftarrow \sum_{k=0}^{\frac{|\mathbf{M}_i|}{2}} P_k$
16 $TotalSpectrum^{\text{pow}} \leftarrow TotalSpectrum^{\text{pow}} + Spectrum_i^{\text{pow}}$
17 **end**
18 **foreach** $i \in \{0, \ldots, \frac{|\boldsymbol{M}|}{w_{\text{size}}} - 1\}$ **do**
19 $nsynth_i \leftarrow \lfloor nsynth_{\text{spectral}} \times \frac{Spectrum_i^{\text{pow}}}{TotalSpectrum^{\text{pow}}} \rfloor$
20 $\mathbf{R}_i \leftarrow$ RandomDoE($\mathbf{M}_i$, $nsynth_i$)
21 $\mathbf{R}_i \leftarrow$ EliminateDuplicate($\mathbf{R}$, $\mathbf{R}_i$)
22 $\mathbf{R} \leftarrow \mathbf{R} \cup \mathbf{R}_i$
23 $\mathbf{F}_i \leftarrow$ HLS($\mathbf{R}_i$)
24 $\mathbf{F} \leftarrow \mathbf{F} \cup \mathbf{F}_i$
25 **end**
26 $M \leftarrow$ Train(RSM, $\mathbf{F}$)
27 $w_{\text{size}} \leftarrow \frac{w_{\text{size}}}{2 \times it}$
28 $\{\mathbf{M}_i\} \leftarrow$ Windowing($\mathbf{M}$, w_{size}), $i \in \{0, \ldots, \frac{|\mathbf{M}|}{w_{\text{size}}} - 1\}$
29 **end**
30 $\mathbf{L} \leftarrow$ ParetoFilter($\mathbf{M}$)

archive $\mathbf{M}$ taken from the solution space, considering all the possible configurations annotated with the predicted metrics (line 4). $\mathbf{M}$ corresponds to the overall HLS signal, which it is partitioned in smaller windows, $\mathbf{M}_i$, according to the partition window size parameter, w_{size} (lines 5–6). At each iteration, the size of the partition window is incrementally refined toward smaller granularities, to enable more localized spectral analysis as the exploration proceeds (lines 27–28).

After acquiring a first prediction of the overall solution space, SPIRIT enters to the iterative refinement phase. The refinement is done on the predicted Pareto design points and on sample points from regions of the solution space that exhibit higher randomness. Regarding the Pareto-aware iterative refinement, we adopt an approach similar to [10] and [13], i.e., the predicted Pareto configurations are evaluated through HLS to acquire the actual metrics, $\mathbf{F}_L$ (line 9) and then inserted in the training database $\mathbf{F}$ for the next refinement of the predictive model $\mathbf{M}$ (line 10). Given the maximum number of HLS evaluations per iteration, $nsynth_{\max}$, and assuming $|\mathbf{L}|$ configurations to be included in the approximate Pareto set, the maximum number of configurations to be used for spectral-aware refinement, $nsynth_{\text{spectral}}$, is calculated (line 12). At each iteration, the $nsynth_{\text{spectral}}$ represents the overall budget of configuration samples to be allocated to the instantiated partition windows of the HLS solution space, $\mathbf{M}_i$. For each partition window, $\mathbf{M}_i$, the power spectrum coefficients, P_k $k \in 0 \ldots |\mathbf{M}_i|/2$, of the corresponding HLS signal are derived (line 14), thus enabling the calculation of the regional, $Spectrum_i^{\text{pow}}$, and total power spectrum of the HLS signal (lines 15–16).

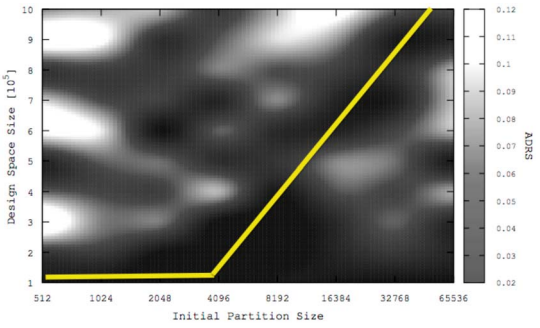

Fig. 2. Impact of $w_{\text{size}_{\max}}$ on ADRS with respect to the size of the design space.

The metric $Spectrum_i^{\text{pow}}/TotalSpectrum^{\text{pow}}$ reflects the contribution that the randomness of each HLS partition window $\mathbf{M}_i$ implies to the overall randomness of the HLS solution space. Since the higher the contribution the harder to predict is the HLS region, we use the aforementioned metric to guide the allocation of configuration samples, $nsynth_i$, in each HLS region (line 19). Each partition window $\mathbf{M}_i$ is randomly sampled with $nsynth_i$ configurations and the new synthesized configurations are inserted into the training database, $\mathbf{F}$ (lines 20–22). The Pareto- and spectral-aware training points of $\mathbf{F}$ are used for refining the RSM and generating the new prediction of the HLS solution space, $\mathbf{M}$ (line 26). The refined $\mathbf{M}$ is incrementally repartitioned according to w_{size} for the next iteration (line 28). At the end of the exploration, the final approximated Pareto set is derived for the designer (line 30).

D. SPIRIT Parameters Analysis and RSM Selection

We evaluate the impact of the: 1) $it_{\max}$ and 2) $w_{\text{size}_{\max}}$ parameters, by considering various problem sizes of the design space defined for the JPEG 2-D-DCT kernel. Specifically, we examine how the aforementioned parameters affect the exploration's accuracy in terms of the average distance from reference set (ADRS) [15] metric.

The analysis showed that, for problem sizes up to 200 K solutions, either a small or a large value of $it_{\max}$ delivers high ADRS values. Moving toward larger problem sizes, there is a benefit regarding ADRS for larger $it_{\max}$, due to the fact that a better refinement is performed. The same trend is also observed in Fig. 2 for the initial $w_{\text{size}_{\max}}$, i.e., the exploration of large design spaces benefits from larger initial partitions. A large $w_{\text{size}_{\max}}$ enables a uniform sampling of the design space to be performed during the first iterations of the algorithm, together with a more concentrated one during the last iterations. On the other hand, small $w_{\text{size}_{\max}}$ concentrates from the very beginning the distribution of HLS evaluations in specific regions of the solution space, increasing the possibility to leave large regions of the design space unexplored.

The RSM selection is based on the minimization of the accuracy error and on the maximization of the convergence behavior. Specifically, we considered six RSM models (four regression-based and two interpolation-based): 1) linear regression [16]; 2) spline-based regression [17]; 3) artificial neural networks (ANNs) [18]; 4) RFs [19]; 5) Shepard-based interpolation [20]; and 6) radial basis functions (RBFs) [21]. The RSMs have been validated by means of the Monte Carlo cross-validation procedures [22]. We trained the selected set of RSMs by using as training-set (known points) a variable number of random configurations, from 200 to 1800, and we evaluated the average normalized error, on both area and delay metrics, computed by the RSMs over the remaining configurations of the entire design space. To further improve the prediction accuracy,

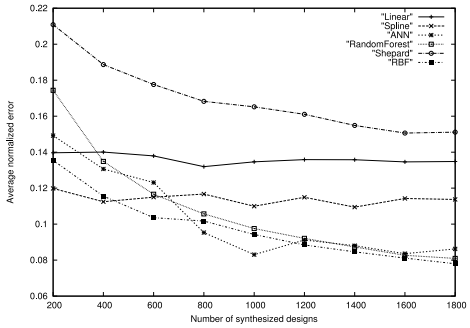

Fig. 3. Comparative prediction accuracy for the best configuration of the selected RSM methodologies.

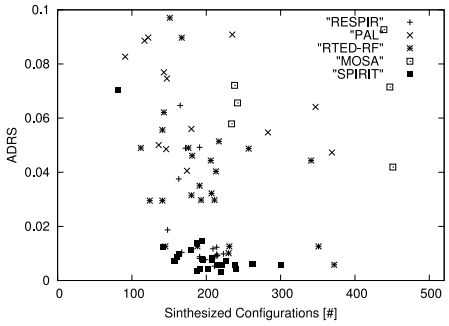

Fig. 4. Comparative results in the ADRS—synthesized configurations space.

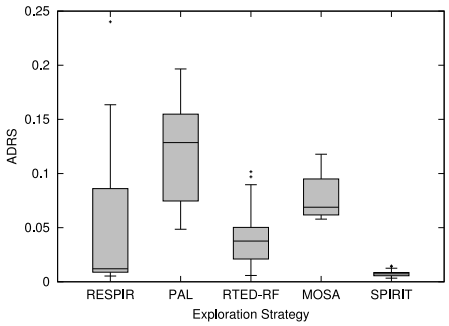

Fig. 5. Robustness of the exploration accuracy (ADRS) in the range of 130–260 synthesized configurations (0.25%–0.5% of the entire design space).

we applied the logarithmic Box-Cox power transform on each of the observed data [16] to reduce the data variation and improve correlation.

Fig. 3 shows the validation results considering the best configuration from each examined RSM with respect to estimation accuracy. For the given set of data, RBFs, RFs, and ANNs present the best estimation accuracy when increasing the size of the training data. However, RBFs always outperform the RFs, and also present a very consistent decrement of the validation error, making its behavior more stable with respect to the ANNs. Thus, our analysis can conclude that the RBF model is the most suitable RSM to be used in the proposed exploration strategy for the target problem. The reason behind this result can be found on the peculiar characteristic of the RBFs in performing an accurate multivariate interpolation even when data are distributed in a nonuniform manner [23].

III. Experimental Results

The SPARK tool [24] has been used as the HLS engine in our DSE framework. However, the proposed methodology is general enough and can be easily retargeted to other HLS tools. Our benchmark suite consists of eight real-life DSP kernels: 1) LMS adaptive filter; 2) 1-D DCT; 3) YUV to RGBA filter; 4) 4×4 matrix multiplication; 5) 2-D DCT kernel of JPEG; 6) 2-D inverse DCT of MPEG-2; 7) Gauss Blur kernel from cavity detector; and 8) 8×8 Sobel edge detection filter. The size of the examined design spaces (considering both compiler- and architectural-level parameters) ranges from 60 K to 1 M configurations, as defined in [8]. We evaluate the efficiency of the SPIRIT strategy in comparison to state-of-art meta-heuristics used for the HLS exploration problem, namely: 1) the multiobjective simulating annealing [25] (MOSA) [26]; 2) the response-surface Pareto iterative refinement [13] (RESPIR) [10]; 3) the Pareto-active learning algorithm [12] (PAL); and 4) the learning-base method presented in [11], where a randomized TED is coupled with random-forest as predictive model (RTED-RF). For a fair comparison, we run the corresponding explorations several times (at least 25) by varying, when available, the parameter configurations and repeating the initial sampling to avoid the possibility of a biased behavior.[1] We present results in an aggregated manner for the selected benchmarks, considering a unified design space across applications composed of 51 840 design alternatives.

[1]MOSA has been configured as follows: #epochs $\in$ {2, 18, 35}, epoch length $\in$ {10, 35, 60}, annealing coefficient 0.75. The Gaussian process models in PAL has been tuned by varying the epsilon parameter $\epsilon \in \{0.8...0.04\}$ with a step of 0.05 and initial training size of 50 points. SPIRIT and RESPIR have been configured by plugging in the RBF model combined with a Random DoE strategy for selecting the initial set of 50 points. Finally, in RTED-RF the initial TED has been applied to a selection of 5000 randomly selected designs to obtain the same initial set of 50 points.

Fig. 4 shows the accuracy in terms of ADRS (up to 10%) with respect to the exact Pareto-front versus the number of synthesized configurations. SPIRIT reaches an ADRS value up to 2% (and close to 0.5% in average) dominating almost completely all the other exploration strategies. For the same or even smaller number of synthesized configurations, SPIRIT delivers design solutions that are closer to the exact Pareto set than the other DSE strategies. Fig. 5 highlights this phenomenon showing the ADRS distribution by using a boxplot graph. The considered solutions are filtered in the area between 130 and 260 synthesized configurations (from 0.25% to 0.5% of the entire design space) thus capturing most of the exploration instances. It is evident that while RESPIR, RTED-RF, and SPIRIT are very close to the Pareto curve, the proposed strategy exhibits higher robustness. This is the result of the enhancement introduced by the spectral-aware refinement over the basic iterative refinement strategies within both RESPIR and RTED-RF. On average, RESPIR and RTED-RF are very accurate in finding the Pareto curve (average ADRS close to 1%), however, it is strongly dependent on the initial random sampling that sometimes can bring to local minima. This is a bit less evident on RTED-RF since it is supported by the initial RTED that helps to generate a better distribution of the initial configurations. However, RTED-RF suffers of a less accurate predictive model (RF) especially for a very low number of training configurations (Fig. 3). MOSA and PAL are less accurate than the other methods. The main reasons are that while MOSA is not supported by any predictive model, PAL requires several design evaluations for enabling the Gaussian Processes to predict an accurate point classification in large design spaces.

IV. Conclusion

In this paper, we addressed the problem of supervised HLS. We proposed, for the first time, the usage of spectral analysis for characterizing the solution space and how effectively can it be incorporated

within an HLS exploration framework to identify hard-to-predict regions. A novel exploration strategy has been developed by exploiting this information during the iterative optimization phase based on RSMs. Extensive experimentation showed that SPIRIT outperforms all the recent HLS-specific exploration heuristics proposed in literature.

REFERENCES

[1] Vivado, (2014). *XILINX*. [Online]. Available: http://www.xilinx.com/products/design-tools/vivado/

[2] (2014). *CyberWorkBench*. [Online]. Available: http://www.nec.com/global/prod/cwb/

[3] (2014). *Legup HLS*. [Online]. Available: http://legup.eecg.utoronto.ca/

[4] A. Gerstlauer *et al.*, "Electronic system-level synthesis methodologies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1517–1530, Oct. 2009.

[5] G. Wang, W. Gong, B. DeRenzi, and R. Kastner, "Exploring time/resource trade-offs by solving dual scheduling problems with the ant colony optimization," *ACM Trans. Design Autom. Electr. Syst.*, vol. 12, no. 4, 2007, Art. ID 46.

[6] B. C. Schäfer and K. Wakabayashi, "Design space exploration acceleration through operation clustering," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 1, pp. 153–157, Jan. 2010.

[7] S. Xydis, K. Pekmestzi, D. Soudris, and G. Economakos, "A high level synthesis exploration framework with iterative design space partitioning," in *VLSI 2010 Annual Symposium* (Lecture Notes in Electrical Engineering), vol. 105. Dordrecht, The Netherlands: Springer, 2011, pp. 117–131.

[8] S. Xydis, K. Pekmestzi, D. Soudris, and G. Economakos, "Compiler-in-the-loop exploration during datapath synthesis for higher quality delay-area trade-offs," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 1, pp. 11.1–11.35, Jan. 2013.

[9] A. C. Murray, R. V. Bennett, B. Franke, and N. P. Topham, "Code transformation and instruction set extension," *ACM Trans. Embedded Comput. Syst.*, vol. 8, no. 4, 2009, Art. ID 26.

[10] S. Xydis, G. Palermo, V. Zaccaria, and C. Silvano, "A meta-model assisted coprocessor synthesis framework for compiler/architecture parameters customization," in *Proc. Conf. Design Autom. Test Eur. (DATE)*, Grenoble, France, 2013, pp. 659–664.

[11] H. Liu and L. Carloni, "On learning-based methods for design-space exploration with high-level synthesis," in *Proc. ACM Design Autom. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–7.

[12] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel, "Active learning for multi-objective optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 28. Atlanta, GA, USA, 2013, pp. 462–470.

[13] G. Palermo, C. Silvano, and V. Zaccaria, "ReSPIR: A response surface-based pareto iterative refinement for application-specific design space exploration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 12, pp. 1816–1829, Dec. 2009.

[14] S. V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*. Chichester, U.K.: Wiley, 2006.

[15] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," in *Proc. Congr. Evol. Comput.*, Canberra, ACT, Australia, 2003, pp. 878–885.

[16] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil, "Construction and use of linear regression models for processor performance analysis," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, Austin, TX, USA, Feb. 2006, pp. 99–108.

[17] B. C. Lee and D. M. Brooks, "Spatial sampling and regression strategies," *IEEE Micro*, vol. 27, no. 3, pp. 74–93, May/Jun. 2007.

[18] S. Fahlman, D. Baker, and J. Boyan, "The cascade 2 learning architecture," Comp. Sci. Dept. CMU, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. TR CMU-CS-TR-96-184, 1996.

[19] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[20] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proc. 23rd ACM Nat. Conf. (ACM)*, New York, NY, USA, 1968, pp. 517–524.

[21] M. J. D. Powell, "The theory of radial basis functions," in *Advances in Numerical Analysis II: Wavelets, Subdivision, and Radial Basis Functions*, W. Light, Ed. New York, NY, USA: Oxford Univ. Press, 1992, pp. 105–210.

[22] R. Picard and D. Cook, "Cross-validation of regression models," *J. Amer. Stat. Assoc.*, vol. 79, no. 387, pp. 575–583, 1984.

[23] D. Lazzaro and L. B. Montefusco, "Radial basis functions for the multivariate interpolation of large scattered data sets," *J. Comput. Appl. Math.*, vol. 140, nos. 1–2, pp. 521–536, Mar. 2002.

[24] S. Gupta, N. Dutt, R. Gupta, and A. Nicolau, "Coordinated parallelizing compiler optimizations and high-level synthesis," *ACM Trans. Design Autom. Electron. Syst.*, vol. 9, no. 4, pp. 441–470, 2004.

[25] K. I. Smith, R. M. Everson, J. E. Fieldsend, C. Murphy, and R. Misra, "Dominance-based multiobjective simulated annealing," *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 323–342, Jun. 2008.

[26] B. C. Schäfer, T. Takenaka, and K. Wakabayashi, "Adaptive simulated annealer for high level synthesis design space exploration," in *Proc. Int. Symp. VLSI Design Autom. Test (VLSI-DAT)*, Hsinchu, Taiwan, Apr. 2009, pp. 106–109.