Towards Generic and Scalable Word-Length Optimization

Van-Phu Ha, Tomofumi Yuki, Olivier Sentieys

Towards Generic and Scalable Word-Length Optimization

Van-Phu Ha
Univ Rennes, Inria, IRISA
Rennes, France
van-phu.ha@inria.fr

Tomofumi Yuki
Inria, Univ Rennes, IRISA
Rennes, France
tomofumi.yuki@inria.fr

Olivier Sentieys
Univ Rennes, Inria, IRISA
Rennes, France
olivier.sentieys@inria.fr

***Abstract*—In this paper, we propose a method to improve the scalability of Word-Length Optimization (WLO) for large applications that use complex quality metrics such as Structural Similarity (SSIM). The input application is decomposed into smaller kernels to avoid uncontrolled explosion of the exploration time, which is known as noise budgeting. The main challenge addressed in this paper is how to allocate noise budgets to each kernel. This requires capturing the interactions across kernels. The main idea is to characterize the impact of approximating each kernel on accuracy/cost through simulation and regression. Our approach improves the scalability while finding better solutions for Image Signal Processor pipeline.**

***Index Terms*—Fixed-Point Refinement, Word-length Optimization, Noise Budgeting, Multiple Kernel Optimization**

I. Introduction

Fixed-Point arithmetic is widely used for implementing Digital Signal Processing (DSP) systems on electronic devices. Since initial specifications are often written using floating-point arithmetic, conversion to fixed-point is a recurring step in hardware design. The primary objective of this conversion is to minimize the cost (energy and/or area) while maintaining an acceptable level of quality at the output. This process, called Word-Length Optimization (WLO), is a time consuming process taking up to $25-50\%$ of design time [1].

In WLO, each variable/operator may be assigned a different fixed-point encoding, which means that the design space grows exponentially as the number of variables increases. This is especially true when targeting hardware accelerators implemented in FPGA or ASIC. Thus, most approaches for WLO involve heuristic search algorithms [2]–[11]. A key component in such search algorithms is the evaluation of output quality. There are two broad evaluation categories: simulations and analytical models. Simulation-based methods suffer from scalability issues as the number of required simulations, as well as the simulation time, increase drastically with the size of the application [2], [3], [5], [6], [8]–[10]. Methods based on analytical models scale well, but are limited by the ability to construct adequate models. Existing techniques are limited to modeling noise power metrics of Linear and Time-Invariant (LTI) systems (with some extensions) [4], [7], [11].

One technique to address scalability issues, called *noise budgeting*, decomposes an application into smaller chunks, or kernels, and assigns separate quality constraints, called *noise budgets*. The smaller sub-problems can be explored much faster, at the cost of ignoring possible inter-kernel interactions. The allocation of noise budgets plays a critical role in this technique, as it is the only parameter that indirectly captures the inter-kernel interactions. However, there is still little work on how to find good allocations of the noise budgets. An existing approach [11] makes heavy use of analytical models, making it difficult to support quality metrics other than noise power and its variants. More complex quality metrics, such as Structual Similarily (SSIM) used for images or Objective Degradation Grade (ODG) used for audio, do not directly correlate with noise power, and are much harder to model.

In this paper, we propose a WLO method to address both scalability (in simulations) and generality (in analytical models). The key in our work is capturing the interactions between approximations applied to a kernel with its cost and approximations applied to other kernels through *empirically* constructed models. Since the models are constructed through simulations, our approach can be used for any quality metric. The constructed models are then used to predict the best allocation of noise budgets. We show that the predicted noise budgets give better solutions than those found by WLO on the whole program (without decomposition) and that it can be used for both Peak Signal to Noise Ratio (PSNR) and SSIM.

II. Background and Related Work

In this section, we introduce WLO and discuss earlier work.

A. Word-Length Optimization

Fixed-point representation contains two parts, integer word-lengths and fractional word-lengths (WL). The integer WL is closely related to dynamic range; the fractional WL controls the precision. In this work, we focus on the WLO for fractional WL, which is the time consuming part of the exploration. The main objective of WLO is to determine a WL configuration that minimizes cost while satisfying quality constraints.

Let $\boldsymbol{W}$ denote a WL configuration. Then the WLO problem is formulated as the following:

$$\min \boldsymbol{C}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W}) \geq \boldsymbol{\lambda}_{obj} \tag{1}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are functions that express cost and quality, respectively, and the target quality is given as $\boldsymbol{\lambda}_{obj}$. How the functions $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are realized varies across work, ranging from simulations to analytical models.

A direct approach to obtain the optimal solution is exhaustive search [12]. However, it is only feasible for small kernels due to exponential growth in possible WL configurations as the number of variables increases.

Many approaches [2], [3], [5], [6], [8]–[10] were proposed based on iterative search using heuristics to address WLO. Most approaches use variants of gradient decent algorithms that evaluate neighboring solutions, typically constructed by changing one (or a few) variables in the current solution, at each iteration. Since the number of neighboring solutions increases as the number of variables increases, the number of solutions that must be evaluated during the iterative search quickly increases. This is the main reason why simulation-based approaches suffer from scalability issues.

Some approaches construct analytical models of noise power [4], [7], [11] to avoid costly simulations during the exploration. These analytical approaches take advantage of a property of errors under linear systems that their propagation do not interfere with each other. Hence, the error introduced at a noise source may be propagated through the system independently and aggregated afterwards. Thus, these approaches cannot be directly extended to handle general programs. Moreover, complex quality metrics, such as SSIM, do not have a direct relationship with noise power.

B. Noise Budgeting

Noise budgeting [11], [13], [14] is a technique to address the scalability of WLO. It decomposes a problem into smaller sub-problems that takes less time to solve, and combines the solutions to sub-problems to form the final solution.

Consider an application that is decomposed into N kernels. The WLO is now formulated as

$$\min \sum_{i=1}^{N} \boldsymbol{C}(\boldsymbol{W_i}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W_i}) \geq \boldsymbol{\lambda_i}, \tag{2}$$

where the WL configuration and the constraint (*noise budget*) for the i-th kernel are denoted as $\boldsymbol{W_i}$ and $\boldsymbol{\lambda_i}$, respectively. The above decomposition gives N smaller sub-problems (subsets of the above for each i may be solved independently), limiting the explosion in number of configurations to explore.

How the quality of each kernel $\boldsymbol{\lambda}(\boldsymbol{W_i})$ is computed, may depend on the approach. In our work, we define the quality of a kernel considered in isolation as the quality of the application output when all other kernels are not approximated (i.e., computed with floating-point).

The main challenge in noise budgeting is in the allocation of the budgets. Decomposition into sub-problems makes it impossible to capture the possible interactions spanning multiple kernels. For example, errors introduced at a kernel may be masked (or amplified) at a later kernel, which affects how much loss in quality in the former kernel is tolerated. Moreover, such interactions make it difficult to tell if the solutions to the sub-problems using noise budgets would satisfy the constraint on application output when the individual solutions are combined.

Parashar *et al.* [11] use analytical models to capture the inter-kernel interactions, and solve problem (2) for the optimal allocations of noise budgets. However, this approach cannot be easily extended to general programs and/or sophisticated quality metrics due to the difficulty of constructing analytical models as discussed in Section II-A.

Another body of work uses similar decomposition into sub-problems, but does not allocate noise budgets [13], [14]. In these work, local WLO is first performed for each kernel to identify designs that expose different quality-cost trade-off. Then, the combinations of the local solutions are explored to find the global solution. The main limitation of this approach is that the final solution space is restricted by the initial local WLO for the kernels. The number of design points available at each kernel is proportional to the amount of time spent on local WLO, and it is difficult to know *a priori* if the "right" design for the global combination has been found. Thus, there is a risk of missing important designs (when local search is too coarse) or having scalability issues beyond more than a few kernels (when local search is too fine). Our approach addresses these limitations by allocating the noise budgets using models constructed after a handful of local WLOs.

III. Approach Overview

Recall the decomposed WLO problem in (2). We are interested in modeling the following functions:

- $\hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i})$: Cost of kernel K_i as a function of the quality constraint at its output.
- $\hat{\boldsymbol{\lambda}}(\boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_N})$: Application output quality as a function of quality constraints at each kernel.

These functions enable the optimal choice of noise budgets to be formulated as:

$$\min \sum_{i=1}^{N} \hat{\boldsymbol{C}}_{\boldsymbol{i}}(\boldsymbol{\lambda_i}) \quad \text{Subject to} \quad \hat{\boldsymbol{\lambda}}(\boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_N}) \geq \boldsymbol{\lambda_{obj}} \tag{3}$$

The functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}$ model the impact of approximating each kernel K_i to cost. This allows us to identify the kernel that gives best savings in cost for some loss in quality. However, how the individual approximations in the kernels interact must be taken into account to determine the impact on the application output. The function $\hat{\boldsymbol{\lambda}}$ models this behavior to optimize the budget allocation.

Our approach empirically constructs the functions $\hat{\boldsymbol{C}}_{\boldsymbol{i}}$ and $\hat{\boldsymbol{\lambda}}$ to obtain optimal allocation of noise budgets as described above. The overview of our approach is as follows:

1) For each kernel K_i, perform a few WLOs with different quality constraints. The Pareto-optimal designs found by these explorations are used as data points to construct the models ($\hat{\boldsymbol{C}}_{\boldsymbol{i}}$).
2) Run simulations for combinations of the designs used above to evaluate the accuracy of combined solutions. These simulations provide the data to construct the model of inter-kernel interactions ($\hat{\boldsymbol{\lambda}}$).
3) Solve (3) for noise budgets.

4) For each kernel K_i, perform a single WLO with the obtained noise budgets. The result of these local WLOs are combined to obtain an initial solution.

One significant advantage of the proposed method is that it is parametric to the WLO algorithm, and how the cost is evaluated. The choice of WLO algorithms strongly influences the quality of designs found; the Pareto frontier is only among the designs explored and is potentially far from the true optimal. Similarly, the cost models used influence the design space and how efficiently an algorithm can explore this space. Thus, the constructed models are specialized for designs that can be found by the given combination of cost model and WLO algorithm. If another model/algorithm is to be used, a separate set of models must be constructed.

IV. Model Construction

In this section, we describe how we construct the models, which is at the core of our approach. We first present how the data points are collected, followed by a description of polynomial fitting we use to construct models.

A. Data Points for Cost Function ($\hat{C}$)

For each kernel K_i, a model of its cost as a function of target quality ($\hat{C}_i$) is constructed. The data points used to construct these models are collected by performing WLO of the kernel using different target quality constraints. Among the designs explored by the WLO runs, we use only those in the Pareto-frontier. This is because we are interested in the best designs (for each accuracy) found by the WLO algorithm in use.

It is necessary to perform multiple instances of WLO, especially when the algorithm is not a greedy search. This is because the designs near the target quality will be explored in detail, finding much better designs only for the region of interest. Figure 1 shows the designs explored during three WLO runs targeting different quality constraints with one of the kernels from our benchmark detailed in Section V. In these explorations we used Tabu search [10], which first performs greedy gradient descent to reach the target quality, followed by local search to minimize cost. It is clearly visible in the figure that the local search significantly reduces the cost without affecting quality. Thus, it is important that we run multiple instances of WLO to gather useful data points to accurately model the relationship between quality constraint and cost.

The design space is explored using three target quality constraints to collect the data points. Given a target quality constraint $\boldsymbol{\lambda}_{obj}$ (for the original problem), and the quality without approximation $\boldsymbol{\lambda}_{max}$, we target the following: $\boldsymbol{\lambda}_{obj}+\delta$, $\boldsymbol{\lambda}_{max}-\delta$, and $0.5(\boldsymbol{\lambda}_{obj}+\boldsymbol{\lambda}_{max})$ where $\delta = 0.1(\boldsymbol{\lambda}_{max}-\boldsymbol{\lambda}_{obj})$. In other words, we target the boundaries and the mid-point under consideration. These targets are motivated by the fact that polynomial fitting over an interval works better when the approximation nodes follow a Chebyshev-like distribution [15], with more nodes towards the boundaries.

The offset δ slightly moves the target constraints inwards. This is because $\boldsymbol{\lambda}_{max}$ is not a realistic target with fixed-point approximation, and because $\boldsymbol{\lambda}_{obj}$ is usually not an appropriate target for any kernel. If a kernel aggressively approximates to the extent that the quality is reduced to $\boldsymbol{\lambda}_{obj}$ by its effect alone, then the combination of such a design with even slight approximations in other kernels usually does not satisfy $\boldsymbol{\lambda}_{obj}$. As an example, if all kernels in an image processing pipeline satisfy an accuracy of SSIM=0.9, it is unlikely that the whole application would have an SSIM=0.9 at its output.

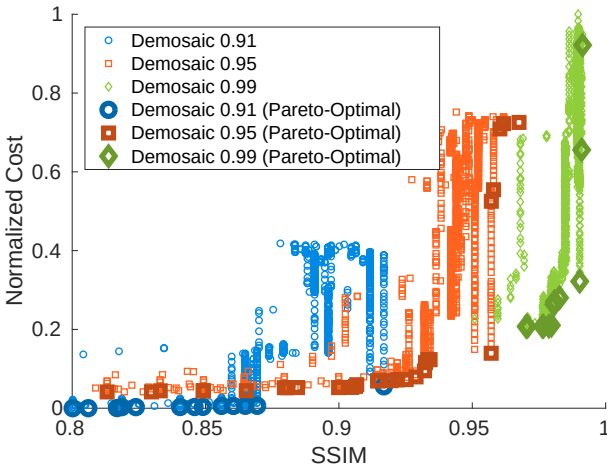

Fig. 1: All explored solutions of three WLO runs for a kernel (Demosaic) in our benchmark. The number in the legend indicates the quality constraint (SSIM) targeted for each run. The Pareto-optimal points (within each WLO run) are emphasized in the plot. More details about the benchmark and the cost are in Section V.

B. Data Points for Quality Function ($\hat{\boldsymbol{\lambda}}$)

The quality function $\hat{\boldsymbol{\lambda}}$ models the output quality of the combined solutions as a function of output qualities under isolated approximations. Thus, simulations of combined solutions are necessary to collect sufficient data points.

In our current implementation, we take a subset of the data points collected for modeling cost functions, and simulate all combinations to collect the data points. We consider X uniform segments of the quality in the range under consideration, and take the best design within each segment. Then the X^N combinations are simulated to collect data points. (We use $X = 5$ in our evaluation.)

This step may be optimized by having a more sophisticated method to select the subset. An initial model may be constructed by using a few samples from each kernel and simulating their combinations. Then, further simulations that would provide important data points may be predicted using the initial model. This process may be repeatedly applied until the improvement in accuracy of the model starts to diminish. The current implementation does not use this optimization.

C. Polynomial Fitting

The functions $\hat{C}_i$ and $\hat{\boldsymbol{\lambda}}$ are constructed with the data points collected using polynomial fitting. These models are expected to be non-linear, especially for complex quality

metrics and/or non-linear systems, motivating the use of non-linear regression. We use polynomial fitting since it does not require a model (template) to be designed. We use Gaussian Process (GP) [16] regression to learn the cost models ($\hat{C}_i$) and least squares for the accuracy model ($\hat{\boldsymbol{\lambda}}$). We use a squared exponential covariance function as the kernel for GP. The least squares polynomial fit degree is manually selected by trying a range of values.

Most of the tuning effort for these models comes from preprocessing of the data. The size of the training data is relatively small, and there is no precise control over its distribution. These limitations make the regression analysis susceptible to common pitfalls (over-fitting, oscillation). We apply the following to fine-tune the models:

- obvious outliers are removed using our insights about the WLO algorithm, and
- designs within $x\%$ of the Pareto frontier are included in training data, where x is selected for each kernel.

Such tuning effort is necessary to improve the model quality. How to fully automate this process is a separate subject on its own, which we do not discuss here. In our evaluation, manual tuning time was within several minutes. The models constructed for our benchmarks are discussed in Section V-C.

V. Evaluation

In this section, we evaluate our approach against global WLO, where all variables are concurrently considered for WLO, by comparing exploration time and quality of solutions.

A. Experimental Setup

All experiments are performed on Linux machine with 2x4 cores Intel Xeon E5640 at 2.67GHz and 4GB memory. Recall that there are two parameters to our approach: WLO algorithm, and cost model. In our evaluation, we have used Tabu search [10], which is a heuristic search algorithm based on gradient descent, to perform WLO. We use an energy model as our cost model. The energy model counts the number of operations performed by each operator, and calculates the total cost based on the energy consumption of an operation. The energy per operation is empirically gathered from several ASIC synthesis, simulation and power estimation for different WLs. An operator is characterized by the WLs of the operands, the WL of the result, and the arithmetic operation performed. Characterization is performed using Synopsys Design Compiler and Prime Time using a 28nm technology.

Our approach is implemented using GeCoS (https://gitlab.inria.fr/gecos/gecos-float2fix) [17], an open-source compiler framework. The WLO is performed at the source-level, where the variables define the granularity of the exploration. We use a set of variables per loop nest so that each loop can run with its own WL configurations. In addition, some of the variables in the code are forced to have the same format, since they are aliases of each other. We use the number of *effective variables*, i.e., the number of individual fixed-point formats being explored, as a measure of the complexity.

We use the constraint solver from Matlab optimization toolbox to solve for the noise budgets. In our experiments, the optimizer returned a solution almost instantaneously.

Each kernel is explored once with the same WLO algorithm using the derived noise budgets. The solutions are then combined to form the final solution. The combined solutions may sometimes overshoot or undershoot the target quality slightly due to inaccuracies in our models. If the quality constraint was not met, we perform a greedy search to find the nearest design that satisfies the constraints. This calibration step takes less than 2% of the total time.

B. Image Signal Processor

We use Image Signal Processor (ISP) to evaluate our approach. This application takes raw data from camera sensors, and applies a sequence of processing kernels to produce a color image. It is an interesting benchmark because it has various filters that naturally serves as kernels, and its primary quality metric is SSIM. We target up to four stages of the ISP pipeline; additional kernels make exploration time for global WLO too long. The four kernels are:

- *NLM*: Non-Local Means denoising filter [18]. This denoising stage takes means of 5 windows, weighted by the distance from the target pixel, to filter noise.
- *Demosaic*: Demosaicing reconstructs a full color image from sensor data that captures color information as a mosaic of primary colors. This stage is also expensive consisting of 7 filters of size 3×3 or 5×5.
- *GC*: Gamma Correction adjusts the brightness of the image to suit human eyes. The image is converted into gray scale to derive the amount of brightness correction, which is then applied to the image.
- *Unsharp*: Unsharp masking sharpens the image by masking it with its blur. It computes a blurred image with a two-pass (vertical + horizontal) filter to use as the mask. It also includes the conversion to/from `YCbCr` color space, since the filter is performed in `YCbCr`.

The number of effective variables to be optimized are 19 for *NLM* and *Demosaic* and 17 for *GC* and *Unsharp*. To evaluate our approach in solving the scalability problem, we create three experiments on ISP with increasing complexity for WLO. Depending on each case, some kernels are selected for WLO while others are kept at highest possible precision. In the first experiment, *NLM* and *Unsharp* are chosen for WLO. In the second experiment, the *Demosaic* kernel is added. The last experiment considers all four kernels.

We use two SSIM targets in our experiments: 0.9 and 0.99. Preserving SSIM = 0.99 is considered to have small impact on human perception, which is suitable for photos. To evaluate our approach, we also target SSIM = 0.9, which has a much larger solution space compared to the 0.99 case and which is representative of, e.g., video capturing.

C. Empirically Constructed Models

It is difficult to evaluate these models since simple metrics, such as (Root) Mean Square Error, are not sufficient to

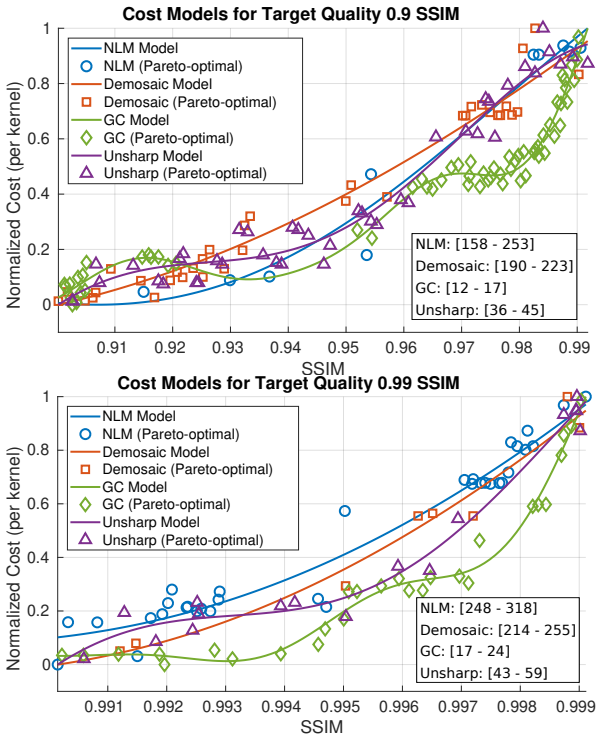

Fig. 2: Cost models constructed for each kernel using Gaussian Process. The costs are individually normalized for each kernel to present the models within a figure. The ranges of unnormalized energy cost (nJ) are shown at the bottom right.

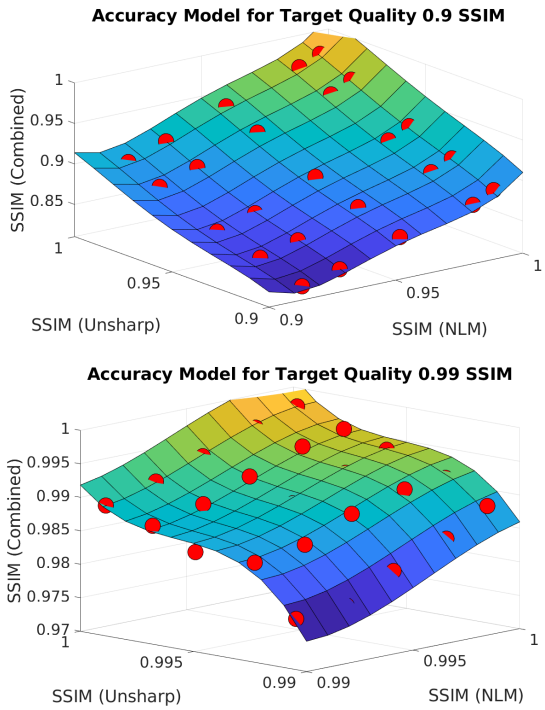

Fig. 3: Models for inter-kernel interaction of quality constraints. These models were constructed with least square fit with degree four and three polynomials for 0.9 SSIM and 0.99 SSIM cases, respectively.

TABLE I: Comparison of solutions for ISP.

		Target 0.9 SSIM		Target 0.99 SSIM	
		SSIM	Cost (nJ)	SSIM	Cost (nJ)
2 Kernels	GWLO	0.901	212	0.990	326
	Ours	0.915	207	0.991	309
3 Kernels	GWLO	0.900	438	0.990	604
	Ours	0.906	427	0.990	587
4 Kernels	GWLO	0.901	474	0.991	695
	Ours	0.907	444	0.990	612

assess their quality with respect to unseen data. In this work, these models are ultimately evaluated by the quality of the noise budgets derived. Figures 2 and 3 present the models constructed for ISP (excluding those that are hard to visualize) as a partial evaluation of their quality.

D. Exploration Time and Quality of Solution

Figure 4 and Table I summarize the exploration of GWLO and our approach for ISP. The model construction time is significant due to the time consuming WLO algorithm, and hence our approach takes longer for smaller problems. However, the difference in scalability becomes clear with more kernels.

The quality of the solutions improves for large problem instances. The explanation is that our approach performs local search around the derived noise budgets, finding better solutions (as explained with Figure 1). There are many local minima in the design space, and GWLO becomes more likely to be stuck with sub-optimal solutions for larger problems.

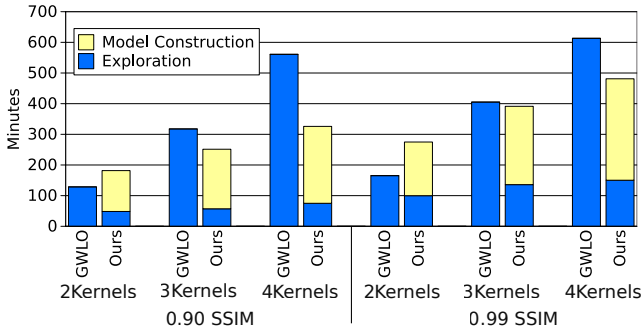

Fig. 4: Comparison of exploration time for ISP.

We have also compared our solutions to those that could be found without empirical models. A number of combined solutions are simulated during the accuracy model construction, which may already include a good design. In such cases, there is no need to perform further exploration, i.e., finding these solutions take the same time as model construction in our approach. We observed that for some accuracy targets, this is indeed the case. For some other accuracy targets, shown in Figure 5, more than 15% improvement in cost may be realized by using the derived noise budgets. These are instances that support our claim in Section II-B.

E. FIR and IIR Filters

We have also applied our approach with cascaded FIR and IIR filters to test how it works for another quality metric and

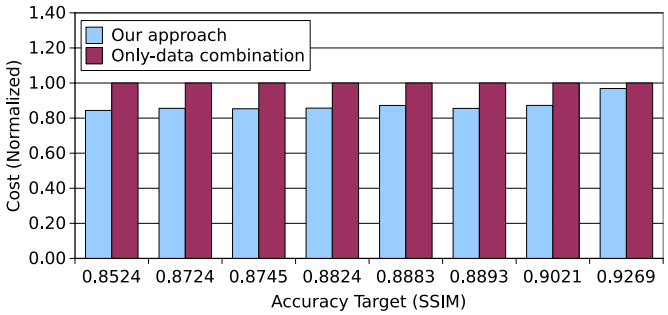

Fig. 5: The quality of solutions by our approach compared to the best combination of the configurations used for accuracy model construction. These results are for ISP with 4 kernels.

TABLE II: Comparison of solutions for FIR and IIR.

		Target 50 dB		Target 60 dB	
		PSNR (dB)	Cost (nJ)	PSNR (dB)	Cost (nJ)
FIR	GWLO	50.0	64.15E-04	60.1	72.95E-04
	Ours	50.2	66.94E-04	60.6	73.76E-04
IIR	GWLO	50.2	9.22E-04	60.1	10.66E-04
	Ours	50.6	9.60E-04	60.7	10.91E-04

linear systems. FIR is decomposed into 2 kernels with 9 and 6 effective variables. IIR is partitioned into 3 kernels with 12, 12, and 7 variables. We used PSNR equal to 50 dB and 60 dB as quality targets. The cost of solutions found by GWLO and our approach are shown in Table II. The cost of our solutions were slightly worse than that of GWLO's solutions, with about 4% for FIR and 2% for IIR. The solution space of FIR and IIR is considered as relatively small and with few sub-optimal solutions compared to ISP. Thus, with considering all kernels during the optimization time, GWLO can find a good solution and avoid local minima. Figure 6 summarizes the exploration time between GWLO and our approach for FIR and IIR with constraints of 50 dB and 60 dB. The important result is that the exploration time of our approach scales much better than GWLO, and that the overall behavior is consistent with ISP.

VI. Conclusion

In this paper, we propose the use of empirically constructed models to solve the generality and scalability problems of WLO in large applications. The key idea in our approach is to characterize the impact of approximating each kernel to accuracy/cost through an empirical model. We show that for sufficiently large applications that justify the time spent on modeling, our approach can significantly reduce exploration time *and* improve the quality of the solutions.

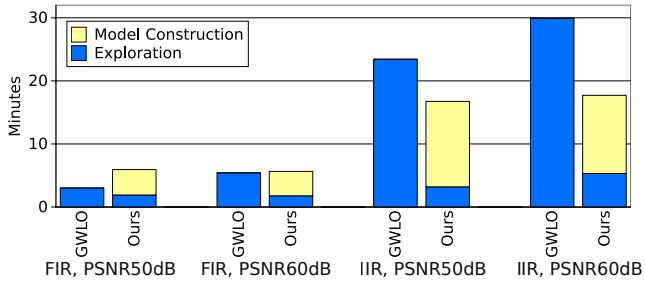

Fig. 6: Comparison of exploration time for filter applications.

Acknowledgment

This work was supported in part by the French National Research Agency; ARTEFaCT project (ANR-RF-2015-01). We thank Silviu-Ioan Filip for his help with regression techniques.

References

[1] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger, "Accelerating fixed-point design for MB-OFDM UWB systems." https://www.design-reuse.com/articles/9559/, 2005.

[2] T. Arslan and D. H. Horrocks, "A genetic algorithm for the design of finite word length arbitrary response cascaded iir digital filters," in *Proceedings of the Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 276–281, 1995.

[3] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *Proceedings of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 53–56, 2001.

[4] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 4, pp. 845–854, 2007.

[5] H. Choi and W. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proceedings of 1994 IEEE Workshop on VLSI Signal Processing*, pp. 198–207, 1994.

[6] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.

[7] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.

[8] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proceedings of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 290–293, 2001.

[9] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[10] H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *Proceedings of the 19th European Signal Processing Conference*, pp. 1944–1948, 2011.

[11] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 638–645, 2013.

[12] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[13] J. Chung and L.-W. Kim, "Bit-width optimization by divide-and-conquer for fixed-point digital signal processing systems," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3091–3101, 2015.

[14] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *Proceedings of the IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 18–23, 2013.

[15] L. N. Trefethen, *Approximation theory and approximation practice*. SIAM, 2013.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2005.

[17] A. Floc'h *et al.*, "Gecos: A framework for prototyping custom hardware design flows," in *IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 100–105, 2013.

[18] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2 of *CVPR'05*, pp. 60–65, June 2005.