Knowledge Transfer in Automatic Optimisation of Reconfigurable Designs

Maciej Kurek, Marc Peter Deisenroth, Wayne Luk and Timothy Todman

***Abstract*—This paper presents a novel approach for automatic optimisation of reconfigurable design parameters based on knowledge transfer. The key idea is to make use of insights derived from optimising related designs to benefit future optimisations. We show how to use designs targeting one device to speed up optimisation of another device. The proposed approach is evaluated based on various applications including computational finance and seismic imaging. It is capable of achieving up to 35% reduction in optimisation time in producing designs with similar performance, compared to alternative optimisation methods.**

I. Introduction

Previous research on automatic optimisation of reconfigurable designs involves optimising both design parameters and Computer Aided Design (CAD) tool parameters. Machine learning can be used to tune CAD tools or design parameters for faster optimisation [1], [2]. Bayesian optimisation can be used to treat noise in benchmark outputs [3], allowing parallelism to speedup optimisation time. Yet, no matter how refined, all of the above approaches are wasteful: When an optimisation finishes, the gathered knowledge is discarded. Manually embedding designers knowledge within an optimisation algorithm is a possibility [4]. However, how can we directly use insight, actionable information derived from previous designs, to optimise new designs? Our solution, presented in Figure 1, is to capture the re-usable information from design synthesis and benchmark outputs for optimising future designs with our novel Auto-Transfer algorithm. The algorithm is based on knowledge transfer, which is related to transfer learning [5]. Transfer learning tackles the problem of transferring knowledge from one problem onto another. The difference between our approach and transfer learning is that we only transfer old data. In transfer learning, the knowledge learned, such as optimised hyperparameters, is also transferred. Our contributions include:

- Statement of the problem of knowledge transfer in reconfigurable design optimisation, inspired by related work [3], [6]. (Section III)
- Presentation of an Auto-Transfer algorithm for knowledge transfer in automatic optimisation of reconfigurable designs. The Auto-Transfer algorithm is based on Bayesian optimisation [3]. (Section IV)
- Evaluation of the Auto-Transfer algorithm using two case studies: a quadrature design for financial computation [7] and a seismic imaging design [8]. (Section V)

II. Background

Developing reconfigurable designs usually starts with coding in a language which can be compiled into hardware descriptions. This is followed by design parameterization, along with constraint and optimisation goal specifications. Lastly, either a set of analytical models are constructed and the design is manually optimised [7], [8] or an automatic tool is used [1], [2], [3]. The manual approach the has advantage of being tailored to a particular problem and making use of designer's experience, while a tool has the advantage of being generic and automated. The main problem in automatic optimisation of reconfigurable designs is hardware generation time. Bayesian optimisation addresses this problem by modeling the target fitness using a Gaussian Process (GP) [9], and has shown promise in the context of reconfigurable designs [3]. In case of GP regression, the target function $f : \mathcal{X} \to \mathbb{R}$ is used to obtain noisy observations y. The goal is to obtain for an unseen point $\mathbf{x}_* \in \mathcal{X}$ the predictive distribution of the modeled function $\mathrm{p}(f|\mathbf{x}, \mathbb{X}, \mathbb{y}, \theta)$, where θ are GP models hyperparameters and the observed data is $\mathbb{X} = \{\mathbf{x}_i\}_1^n$ and $\mathbb{y} = \{y_i\}_1^n$. The designer chooses a suitable model, and its hyperparameters are typically optimised by maximizing the marginal likelihood [9].

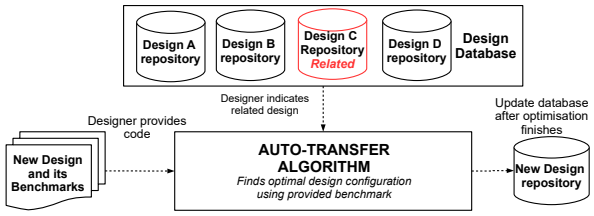


Fig. 1: Knowledge transfer approach.

The advantage of GP over probabilistic regression techniques in the global optimisation context is the prediction uncertainty encapsulated in the predictive distribution. In Bayesian optimisation, this information is used to define an acquisition function $a : \mathcal{X} \to \mathbb{R}$, which is used to determine designs to evaluate. It automatically manages exploration and exploitation of the parameter space. Exploration happens when areas of high uncertainty are evaluated, and exploitation when areas with high expected utility are evaluated. Parallel asynchronous Expected Improvement (EI) can be used as an acquisition function, $E[I^{(\mu,\lambda)}(\mathbf{X}_\lambda)]$ [10]. At any given time μ nodes are busy and λ nodes are idle. The goal is to find a set of designs $\mathbf{X}_\lambda = \{\mathbf{x}_i\}_{i=1}^{\lambda}$ with highest expected improvement $I^{(\mu,\lambda)}(\mathbf{X}_\lambda)$ over the currently best found solution and to evaluate them using the idle nodes, while μ designs are being evaluated on the busy nodes. Automatic Reconfigurable Design Efficient Global Optimization (ARDEGO) uses a constrained acquisition function $E[I_{\mathfrak{v}}^{(\mu,\lambda)}(\mathbf{X}_\lambda)]$ [3], based on the constrained improvement function:

$$I_{\mathfrak{v}}^{(\mu,\lambda)}(\mathbf{X}_\lambda) = \begin{cases} I^{(\mu,\lambda)}(\mathbf{X}_\lambda), & \forall \mathbf{x}_i \in \mathbf{X}_\lambda : d(\mathbf{x}_i) = 0 \\ 0, & \text{otherwise} \end{cases},$$

where the decision function $d : \mathcal{X} \to \mathcal{T}$ predicts constraint failures, and $d(\mathbf{x}_i) = 0$ indicates a valid design. A support vector machine (SVM) classifier is suggested to build d [3], trained using $\mathbb{X}$ and the target labels $\mathbb{t}$, where $\mathbb{t} = \{t_i\}_1^n$.

III. Problem Statement

The problem of optimizing reconfigurable design can be considered as the optimisation of noisy black box functions defined over the design's parameter space $\mathcal{X}$. The parameter space is a discrete metric space with D parameters $\mathcal{X} \subseteq \mathbb{R}^D$. A design configuration $\mathbf{x}$ resides in the design parameter space $\mathbf{x} \in \mathcal{X}$. A benchmark function $f : \mathcal{X} \to \mathbb{R}$ is used to obtain noisy observations of design performance $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, {\sigma_n}^2)$ is Gaussian i.i.d. noise with variance σ_n^2. A reconfigurable deign can fail to meet various constraints, like accuracy, timing or resource limitations. Constraint violation is encoded in an exit code using the constraint function $c : \mathcal{X} \to \mathcal{T}$, where $\mathcal{T}$ is the set of possible exit codes. For example, exit code $t = 0$ indicates that all constraints are satisfied, while $t = 1$ indicates inaccurate design and $t = 2$ failed timing constraints. For simplicity, due to the effect of random processes like Place and Route (PAR), it is assumed that c is a random process that consists of a number of nonidentical independent random variables, each following a distribution with an unknown probability density function $\mathfrak{p}_\mathbf{x}(t) = \mathrm{p}(c(\mathbf{x}) = t)$. Exit code depends on a set of latent constraint functions: the functions $h_i : \mathcal{X} \to \mathcal{R},\ i = 1, .., k$, such as resource consumption or accuracy, and the binary functions $g_j : \mathcal{X} \to [0, 1],\ j = 1, .., r$, for example PAR or timing success. The region of the parameter space that has a non-zero probability of satisfying all constraints is called the valid region $\mathcal{V} = \{\mathbf{x} | \exists \mathbf{x} \in \mathcal{X} : [\mathfrak{p}_\mathbf{x}(1) > 0]\}$. The configurations, which have a 0% probability of meeting constraints, belong to the invalid region $\mathcal{I} = \{\mathbf{x} | \exists \mathbf{x} \in \mathcal{X} : [\mathfrak{p}_\mathbf{x}(1) = 0]\}$. A good example is a configuration that overmaps on resources. The goal is to find the configuration $\mathbf{x}_{opt}$ that satisfies all constraints and has the highest possible fitness $\mathbf{x}_{opt} = \underset{\mathbf{x} \in \mathcal{V}}{\operatorname{argmax}} f(\mathbf{x})$.

A designer often has the opportunity to create a design similar to an existing one, using their prior experience to improve its efficiency. The same process can be adopted in automated optimisation. There are two designs, an *old* design and a *new* one. Can both the information learned and the old data be used to speed up new design optimisations? Three challenges have to be addressed to transfer knowledge: (a) "what to transfer", (b) "when to transfer" and (c) "how to transfer" [5]. The old design was optimised with an old design benchmark function f_{old} and old valid region $\mathcal{V}_{old}$. It is based on the old latent constraint functions $h_{old,i}$ and $g_{old,j}$.

IV. Auto-Transfer Algorithm

The goal of the Auto-Transfer algorithm is to enable faster optimisation of reconfigurable designs by extracting valuable information from a previous optimisation attempt. The approach is illustrated in Figure 1. It requires two inputs. The first input is an old design optimisation result. The task of selecting an old design from the database suitable for knowledge transfer is not currently automated, an old design believed to be related to the new design is manually selected. The second input is a script. For a given parameter configuration of the new design, the script builds the design for a specific Field Programmable Gate Array (FPGA) device, compiles benchmarks to assess performance of that configuration. Depending on the outcome, the script outputs an exit code indicating whether the design with a given configuration meets all the design constraints or not. Those constraints range from resource constraints to design specific constraints such as output accuracy. The script produces performance metrics — like execution time or power consumption. This packaging of the design generation and assessment processes allows parallel optimisation of a range of designs with different toolchains and test data.

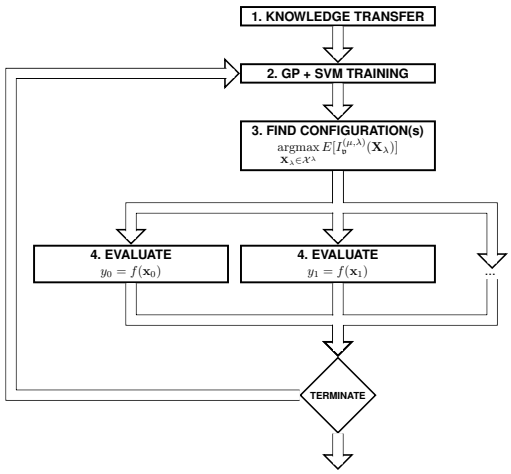
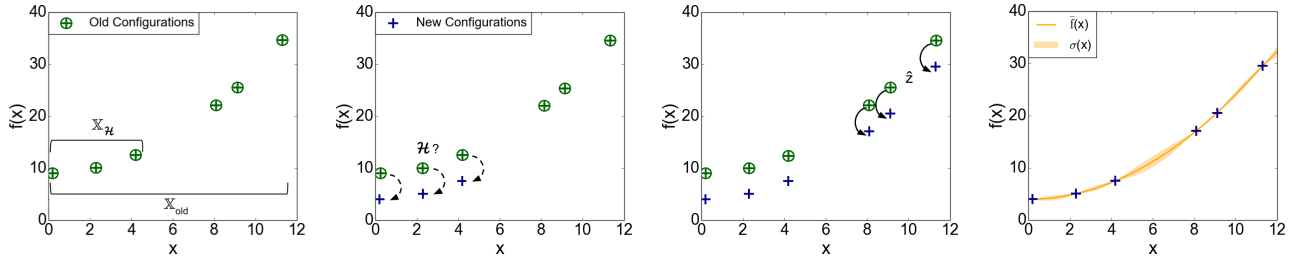


Fig. 2: The Auto-Transfer algorithm.

The Auto-Transfer algorithm (Figure 2.) starts with the knowledge transfer step, during which an initial GP and Support Vector Machines (SVM) surrogate model of the design behavior is constructed, based on the old design experiments. Then the algorithm iteratively improves the surrogate model through experiments indicated by the acquisition function $E[I_{\mathfrak{v}}^{(\mu,\lambda)}(\mathbf{X}_\lambda)]$, driving optimisation. The function accounts for constraints and allows for asynchronous parallelism during optimisation.

The knowledge transfer step has three phases, each addressing one of the previously mentioned challenges. An example is shown in Figure 3 and the step is presented in Figure 4. The step tests if there is a relationship between performance f and each of the latent constraint functions h_i, such as accuracy or resource utilization. If any are identified, the previously collected data are reused accordingly. For example, it is likely that if a design is ported across devices, the Lookup Table (LUT) utilization is going to grow in a similar fashion on both devices. If the relationship between utilization is uncovered, it can be possible to predict resource overmapping on the new device more accurately.



(a) *Phase 1* (b) *Phase 2* (c) *Phase 3* (d) *Optimisation*

Fig. 3: Knowledge transfer step. There are six designs in $\mathbb{X}_{old}$, of which three are used for hypothesis testing (a). In (b) three new configurations are evaluated taking substantial amount of time, and the relationship hypothesis is tested. As hypothesis is not rejected, and cheap regression $\hat{z}$ is constructed, old data are mapped and treated as if they come from new experiments (c). The benefit is clear when an accurate model is constructed with six data points, after only three configuration evaluations (d).

Phase 1
Choose a random set $\mathbb{X}_{\mathcal{H}}$ of designs from $\mathbb{X}_{old}$ for hypothesis testing
Evaluate $\mathbb{X}_{\mathcal{H}}$ designs using benchmark
Phase 2
for f, all g and h **do**
  Test $\mathcal{H}_l$ using $\mathbb{v}_{old,i}$ and $\mathbb{v}_i$
  Test $\mathcal{H}_m$ using $\mathbb{v}_{old,i}$ and $\mathbb{v}_i$
Phase 3
for f, all g and h **do**
  if $\mathcal{H}_l$ not rejected at significance level α **then**
    Calculate linear least square regression $\hat{z}_i$
    Map $\mathbb{y}_{old}$ or $\mathbb{v}_{old,i}$ using the regression $\hat{z}_i$
      a.) If f, insert the results into the vector $\mathbb{y}$
      b.) If h_i, evaluate constraints using the mapping and insert exit codes into the vector $\mathbb{t}$
  else if $\mathcal{H}_m$ not rejected at significance level α **then**
    Calculate Isotonic regression $\hat{z}_i$
    Map $\mathbb{y}_{old}$ or $\mathbb{v}_{old,i}$ using the regression $\hat{z}_i$
      a.) If f, insert the results to the vector $\mathbb{y}$
      b.) If h_i, evaluate constraints using the mapping and insert exit codes into the vector $\mathbb{t}$

Fig. 4: Knowledge Transfer, Step 1 of Auto-Transfer algorithm.

1) Phase 1: "What to Transfer" – The goal is to identify the relationship between the old and the new design using as few design evaluations as possible. There are two relatively easily verifiable relationships. The algorithm allows for either linear or monotonic relationships between the old and new f and h_i. The process starts by re-evaluation of a random subset of $n_{\mathcal{H}}$ parameter settings $\mathbb{X}_{\mathcal{H}}$, which are present in $\mathbb{X}_{old}$; they can be evaluated on the new platform. This subset is used for hypothesis testing. If there is no such subset, then knowledge transfer cannot proceed. The information re-used will typically consists of LUT, flip-flop, Block RAM (BRAM), Digital Signal Processor (DSP) block utilization and design performance.

2) Phase 2: "When to Transfer" – First a test is performed to verify if there is a linear relationship between any of the old and new functions f and h_i. To test the linear relationship hypothesis $\mathcal{H}_l$, the Pearson product-moment correlation coefficient [11] is calculated between old and new data sets for a given function. For example, accuracy of the results for the same configuration on two devices might be linearly mapped between each other. If the linear relationship hypothesis $\mathcal{H}_l$ is rejected at an α-significance level or indicates a weak correlation, a monotonic relationship hypothesis $\mathcal{H}_m$ is tested. To test the hypothesis $\mathcal{H}_m$ the Spearman rank correlation is calculated [12]. Similar to the Pearson correlation, the double sided p-value is calculated. If $\mathcal{H}_m$ is rejected at an α-significance level or there is a weak correlation, it is assumed that the null hypothesis $\mathcal{H}_0$ holds and that there is no relation between the two tested functions. Each function f and h_i is tested separately, as for example it is possible that two devices follow a similar flip-flop utilization pattern but not LUT.

3) Phase 3: "How to Transfer" – Knowledge is transferred differently depending on which hypothesis holds. If the linear relationship hypothesis $\mathcal{H}_l$ is accepted, the old data collected for one of the function f or h_i are mapped to the new design. This is done by calculating a least-squares linear regression to recover a mapping function z_i for either f or h_i. The mapping function regression $\hat{z}_i \approx z_i$ becomes $\hat{z}_j(v) = av + b$, which is then used for the mapping $\hat{z}_j(v_{old,i}) = v_i$ treating the data from previous optimisations as if they originated from the new design. If the linear relationship hypothesis $\mathcal{H}_l$ does not hold, but the monotonic $\mathcal{H}_m$ does, isotonic regression is performed [13]. For example, if flip-flop consumption grows slightly differently between the two related designs, $v_{old} = [0.1, 0.2, 0.3]$ and $v = [0.1, 0.4, 0.9]$ the trend is similar but $\mathcal{H}_l$ does not hold. At the same time $\mathcal{H}_m$ holds and an isotonic regression is used to map the old data.

V. Evaluation

To assess the benefit of knowledge transfer, a comparison is made between the new Auto-Transfer algorithm and ARDEGO [3]. Two benchmarks are used: a quadrature-based financial design with customizable precision [7] and a high performance Reverse Time Migration (RTM) design with seven parameters [8]. Each design is first optimised for the Maxeler MPC-X1000 system with Xilinx Virtex-6 XC6VSX475T FPGA. Afterwards, optimisation for Maxeler MPC-X2000 with an Altera Stratix V GS 5SGSD8 FPGA follows. The Auto-Transfer algorithm uses up to 5 D parameter settings to construct $\mathbb{X}_{\mathcal{H}}$. α is set to 1%, and correlations weaker than ± 0.95 are rejected.

4) Quadrature-based Financial Computation: The quadrature-based financial design [7] has three parameters and an associated throughput benchmark. It can be used to compute integrals for various financial applications. The design offers a trade-off between accuracy and throughput. The parameters are mantissa width m_w of the floating point

operators, the number of computational cores κ and the density factor d_f. A larger number of m_w bits increases computation accuracy, but limits κ due to the increased size of an individual core and the resource constraints. The number of quadratures used for integral estimation is regulated by d_f. By increasing the parameter a larger number of quadratures is used for an estimation of an integral. This increases both the computation cost and the accuracy of the results. The optimization goal is to find the design with the highest throughput using the provided benchmark. A latent constraint accuracy function exists which specifies the maximum acceptable error ϵ_{rms}.

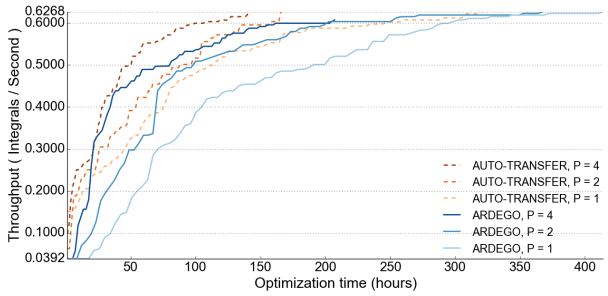


Fig. 5: Optimisation of the quadrature-based financial design throughput benchmark for $\epsilon_{rms} = 0.001$.

Transfer knowledge speeds up optimisation in nearly all of the cases during the early stages of optimisation as seen in Figure 5. This is due to transferred knowledge induced inaccuracies. When $\epsilon_{rms} = 0.001$ and $P = 4$ the average optimization time reduces by 35%, from 133 hours to 86 hours. The reduction for $P = 2$ and $P = 1$ is 33% and 35% respectively. Although both designs run at the same clock frequency, and the accuracy constraint functions of the design on both platforms are identical, this is not the case for the fitness function. As the d_f parameter is decreased, the problem becomes communication bound instead of compute bound. This is especially prominent when $\epsilon_{rms} = 0.1$ or $\epsilon_{rms} = 0.01$ as transfer knowledge does not offer as big an improvement.

5) Reverse Time Migration: The RTM design [8] is a stencil-based design used for seismic imaging and it involves seven parameters. The parameters determine parallelism, numerical precision and balancing of computation and communication through changing of the design architecture. They are bit-width optimisation ratio B, dimension and kernel parallelism P_t, P_{knl} and P_{dp} as well as blocking ratios α and β and arithmetic operation transformation ratio T. Depending on the device there are 20 or 81 million possible configurations. The globally optimal configuration offers execution time over 100 times shorter than the basic configuration. Latent constraint functions involve resource utilization and communication bandwidth.

Figure 6 shows there is a clear benefit from using information derived from the old designs. The improvement is most prominent when $P > 1$. The improvement when $P = 4$ is 45% in design performance, as well as 11% shorter optimization time. The initial large benefit comes from reevaluation of old designs, which are all built on the old platform, and from knowledge transfer.

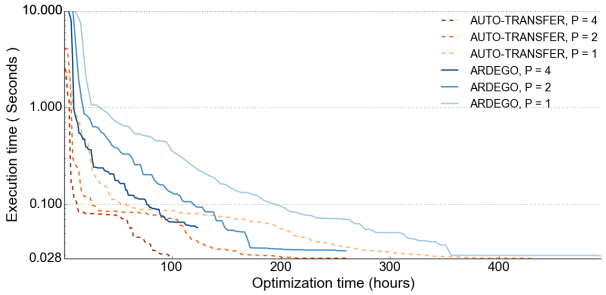


Fig. 6: Optimisation of execution time of the RTM design.

VI. Conclusion

We present a new Auto-Transfer algorithm, which offers substantial reduction in optimisation time. For the quadrature-based financial design and the reverse time migration designs we observe a reduction in optimisation time of up to 35% compared to [3]. For the quadrature based design, the knowledge transfer step helps improve optimisation speed despite the large amount of noise introduced by the new platform. Current and future work includes automating the selection of related previous designs, and extending the evaluation of the proposed approach by a wide range of applications.

Acknowledgement. This research has received funding from European Union H2020 Programme with project reference 671653. The support of UK EPSRC (grant references EP/I012036/1 and EP/L00058X/1), HiPEAC NoE, the Maxeler University Program, Altera and Xilinx is gratefully acknowledged. MPD has been supported by a Google Faculty Research Award.

References

[1] N. Kapre, B. Chandrashekaran, H. Ng, and K. Teo, "Driving timing convergence of FPGA designs through machine learning and cloud computing," in *FCCM*, 2015.

[2] A. Mametjanov, P. Balaprakash, C. Choudary, P. Hovland, S. Wild, and G. Sabin, "Autotuning FPGA design parameters for performance and power," in *FCCM*, 2015.

[3] M. Kurek, T. Becker, T. C. Chau, and W. Luk, "Automating optimization of reconfigurable designs," in *FCCM*, 2014.

[4] M. K. Papamichael, P. Milder, and J. C. Hoe, "Nautilus: Fast automated IP design space search using guided genetic algorithms," in *DAC*, 2015.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *TKDE*, vol. 22, no. 10, pp. 1345–1359, 2010.

[6] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *UAI*, 2014.

[7] A. H. Tse, G. C. Chow, Q. Jin, D. Thomas, and W. Luk, "Optimising performance of quadrature methods with reduced precision," in *ARC*, 2012.

[8] X. Niu, Q. Jin, W. Luk, Q. Liu, and O. Pell, "Exploiting run-time reconfiguration in stencil computation," in *FCCM*, 2012.

[9] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[10] J. Janusevskis, R. Le Riche, D. Ginsbourger, and R. Girdziusas, "Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges," in *LION 12*, 2012.

[11] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. Roy. Soc.*, vol. 58, pp. 240–242, 1895.

[12] D. Zwillinger and S. Kokoska, *CRC standard probability and statistics tables and formulae*. CRC, 1999.

[13] N. Chakravarti, "Isotonic median regression: A linear programming approach," *Mathematics of Operations Research*, vol. 14, no. 2, pp. 303–308, 1989.