





Region of Interest-Based Parameter Optimization for Approximate Image Processing on FPGAs

Manu Manuel^{}, Nguyen Anh Vu Doan^{}, Walter Stechele^{}
Technical University of Munich, Munich, Germany
Email: {manu.manuel, anhvu.doan, walter.stechele}@tum.de

Arne Kreddig^{}
SmartRay GmbH, Wolfratshausen, Germany
Email: arne.kreddig@smartray.com

Simon Conrady^{}
Arnold & Richter Cine Technik, Munich, Germany
Email: SConrady@arri.de

Received: February 15, 2021
Revised: May 1, 2021
Accepted: May 31, 2021
Communicated by Susumu Matsumae

Abstract

In recent years, technological advancements in computer hardware systems have been lagging behind the demand for increased computational power, especially in application domains such as signal and image processing. Approximate computing is a design paradigm for efficient system design to overcome this bottleneck by exploiting the resilience of such applications to inaccuracy in their computations and trading off quality for hardware resource savings. Over the years, many approximation techniques have been proposed on various abstraction layers and demonstrated their effectiveness in different applications. Combining multiple methods in a larger system can further increase the resulting benefits. However, this often leads to a non-trivial optimization task of finding the best parameterization across all employed methods. The interaction and influence of error propagation between individual components demand a global optimization of parameters that simultaneously considers all the parameters for each of the approximation techniques used. In this work, we propose a methodology for exploring such highly complex design spaces using a multi-objective genetic algorithm in an FPGA-based system. Simple models are used for the estimation of resource demands in terms of power together with the anticipated quality degradation. The optimization is carried out to determine the trade-off between these objectives. We demonstrate the effectiveness of our approach on a typical color processing pipeline by tailoring the encoding and genetic operations to the needs of this application. To focus the optimization into a relevant region of interest, we propose ROI-NSGA, a novel variant of nondominated solution selection, and compare its optimization efficiency with the traditional NSGA-II approach for the examined case study. Our results show that the models are able to guide the optimization, and that the genetic operations and selections are capable to find Pareto-optimal solutions, among which the desired quality-resource trade-off can be chosen. Besides, the ROI-NSGA based optimization outperforms the results obtained for the case study using the NSGA-II approach within the region of interest.

Keywords: FPGA, Approximate Computing, Design Space Exploration, Multi-Objective Optimization, NSGA-II, Genetic Algorithm, Image Processing.

1 Introduction

Increasing demands for processing power and bandwidth on computational systems are outpacing the technological improvements of embedded computing systems such as field-programmable gate arrays (FPGAs). Designing efficient implementations of such systems is challenging, especially when targeting mobile devices that have to fulfill significant requirements on power consumption. Addressing this issue, the research field of approximate computing provides a novel design paradigm that exploits the inherent error resilience of certain application domains such as image processing, signal processing, data analysis, and data mining. It assumes that such applications can tolerate some inexactness in their computations which can be leveraged for a more efficient usage of computational resources. However, to ensure acceptable quality of experience in the application, the quality degradation resulting from approximations needs to be controlled which leads to a non-trivial parametrization task.

Many approximation methods focusing on embedded computing hardware have been proposed [2, 5, 8, 15, 16, 17, 20, 23, 25, 27, 31, 34, 36, 37, 38, 41], comprising single-purpose methods such as approximate arithmetic units or precision scaling as well as general-purpose methods for automatically approximating combinatorial circuits. Such individual methods have been demonstrated in isolation, but combining several of them could yield stronger benefits in complex applications. However, when multiple approximations are used in conjunction, both their collective influence on the quality and the propagation of errors through the system need to be considered.

Because resource usage and application quality are conflicting objectives, the designer is interested in analyzing the potential trade-offs between Pareto-optimal solutions. This leads to a multi-objective design space exploration (DSE) problem for which brute force searches become infeasible due to the size of the design space growing exponentially with the number of employed methods. The search must therefore be guided by an appropriate optimization algorithm, and the probing of possible solutions with respect to resource usage and quality degradation must be both accurate and fast. To ensure timely evaluation of the solutions, the estimation of area and power consumption should avoid time-consuming synthesis of the system for each parametrization. Furthermore, in order for the designer to be able to make valid choices regarding the acceptable quality degradation, the employed quality model should be suitable for the targeted application and interpretable for the designer.

To solve the problem of finding Pareto-optimal parametrizations for FPGA-based image processing systems that combine multiple approximation techniques, we propose an approach that uses multi-objective optimization (MOO) in combination with fast models for the estimation of application quality and resource usage without the need for time-consuming synthesis during the optimization. The proposed method can be used to select and combine arbitrary methods that expose parameters to tune the strength of approximation. It considers and optimizes all exposed parameters in conjunction and therefore implicitly takes interactions between different system components and error propagation into account.

In our approach, the MOO for the DSE makes use of the Nondominated Sorting Genetic Algorithm-II (NSGA-II) [6] which globally explores the highly complex design space with a reasonable number of fitness evaluations and obtains Pareto-optimal or near Pareto-optimal solutions. This approach inherently maintains the convergence and diversity in the obtained Pareto solutions. In applications where acceptance thresholds in the objectives are known, a global optimization might spend unnecessary efforts to find solutions outside of the region of interest (ROI) which are irrelevant for practical implementation. In order to guide the optimization to concentrate its search pressure on a relevant ROI defined by the designer for a specific application, we propose a novel variant of nondominated sorting selection, named Region of Interest Nondominated Sorting Genetic Algorithm (ROI-NSGA), inheriting from NSGA-II based optimization.

We demonstrate the usability of our proposed methodology with both of these optimization approaches experimentally on an image processing pipeline commonly used by digital cameras to adapt images for display on a specific monitor, employing a combination of precision scaling, approximate arithmetic units, and sparse look-up tables. Our results show that the proposed DSE with both optimization approaches is able to construct a suitable Pareto front with respect to the quality-

resource trade-off. Selected parameter configurations from the Pareto front are synthesized and fitted using a standard FPGA design flow in order to validate the potential savings. Furthermore, the performance of each optimization approach is analyzed individually based on the hypervolume indicator which reflects both convergence and diversity of evolved Pareto-optimal solutions, showing that with the novel ROI-NSGA, similar results to NSGA-II can be achieved with less than 50% of the computational effort.

The rest of the paper is organized as follows. Section 2 gives an overview on the related work. In Section 3, the tackled DSE problem is introduced in detail. Section 4 explains our general methodology, including the modeling for resource, power, and quality as well as the exploration method, introducing the ROI-NSGA selection algorithm. In Section 5, we present an image color processing case study suitable for demonstrating the use of our method. Section 6 describes the experimental setup and presents the results obtained from the experiments. Finally, Section 7 summarizes our findings and concludes this paper.

2 Related Work

Over the last decades, numerous approximation techniques have been proposed on different abstraction layers, targeting various applications. A survey on approximate computing trends by Barua and Mondal [2] and another survey by Mittal [20] give a broad overview of main research categories and proposed techniques in this field.

Many of the methods proposed for hardware-based signal processing systems aim at approximating a specific component of the design. This includes precision scaling [16, 25], approximate arithmetic units such as adders [8, 17] and multipliers [34], table-based methods [5, 15, 27] and memoization techniques [31]. Furthermore, general-purpose techniques have been proposed to approximate arbitrary combinatorial circuits. Specifically targeting FPGA systems, Wu et al. proposed a method for functional approximation by wire removal [41]. Vasicek and Sekanina have shown that cartesian genetic programming (CGP) can be used to automatically approximate circuits and that the benefits are preserved by FPGA design tools [36]. These methods can act on the circuit level only and are not suited for higher-level methods and approximations for data-based calculations. Other approximate logic synthesis methods like SALSA [38], SASIMI [37] and ABACUS [23] have been proposed for application specific integrated circuit (ASIC) designs. However, most of these methods can also translate well for FPGAs as target platform.

The choice of approximation methods depends strongly on the target application. Depending on the specific application, different methods might be most useful. For example, when signal calibrations or complex non-linear functions are implemented using pre-calculated data, table-based approximations are highly efficient especially for reducing memory consumption. On the other hand, systems that perform many linear calculations benefit most from approximate arithmetic units and precision scaling. Many signal and image processing systems contain both of these types of calculations in conjunction. Hence, neither the single-purpose methods alone nor the general-purpose methods, which are restricted to the circuit level, can exploit all of the potential for approximation. A combination of specialized single-purpose methods, manually chosen by the designer and tailored to the specific application, on the other hand, can explore the quality-resource trade-off across multiple abstraction layers.

When multiple approximations are combined, however, the number of possible configurations grows exponentially with the number of exposed parameters. Many proposed approaches are addressing this problem. A multi-level approximate accelerator synthesis approach has been proposed by Zervakis et al. [45] in which a library of arithmetic units is formed by combining algorithmic techniques with circuit-level approximations and Voltage Over-Scaling (VOS) on the device level. Their work targets ASIC designs, and while some methods might be applicable to FPGA-based systems, usage of VOS with locally different voltages is not possible on current commercial FPGAs. Furthermore, their quality model involves the use of an artificial neural network (ANN) to estimate the overall quality. The model is trained with output errors obtained by simulation on multiple synthesized designs, making their approach fairly complex. Starting from the behavioral description,

Xu and Schafer propose a multi-level approximation method that enables to apply approximations in different phases of the design flow [43]. However, their methodology works with high-level synthesis (HLS) and is not applicable to HDL-based design flows.

Furthermore, autoAx by Mrazek et al. [22] and ApproxFPGAs by Prabakaran et al. [26] combine different state-of-the-art approximated circuits by replacing corresponding arithmetic operations in the target design. While autoAx specifically targets ASIC designs, ApproxFPGAs is an extension of the methodology for FPGA designs. Both of these approaches use machine learning techniques for quality and resource models which are trained with data from randomly selected configurations. Castro-Godínez et al. proposed AxHLS that selects, configures, and optimizes approximate arithmetic circuits using a Tabu Search-based DSE during HLS [3]. As the core of all these approaches lies in the combination of different arithmetic units, they are also restricted to the circuit level.

A characteristic common to many publications in approximate computing is that they use basic signal difference metrics such as the absolute or relative error, the error rate, or the hamming distance. While these metrics are capable of assessing the general trade-off between accuracy and resource demand, their relevance for real-world applications is often limited. For the quality model to be valid for design decisions, it should therefore be linked to the application instead of relying on simple signal error metrics. In specific application domains such as image processing, many application-specific metrics have matured over the last decades, providing a better understanding of quality in relation to the targeted field. For color processing systems, the CIELAB ΔE is used to measure color differences in a perceptually uniform color space [14]. For systems that perform tasks such as filtering or noise reduction, metrics that consider structural differences like the structural similarity measure (SSIM) [40] are better suited.

Comprehensive quality-resource trade-off analysis is often missing in the state-of-the-art approximate computing methods which combine multiple approximation techniques. This limits the exploitation of possible benefits from the combined approximation methods. Overcoming this limitation, the approach used in autoAx [22] and ApproxFPGAs [26] identifies the Pareto front for the quality-resource trade-off with a two-phase optimization approach. However, the core of this approach is based on a heuristic hill climbing algorithm that employs a local search. Similarly, the principle behind the Tabu Search-based DSE in AxHLS [3] is also a local search. Hence, these approaches do not inherently try to ensure diversity in the evolved Pareto solutions during the exploration which might hinder trade-off analyses. A fast and elitist multi-objective optimization, NSGA-II, proposed by Deb et al. is an established method for an effective exploration of the design space to form a suitable Pareto front while trying to ensure diversification of the results [6]. However, in DSE problems, NSGA-II explores and tries to optimize the entire search space which is not always necessary in practical applications. Many variants of NSGA have been proposed to address this problem [7, 9, 29, 39]. However, all of these methods explore the targeted space based on a reference direction or reference solutions of the objective function. In many cases, these points or directions have a high influence on the explored region and, the designer has to provide this information before the optimization. But, in some real-world optimization problems such as the case study examined in this paper, it might be difficult to provide such information for all the objectives in advance due to the lack of knowledge about final optimality. On the other hand, boundaries in the objective space can be defined based on acceptable threshold levels.

Our approach differs from previous works in several ways, addressing some of their shortcomings. The proposed methodology directly targets FPGA designs using established design flows, taking into account the specific architectural characteristics that these devices provide, including the use of dedicated digital signal processing (DSP) units, block RAMs (BRAMs) and adaptive look-up tables (LUTs). Furthermore, this work makes use of the designer's domain knowledge for selecting suitable approximation methods. Instead of relying on optimizations on the circuit-level only, we allow the designer to select and combine appropriate single-purpose approximation methods across different abstraction layers and then globally optimize their joint parameterization. While for each method, the usage of FPGA architectural units has to be quantified by the designer depending on the parameters, the power consumption is predicted directly from this estimated resource, removing the need for time-consuming iterative re-synthesis or training of machine-learning models. Moreover, we allow the designer to use their preferred quality metric, so that they can rely on established

application-specific metrics. The optimization approaches enable the designer to analyze the quality-resource trade-off on the desired region and exploit the maximum benefit in the selection of a solution from the Pareto front for implementation.

3 Problem Formulation

The overall goal of the approximate computing paradigm is to explore the trade-off between application quality and resource usage. As this is a multi-objective task, there is no single optimal solution, and the set of Pareto-optimal points is of interest to the designer. In this work, we consider the optimization of FPGA designs that employ a combination of different single-purpose approximation methods. The targeted system is described by a data flow graph where the nodes represent components or operations of the reference design, denoted by $c \in C$. For each component, the designer can select one or more suitable approximation method that is considered by the DSE. The approximation of each component exposes a set of approximation parameters $s_c \in S$ used to a) select among different approximate units, if applicable, and b) tune the strength of the approximation. Because of possible interactions of the components and error propagation effects, these parameters have to be optimized globally. The design space, comprising all possible parameter configurations across S , thus exponentially grows with the number of components and the parameterization complexity, making brute force exploration infeasible. To determine the quality-resource trade-off, resource usage is quantified by system power consumption while quality degradation is assessed by a user selectable metric. The objective of this work is to effectively explore this trade-off with a focus on a desired ROI for a given targeted FPGA-based system which consists of a set of components C and a selection of suitable approximation methods exposing a set of approximation parameters S .

4 Methodology

As stated in Section 3, the proposed approach splits the system into simpler components and employs a divide-and-conquer strategy. The overall system is represented as a data flow graph in which each node represents a single component of the system. This allows for modeling the resource and power consumption of every component individually and therefore facilitates the design space exploration on complex FPGA systems. However, for quality, this is not possible, as the effect of error propagation on the application quality must be considered during the design space exploration.

To assemble the whole system, approximate components are selected from a predefined library to replace accurate components in the system. The approximate components in this library are implementations of various single-purpose approximation techniques created by the designer or taken from existing literature. Furthermore, every component in the library comes with a set of parameters to tune the approximation strength. The resource usage and power consumption of the components depend on these parameters.

The employed power model directly derives the power consumption from the estimated resource usage of the system in an approach similar to HAPE [18]. To account for error propagation in quality assessment, a software simulation is used together with a user selectable metric which is evaluated over a training data set. For the DSE, our method uses genetic algorithm (GA)-based MOO with a novel modification of the state-of-the-art NSGA-II selection algorithm called ROI-NSGA in further sections.

The following subsections provide further details about the modeling of resource usage, power consumption, and quality. Furthermore, the functionality of the employed optimization procedure is explained.

4.1 Resource Model

To model the resource consumption of the whole system, the resources consumed by each component are modeled individually first. Then, the overall system resources can be expressed as a sum of the

resources from all the individual components $c \in C$:

$$R_{total} = \sum_{c \in C} r_c. \quad (1)$$

Such a divide-and-conquer approach enables us to easily use the resource model during design space exploration, and we do not have to synthesize the complete system for every possible combination of approximation parameters.

For each component, a specific resource model is stored in the component library. Each of these models must offer a function to calculate the resource consumption of the component r_c depending on the respective approximation parameters s_c :

$$r_c = f(s_c). \quad (2)$$

These models must be selected or created for the targeted FPGA family. Such models can be taken from state-of-the-art libraries such as [30] or [34]. Alternatively, the designer can estimate the resource usage taking the parameters and their influence on the component instantiation into account. For example, the memory consumption of sparse look-up tables [15] can be directly calculated from the number of sparse grid points, as shown in Section 5.2.1. Otherwise, the tools and information from the FPGA manufacturer need to be used to derive the resource consumption. Therefore, the approximate component must be synthesized in different configurations. Afterwards, the impact of the approximation parameters on the resource consumption must be observed. However, due to the characteristics of the FPGA technology, the resource consumption might not increase continuously when tuning an approximation parameter. This usually happens when the synthesis only uses BRAMs or DSPs partially. In this case, the resource consumption only increases once the synthesis filled one unit completely and requires an additional BRAM or DSP unit. Consequently, discrete jumps in the resource consumption can be seen instead of a continuous increase. The exact behavior of these units depends on the targeted FPGA family.

FPGAs have different types of resources available, namely DSPs, BRAMs, LUTs, and registers which serve different functional purposes. While DSPs are used to calculate arithmetic operations such as multiply or multiply-accumulate, LUTs can implement fully customized logic. Furthermore, BRAMs and registers are used to store data. While BRAMs can store big chunks of data to implement e.g. First-in First-out (FIFO) buffers, registers are placed close to the LUTs and usually store intermediate results. Because the maximum number of units available on a single FPGA device and their power consumption properties vary strongly between the different resource types, the resource model lists the estimated number of units separately for each type. Consequently, the resources consumed by one component are represented by a vector:

$$r_c = \{r_{c,dsp}; r_{c,bram}; r_{c,lut}; r_{c,reg}\}. \quad (3)$$

4.2 Power Model

The power of each component is derived from the resources used by each component. Like the resource model, the total power consumption of the system is the sum of the power consumption of all components:

$$P_{total} = \sum_{c \in C} p_c. \quad (4)$$

As all FPGA units contribute to the overall power consumption with a specific static power consumption and a specific dynamic power consumption which depends on the switching activity, the static and dynamic power consumption of each component is calculated separately.

The power model calculates the power for the different FPGA resource types separately. For every component, the power is calculated as the sum of the static and the dynamic power consumption:

$$p_{c,t} = p_{c,t,stat} + p_{c,t,dyn}, t \in \{dsp, bram, lut, reg\}. \quad (5)$$

First, the static power consumption of each resource type can be calculated as

$$p_{c,t,stat} = r_{c,t} * q_{t,stat}, \quad t \in \{dsp, bram, lut, reg\}, \quad (6)$$

where $q_{t,stat}$ is the static power consumption per unit of the given resource type, and $r_{c,t}$ is the number of units of the given type that are used by the component.

The dynamic power consumed by the FPGA depends on the switching activity, expressed as signal transitions per second. To simplify the handling of the model, the signal transitions per second are split into a toggle rate α and a clock frequency f_{clk} , so the number of signal transitions per second is $f_{clk} * \alpha$.

The values for clock frequency and toggle rate have to be set by the designer. Calculating exact toggle rates for each configuration would be very time-consuming. Therefore, the toggle rate is estimated using the reference design without approximations, assuming that the values are similar in approximated systems. This can be done by simulating common input data for the system and taking the average of the measured toggle rates. Using this global toggle rate α_{global} , the dynamic power consumption for DSPs, LUTs, and registers calculates as

$$p_{c,dsp,dyn} = r_{c,dsp} * f_{clk} * \alpha_{global} * q_{dsp,dyn}, \quad (7)$$

$$p_{c,lut,dyn} = r_{c,lut} * f_{clk} * \alpha_{global} * q_{lut,dyn}, \quad (8)$$

$$p_{c,reg,dyn} = r_{c,reg} * f_{clk} * \alpha_{global} * q_{reg,dyn}, \quad (9)$$

where $q_{dsp,dyn}$ is the power consumption of a DSP unit per MHz, and $q_{lut,dyn}$ and $q_{reg,dyn}$ are the power consumption of the LUTs and registers per unit per MHz, including average routing resources.

Furthermore, the dynamic power consumption of the BRAMs must be calculated. Therefore, $p_{c,bram,dyn}$ is split into the power consumed for accessing the memory in write and in read mode respectively:

$$p_{c,bram,dyn} = p_{c,bram_wr,dyn} + p_{c,bram_rd,dyn} \quad (10)$$

which calculate as

$$p_{c,bram_wr,dyn} = r_{c,bram} * f_{clock} * \alpha_{bram_wr} * q_{bram_wr,dyn}, \quad (11)$$

$$p_{c,bram_rd,dyn} = r_{c,bram} * f_{clock} * \alpha_{bram_rd} * q_{bram_rd,dyn}, \quad (12)$$

given the BRAM read and write rates α_{bram_rd} and α_{bram_wr} which also have to be set by the designer according to the application.

To extract the model parameters, the Intel Early Power Estimator (EPE) is employed [12]. Analogously, the Xilinx Power Estimator (XPE) can be used to create a power model for Xilinx FPGAs [42]. These tools are spreadsheet-based and allow for estimating the power consumption of an FPGA design based on the number of instantiated units. To include this model into our DSE workflow, targeting an Intel FPGA, we extract the model parameters $q_{t,stat}$ for $t \in \{dsp, bram, lut, reg\}$ and $q_{t,dyn}$ for $t \in \{dsp, bram_rd, bram_wr, lut, reg\}$ from the EPE by instantiating only units of the same type at a time and analyzing the reported power consumption. In the EPE, some basic settings need to be configured, such as the expected FPGA temperature as well as the targeted FPGA device. The temperature is set to 50 °C in the EPE, as this is a realistic temperature that can commonly be found in real-life use-cases [44].

4.3 Quality Model

In any approximated system, the estimation of quality is an integral part needed for making valid choices regarding the acceptable approximation strength. The choice of a suitable quality model has significant impact on the designer's ability to set meaningful quality thresholds. Hence, the model should be accurate, interpretable for the designer, and suitable for the targeted application. In our proposed approach, any state-of-the-art reference metric like for example peak signal-to-noise ratio (PSNR), CIELAB ΔE [14], or SSIM [40] can be employed. Therefore, the quality estimation

can be adapted according to the designer's experience and the individual needs of the application. For example, in color processing applications used in professional camera systems, CIELAB ΔE can be a good choice to account for the perceptual properties of the human visual system.

For the calculation of any reference metric, both the golden output from the reference system and output from the approximated version are required. In order to perform the DSE effectively and obtain these output values, a training data set must be chosen carefully covering typical use cases as well as corner cases from the selected application. For color processing tasks, this could be a selection of common scenes or a set of colors sampled across the input color space. Initially, the training set is processed with the accurate reference system, and the golden output is stored. During the DSE, the quality model calculates the approximated output for any probed approximation configuration and compares it to the golden output using the chosen reference metric. For this, a configurable software simulation of the application is used which can bit-accurately simulate the effects of using the approximate components within the system. Therefore, the estimation can be done on-the-fly during the design space exploration and does not require repeated synthesis of the corresponding FPGA designs.

4.4 Design Space Exploration

Our approach uses the previously described models for analysis and evaluation of feasible solutions. The design space is searched and pruned using a multi-objective metaheuristic. Because of potential interactions between different approximation parameters and error propagation effects, the parameters cannot be optimized individually, leading to a highly complex design space. We use the GA as a population-based metaheuristic approach which maintains a balance between selection pressure and population diversity for highly explorative and exploitive search. The multi-objective approach excludes the difficulty of defining suitable weights to form an aggregated cost function. As a result, we construct the Pareto-optimal set with nondominated configurations regarding the quality-resource trade-off. These configurations include both the combination of possible approximations and the approximation parameters of each module. The Pareto-optimal set helps to determine the desired configuration with maximum benefits while keeping the application quality above a certain threshold.

4.4.1 Genetic Algorithm

Figure 1 shows an overview of the DSE procedure based on the GA and nondominated sorting selection. First, suitable approximation methods with exposed parameters $s_c \in S$ are selected by the designer for individual components $c \in C$ depending on the target application. For simplicity, we use real value encoded genes to form individuals by concatenating the parameters s_c from all components C in the system. The initial population of individuals is created with randomly generated genes by taking the parameter interdependencies and design constraints of the individual components c in the targeted system into account.

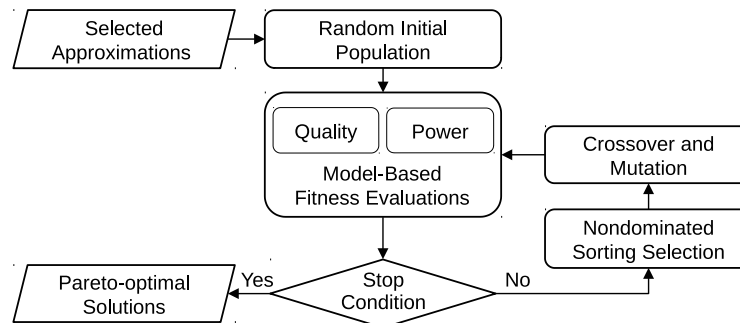


Figure 1: Overview of the model-based multi-objective design space exploration

The initial population is evaluated in the solution space according to the proposed resource and quality models to determine their fitness. In the selection phase, the individuals are then sorted

into different Pareto ranks by considering the dominance of each configuration based on their fitness values, and candidates from Pareto ranks are selected to perform mutation and crossover. For the selection, we are proposing a novel variant called ROI-NSGA, which is described in the following subsection, as well as the traditional NSGA-II for comparison. The genetic operations must be tailored to consider parameter interdependencies and design constraints of the targeted application. In Section 5, we describe how the genetic operations can be adjusted to respect the demands of an exemplary color processing system. Closing the loop, the newly generated offspring are evaluated again regarding their fitness, and this cycle will repeat until the stop condition is satisfied, resulting in an evolved set of Pareto-optimal configurations.

4.4.2 Region of Interest Nondominated Sorting Genetic Algorithm (ROI-NSGA)

The idea of NSGA-II is that, first, it performs the nondominated sorting on all the individuals in a generation based on their fitness values and is arranged into different Pareto ranks which can have different numbers of individuals according to their dominance level. From these Pareto ranks, the desired number of individuals is selected, starting from the first nondominated front.

However, in the final Pareto rank from which individuals are selected, there might be more individuals than the actual required number. In this case, the required number of individuals from the final Pareto rank are selected based on the crowding distance between each other in a way that it spreads the Pareto front. In NSGA-II, the first nondominated selection phase aims at converging to the optimal solution, and the second crowding distance-based selection from the ambiguous front will ensure diversity.

The classical NSGA-II aims to explore the entire design space and stretch the Pareto-optimal solutions with good diversity in each fitness dimension. Hence, it tries to improve the convergence to the optimal values equally all over the Pareto front. This is not always required in some applications when the designer has a certain threshold in mind to make valid choices regarding the acceptable loss of quality. Based on these threshold values, the designer selects the final solutions in the application. In such cases, all optimization efforts in the region beyond this threshold level are a loss of time and computing resources. To tackle this issue, we are proposing the novel ROI-NSGA in which the optimization effort is concentrated on a certain region of the solution space. Hence, the designer can define an ROI by setting boundaries in the target objectives, separating the solutions worth considering for final implementation from the rest.

The ROI-NSGA is inherited from the idea of NSGA-II. During the selection process, all Pareto ranks except the final Pareto rank are selected in a similar way in both approaches. The difference is associated with the selection of individuals from the final Pareto rank where NSGA-II always selects based on the crowding distance.

ROI-NSGA consists of 3 phases. In the first phase, it performs a normal NSGA-II selection until more than one solution has appeared on the final Pareto rank within the ROI. The duration of phase one depends on the size of the ROI in comparison to the entire design space. The probability of finding a solution early within the ROI increases with its size.

In the second phase, once more than one point is identified, a dynamic reference point is calculated. This is done by taking the mean between the ROI boundary points and either the minimum in minimization problems or the maximum in maximization problems in each fitness dimension from all the points within the ROI. Then, for all individuals in the final Pareto rank, the normal crowding distance calculation is replaced with the Euclidean distance from the reference point after normalization with the ROI boundary values. This phase would help to concentrate the selection and gather points within the ROI. In phase three, once enough points are available within the ROI for the selection from the final Pareto rank, the algorithm switches back to the NSGA-II selection. However, in this phase, the crowding distance calculation is performed only for the individuals within the ROI points. In general, these three phases alternate during the optimization process depending on the number of points gathered within the ROI. A comparison of results from both approaches is discussed with our case study in Section 6.2.

5 Image Color Processing Case Study and Approximation Techniques

This section introduces an image color processing application, based on the one proposed in [4], which serves as case study for the evaluation of the proposed approach. Furthermore, details on the employed approximation methods and the estimation of application-specific model parameters are given. Finally, the adaptation of the genetic encoding as well as the genetic operations to reflect the characteristics of the case study in the DSE is presented.

5.1 Display Rendering Pipeline

This work experimentally evaluates the proposed methodology with a display rendering pipeline shown in Figure 2 which is a subclass of image processing systems in which individual pixels are processed without any spatial correlation. The display rendering pipeline adapts an input image for display on a monitor as used by digital cameras [1]. It consists of three steps to adjust the image colors to the specification of the targeted monitor in terms of dynamic range, color space, and the electro-optical transfer function (EOTF). Therefore, a perceptually uniform color difference measure such as the ΔE metric is most suitable for assessing the application quality. A detailed description of each stage will be given in the following subsections.

In general, due to the non-linear functions in tone mapping and EOTF compensation, the system partially relies on computations with look-up tables, while the color space conversion is made up of arithmetic functions. Hence, this case study gives enough room to demonstrate the effectiveness of our approach by employing various approximations. In professional image processing cameras, the pipeline bitwidth is ranging from 8 bit in normal applications to 16 bit in high-end applications like movie cameras. For our experiments, we use a bit depth of 12 bit per color channel which is a typical value for this application. The experiments are targeted towards an implementation on the Intel Arria 10 10AS066N3F40E2SG FPGA [11] running at 266 MHz.

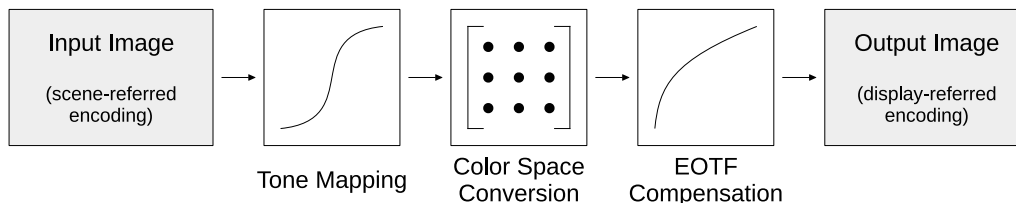


Figure 2: Display rendering pipeline used for the case study

5.1.1 Tone Mapping

Tone mapping is the first phase in the display rendering pipeline which transforms the input luminance for the natural reproduction of scenes on a display with a different dynamic range.

For this task, we use a global sigmoidal operator as proposed by Reinhard and Devlin [28]. The tone mapped values are calculated as

$$X_{out} = \frac{\text{enc}^{-1}(X_{in})}{\text{enc}^{-1}(X_{in}) + (hI_a)^k} u + v, \quad (13)$$

where $X \in \{R, G, B\}$ represents the luminance in any color channel, and h, k , and I_a are model parameters used to control the overall luminance and contrast. Because the employed tone mapping function is defined for linear intensities, the input values have to be linearized if the input image uses any non-linear intensity encoding $\text{enc}()$. With u and v , the output is scaled and shifted to the desired numerical range. The employed tone mapping curve is plotted in Figure 3.

5.1.2 Color Space Conversion

Once tone mapped, the colors are transformed into the target color space by multiplication of the RGB triplet with a 3-by-3 conversion matrix:

$$\begin{bmatrix} R_{out} \\ G_{out} \\ B_{out} \end{bmatrix} = \underbrace{\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} R_{in} \\ G_{in} \\ B_{in} \end{bmatrix}. \quad (14)$$

5.1.3 EOTF Compensation

The final step is the display's EOTF compensation by applying its inverse transform. We use the transfer function defined in IEC 61966-2-1 Amendment 1 [10]:

$$Y_{out} = \begin{cases} 12.92 Y_{in} & , \text{ for } Y_{in} < Y_{th}, \\ 1.055 Y_{in}^{1/2.4} - 0.055 & , \text{ for } Y_{in} \geq Y_{th}, \end{cases} \quad (15)$$

where Y represents any of $\{R, G, B\}$ and $Y_{th} = 0.0031308$.

5.2 Considered Approximations

The pipeline is implemented in fixed-point arithmetic. In order to trade in quality for power benefits, we apply a variety of suitable approximation methods as described below to each component c in the three steps of the pipeline. All of these methods expose parameters s_c that have to be optimized in conjunction to optimally explore the quality-power trade-off.

5.2.1 Sparse Look-Up Tables

The tone mapping and EOTF compensation are implemented as look-up tables, as these are highly non-linear functions. In the reference design of the pipeline, the tables contain the respective outputs for all possible input values, consuming a lot of BRAM. In order to reduce the memory usage, we approximate these steps by replacing the full look-up tables with sparse ones. To control the trade-off between function accuracy and memory consumption, we adopt a hierarchical segmentation approach from Lee et al. [15] with an optional interpolation. In our segmentation scheme, two uniformly distributed levels are used as illustrated in Figure 3.

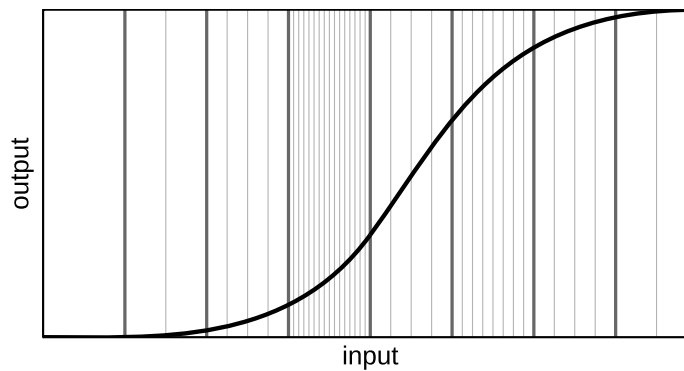


Figure 3: Function plot for the tone mapping step (parametrized as $h = 9$, $k = 0.6$, $I_a = 0.4$ and $\text{enc}^{-1}(x) = 2^x$) and exemplary hierarchical segmentation ($N_{\text{sec}} = 8$ and $N_{\text{seg}} = [1, 2, 4, 16, 4, 8, 4, 2]$)

First, the input range is split into N_{sec} uniformly distributed sections, as separated by the thick vertical lines in the figure. Then, each section $i \in [1, N_{\text{sec}}]$ is further subdivided into a variable number $N_{\text{seg}}(i)$ sub-segments. In the resulting sparse table, only one value is stored per

sub-segment. Consequently, the total number of words per sparse table can be derived directly from the approximation parameters:

$$N_{\text{total}} = \sum_{i=1}^{N_{\text{seg}}} N_{\text{seg}}(i). \tag{16}$$

5.2.2 Color Space Conversion

The data flow of the color space conversion module is illustrated in Figure 4. To approximate this step, we apply precision scaling both for the fractional bandwidth of the matrix coefficients in \mathbf{M} and for the multiplication results. The reduction in precision leads to a smaller number of bits used for the signals which reduces the number of registers in FPGA designs and decreases the size of subsequent arithmetic operations. The last bit-shift operation in Figure 4 is not subject to precision scaling as the output bitwidth of the module is fixed to 12 bits.

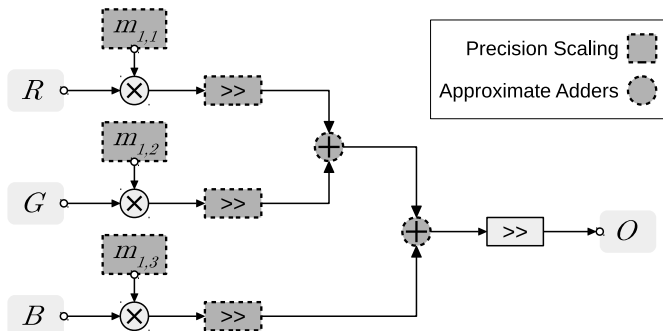


Figure 4: Data flow graph of the color space conversion step for one output channel with multipliers (\times), adders ($+$), and bit-shifts (\gg)

Additionally, the employed arithmetic units can also be replaced with corresponding approximated FPGA implementations. In this case study, we focus on the approximation of the adders and employ the dedicated DSPs for the multiplications. On FPGAs, approximate multipliers have to be implemented in logic and are not as fast as dedicated multiplication circuits for many input sizes. In state-of-the-art libraries such as [26, 34, 35], even small 8×8 multipliers show delays between 5ns and 12.5ns, which corresponds to a maximum operating frequency between 80MHz and 200MHz. The targeted color space conversion employs larger multiplications of size $12 \times x$, likely limiting performance even further. Furthermore, these libraries are generally restricted to a few specific input sizes, typically 4×4 , 8×8 and 16×16 , which restricts their adaptability to generic input sizes and therefore their potential for resource savings.

In contrast, approximate adders can easily be parameterized for various bitwidths. We use two different versions of approximate adders, namely lower-OR adder (LOA) [17] and lower-select adder (LSA), which split the carry chain and apply approximations in the least significant bit (LSB) part. In LOA, the LSB part is calculated with a simple OR operation instead of precise addition. This shortens the carry path and will result in reduced routing resources during the synthesis. In LSA, the LSB part of one of the operands is forwarded directly to the output without any additional operations. Hence, it shortens the carry chain and removes any computation in the LSB part completely. For any of the actual adders in the color space conversion module, one of these two approximated versions can be chosen, and its parameters can be varied to tune the approximation strength.

5.3 Application-Specific Model Parameters

The dynamic part of the power model requires a toggle rate estimation for the considered application. In order to estimate the toggle rate for the targeted display rendering pipeline from a set of realistic

input data, we used an image set published by Andriani et al. [1] after truncating it to 12 bit. All the images in the dataset are processed with the reference implementation and the estimated toggle rate values vary from 0.1980 to 0.2934 with an average of $\alpha_{global} \approx 0.25$.

Because the BRAMs as memory for the look-up tables in the tone mapping and EOTF compensation steps are read-only during runtime, we set the BRAM read and write rates to $\alpha_{bram_rd} = 1$ and $\alpha_{bram_wr} = 0$.

5.4 Design Space Exploration

The described approximation possibilities for the display rendering pipeline lead to a complex design space exploration problem. Table 1 shows a general overview of possible approximation methods and parameters s_c together with their bounds for all the system components C in the pipeline with bit depth b . To differentiate between the parameters s_c , we use different variables in the table. For a 12-bit pipeline, the number of design configurations would be approximately 10^{78} .

In general, the optimization problem is formulated as maximizing the image quality while minimizing the total power consumption of the system. In order to find the optimal configurations from this highly complex design space, we use MOO with GA and NSGA-II selection. As an alternative optimization approach that concentrates its search on the ROI defined by the designer, we also employ ROI-NSGA selection.

5.4.1 Objective Function

The optimization objectives are defined from power usage of the system and the quality degradation obtained from the models. As described in Section 4.2, the total power consumption of an individual system configuration is estimated from the total number of resource components. Due to the fact that the system power is reflecting the total resource consumption, we use only power as an objective among both values. However, the DSE keeps track of the resource estimation from models to ensure that the estimated resources do not exceed the available FPGA resources. Since the display rendering pipeline deals with color transformation, the quality model uses ΔE as the most suitable metric.

Table 1: List of approximation parameters of the display rendering pipeline

Parameters	Symbols and Ranges
Sparse Tables	
Interpolation types	$I \in \{None, Linear\}$
No. of sections	$N_{sec} = 2^p$, where $p \in [0, 5]$
No. of sub-segments	$\forall i \in [1, N_{sec}]: N_{seg}(i) = 2^{q_i}$, where $q_i \in [0, \log_2(\text{sizeof}(N_{sec}))]$, $\sum_{i=1}^{N_{sec}} N_{seg}(i) \geq 16$
Matrix Multiplication	
No. of fractional bits of matrix entries	$\forall i, j \in [1, 3]: F_{co}(i, j) \in [0, F_{co,ref}]$, where $F_{co,ref} = 13$
No. of fractional bits of intermediate results	$\forall i \in [1, 3]: F_{in}(i) \in [0, \max(F_{co}(i, [1, 3]))]$
Approximate adder types	$\forall i \in [1, 3], j \in [1, 2]: A_t(i, j) \in \{LOA, LSA\}$
Approximate adder LSB input selects	$\forall i \in [1, 3], j \in [1, 2]: A_s(i, j) \in \begin{cases} [0, 1], & \text{if } A_t(i, j) = LSA \\ \{0\}, & \text{otherwise} \end{cases}$
Approximate adder split points	$\forall i \in [1, 3], j \in [1, 2]: A_p(i, j) \in [0, (b + F_{in}(i))]$

The maximum ΔE calculated across the training set reflects the worst-case error generated by the approximated pipeline and is of the highest interest for the designer who typically needs to bound the quality degradation. However, during the exploration, multiple configurations might yield the same maximum error if these configurations behave the same for the worst-case input. In such a case, using the mean ΔE allows for a differentiation between these solutions, helping the optimization to select a better configuration and allowing for moving away from a plateau of configurations with the same worst-case error. Therefore, although usually not the main target of the designer, the mean ΔE estimation is included as an additional objective to guide the optimization to make finer improvements towards optimal solutions. Hence, the chosen fitness values are power, mean ΔE and maximum ΔE , and the objective function is:

$$f_{obj} = \text{minimize}(\text{maximum}(\Delta E), \text{mean}(\Delta E), \text{power}). \quad (17)$$

5.4.2 Gene and Chromosome Representation

The parameters s_c used in a given approximation component are encoded explicitly in an individual gene. A chromosome or an individual is represented as a list containing 3 sub-lists, each corresponding to the parameterization of a single step in the processing pipeline. We encode the interpolation type I of the sparse tables as $None = 0$ and $Linear = 1$. Furthermore, we encode the approximate adder type $A_t(i, j)$ of the matrix multiplication as $LOA = 0$ and $LSA = 1$. As an example for the encoding of each pipeline step:

1. Tone mapping: [1, 1, [512]] (describing 1 section of 512 sub-segments, and *Linear* interpolation)
2. Color space conversion: [[[7, 1, 3], [8, 0, 13], [8, 1, 14]], [4, 0, 2], [[1, 0], [1, 0], [0, 1]], [[1, 0], [1, 0], [0, 0]], [[10, 0], [1, 16], [6, 0]]] (each nested list describing respectively the parameters for the matrix multiplication in the order given in Table 1)
3. EOTF compensation: [0, 4, [64, 1, 64, 256]] (describing 4 sections of respectively 64, 1, 64, 256 sub-segments, and not using interpolation)

5.4.3 Initial Population

The initial population for the GA is created by concatenating a random selection of parameters s_c for each section in the table. Hence, a typical individual is:

$$\text{random}([[I^{\text{TM}}, N_{\text{sec}}^{\text{TM}}, N_{\text{seg}}^{\text{TM}}], [F_{\text{co}}, F_{\text{in}}, A_t, A_s, A_p], [I^{\text{EOTF}}, N_{\text{sec}}^{\text{EOTF}}, N_{\text{seg}}^{\text{EOTF}}]]).$$

5.4.4 Genetic Operations

5.4.4.1 Crossover The crossover operation is defined independently for each sub-list by considering the properties of each module. However, the crossover has to be done on the 3 modules simultaneously in each generation.

In the case of tone mapping and EOTF compensation, a direct crossover operation is infeasible since the size of the parent chromosomes might be different, i.e. the selected parents may have a different number of sections N_{sec} and hence also a different number of sub-segments N_{seg} . In order to avoid this problem, we either upsample or downsample the parents to the same number of sections N_{sec} so that the length of both parents would be the same. For example, as shown in Figure 5, an individual with 2 sections of 256 and 512 sub-segments can be directly upsampled to 4 sections of 128, 128, 256, and 256 sub-segments. However, downsampling needs to be performed carefully to create only feasible design points. Here, the downsampling of 2 to 1 section leads to 768 sub-segments which need to be rounded to the next higher value of 1024 in order to meet the constraint of having a power of 2. Similarly, for the upsampling, no value below 1 will be created for any sub-segment. These operations will modify the parents to the same size without losing their genetic characteristics, and a classical single-point crossover operation can then be performed directly on these parents.

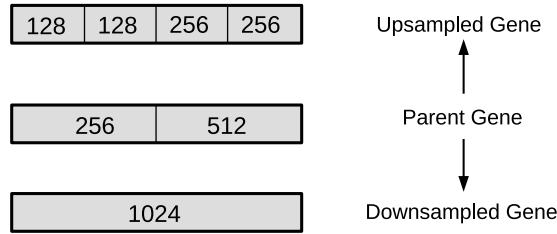


Figure 5: Upsampling and downsampling operations

In the case of the color space conversion module, individuals will always have the same list sizes and no modification is required on the parents for the classical crossover operation. However, the single-point crossover cannot be used on the sub-lists directly due to parameter dependencies in the individual color channels. The first 1x3 elements ([7, 1, 3]) in the exemplary configuration are the number of fractional bits of the matrix entries in the red channel output F_{co} . According to Table 1, the next parameter, the number of fractional bits of subsequent intermediate results F_{in} depends on the maximum of previous parameter F_{co} in the same channel. Similarly, any of the LOA or LSA adders can be chosen as adder type A_t instead of exact adders which are represented as 0 and 1 respectively in the gene. The subsequent approximate adders' LSB input selects A_s have relevance only if the selected adders are LSA. Finally, the maximum splitting point A_p also depends on previous intermediate fractional bits F_{in} in the same channel and the overall image bit depth b . Due to all these channel dependencies in the color space conversion module, a single-point crossover on the sub-lists with three channels is performed in a way that splits the parents between the channels to create only feasible offspring.

5.4.4.2 Mutation For the mutation operation, one of the three modules is randomly selected. Then, this module is mutated either fully or partially by random generation with equal probability in a way that ensures a feasible combination.

6 Evaluation

In this section, the proposed methodology is evaluated experimentally with the presented case study. First, the setup for the experiments is described in Section 6.1, including the parameters used in the DSE as well as the choice of the training data. Then, results from the experiments are reported and the findings are discussed in detail in Section 6.2.

6.1 Experimental Setup

6.1.1 Exploration Parameters

In the design space exploration setup, the optimization objectives, GA parameters, and the genetic representation and operations are the same for both of the optimization approaches with NSGA-II and ROI-NSGA. We used the $(\mu + \lambda)$ evolution strategy [33]. The population size μ is set to 50, and the number of offspring produced λ is set as 100. The crossover and mutation probabilities are set to commonly-used values 0.7 and 0.3 respectively [24]. As stopping criteria, we chose to terminate the evolution for both approaches when at least 100 000 evaluations or 1000 generations have been performed. These numbers have been selected empirically after analyzing the convergence of individual NSGA-II based experiments. We used the same number of generations for ROI-NSGA based experiments for performance comparison, which will be explained in Section 6.4.

6.1.2 Hypervolume References and Region of Interest

In order to analyze the convergence of both optimization approaches, the hypervolume indicator is estimated from all solutions identified in each generation. The hypervolume is a hybrid indicator measuring both convergence and diversity of the Pareto front [33]. Figure 6 illustrates the basic principle of the hypervolume indicator, which calculates the volume bounded by the solutions on the Pareto front and a reference point.

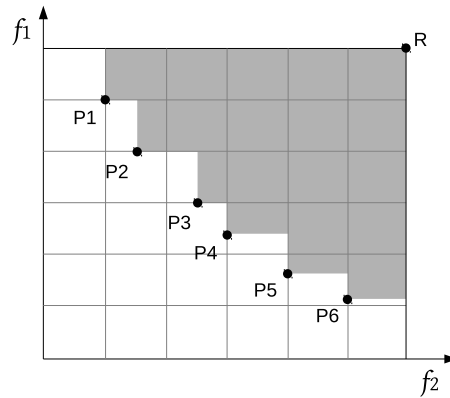


Figure 6: Example of the hypervolume indicator which is represented by the shaded area for Pareto points P1 to P6 with respect to the reference point R

Since the NSGA-II algorithm tries to explore and identify the Pareto-optimal solutions from the entire design space, a global-hypervolume reference is defined for the performance analysis. The global-hypervolume reference is set experimentally to 231.36, 84.85, and 1.18 for maximum ΔE , mean ΔE , and power respectively, which are 10% higher than the maximum identified values for each objective from multiple runs.

Studies have shown that the visual perceptibility threshold of color differences in human lies approximately at ΔE of 2.15 [32]. According to Mokrzycki and Tatol [21], a $\Delta E \leq 2$ can only be perceived through close observation and a $\Delta E \leq 1$ is imperceptible. Hence, the ROI where the designer can realistically select the configurations for the display rendering application is only a small part of the overall Pareto-optimal front resulting from the NSGA-II. We selected ROI boundary values as 5 for maximum ΔE which represents the worst-case error, 2.15 for mean ΔE , and 55.46 mW for the power which is the power consumption of the reference implementation. Hence, the performance of ROI-NSGA is estimated in terms of hypervolume from this ROI, and these boundary points are set as ROI-hypervolume reference. Finally, due to the fact that the points outside the ROI are not useful for the display rendering application, both the NSGA-II and the novel ROI-NSGA optimization approaches are compared based on the evolved Pareto-optimal solutions within the ROI in terms of hypervolume with respect to the same ROI-hypervolume reference.

6.1.3 Training Data

The error of an approximated system depends on the data it processes. When assessing the quality of a color processing pipeline, guarantees on the maximum error can only be given when all possible input colors are probed. For a bit depth of 12 bit per channel, the input space contains $2^{36} = 6.872 \times 10^{10}$ colors, and using such a large training data set makes the optimization take a prohibitively long time due to the extensive time required for the quality calculation. Hence, we use a representative subset of the input space by sampling the color space in 128 steps which uses 7 bits in the most significant bit (MSB) part in each channel, creating a training data set of $128^3 = 2.097 \times 10^6$ colors evenly spread across the color space.

Additionally, we add random noise smaller than the sampling step size to every data point in order to avoid constant data in the lower bits, thus minimizing potential effects of perfectly even

sampling. Otherwise, the DSE might find solutions that do not consider the lower bits of the input data. Instead, such solutions might just generate output data corresponding to the constant lower bits of the training data. Using such a reduced training data set significantly decreases the computational effort for quality estimation while it still covers the entire color space. It constitutes a compromise between the accuracy of the quality estimation and the time needed for it. The resulting training image is shown in Figure 7.

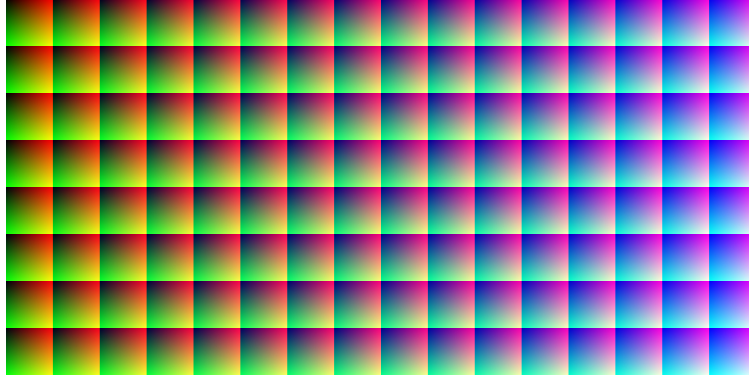


Figure 7: Training data with 128 steps used for quality estimation

6.2 Experimental Results

We performed the DSE to optimize the combined parameter settings of the display rendering pipeline with both the NSGA-II and ROI-NSGA optimization approaches as described in Section 4.4. The NSGA-II implementation is the same as employed in [19] for the same case study. As a result, a three-dimensional Pareto front with hundreds of nondominated solutions is obtained from each individual run with the objectives power, maximum ΔE , and mean ΔE . We repeated the experiments 50 times with both optimization approaches in order to ensure the consistency of the results.

Since three objectives are optimized simultaneously with the DSE, the results are depicted as two-dimensional projection in Figure 8, which shows the most relevant trade-off between estimated power consumption and maximum ΔE . The solutions depicted in this plot contain results from 6 randomly selected experiments for each of the NSGA-II and ROI-NSGA approaches. These Pareto fronts are representative of the results we obtained across all experiments. All irrelevant Pareto-optimal points with a maximum ΔE higher than 50 and the points which consume more power than the reference implementation due to the approximation overhead are not shown in the plot. In the figure, it can be seen that the estimated power consumption of configurations with more than 50mW does not have any significant improvement in maximum ΔE . Similarly, for a maximum ΔE greater than 10, the potential for power savings is also limited.

For analyzing the quality-power trade-off and for the validation of our models, 6 points are sampled from the ROI as defined in Section 6.1.2. The ROI is magnified in the inset in Figure 8, and the selected points are marked as S1 to S6 in the plot. Each of these points was synthesized, and the post-synthesis power consumption was estimated with the Intel Power Analyzer [13] after synthesis and place & route. To ensure a valid comparison, the operation temperature and the average toggle rate are set to 50 °C and 0.25 respectively in the Intel Power Analyzer to mimic the settings used for the model. For checking the quality estimation, the chosen solutions were validated with a set of high quality test images published by Andriani et al. [1].

Table 2 contains detailed information on the selected points. It lists the resource utilization for each FPGA resource type, power consumption estimated with our model and obtained after synthesis, and the quality assessed using both the training set and the test set. We also synthesized the reference design without any approximation and obtained the post-synthesis power consumption, which is 48.52 mW. The selected approximated designs save between 23.06% and 46.02% of power while preserving acceptable quality on the test set.

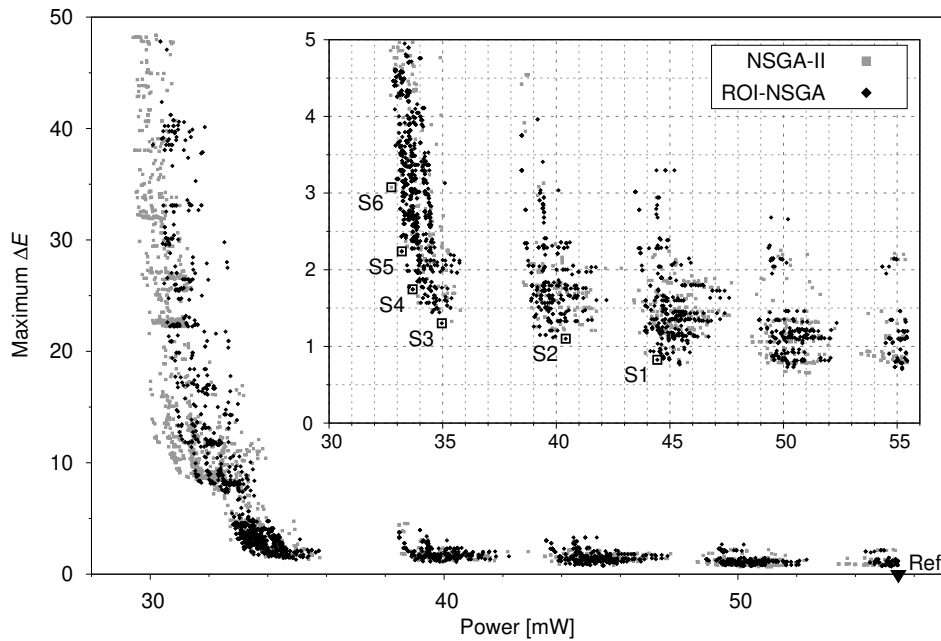


Figure 8: Pareto fronts resulting from the experiments, depicted as projection of maximum ΔE vs. power. The reference design without any approximation is indicated as Ref.

Table 2: Resource usage, power consumption and quality data of the selected points

Sel. Point Index	FPGA Resources				Power		Quality			
	DSP [Units]	BRAM [Units/Bits]	LUT [Units]	Reg [Units]	Model [mW]	Synth. [mW]	Train [max. ΔE]	Test [mean. ΔE]	Train [max. ΔE]	Test [mean. ΔE]
S1	9	12 / 182 016	250	785	44.45	37.33	0.83	0.82	0.15	0.22
S2	9	9 / 110 592	302	832	40.41	33.67	1.10	1.25	0.20	0.25
S3	9	6 / 73 728	276	795	34.96	27.93	1.30	1.39	0.21	0.27
S4	9	6 / 73 728	261	699	33.68	27.18	1.74	1.72	0.36	0.42
S5	9	6 / 73 728	199	695	33.20	26.19	2.24	2.50	0.41	0.46
S6	9	6 / 55 296	189	661	32.74	26.66	3.08	3.52	0.39	0.63
Ref	9	18 / 294 912	292	865	55.46	48.52	–	–	–	–

6.3 Model Validation

Due to the need for evaluation of many solutions during the DSE, our models are intentionally simple to allow for a fast evaluation of all probed configurations. Our power model does not involve any time-consuming synthesis and the quality model uses a relatively small training set. Running on a computer system with an Intel Core i7-8550U and 16GB DDR4 RAM, evaluating a single solution using these models takes approximately 0.9 seconds. In contrast, performing a single synthesis on the same system takes about 22 minutes. However, because of the simplifications and abstractions used in our model, their suitability for guiding the DSE needs to be validated. For that reason, the values estimated with the models are compared to the post-synthesis power report and the results from the test set for quality.

6.3.1 Power

Regarding power estimation, our proposed model omits the synthesis step to enable a fast evaluation of solutions. The results on power consumption reported in Table 2 show that our model overesti-

mates the power consumption as compared to the values obtained post-synthesis by approximately 24%. In contrast to the proposed model, the post-synthesis values obtained from the Intel Power Analyzer were calculated considering the complete netlist, including the exact routing. Furthermore, our model only considers integer unit counts for all resource types of the FPGA. The power consumption of individual DSP slices and BRAM units, however, might vary with the input bitwidths for DSP slices and the actual memory utilization in terms of bits for the BRAM units.

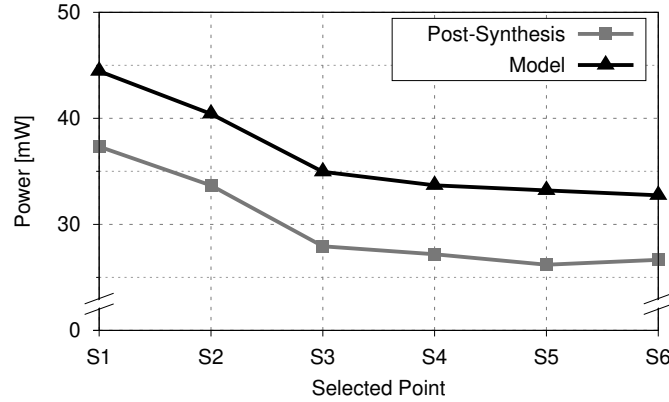


Figure 9: Comparison of the estimated power consumption with the post-synthesis results

Nevertheless, the most important aspect for the optimization procedure is whether the model retains the correct relation between different solutions, i.e. if one probed solution consumes less power than another one in the estimation, it should also consume less power in the post-synthesis results. To visualize the relations, we have plotted the power values estimated from our model together with the respective post-synthesis values in Figure 9. The plot shows that while overestimating the consumed power, the proposed model follows the same trend across the selected points. There is a small irregularity between points 5 and 6, where the model estimated point 6 to consume slightly less power, but the post-synthesis results contradict this. However, looking at the resource usage reported in Table 2, it can be seen that point 6 actually needs the same or less of every resource type. Hence, the difference in power is likely caused by differences in routing applied during synthesis. Nevertheless, the difference is very small and overall, the model is able to guide the optimization and to explore the quality-power trade-off.

6.3.2 Quality

The proposed quality assessment estimates the error with a reduced training set (cp. Section 6.1.3). Therefore, guarantees for the maximum ΔE bounds cannot be given for all inputs potentially fed to the system. However, the estimation needs to be accurate enough for the designer to be able to make a valid decision when choosing a solution. For a more detailed analysis, we plotted the estimated maximum ΔE and mean ΔE values together with the range of the respective results obtained across all 12 images from the test set for each of the selected points in Figure 10.

Comparing the maximum ΔE values in Figure 10a), we can see that the values estimated from the training set are closely matching the results obtained with the test images. In some cases, the values obtained from the test set slightly surpass the estimated bounds. However, the largest ΔE overshoot stays within 0.44 which is acceptable for practical applications.

On the other hand, regarding the mean ΔE , we observe that the results estimated with the training set translate less well to the test set. However, all of the values reported from the test set are relatively low. The mean ΔE strongly depends on the content of the processed image, i.e. if the image has lots of inputs that lead to high errors, the average error will also be high. Given that the estimation of mean ΔE values is included primarily to help the optimization to improve the solutions in finer granularity (cp. Section 5.4.1), this behavior is expected and acceptable.

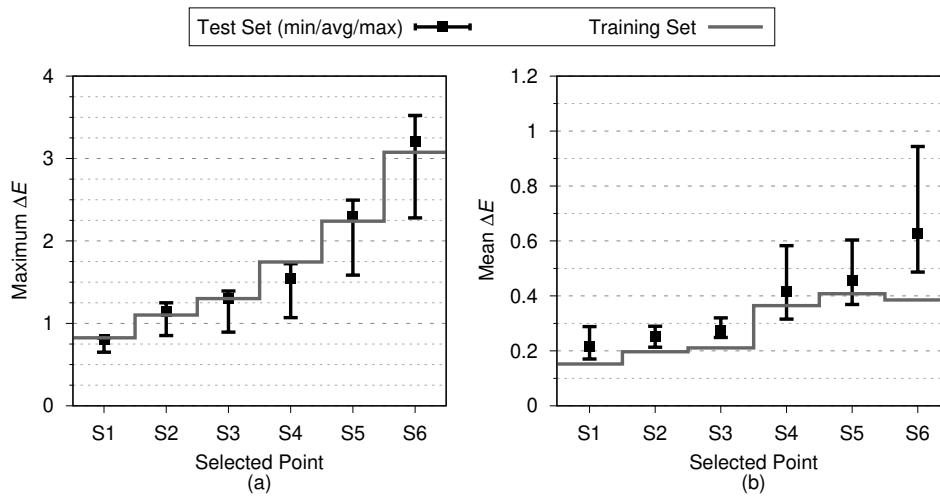


Figure 10: Validation of the quality estimation for the selected points with the test set [1]

To ensure consistent quality across different inputs, the designer’s decision on which of the considered solutions to implement is driven by the estimated maximum ΔE bound while the mean error can be considered with secondary priority.

6.4 Optimization Analysis

The validation of the proposed models with the selected points from the ROI showed that the optimization approaches are able to deliver reasonable savings in power by maintaining acceptable quality values for practical application. However, to evaluate the performance of both optimization approaches in terms of convergence and diversity, we analyze the hypervolume values obtained from the solutions evolved in each generation independently of individual optimization runs as described in Section 6.1.

6.4.1 Performance Analysis of NSGA-II Optimization

To demonstrate the performance of the state-of-the-art NSGA-II based optimization approach which equally explores and optimizes the entire design space, we plot the hypervolume from a single run with respect to the global-hypervolume reference as an example in Figure 11. From the plot, we can observe multiple phases of convergence, and the final phase of convergence starts around 850

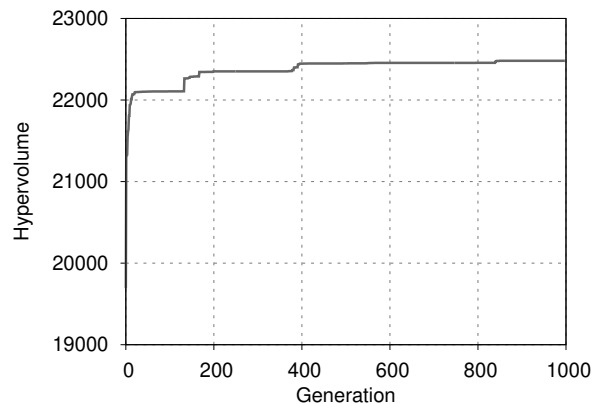


Figure 11: Convergence of the NSGA-II optimization process

generations. We empirically analyze all the hypervolume values from each individual run and set 1000 generations as the stopping criteria.

6.4.2 Performance Analysis of ROI-NSGA Optimization

The performance of ROI-NSGA based optimization which optimizes the ROI-relevant solutions can also be analyzed with hypervolume values. However, this has to be done with respect to the ROI-hypervolume reference. Figure 12 shows the hypervolume from a single run with ROI-NSGA based optimization. From this example, we can observe that the final phase of convergence is reached at around 500 generations within the ROI. By analyzing all the hypervolume values from 50 independent runs, we observed that convergence is reached in the majority of runs after 800 generations which is a good number for the optimization termination.

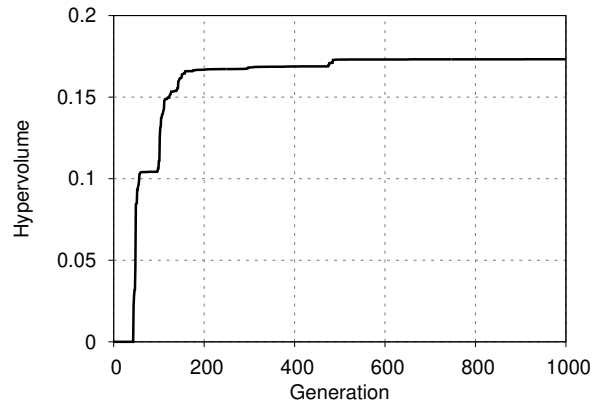


Figure 12: Convergence of the ROI-NSGA optimization process

6.4.3 Performance Comparison Between ROI-NSGA and NSGA-II

The performance analysis above describes the effectiveness of each optimization approach. However, a direct comparison is not feasible based on these numbers. Due to the fact that the applicability of the explored Pareto-optimal solutions is limited to the ROI specified by the designer for the case study, we compare the performance of both optimization approaches based on ROI-relevant solutions with respect to the ROI-hypervolume reference. To enable a consistent comparison, we plot the average hypervolume values from 50 individual optimization runs with both the ROI-NSGA and NSGA-II in Figure 13. The plot shows that the ROI-NSGA outperforms the NSGA-II in exploring the application relevant solutions from the ROI. Comparable ROI-relevant solutions from 1000 generations of NSGA-II can be efficiently obtained with less than 500 generations of ROI-NSGA, reducing the number of necessary iterations by more than 50%. In our experiments, the same hypervolume was reached with 406 generations on average. When running the experiments, we observed that the additional overhead in computation time introduced by the modified sorting algorithm was multiple orders of magnitude smaller than the time needed to assess the fitness of the individuals. Therefore, it can be assumed that the potential time savings are proportional to the reduction in the necessary number of generations.

By analyzing all the optimization results, it's obvious that even though the hypervolume of NSGA-II saturates based on the global-hypervolume reference, there is still room for improvement in the ROI. However, since it equally tries to optimize the entire design space, the possibility of obtaining further improvement would be time-consuming. At the same time, the ROI-NSGA can deliver a similar performance within half of the generations required for the NSGA-II optimization. In fact, the independent performance analysis of ROI-NSGA recommends running the optimization for at least 800 generations to achieve convergence in the ROI and a good solution close to the optimum.

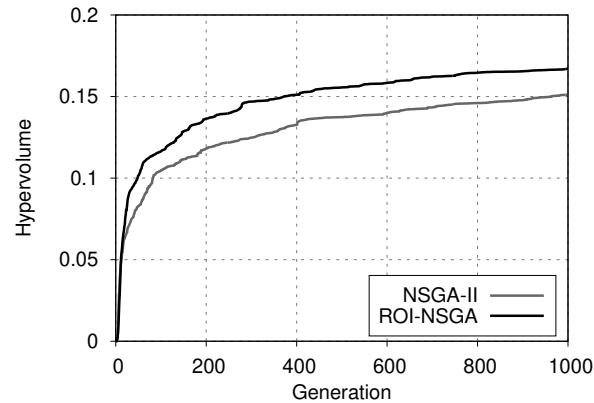


Figure 13: Average hypervolume values from 50 individual experiments with NSGA-II and ROI-NSGA

7 Conclusion

In this paper, we address the problem of optimally configuring parameters for FPGA-based signal and image processing systems that employ a combination of multiple single-purpose approximate computing techniques. Taking the architectural structure of FPGAs into account, we introduced fast and simplified models to estimate the fitness of configurations in terms of area, power, and quality without the need for time-consuming synthesis. In order to explore the trade-off provided by the approximations, we employ multi-objective optimization by means of the genetic algorithm to find a Pareto-optimal set regarding the specified goals. The optimization directly tunes the parameters exposed by the approximations which enables the approach to employ combinations of arbitrary approximation methods. To avoid spending effort on finding solutions that are irrelevant for practical implementation in applications where global search is not necessary, we introduce ROI-NSGA to guide the optimization into an ROI specified by the designer according to known acceptable thresholds in the objectives. We demonstrated the usability of our methodology on a case study of a color processing pipeline typical for digital cameras, showing that the proposed models are appropriate to assess the quality-resource trade-off and that the DSE is able to find solutions relevant for practical implementations. A comparison of the ROI-NSGA with the classical NSGA-II shows that for the examined case study, the ROI-NSGA can achieve similar hypervolume levels relative to the ROI with less than 50% of generations needed by the NSGA-II. In future work, the performance of the ROI-NSGA should be evaluated on a broader set of benchmark applications and compared with a wider range of other optimization approaches to evaluate its general usability.

Acknowledgment

This work is supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy under Grant No.: IUK574.

References

- [1] Stefano Andriani, Harald Brendel, Tamara Seybold, and Joseph Goldstone. Beyond the Kodak image set: A new reference set of color image sequences. In *2013 IEEE International Conference on Image Processing*, pages 2289–2293. IEEE, Sep 2013.
- [2] Hrishav Bakul Barua and Kartick Chandra Mondal. Approximate Computing: A Survey of Recent Trends—Bringing Greenness to Computing and Communication. *Journal of The Institution of Engineers (India): Series B*, 100(6):619–626, December 2019.

- [3] Jorge Castro-Godínez, Julián Mateus-Vargas, Muhammad Shafique, and Jörg Henkel. AxHLS: Design space exploration and high-level synthesis of approximate accelerators using approximate functional units and analytical models. In *Proceedings of the 39th International Conference on Computer-Aided Design, ICCAD '20*, pages 1–9, New York, NY, USA, November 2020. Association for Computing Machinery.
- [4] Simon Conrady, Manu Manuel, Arne Kreddig, and Walter Stechele. LCS-based automatic configuration of approximate computing parameters for fpga system designs. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '19*, page 1271–1279, New York, NY, USA, 2019. Association for Computing Machinery.
- [5] F. de Dinechin and A. Tisserand. Multipartite table methods. *IEEE Transactions on Computers*, 54(3):319–330, March 2005.
- [6] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, Lecture Notes in Computer Science, pages 849–858, Berlin, Heidelberg, 2000. Springer.
- [7] Kalyanmoy Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 635–642. Association for Computing Machinery, 2006.
- [8] Jorge Echavarría, Stefan Wildermann, Andreas Becher, Jürgen Teich, and Daniel Ziener. FAU: Fast and error-optimized approximate adder units on LUT-Based FPGAs. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 213–216, December 2016.
- [9] Ernestas Filatovas, Olga Kurasova, J. Redondo, and J. Fernández. A reference point-based evolutionary algorithm for approximating regions of interest in multiobjective problems. *TOP*, 28, November 2019.
- [10] IEC 61966-2-1:1999/AMD1:2003. Multimedia Systems and Equipment - Colour Measurement and Management Part 2-1: Colour Management - Default RGB Colour Space - sRGB - Amendment 1. Standard, International Electrotechnical Commission, Geneva, CH, January 2003.
- [11] Intel Arria 10 FPGAs. <https://www.intel.com/content/www/us/en/products/details/fpga/arria/10.html>, accessed April 26, 2021.
- [12] Intel Early Power Estimator. <https://www.intel.com/content/www/us/en/programmable/support/support-resources/operation-and-testing/power/pow-powerplay.html>, accessed April 26, 2021.
- [13] Intel Power Analyzer. <https://www.intel.com/content/www/us/en/programmable/support/support-resources/operation-and-testing/power/sof-qts-power.html>, accessed April 26, 2021.
- [14] ISO/CIE 11664-4:2019. Colorimetry — Part 4: CIE 1976 L*a*b* colour space. Standard, International Organization for Standardization, Geneva, CH, July 2019.
- [15] D. U. Lee, R. C. C. Cheung, W. Luk, and J. D. Villasenor. Hierarchical Segmentation for Hardware Function Evaluation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(1):103–116, January 2009.
- [16] Seogoo Lee and Andreas Gerstlauer. Fine Grain Precision Scaling for Datapath Approximations in Digital Signal Processing Systems. In *VLSI-SoC: At the Crossroads of Emerging Trends*, IFIP Advances in Information and Communication Technology, pages 119–143. Springer, Cham, October 2013.

- [17] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas. Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(4):850–862, April 2010.
- [18] Mariem Makni, Smail Niar, Mouna Baklouti, and Mohamed Abid. HAPE: A high-level area-power estimation framework for FPGA-based accelerators. *Microprocessors and Microsystems*, 63:11–27, November 2018.
- [19] M. Manuel, A. Kreddig, S. Conrady, N. Anh Vu Doan, and W. Stechele. Model-based design space exploration for approximate image processing on fpga. In *2020 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pages 1–7, 2020.
- [20] Sparsh Mittal. A Survey of Techniques for Approximate Computing. *ACM Computing Surveys*, 48(4), March 2016.
- [21] Wojciech Mokrzycki and Maciej Tatol. Color difference delta e - a survey. *Machine Graphics and Vision*, 20:383–411, 04 2011.
- [22] Vojtech Mrazek, Muhammad Abdullah Hanif, Zdenek Vasicek, Lukas Sekanina, and Muhammad Shafique. autoAx: An Automatic Design Space Exploration and Circuit Building Methodology utilizing Libraries of Approximate Components. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2019.
- [23] Kumud Nepal, Yueting Li, R. Iris Bahar, and Sherief Reda. ABACUS: A technique for automated behavioral synthesis of approximate computing circuits. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.
- [24] Nikhil Padhye and Subodh Kalia. Rapid prototyping using evolutionary approaches: part 1. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2725–2728, New York, NY, USA, 2009. ACM.
- [25] J. Park, J. H. Choi, and K. Roy. Dynamic Bit-Width Adaptation in DCT: An Approach to Trade Off Image Quality and Computation Energy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(5):787–793, May 2010.
- [26] Bharath Srinivas Prabakaran, Vojtech Mrazek, Zdenek Vasicek, Lukas Sekanina, and Muhammad Shafique. ApproxFPGAs: Embracing ASIC-Based Approximate Arithmetic Components for FPGA-Based Systems. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20, pages 1–6, Virtual Event, USA, July 2020. IEEE Press.
- [27] Arnab Raha and Vijay Raghunathan. qLUT: Input-Aware Quantized Table Lookup for Energy-Efficient Approximate Accelerators. *ACM Trans. Embed. Comput. Syst.*, 16(5s):130:1–130:23, September 2017.
- [28] Erik Reinhard and Kate Devlin. Dynamic range reduction inspired by photoreceptor physiology. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):13–24, Jan 2005.
- [29] L. Ben Said, S. Bechikh, and K. Ghedira. The r-Dominance: A New Dominance Relation for Interactive Evolutionary Multicriteria Decision Making. *IEEE Transactions on Evolutionary Computation*, 14(5):801–818, October 2010. Conference Name: IEEE Transactions on Evolutionary Computation.
- [30] Muhammad Shafique, Waqas Ahmad, Rehan Hafiz, and Jörg Henkel. A Low Latency Generic Accuracy Configurable Adder. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 86:1–86:6, New York, NY, USA, 2015. ACM.
- [31] S. Sinha and W. Zhang. Low-Power FPGA Design Using Memoization-Based Approximate Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(8):2665–2678, August 2016.

- [32] Mike Stokes, Mark D. Fairchild, and Roy S. Berns. Precision requirements for digital color reproduction. *ACM Trans. Graph.*, 11(4):406–422, October 1992.
- [33] El-Ghazali Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009.
- [34] S. Ullah, S. S. Murthy, and A. Kumar. SMAApproxLib: Library of FPGA-based Approximate Multipliers. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2018.
- [35] Salim Ullah, Semeen Rehman, Bharath Srinivas Prabhakaran, Florian Kriebel, Muhammad Abdullah Hanif, Muhammad Shafique, and Akash Kumar. Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators. In *Proceedings of the 55th Annual Design Automation Conference, DAC '18*, pages 1–6, New York, NY, USA, June 2018. Association for Computing Machinery.
- [36] Zdenek Vasicek and Lukas Sekanina. Search-based synthesis of approximate circuits implemented into FPGAs. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, Lausanne, Switzerland, August 2016. IEEE.
- [37] S. Venkataramani, K. Roy, and A. Raghunathan. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1367–1372, March 2013.
- [38] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan. SALSA: Systematic logic synthesis of approximate circuits. In *DAC Design Automation Conference 2012*, pages 796–801, June 2012.
- [39] Y. Vesikar, K. Deb, and J. Blank. Reference Point Based NSGA-III for Preferred Solutions. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1587–1594, November 2018.
- [40] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.
- [41] Y. Wu, C. Shen, Y. Jia, and W. Qian. Approximate logic synthesis for FPGA by wire removal and local function change. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 163–169, January 2017.
- [42] Xilinx Power Estimator. <https://www.xilinx.com/products/technology/power/xpe.html>, accessed April 26, 2021.
- [43] Siyuan Xu and Benjamin Carrion Schafer. Exposing Approximate Computing Optimizations at Different Levels: From Behavioral to Gate-Level. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(11):3077–3088, November 2017. IEEE Transactions on Very Large Scale Integration (VLSI) Systems.
- [44] Wen-Juan Yu, Shi-Wei Feng, Shi-Jie Pan, Ya-Min Zhang, and Bang-Bing Shi. Temperature distribution measurement for chips based on FPGA. In *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pages 1–3, October 2018.
- [45] Georgios Zervakis, Sotirios Xydis, Dimitrios Soudris, and Kiamal Pekmestzi. Multi-Level Approximate Accelerator Synthesis Under Voltage Island Constraints. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(4):607–611, April 2019. Conference Name: IEEE Transactions on Circuits and Systems II: Express Briefs.