HAL
open science

A Genetic-algorithm-based Approach to the Design of DCT Hardware Accelerators

Mario Barbareschi, Salvatore Barone, Alberto Bosio, Jie Han, Marcello Traiola

To cite this version:

Mario Barbareschi, Salvatore Barone, Alberto Bosio, Jie Han, Marcello Traiola. A Genetic-algorithm-based Approach to the Design of DCT Hardware Accelerators. ACM Journal on Emerging Technologies in Computing Systems, 2022, 18 (3), pp.1-25. 10.1145/3501772 . hal-03553505

HAL Id: hal-03553505

https://inria.hal.science/hal-03553505v1

Submitted on 2 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Genetic-Algorithm-Based Approach to the Design of DCT Hardware Accelerators

MARIO BARBARESCHI and SALVATORE BARONE, Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Italy

ALBERTO BOSIO, Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, France

JIE HAN, Department of Electrical and Computer Engineering, University of Alberta, Canada

MARCELLO TRAIOLA, University of Rennes, Inria, CNRS, IRISA, UMR 6074, France

As modern applications demand an unprecedented level of computational resources, traditional computing system design paradigms are no longer adequate to guarantee significant performance enhancement at an affordable cost. Approximate Computing (AxC) has been introduced as a potential candidate to achieve better computational performances by relaxing non-critical functional system specifications. In this paper, we propose a systematic and high-abstraction-level approach allowing the automatic generation of near Pareto-optimal approximate configurations for a Discrete Cosine Transform (DCT) hardware accelerator. We obtain the approximate variants by using approximate operations, having configurable approximation degree, rather than full-precise ones. We use a genetic searching algorithm to find the appropriate tuning of the approximation degree, leading to optimal trade-offs between accuracy and gains. Finally, to evaluate the actual HW gains, we synthesize non-dominated approximate DCT variants for two different target technologies, namely Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). Experimental results show that the proposed approach allows performing a meaningful exploration of the design space to find the best trade-offs in a reasonable time. Indeed, compared to the state-of-the-art work on approximate DCT, the proposed approach allows an 18% average energy improvement while providing at the same time image quality improvement.

CCS Concepts: • **Hardware** → **Circuit optimization**; **Circuits power issues**; *Switching devices power issues*; *Application specific integrated circuits*; *Full-custom circuits*; • **Applied computing** → **Multi-criterion optimization and decision-making**.

Additional Key Words and Phrases: Code Mutation, Generic Algorithm, Approximate Computing Techniques, Design Space Exploration, JPEG, Discrete Cosine Transform.

ACM Reference Format:

Mario Barbareschi, Salvatore Barone, Alberto Bosio, Jie Han, and Marcello Traiola. 2022. A Genetic-Algorithm-Based Approach to the Design of DCT Hardware Accelerators. *ACM J. Emerg. Technol. Comput. Syst.* 18, 3, Article 50 (January 2022), 25 pages. https://doi.org/10.1145/3501772

© ACM 2022. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in Journal on Emerging Technologies in Computing Systems (JETC), http://dx.doi.org/10.1145/3501772.

Authors' addresses: Mario Barbareschi; Salvatore Barone, firstname.lastname@unina.it, Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Via Claudio, 21, Naples, Italy; Alberto Bosio, alberto.bosio@ec-lyon.fr, Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, avenue Guy de Collongue, 36, Ecully, France; Jie Han, jhan8@ualberta.ca, Department of Electrical and Computer Engineering, University of Alberta, 116 Street NW, 9211, Edmonton, Canada; Marcello Traiola, marcello.traiola@inria.fr, University of Rennes, Inria, CNRS, IRISA, UMR 6074, Campus de Beaulieu, 263 Av. Général Leclerc, Rennes, France.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

Manuscript submitted to ACM

1 INTRODUCTION

With the increasingly fast-growing amount of information processed by modern computing systems, energy efficiency has become a stringent requirement. Therefore, design activities are becoming more challenging and traditional techniques seem more and more unsuitable. An increasingly popular solution is the Approximate Computing (AxC) design paradigm. AxC exploits the gap between the high accuracy level provided by a computer system and the moderate accuracy required by a given application to achieve performance gains or energy savings by carefully reducing accuracy. In this perspective, AxC is most effective when applied to applications dealing with redundant data, end-user perceptual limitations, or error resilient algorithms[12, 13, 39].

AxC has raised many design challenges due to the variety of Approximate Computing Techniques (AxCTs) [25]. Each AxCT can introduce different approximation degrees. This leads to a huge amount of different approximate system configurations. Evaluating the approximation impact on the output quality at the application level is often quite time-consuming and not trivial. Indeed, it is usually achieved through simulations or executions of the whole approximate application [12] to ensure that the quality specifications are met. An additional challenge is related to the lack of a general automation tool and a methodology for the Design Space Exploration (DSE).

One of the main fields of application for AxC is image processing: imperceptible reduction of image quality can lead to important computational resources savings. Most of the research work focuses on the JPEG compression, either considering the algorithms as a whole or its individual computational steps. Concerning the design of hardware accelerators, researchers focused on the approximation of Discrete Cosine Transform (DCT) accelerators, mainly targeting figures of merit such as circuit complexity, delay, area and power dissipation [2, 3, 21].

Unfortunately, the effect of the different approximation techniques and relative configurations (i.e., approximation degrees) are only analyzed individually and without a supporting methodology.

Conversely, in this work, we assess the impact of approximation – for different approximation degrees – on the DCT computation by performing a fully automated DSE. Starting from the DCT algorithm, we first perform an Abstract Syntax Tree (AST) analysis to gather information on the operations suitable for approximation. Then, we generate parametric approximate versions. Through the approximation parameters, we can tune the approximation degree. Finally, we build a Multi-objective Optimization Problem (MOP) to find the Pareto-optimal values for the aforementioned approximation parameters. To converge towards a Pareto front, we use a Genetic Algorithm (GA). The fitness functions driving the GA are the *quality reduction minimization* and the *gain maximization*. To perform the DSE in an acceptable time, we operate at high-abstraction level, by modeling the hardware approximation gains with an estimation function. We measure the quality reduction by evaluating the JPEG execution over a large image dataset. Finally, after the DSE, we synthesize the obtained approximate DCT configurations on both Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuits (ASICs) target technologies and demonstrate the effectiveness of the approach w.r.t. the state of the art.

The remainder of this paper is structured as follows: Section 2 reviews the existing related work; Section 3 provides preliminary technical background, while Section 4 describes in detail the proposed workflow. In order to evaluate the proposed approach, Section 5 describes the experimental setup, reports result and compares our approach with previous studies. Finally, Section 6 draws the conclusions.

2 RELATED WORK

The effectiveness of imprecise computation for error-resilient applications has been demonstrated both for software and hardware components implementing inexact algorithms [12, 39]. The use of approximation is sometimes intrinsic, e.g., in digital signal processing, where analog signals are discretized and quantized. In many scenarios, the voluntary introduction of approximation turns out to be beneficial. For instance, the perceptual limitations of human senses can be leveraged to reduce the data precision, thus to reduce the storage requirements [31] or to improve performances of multimedia and signal processing applications [34]. Likewise, executing iterative-refinement algorithms with a reduced precision of intermediate computation can improve performances, with little or even no effects on the quality of results [30]. Different AxCTs have been proposed in the scientific literature [25]. Some examples use bit-width optimization [19] and loop-perforation [36].

Employing the AxC full potential requires dealing with several challenges. (i) Approximation error-assessment: since compliance with output quality constraints has to be guaranteed, error-assessment is crucial. The actual error assessment may require the simulation of both exact and approximate applications, which is very costly. Nevertheless, in order to reduce the required effort, an estimation can be obtained by resorting to a representative workload. On the other hand, a different approach, based on Bayesian inference [37, 38], analytical models [11], machine-learning [26], and even neural networks [43] have been proposed in order to avoid simulations. (ii) The choice of metrics for error estimation: they may strongly depend on the particular application. Thus, choosing the right metric, or the right combination of metrics, is of major concern, and it is not a trivial task. (iii) Approximate configuration choice: a given application can be approximated in many different ways. Among all the different approximate configurations, choosing those providing the best trade-off between accuracy loss and resource gains is one of the major challenges in AxC. Indeed, low error and high benefits are conflicting goals.

Concerning hardware design, initial approaches – although not efficient nor scalable – addressed the aforementioned issues by manually identifying approximable sub-parts of a given application. Afterwards, there have been several attempts to define systematic and automated approaches. Ranjan et al. [33], for instance, identify combinational sub-circuits – such as arithmetic units – by looking at the Register Transfer Level (RTL) description of the circuit to be approximated, selecting the optimal quality/energy operating-point using stochastic gradient-descent, in order to maximize energy savings. Nepal et al. [27] perform AST manipulations to generate several approximate versions of a given circuit, starting from either behavioral or RTL descriptions. The variant providing the optimal trade-off between accuracy and gains is selected by means of a stochastic greedy algorithm.

The synergetic effects of multiple approximation techniques, acting at different levels, have been also investigated in the scientific literature. In [43], for instance, the precision-scaling technique is exploited in conjunction with voltage-scaling, since the former usually leads to lower circuit delay, paving the way for aggressive voltage scaling, which provides higher impacts on energy savings.

However, all the aforementioned approaches either combine multiple design objectives in a single-objective optimization problem or optimize a single parameter while keeping the others fixed. Therefore, the resulting solutions are centered around a few dominant design alternatives and do not cover the whole Pareto front [15]. Researches in [35] addressed the problem by using Cartesian genetic programming to search for Pareto-optimal approximate circuit implementations requiring progressively fewer hardware resources. Unfortunately, such approaches did not focus on complex systems, rather on arithmetic components, such as adders and multipliers, which are used as building blocks for hardware accelerators.

In [26], circuits from a library of approximate components are selected to generate an approximate accelerator for a given application. On the basis of contributions from single components, machine learning techniques are adopted to estimate the overall quality and hardware cost of the accelerator, without requiring simulations and synthesis. A similar approach has been presented in [11]. A set of analytical models of quality and resource requirements are derived for a library of approximate components. Then these are used to estimate resource needed and accuracy of accelerator designed through high-level synthesis of C language description.

In this work, we focus on a complex system, performing image processing. Specifically, we focus on the JPEG compression. In [3], a framework relying on inexact computing to perform the DCT computation for the JPEG has been proposed. The framework acts on three levels: (i) at the application level, it exploits human insensitivity to high-frequency variation to use a filter and discard high-frequency components; (ii) at the algorithmic level, multiplier-less fast algorithms are employed for the actual DCT computation on integer coefficients; (iii) at hardware level, rather than using a simple truncation for adder circuits, authors used Inexact-Adder Cells (IACs) to compute less significant bits instead of the Full-Adder Cells (FACs). Therefore, firstly, the JPEG quantization step is performed only low-frequency components of an image block; thus the high-frequency filter implementation comes down to simply setting some DCT coefficients to zero. Then, at algorithmic level, since the DCT is the most effort-demanding step in JPEG, fast DCT algorithms have been used [7–10, 14, 28, 29]. Those algorithms reduce the DCT complexity from $O(N^2)$ to $O(N)$ and require only integer additions. Finally, at the hardware level, three different IAC families are considered in [3], i.e., the Approximate Mirror Adder (AMA) [21], the Approximate XOR-based Adder (AXA) [42] and the IneXact Adder (InXA) [2]. The framework in [3] mainly aims at assessing the joint impact of those three levels of approximation. However, it presents two main weaknesses: (i) approximation is introduced by manually tuning the individual approximation parameters and (ii) output quality is assessed over only four images. In this paper, we propose a methodology to overcome the highlighted problems by defining a systematic and automatic approach for approximate hardware design.

3 PRELIMINARY TECHNICAL BACKGROUND

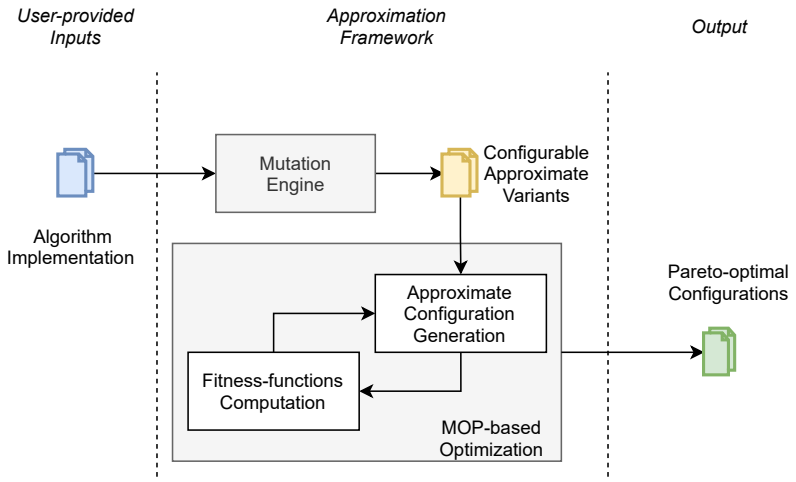


Fig. 1. The workflow of our approach.

In this Section we discuss the proposed workflow, which is depicted in Figure 1, while providing the required technical background. In particular, we firstly detail the proposed generation process of approximate variants. Secondly, we describe how we perform a DSE to converge towards the Pareto-optimal approximate variants.

3.1 Approximate variants generation

Given an algorithm implementation, in order to automatically generate approximate variants while having control on the error, gathering information on the operations suitable for approximation is necessary. We obtain this information through the analysis of the algorithm implementation's AST. Hence, we employ *mutators* [5, 6], to make systematic modifications to the AST, producing altered implementations. Mutators are defined as a set of rules to search and modify the AST. The rule definition is generally application-independent and does not require the user to know the algorithm or its specific implementation. In particular, we consider, as an approximation technique, the replacement of exact sums with approximate ones. We produce a mutator that automatically replaces exact sums with the approximate counterparts, without knowing the particular algorithm or its implementation. In this way, we obtain *approximate variants* of the original algorithm. Moreover, the approximate sums are configurable in their degree of approximation.

The number of variants and the number of configurations grow quickly with the number of operations suitable for approximation. Consider, for instance, an algorithm implementation with n approximable operations, each allowing k different degrees of approximation: $\binom{n}{j}$ different approximate variants can be defined by simultaneously approximating j operations, and k^j different approximate configurations can be defined for each of the variants. Therefore, the total number of approximate configurations is $\sum_{i=1}^{n} k^i \times \binom{n}{i}$. At this point, the main challenge is to find values for the approximation parameters leading to the Pareto-optimal trade-offs between performance gains and accuracy losses, i.e., to perform a DSE.

3.2 Design space exploration

As mentioned in the introduction, we model the DSE as a MOP. Basically, a MOP consists of a set of *fitness-functions* to minimize/maximize at the same time and a set of *constraints* to be met, as reported in (1),

$$\begin{aligned} &\Gamma = \{\gamma_i : A \rightarrow \mathbb{R},\ i = 1 \cdots k\} \\ &\Psi = \{\psi_j : A \rightarrow \{0,1\},\ j = 1 \cdots l\ \} \\ &A \subseteq \mathbb{R}^n \end{aligned} \tag{1}$$

where Γ and Ψ are the set of fitness-functions and the set of constraint-functions, respectively. While the functions of the former set assume values in $\mathbb{R}$, or its subset, the constraint functions assume either the value 1 or 0 to indicate that the constraint is or is not met, respectively.

Equation (2) describes the set of solutions for (1). For non-trivial MOPs, $|X| > 1$, where $|\cdot|$ expresses the size of the set, i.e., the number of elements it contains.

$$X = \{x \in A : \gamma_i(x) \leq \gamma_i(x'),\ x \vdash \psi_j,\ x' \neq x,\ i \in [1,k]\,,\ j \in [1,l]\} \tag{2}$$

Indeed, since different objectives (i.e., fitness functions) often represent conflicting goals, the DSE goal is to seek for a set of equally good solutions being close to the *Pareto-front* (2). Let us consider two solutions, $x, y \in X : x \neq y$, x is said to *dominate* y i.f.f. (3) holds, i.e., x shows better or equally good objective values than y in all objectives and at least better in one objective. If a solution is not dominated by any others, it is called a *Pareto-optimal* solution.

$$x \prec y \iff \gamma_i(x) \leq \gamma_i(y)\, \forall i \in [1,k] \;\wedge\; \exists j \in [1,k] : \gamma_j(x) < \gamma_j(y) \tag{3}$$

Due to the rapid growth of the size of the solution space as the number of decision variables, fitness functions and constraints increases, using exact solving algorithms turns out to be very computation-intensive. Consequently, a variety of heuristics aiming at producing an approximation of the Pareto-front have been proposed in the scientific literature. Some well-known heuristics are hill-climbing, ant-colony [18] and Evolutionary Algorithms (EAs).

GAs, a subclass of EAs, have been largely used in the literature to find Pareto-fronts for MOPs [16]. GAs are inspired by and also borrows terminology from the evolutionary theory: *mutation*, *crossover* and *selection* mechanisms cause the extinction of weak and unfit species while strong ones have greater opportunities to survive and pass their genes to future generations. In particular, we resort to Multi-Objective Evolutionary Algorithms (MOEAs). MOEAs operate on a set of *individuals*, called *population*, that evolves and, eventually, converges to a set of Pareto-dominant solutions. Each individual is represented as a *chromosome*, i.e., a data structure encoding the search space. During the evolution process, new offspring is generated either through or in combination of *crossover* and *mutation* [17]. A *crossover* takes two parent chromosomes to produce a new chromosome. Thence, we resort to the Non-dominated Sorted Genetic Algorithm-II (NSGA-II), which is widely employed as a standard approach [16].

It follows an elitist principle, i.e., the elites of a population are given the opportunity to be carried to the next generation. The evolution process starts from an *initial population*, which is typically randomly generated or seeded in areas where optimal solutions are likely to be found, and it goes through the following steps.

At the beginning, the NSGA-II builds a random initial population P_0 of size N, and sorts the latter using *non-domination sorting*, which will be presented below. Thus, each individual is assigned a *rank* equal to its non-domination level. At the first iteration, usual crossover, mutation and binary tournament selection operator are used to build the Q_0 offspring population.

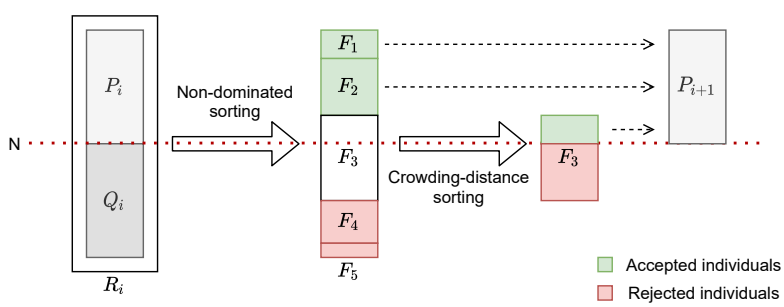


Fig. 2. The NSGA-II selection strategy.

The behavior of the algorithm during a generic i-th iteration is summarized in Figure 2: (i) a combined population $R_i = P_i \cup Q_i$ is formed exploiting crossover and mutation, then sorted on the basis of non-domination, and finally split in subsets $\{F_1, \cdots, F_m\}$; (ii) individuals from the highest rank subsets F_1 are preserved in the next iteration; if the amount of surviving individuals from F_1 is less than N, individuals from lower rank subsets $F_2 \cdots F_l$ are also preserved, and they survive for the next iteration; (iii) in order to select exactly N individuals from R_i, individuals from the last non-rejected non-domination subset F_l are sorted on the basis of the *crowding distance*, and selected using the *crowded selection operator* (4).

In order to perform non-domination sorting, for each individual $p \in P$, the NSGA-II computes (i) the non-domination count $n_p = |D_p|$, $D_p = \{q \in P : q \prec p\}$, i.e., the amount of individuals q which dominate p, and (ii) the set $S_p = \{q \in P : p \prec q\}$, i.e., the set of individuals q which are dominated by p. Individuals in the highest rank subset of R_i, i.e. F_1, have $n_p = 0$. Then, for each $p : n_p = 0$, each $q \in S_p$ is visited and n_q is decremented by one. If the term n_q becomes zero, q is placed in the second-highest rank subset of R_i, i.e. F_2. The procedure is iterated until each subset is fully identified.

Along with convergence to the Pareto-front, it is also desired that am EA maintains a good spread in the obtained set of solutions. In order to preserve diversity, the NSGA-II adopts the *crowding distance* as a metric for the density of solutions surrounding a particular solution. Such distance is the average distance of two points on either side of the considered solution, along each of the fitness functions. The computation requires sorting the population M times, according to each fitness function, in ascending order. Each time the population is sorted, the boundary solutions – i.e., solutions with the smallest and the largest fitness – are assigned an infinite distance, while all other solutions are assigned with a crowding distance equal to the absolute normalized difference of fitness of the two adjacent solutions. The overall crowding distance is the sum of individual distances corresponding to each fitness function. Using crowded distance, the usual definition of Pareto dominance (3) is slightly modified as in (4), where $x \approx y$ denotes x and y do not dominate each other and $d(\cdot)$ is the crowding distance of a given solution. The *crowded selection operator* preserves diversity by promoting solutions located in less crowded areas of the solution space.

$$x \prec_n y \iff x \prec y \cup (x \approx y \wedge d(x) > d(y)) \tag{4}$$

Modeling a specific optimization problem so that it can be resolved by a MOEA is not trivial and there are no general rules. In our case, as mentioned in Section 3.1, we approximate a number of operations in the DCT algorithm. According to their approximation degree, each of the operations contributes to error and gains differently. On the one hand, it may be easy to analytically define a relationship between the degree of approximation and the corresponding gains, by using properties of the specific approximation. On the other hand, doing the same with the error is not trivial, due to its propagation through the data-flow. Therefore, fitness functions should be defined case by case. Finally, regarding the constraints, identifying acceptable variation ranges for one or more genes or identify an error threshold is generally feasible. For instance, for an n-bits adder, the variation interval of the gene could be $\{0, 1, \cdots, n\}$ to cover the degree that goes from "no-approximation" to "totally approximate".

Finally, from a technical standpoint, approximate configurations can be represented by using a vector – i.e., a chromosome, in the MOEA context – having as many elements – i.e., genes – as operations. The value of each element represents the approximation degree for the corresponding operation.

Being the goal of the DSE to find the Pareto-optimal approximation degree for all the operations, the NSGA-II provides a suitable tool to perform the exploration of such a large design space, in a reasonable time.

4 TOWARDS APPROXIMATE DCT

In this section, firstly we discuss the mathematical steps leading to a set of linear equations to compute DCT coefficients using only additions. These equations are general and independent of the particular DCT algorithm. Secondly, we present the approximate DCT variants generation and how we model the DSE as a MOP so that it can be resolved by using the NSGA-II.

Table 1. Comparison among DCT algorithms in terms of number of operations

Method	Additions	Multiplications	Shifts	Total operations
DFT (definition)	432	192	0	624
FFT	58	6	0	64
DCT (definition)	56	64	0	120
Arai algorithm	29	5	0	34
BAS08 [9]	18	0	2	20
BAS09 [10]	18	0	0	18
BAS11 [8] (a=0)	16	0	0	16
BAS11 [8] (a=1)	18	0	0	18
BAS11 [8] (a=2)	18	0	2	20
CB11 [14]	22	0	0	22
BC12 [7]	14	0	0	14
PEA12 [29]	24	0	6	30
PEA14 [28]	14	0	0	14

4.1 Addition-based computation of DCT coefficients

As we already mentioned, the DCT computation is known to have $O(N^2)$ complexity and requires resource-intensive functional units, such as floating-point arithmetic modules. The algorithm proposed in [24] requires 11 multiplications and 29 additions to compute the one-dimensional eight-point DCT needed by the JPEG compression. It is considered the most efficient exact algorithm, since the lower bound on the number of multiplications required for such DCT computation has been proven to be 11 [20]. In order to achieve an additional reduction in resource requirements, authors of [4] moved parts of the DCT computation to the JPEG quantization step. Furthermore, transformed coefficients can be scaled and rounded such that floating-point operations can be superseded by integer ones: the resulting algorithms are significantly faster, and they find extensive use in practical applications. However, integer multiplication is still complex and resource intensive; thus, many low-complexity multiplier-less algorithms have been proposed [7–10, 14, 28, 29]. As in [4], all of these algorithms split the DCT computation into two consecutive steps: the first one is referred to as approximate-DCT, which involves only integer operations, while the second step is embedded into the quantization and takes advantages of floating-point operations the latter requires. Moreover, they all avoid computing DCT coefficients separately or iteratively. Instead, they extensively use matrix algebra and its properties. To show how the above-mentioned algorithms work, let X be an input image tile, which is a 8×8 matrix; its two-dimensional DCT transform, from now on simply DCT, is described by the following equation:

$$F = C \cdot X \cdot C', \tag{5}$$

where C is referred to as *DCT matrix*. C contains the cosine function values at the needed frequencies. The X and the F matrices have the same dimensions. The elements in F represent the DCT coefficients as the frequency progressively increases: low-frequency components are closer to the top-left corner, while high-frequency ones are placed close to the bottom-right corner. The C matrix can be split into two matrices, T and D, as reported in Equation (6).

$$F = C \cdot X \cdot C' = D \cdot (T \cdot X \cdot T') \cdot D \tag{6}$$

Different algorithms define T and D in different ways, so the number of computation operations may vary from algorithm to algorithm, as reported in Table 1. Splitting C allows integers-only matrix multiplications. Indeed, T contains only the values $\{0, \pm\frac{1}{2}, \pm 1, \pm 2\}$ and it is orthogonal, i.e. $T' = T^{-1} \Rightarrow TT' = T'T = I$, where I is the identity matrix. Note that multiplying by $\frac{1}{2}$ or 2 comes down to shifting to the right or to the left, respectively; this, at hardware level, is reduced to simple wiring. This means T allows computing the DCT using only additions. Nevertheless, the

multiplication of D in (6) still requires floating-point operations. In fact, D is a diagonal matrix consisting of values in the $[-1, 1]$ range, with $\{\frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{8}}\}$ being typical values. For this reason, resorting to properties of diagonal matrices allows obtaining the following equation:

$$F = T \cdot X \cdot T' \circ (diag(D) \cdot diag(D)'), \tag{7}$$

where $\circ$ is the Hadamard product, i.e., an element-wise multiplication. Thus, the integer null-multiplicative part $T \cdot X \cdot T'$ can be isolated from floating-point operations required by D. Afterwards, floating-point operations can be performed outside the DCT, and embedded into the JPEG quantization step, as shown in the following equation.

$$F_Q = \lceil F \oslash Q \rfloor = \lceil T \cdot X \cdot T' \circ (diag(D) \cdot diag(D)') \oslash Q \rfloor = \lceil T \cdot X \cdot T' \circ \hat{Q} \rfloor = \lceil (T \cdot (T \cdot X')') \circ \hat{Q} \rfloor \tag{8}$$

$$\hat{Q} = (diag(D) \cdot diag(D)') \oslash Q, \tag{9}$$

where $\hat{Q}$ in (9) is the *complete quantization matrix* and the $\oslash$ operator is the Hadamard division, i.e., an element-wise division. From (8), it follows $F = (T \cdot (T \cdot X')')$, which means that the approximate two-dimensional DCT transform can be computed using the one-dimensional DCT transform twice, reducing the complexity from quadratic to linear.

The only substantial difference between the different multiplier-less DCT algorithms is the T matrix. Hence, it is straightforward to derive a set of equations to calculate the one-dimensional DCT coefficients. Equations in (10), for instance, refers to the BC12 algorithm [7].

$$\begin{aligned} f_0 &= x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 & f_1 &= x_0 - x_7 \\ f_2 &= x_0 - x_1 - x_2 + x_3 + x_4 - x_5 - x_6 + x_7 & f_3 &= x_4 - x_3 \\ f_4 &= x_0 - x_3 - x_4 + x_7 & f_5 &= x_5 - x_2 \\ f_6 &= x_2 - x_1 + x_5 - x_6 & f_7 &= x_6 - x_1 \end{aligned} \tag{10}$$

Furthermore, some terms – for instance $(x_0 + x_7)$ – are involved in the computation of multiple f_i coefficients, which allows to further reduce the amount of operations.

4.2 Approximate variants generation and MOP modeling

Once the addition-based equations for the DCT coefficients are defined, simple implementations for the DCT computation algorithm can be derived, such as BAS08 [9], BAS09 [10], BAS11 [8], BC12 [7], CB11 [14], PEA12 [29] and PEA14 [28]. Within those, we introduce further approximation by replacing exact sums by configurable approximate ones. Such approximate sums allow setting two parameters, i.e., the Number of Approximate Bits (NAB) and the type of adder hardware cell to use (namely, a classic FAC or an IAC). This was the same approach adopted in [3]. However, while in [3] the approximation was manually introduced, we propose to automate the replacement process by considering the AST of the algorithm implementation. Moreover, in [3], approximation parameters were tuned manually and one at a time, keeping the others at fixed values. Unfortunately, this does not necessarily lead to Pareto-optimal solutions. Conversely, we propose to find the optimal value for the approximation parameters through an automatic DSE. We model the DSE as a MOP so that it can be resolved by using a MOEA. In the next subsections, we detail the MOP modeling.

4.2.1 Chromosome encoding. To enable the MOEA to resolve the MOP, each approximate configuration is modeled as a chromosome. Chromosome's genes represent the two aforementioned approximation parameters, i.e., the NAB

value and the type of adder hardware cell to use. Thus, if N_{op} is the number of addition required, each chromosome is composed of $2 \cdot N_{op}$ different genes. Chromosomes are provided with an additional gene representing the approximation degree for the high-frequency filter. Thus, each chromosome is composed of $2 \cdot N_{op} + 1$ genes.

Let us consider an approximate configuration from the population and its corresponding chromosome. Suppose a mutation occurs: depending on the particular gene being altered, a different approximate configuration will be generated. The latter will differ from the origin one in (i) the number of approximate bits for a certain sum operation, (ii) the adder cell to be used for a certain sum, or (iii) the number of frequencies discarded by the high-frequency filter. Despite the fact that it involves multiple genes and, hence, multiple characteristics of a configuration, the same reasoning also applies to crossover. The NSGA-II selection operator will determine which of the individuals will survive, on the basis of fitness. Fitness functions driving the DSE are detailed in the following sections.

4.2.2 Error fitness-function. For the MOEA to be able to evaluate the error entailed by the approximations, we need to define and error fitness-function to minimize.

In [2], the authors computed, through exhaustive simulations, the error-rate for different numbers of erroneous bits, for IAC adders. Unfortunately, this kind of measurements is not suitable for complex algorithms, such as the DCT, as it does not take into account the error propagation. Therefore, we resort to the whole JPEG compression, performed on a representative data set, to estimate the error. Section 5 discusses the experimental setup in more detail.

Regarding error metrics, in [3] the Maximum Difference (MD), the Average Difference (AD), the Mean Square Error (MSE) and the Peak Signal-to-Noise Ratio (PSNR) have been considered. Unfortunately, since all of them consider single pixels, these metrics turn out to be too sensible to noise [32]. Therefore, they are not particularly suited to evaluate the effectiveness of approximations on image processing algorithms. To overcome this issue, we resort to the Structural SIMilarity (SSIM) [41] to evaluate differences among images. Its formal definition is reported in Equation (11), where X and Y are two sets of data (i.e., the images), μ_X and μ_Y are their mean values, σ_X^2 and σ_Y^2 are their variances, σ_{XY} is their co-variance, L is the value range in which elements of X and Y can vary, and k_1 and k_2 are tuning parameters (typically equal to 0.01 and 0.03, respectively). Values of $SSIM(X, Y)$ span in the range $[-1, 1]$. Values of $SSIM(X, Y) \approx 1$ mean that X and Y are structurally similar, while values of $SSIM(X, Y) \approx 0$ mean that there is no similarity between the two images. Values smaller than zero are meaningless [40].

$$SSIM(X, Y) = \frac{(2\mu_x\mu_y + k_1) \cdot (2\sigma_{xy} + L \cdot k_2)}{(\mu_x + \mu_y + k_1) \cdot (\sigma_x^2 + \sigma_y^2 + L \cdot k_2)} \tag{11}$$

The effectiveness of image quality assessment increases if (11) is applied locally rather than globally since (i) image statistical features are usually highly spatially non-stationary, (ii) image distortions, which may or may not depend on the local image statistics, may also be space-variant, and (iii) only a local area in the image can be perceived with high resolution by human observers [41]. Since, in practice, a single overall quality measure of the entire image is required, the Mean SSIM (MSSIM) from Equation (12) is adopted. There, X and Y are the reference and the distorted images, respectively, x_j and y_j are the image contents at the j-th *local window*; and M is the number of local windows in the image. Typically, the MSSIM index is computed considering 11×11 Gaussian weighted circular windows rather than on 8×8 square tiles [41].

$$MSSIM(X, Y) = \frac{1}{M} \sum_{j=1}^{M} SSIM(x_j, y_j) \tag{12}$$

As SSIM, the lower the MSSIM index the lower the similarity between X and Y sets; thus, in order to define a suitable fitness-function for the MOEA to minimize error, we adopt the Structural DISSIMilarity (DSSIM) – $DSSIM(X, Y) = 1 - MSSIM(X, Y)$. In particular, we compute the $DSSIM$ between a standard JPEG compressed image X and an image Y which is obtained by using a certain approximate configuration of a given approximate algorithm. Both X and Y originate from the same non-compressed source image. We perform this operation for several images and use the average $DSSIM$ as final error fitness-function.

4.2.3 The reward fitness function. In order to accurately assess resource savings, area, power consumption and maximum clock speed should be measured. Unfortunately, this would require the synthesis and simulation of large hardware designs. Thus, performing a hardware synthesis for each design explored in the DSE is a very time-consuming process. Therefore, we resort to a gain estimation to drive the DSE. In particular, we estimate the gain (from now on 'reward') from the number of transistors required to implement an inaccurate cell, using the data from [3]. This constitutes a good predictor for area and power gains, as experimentally evaluated in Subsection 5.2, for both ASIC and FPGA. For convenience, in Table 2 we report, from [3], the number of transistors required to implement inaccurate cells of the mentioned IACs. Concerning the operating frequency, in this case the approximation does not entail any change, as further explained in Subsection 5.1.

Table 2. Transistor count from [3] for inexact-adder cells

Cell	Full Adder	AMA1	AMA2	AMA3	AMA4	AXA1	AXA2	AXA3	InXA1	InXA2	InXA3
Transistors	58	20	14	11	14	8	6	8	6	8	6

Let us detail the reward function. Let N_{op} be the number of operations required to compute the single-dimensional DCT and let nab_i be the NAB for the i-th addition. We compute the total number of saved transistors as

$$\sum_{i=0}^{N_{op}-1} nab_i \cdot (T_{FA} - T_{IAi}), \tag{13}$$

where T_{FA} and T_{IAi} are the number of transistors required by the FAC and the i-th IAC, respectively. Finally, since the number of additions required by each algorithm varies, we use a normalized measure, as reported in the following equation:

$$\rho = \frac{1}{2 \cdot N_{bits} \cdot N_{op} \cdot T_{FA}} \sum_{i=0}^{N_{op}-1} nab_i \cdot (T_{FA} - T_{IAi}) \tag{14}$$

where N_{bits} and N_{op} are the number of bits on which each of the sums is expressed and the number of sums required for the DCT computation, respectively.

Thanks to the so-defined reward function estimation, we are able to execute the DSE without performing any time-consuming syntheses of the approximate designs explored. This allows us to explore a lot more approximate designs, which would be infeasible otherwise. As a result of the DSE, we obtain the Pareto-optimal approximate designs in terms of accuracy and gains. Hence, we actually synthesize these designs to a particular technology.

5 EVALUATION AND EXPERIMENTAL RESULT

In this section, we firstly describe the hardware implementation of the DCT algorithm. Secondly, we validate the suitability of the chosen reward fitness function. Thirdly, we show our experimental setup and related result. Finally, we perform a comparison with previous work.

5.1 DCT algorithm hardware implementation

In order to be able – at the end of the DSE – to measure the final gains, we encoded all the above-mentioned DCT algorithms in VHDL. Such implementations guarantee high flexibility: they handle the configuration of both the type of adder cells to use for each addition and the number of bits to approximate (NABs). This allows the synthesis of any solution eventually found in the DSE process. VHDL implementations follow Equation (8), which allows splitting the two-dimensional DCT into two consecutive one-dimensional DCTs, separated by a transposition block, which transposes the signals. The transposition block implementation in hardware comes down to being just wiring.

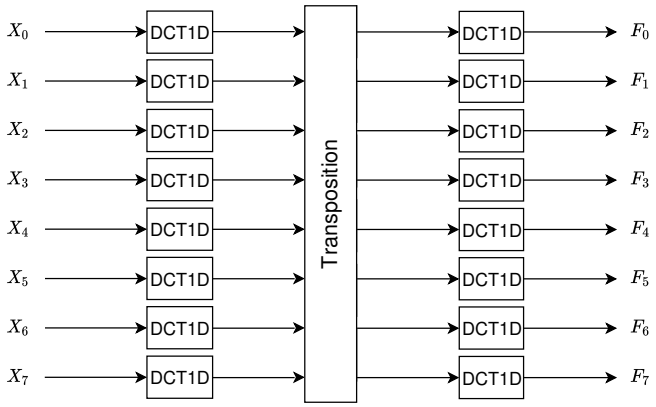


Fig. 3. RTL block schema for the BC12-2D hardware implementation

A block schema of the two consecutive one-dimensional DCTs is depicted in Figure3: $X_1, \cdots, X_7$ represent the rows of the image tile being transformed, while $F_1, \cdots, F_7$ represent the rows of the transformed block. A RTL schema of the single dimensional DCT computation block (DCT1D) is shown in Figure 4; without loss of generality, the schema refers to BC12 [7], since the differences between different algorithms are negligible.

The architecture of the one-dimensional DCT computing block is pipelined, with pipe registers separating the adders needed for the partial-sums computation. The one-dimensional DCT has three clock cycles latency; thus, the whole two-dimensional DCT block is six clock cycles latency. Each of the partial sums is performed using a configurable approximate adder. The scheme of a configurable approximate adder is depicted in Figure 5: it is a ripple-carry adder whose least significant bits are computed by IACs, while the most significant ones are computed by classical FACs. The number of approximate sums, i.e. IACs, is configurable by means of the NAB parameter.

The DCT is computed on 8×8 image tiles, each one made of three different color channels. Each element's value spans from 0 to 255. For this reason, each one of the single-dimensional DCT output terms can be expressed, at most, as the sum of eight elements. Therefore, the maximum value for the single-dimensional DCT terms is $8 \times 255 = 2040 < 2048 = 2^{11}$. As a consequence, the two-dimensional DCT output terms can have a maximum value of $8 \times 2040 = 16320 < 16384 = 2^{14}$.

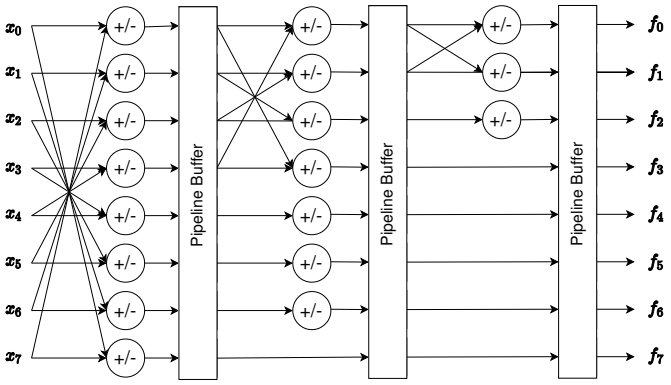


Fig. 4. RTL block schema for the BC12-1D hardware implementation

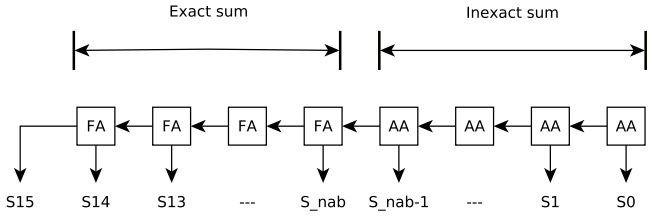


Fig. 5. Inexact ripple-carry adder

As a result, 14 bits turn out to be sufficient to represent the DCT frequency coefficients. It is also worth highlighting that replacing FACs with IACs leaves the overall structure unchanged. As anticipated in Subsection 4.2.3, this means that the approximation does not affect the latency nor the operating frequency of the device. Therefore, we can conclude that the maximum frequency of operation depends entirely on the target technology. In order to compute the maximum operating frequency for our hardware designs, we performed a preliminary FPGA and ASIC synthesis, varying NABs and IACs, while targeting a Xilinx Zynq-7020 FPGA and a 65nm Fin Field-Effect Transistor (FinFET) technology. Preliminary synthesis confirmed our aforementioned observation: approximation does not impact on the operating frequency, which depend only on the target technology. Synthesis tools reported a maximum operating frequency in the 250MHz and 770MHz range for FPGA and ASIC, respectively.

5.2 Reward fitness function evaluation

As mentioned in Section 4.2.3, the reward fitness-function described in Equation (14) performs an estimation of the approximate designs gains. As the main advantage, such an approach enables an immediate gain estimation, without the need to resort to highly time-consuming circuit syntheses. This paves the way to a much wider exploration, thus to a more thorough DSE.

Despite such an advantage, it is not trivial to reveal the correlation between transistor reduction and FPGA resource savings. In order to validate Equation (14) as suitable gain estimator for FPGA, we conducted two preliminary experiments: (i) we synthesized several configurations of approximate adder by varying the IAC to be used and NABs in order

to appreciate programmable resources overhead; (ii) we further synthesized several approximate configurations of DCT hardware accelerators on both a Xilinx Zynq-7020 FPGA and 65nm FinFET [22] technologies. In these experiments, we did not measure the accuracy, since we are only interested in resource savings.

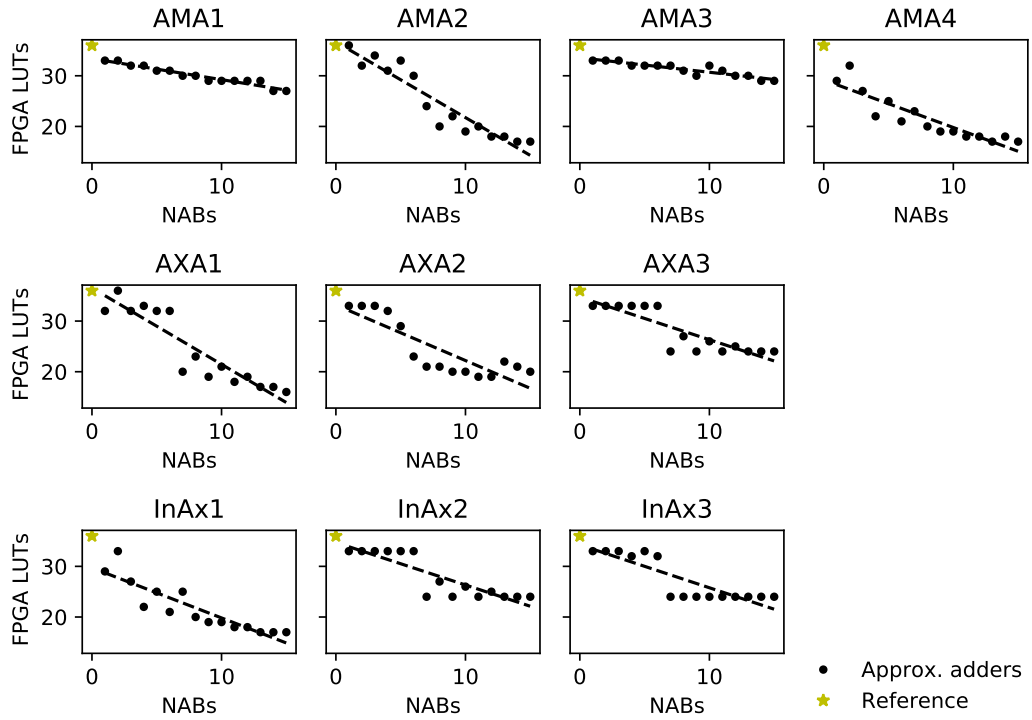


Fig. 6. Area requirements of 16 bits adders on FPGA, varying the NABs

As reported in Figure 6, there is an appreciable overhead reduction, in terms of occupied Look-Up Tables (LUTs), for any adder using IAC as NABs increase. Indeed, Boolean minimization process, performed during synthesis, gets more possibilities to reduce the size of circuits in terms of literals, fan-in and number of LUTs since truth tables of IACs are designed to simplify Boolean expressions [3].

Furthermore, we used different DCT algorithm hardware accelerators from the literature (BAS08 [9], BAS09 [10], BAS11 [8], BC12 [7], CB11 [14], PEA12 [29] and PEA14 [28]). For each accelerator, we gradually increased the NABs to observe the corresponding hardware resources trend.

Figures 7 and 8 report the amount of LUTs and a comparison between reward estimation given by Equation (14) and the actual reward on FPGA, respectively, for each of the considered DCT algorithms. As for the former, an appreciable resource overhead reduction is observable; as for the latter, the actual reward trend is sufficiently close to the predicted one. As one can notice, the predicted reward from Equation (14) has the same slope as actual reward trend. Furthermore, the predicted reward is almost a lower bound for the actual reward, which is significantly relevant in order to achieve a fair DSE.

Finally, we measured the Mean Absolute Percentage Error (MAPE) between the reward values predicted by Equation (14) and the actual ones from the syntheses, for both FPGA and ASIC. The measured MAPE ranges between [0.33%,

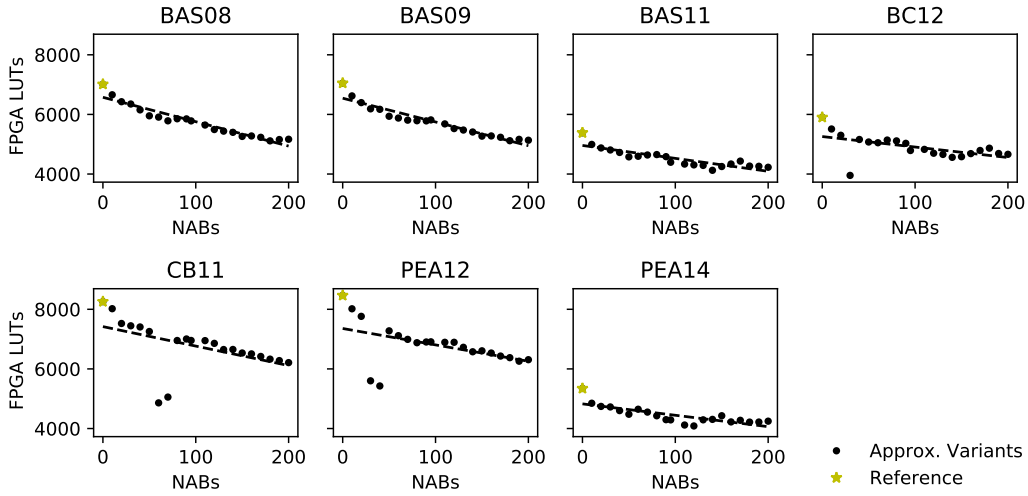


Fig. 7. Area requirements of DCT algorithms on FPGA, varying the NABs

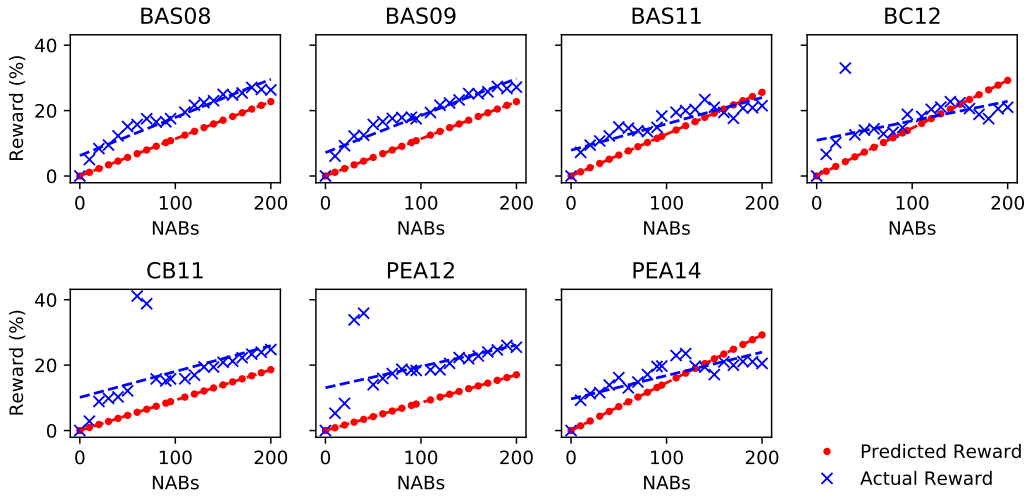


Fig. 8. Expected vs. actual reward for DCT algorithms on FPGA, varying the NABs

0.65%] and [0.82%, 2.44%], respectively for ASIC and for FPGA. Such low values of MAPE reveal that Equation (14) is an accurate estimator for the involved reward. Therefore, we concluded that Equation (14) is a suitable estimation to forecast approximation gains.

5.3 Experimental Setup

In this Section, we describe the experimental setup that we used in order to evaluate our approach. We considered seven different DCT algorithms and ten types of IACs. As for the DCT algorithms, we considered BAS08 [9], BAS09 [10],

BAS11 [8], BC12 [7], CB11 [14], PEA12 [29] and PEA14 [28]. As for the IACs families, we considered AMA [21], AXA [42] and InXA [2]. The considered DCT algorithms and approximate adders are the same considered by the authors of [3].

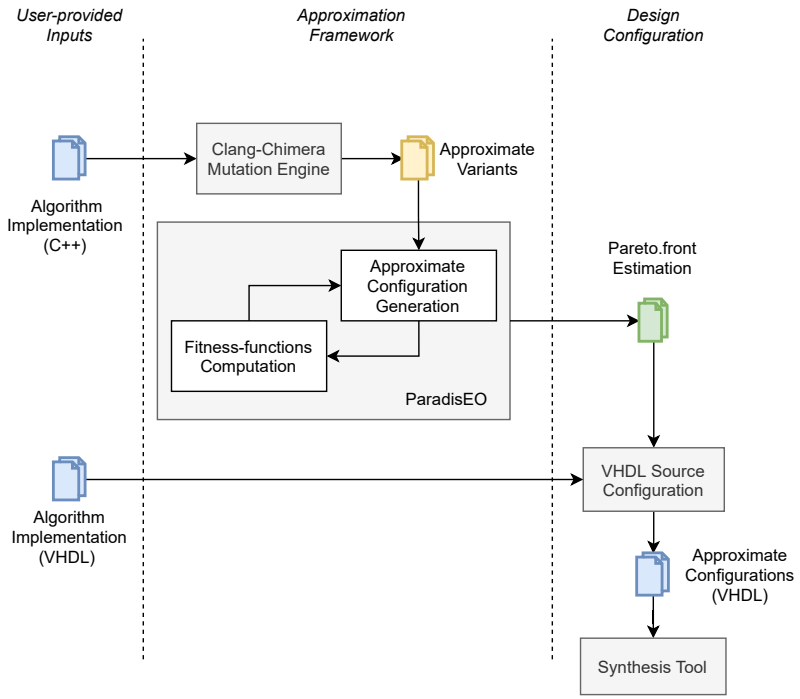


Fig. 9. Actual workflow of our approach.

Figure 9 sketches our workflow as a whole: to speed up the simulation process, we modeled each of the above-mentioned multiplier-less DCT algorithms by using C/C++ implementations straightly derived from equations (8) and (10). Starting from such implementations, the generation of approximate variants is performed using the Clang-Chimera tool, which is a Clang/LLVM-based C/C++ source-to-source mutation engine part of the 𝕀DE𝔸 framework [5, 6]. For each DCT algorithm, the Clang-Chimera tool produces mutated sources which allow configuring, for each of the sums, both the NABs and type of adder cell to use (i.e., either FAC or IAC). Furthermore, as mentioned in subsection 4.2, we modeled the DSE as a MOP. The MOP resolution is performed by using the ParadisEO framework, a template-based evolutionary computation library [23]. In order to perform the hardware synthesis, non-dominated solutions from the DSE are employed to configure the VHDL implementation discussed in Section 5.1. Finally, we measure actual gains.

To find a suitable MOEA configuration, we conducted several DSE campaigns with different MOEA parameters. As a result, we deduced two things: (i) to obtain a populous frontier and avoid local sub-optimum, we need to increase the initial population size as much as possible; (ii) to avoid long-run exploration around local sub-optimum, mutations have to take place frequently. Hence, we set our MOEA parameters as follows: initial population equals to 2000 individuals, mutation and crossover probabilities set to 0.7 and 0.9, respectively. We did not set any maximum error threshold.

Details on the fitness-functions employed for the DSE have been provided in Section 4.2.2 and Section 4.2.3. As mentioned in Section 4.2.2, we resort to the whole JPEG compression, performed on a representative data set, to estimate

the error. The considered data set [1], consists of 44 different images, covering a wide set of common features, including among others a flat gray scale, foreground subject with a messy background, and high contrast images.

5.4 Experimental Result

In this Section, we report experimental results. Firstly, we discuss the DSE results. Secondly, we discuss both silicon die area and power consumption reductions we achieved for designs resulting from DSE, synthesized on both FPGA and ASIC.

5.4.1 DSE results. Figure 10 reports the Pareto front provided by the MOEA for all the considered algorithms. The reference is, for each algorithm, its non-approximate implementation, depicted as a gold star. It is important to bear in mind that such algorithms are non-exact DCT versions (see Section 4) and that the JPEG implemented with a non-exact DCT algorithm produces lower-quality images compared to its exact version. For this reason, the reported non-approximate solutions exhibit already some error. Their reward value is zero, since they do not use any IACs, so they do not achieve any approximation gain according to Equation 14. As envisioned, the graphs highlight increasing expected rewards as the error increases.

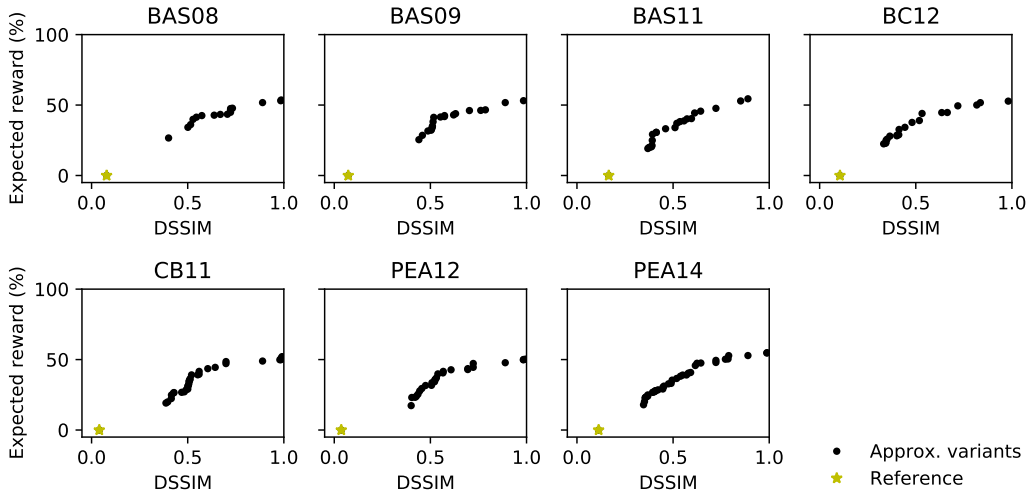


Fig. 10. Pareto-front estimation provided by the MOEA

Concerning the exploration time, the DSE has been conducted on a host PC equipped with 16 GB of RAM and an Intel i7-3770 CPU running at 3.9 GHz. On that hardware platform, the DSE for each algorithm took about 20 hours to complete. Therefore, thanks to the proposed fitness-function (Equation 14), we only needed few days to complete the exploration for all the algorithms. Indeed, by simply evaluating the proposed fitness-function, we avoided performing a circuit synthesis to compute the reward. In fact, on the same hardware platform, the ASIC synthesis tool requires, on average, of 22 minutes to accomplish a single synthesis, while the FPGA tool needs 30 minutes (17 for synthesis and 13 for technology mapping), on average. Considering that the MOEA configurations had a population of 2000 individuals (i.e., 4000 circuit variants to synthesize at each iteration), performing the first of the seven iterations alone would have required $22 \times 4000 = 88000$ minutes – i.e., about 60 days – for ASIC technology.

It is worth noting that exhaustive DSE is undoubtedly unfeasible, even in the case evaluating a single solution requires negligible time, since the size of solution spaces ranges between $2.66 \times 10^{49} \approx 2^{164}$ and $1.081 \times 10^{80} \approx 2^{265}$.

After the DSE, to correctly evaluate the final gains, we synthesized the obtained approximate configurations to both ASIC and FPGA technologies. Over all the algorithms, the total number of obtained non-dominated approximate configurations to synthesize was 164, i.e., $\approx$ 24 per each algorithm, on average. For the reader convenience, in the following figures we plotted the experimental result data along with the corresponding first-order interpolation to highlight the trend.

5.4.2 ASIC Synthesis. We synthesized all the obtained non-dominated approximate configurations to ASIC, by using the 65nm FinFET [22] technology and the Cadence *Genus Synthesis Solution* tool. We resorted to the synthesis reports for the silicon die area of the approximate configurations. In Figure 11, we report the result. Concerning the power

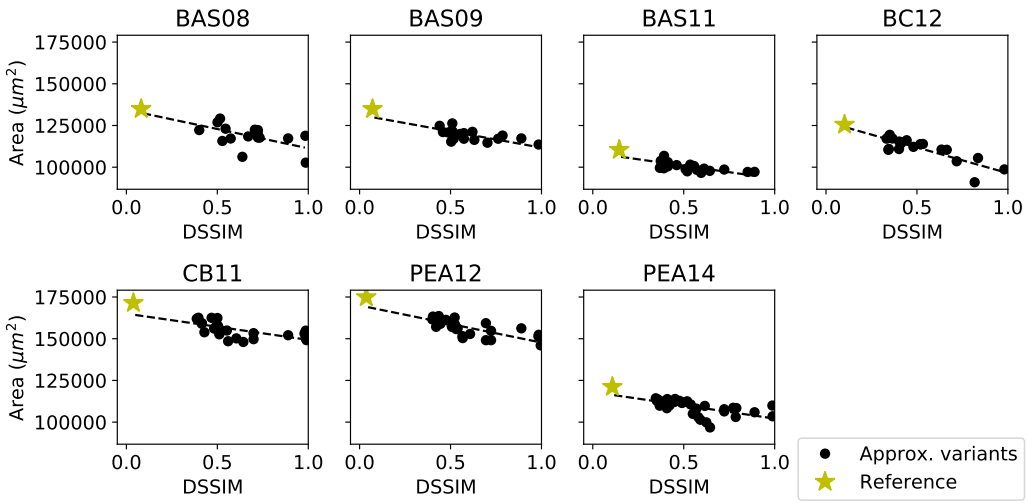


Fig. 11. ASIC silicon die area (μm^2)

consumption, to determine whether the synthesis power report provides a satisfying accuracy, we simulated the whole workload for two algorithms (BAS08 and BAS09) and collected the resulting power consumption. As a result, we realized that the difference between the power consumption resulted from the workload simulation and that estimated by the synthesis tool only differed by 5%, on average. We considered the synthesis report accuracy sufficient, thus in Figure 12 we show the power results from the synthesis report. Kindly note that the scale on the left axis (static power) is different from the scale on the right axis (dynamic power).

Power savings are achieved due to both the reduced area and the lower switching activity that IACs exhibit w.r.t FACs, as also reported in [3].

It is worth highlighting that the trends shown in Figures 11 and 12 are perfectly in line with the trend predicted by our approach (see Figure 10). Indeed, higher reward in Figure 10 corresponds to lower area/power in Figures 11 and 12.

For the reader convenience, Table 3 reports a summary of the minimum and maximum area/power savings we achieved during the experimental campaign while targeting ASIC.

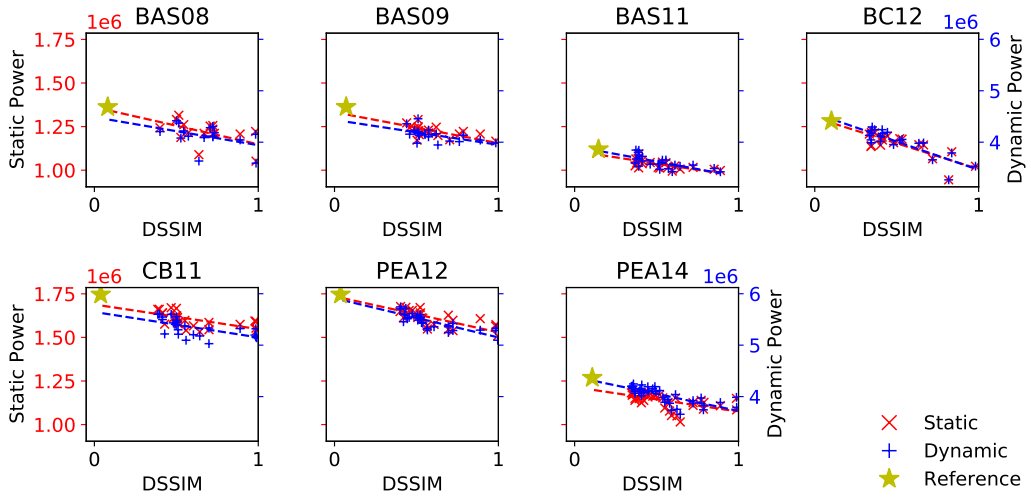


Fig. 12. Power consumption estimation for ASIC (nW)

Table 3. Minimum and maximum savings while targeting ASIC

Algorithm	Area Savings (%)		Power Savings (%)	
	min	max	min	max
BAS08	9	25	5	20
BAS09	5	15	5	15
BAS11	5	12	9	13
BC12	6	27	6	25
CB11	5	17	3	10
PEA12	7	17	5	15
PEA14	5	23	4	18

5.4.3 FPGA Synthesis. We synthesized all the obtained non-dominated approximate configurations to a Xilinx Zynq-7020 MPSoC. To get a fair estimation of hardware requirements, we used only its embedded FPGA and inhibited Digital Signal Processors (DSPs) usage.

Figure 13 reports synthesis result in terms of number of LUTs for all the considered algorithms. As expected, approximate solutions require less resources than the precise implementation, as highlighted by the decreasing general trend.

In order to correctly evaluate energy savings, we performed a post-synthesis timing simulation, using the Dynamic Power Analysis tool provided by the Xilinx Vivado. In this case, since the synthesis report has a very low confidence level for power consumption estimation, we resorted to a workload simulation for all the solutions the DSE provided, for all the algorithms. In this way, we achieved a high confidence level power estimation.

Figure 14 shows static and dynamic power consumption for all the algorithms.

The static power of the FPGA is largely caused by the fabric of the device and does not directly depend on used resources, while dynamic one is directly linked to the user design, due to the input data pattern and the design internal activity. Being our hardware implementations of approximate DCT characterized by low overhead, i.e., device resources usage falls between 6 and 13%, it is necessary to split power consumption in static and dynamic since the former turned out to be about an order of magnitude greater than the latter one for the target FPGA device.

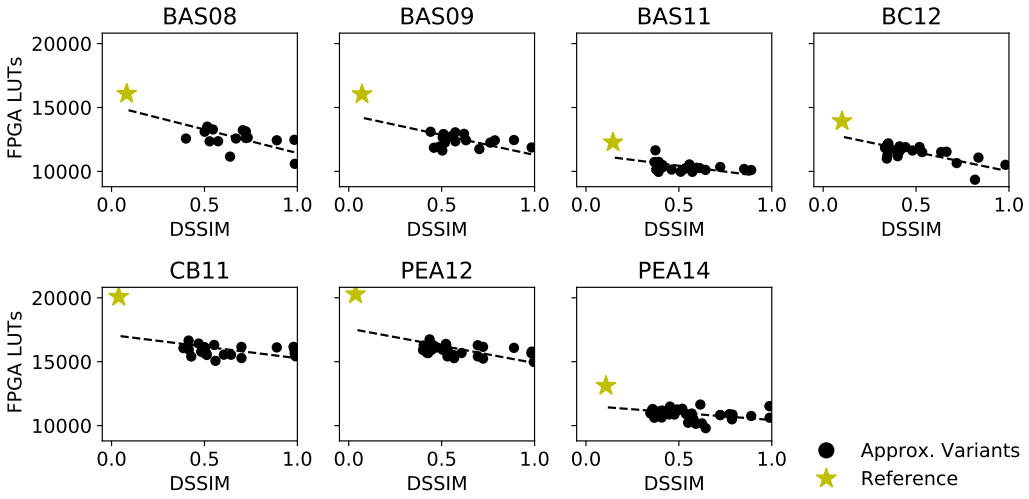


Fig. 13. FPGA resource requirements

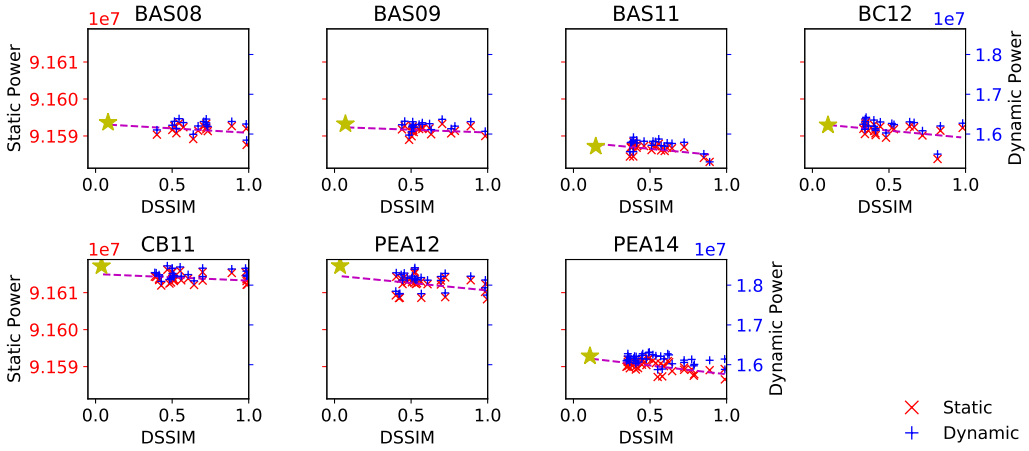


Fig. 14. Power consumption estimation for FPGA (nW)

Also in this case, power savings are achieved thanks to both the reduced total area and the logical structure of IACs: FPGA LUTs implementing IACs have a lower switching activity than those implementing FACs, as reported in [3].

As in the ASIC case, also for FPGA the trends shown in Figures 13 and 14 are perfectly in line with the trend predicted by our approach (see Figure 10). Indeed, higher reward in Figure 10 corresponds to lower area in Figures 13.

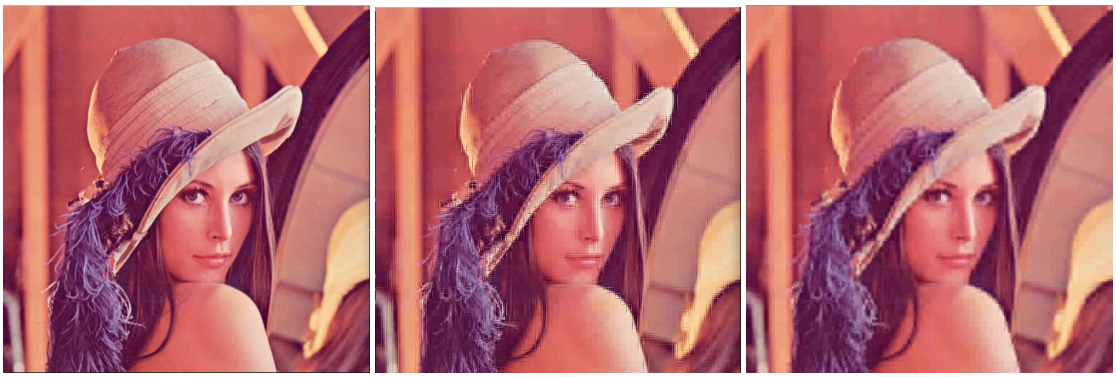
As done for ASIC, we report a summary of the minimum and maximum area/power savings we achieved during the experimental campaign while targeting FPGA in Table 4.

5.4.4 Visual Test. Since JPEG belongs to image processing domain, we also provide a visual test: Figure 15 shows, from left to right, the standard JPEG-compressed image of Lena and Baboon, the same images compressed using the exact

Table 4. Minimum and maximum savings for FPGA synthesized approximate configurations

Algorithm	LUTs Savings (%)		Dynamic Power Savings (%)	
	min	max	min	max
BAS08	27.5	48.2	0.1	3.4
BAS09	32.6	42.5	0.3	1.8
BAS11	30.6	40.4	0	2.5
BC12	31.1	50.8	0	5.2
CB11	30.9	42.7	0.5	2.6
PEA12	31.3	42.7	0.7	4.4
PEA14	29.2	44.5	0.1	2.8

version of the BC12 algorithm [7] – which exhibit a DSSIM of 0.10, and requires 125473.92 μm^2 and 5691946 μW when implemented on ASIC, or 5902 LUTs and 107933980 μW while targeting FPGA – and, finally, the ones compressed with its approximate variant having 0.33 as DSSIM value and 0.22 of reward, which correspond to 8362.64 μm^2 and 352.711μW saved for ASIC and 1846 LUTs and 94506.744μW saved for FPGA. As the reader can easily figure out, the quality differences are barely perceivable.



(a) Visual test with Lena

(b) Visual test with Baboon

Fig. 15. Visual test

5.5 Comparison with previous work

In this subsection, we compare the results obtained with our approach with those obtained in the work in [3]. Authors of [3] estimated gains G through the following equations:

$$G = \frac{V_i - V_e}{V_e} \tag{15}$$

$$V_i = P_i \cdot nab + (N - nab) \cdot P_e \tag{16}$$

where V_i and V_e represent the average energy required to perform an addition, by an inexact N-bits adder and by an exact N-bits adder respectively, P_i and P_e represent the average energy required by a single IAC and by a full-adder cell, respectively. Values of P_i and P_e used in [3] were measured by using the 45nm Complementary Metal-Oxide Semiconductor (CMOS) technology and are reported in Table 5. Such equations have the same goal as Equation (14), i.e., predicting the gains achieved thanks to the approximation. While Equations (15) and (16) take into account the energy consumption parameters of the individual adder cell, Equation (14) takes into account only the number of transistors.

Table 5. Energy consumed by a single adder cell from [3]

Cell		FullAdd	AMA1	AMA2	AMA3	AMA4	AXA	InXA1	InXA2	InXA3
Energy(fJ)	Avg.	0.9267	0.513	0.6631	0.6649	0.478	0.4042	0.1535	0.0563	0.3409
	Max.	2.3668	0.9794	0.7203	0.7116	0.6271	0.8924	0.2096	0.1291	0.4211

In [3], authors performed a manual exploration. In particular, firstly they tried different IACs and decided to always resort to InXA2 in their experiments, based on its energy delay product. Then, they tried different NAB values for the InXA2 adder and finally set it to 4 for all the experiments (i.e., for all the DCT algorithms). Besides, they used the PSNR metric to measure the JPEG error entailed by the approximate DCT variants. Conversely, we adopted the DSSIM index as error metric – which is more suitable for image processing – and we let the MOEA decide which inaccurate cell to use and how many bits to approximate (i.e., the NAB parameter) for each of the sums. A minor difference concerns the implementation of the adders: while 32-bit adders were considered in [3], we considered 14-bit adders.

In order to effectively compare the two studies, it is necessary to place them under the same conditions. Thus, we executed the JPEG algorithm on the same four images considered in [3] – i.e., Lena, Cameraman, Boat and Pepper – by using the approximate DCT variants obtained with our approach and computed the PSNR metric. Hence, we computed energy savings according to Equation (15), considering 32-bit adders. Figure 16 shows the obtained results. Concerning both energy consumption and PSNR, our approach allowed a significant improvement for all the considered algorithms compared to the approach adopted in [3]. In detail, our approach allowed an absolute improvement spanning from 15.69% to 20.15% (average 18.38%) concerning the energy gain and from 5.24 dB to 14.88 dB (average 7.91 dB) concerning the PSNR. Therefore, with our approach we were able to produce higher quality images, i.e., with less error, while consuming less energy. This is the result of the thorough DSE made possible by the proposed approach. Indeed, using a MOEA allows performing a multi-objective optimization more efficiently and automatically. Moreover, not needing to synthesize each approximate variant allows exploring more extensively the design space in a reduced time.

6 CONCLUSION AND FUTURE WORK

In this manuscript, we presented a novel generic and fully automatic approach for the approximation of DCT hardware accelerators. It enables automatic approximate DCT variant generation and automatic space exploration, by using the GA approach.

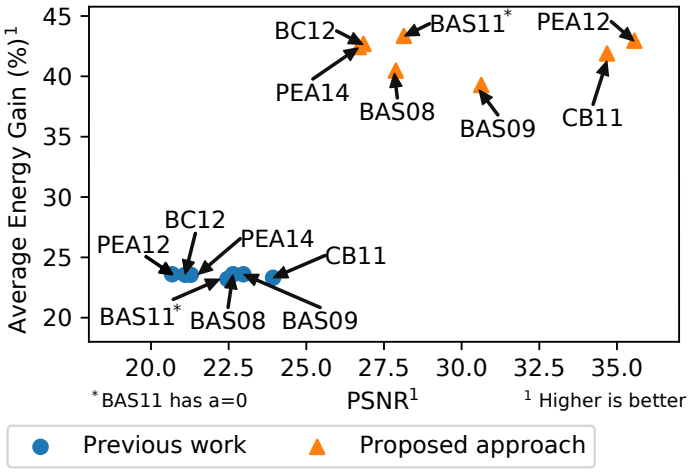


Fig. 16. Comparison with results from [3]

We analyzed and modeled several algorithms from the literature to compute a fast and lightweight version of the DCT. For each algorithm, we applied approximation by substituting full-precise adders with several approximate ones from the literature having configurable approximation degree. In this way, we can obtain different approximate configurations of the algorithms, depending on the chosen approximate adders and their approximation degree. Approximate adders introduce inaccuracy in the computation, but also achieve gains in terms of area and power consumption. For each algorithm, we performed a DSE to find the non-dominated approximate designs in terms of trade-off between inaccuracy and gains. We modeled the DSE as a MOP and we used a GA to solve it.

After the DSE, we synthesized the obtained designs by targeting both FPGA and ASIC. To do so, we implemented all the algorithms as re-configurable hardware designs. Finally, we evaluated the actual gains in terms of area and power consumption. Experimental results clearly showed that, with the proposed approach, it is possible to perform a meaningful DSE to find the best trade-offs between output accuracy and resource gains in a reasonable time. Finally, the comparison performed with previous work clearly showed the advantages of the proposed approach.

ACKNOWLEDGMENTS

Authors would like to thank Andrea Aletto for his technical support during experimental campaigns.

REFERENCES

[1] [n.d.]. SIPI Image Database. https://sipi.usc.edu/database/.

[2] Haider A.F. Almurib, T. Nandha Kumar, and Fabrizio Lombardi. 2016. Inexact Designs for Approximate Low Power Addition by Cell Replacement. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 660–665.

[3] Haider A.F. Almurib, Thulasiraman Nandha Kumar, and Fabrizio Lombardi. 2018. Approximate DCT Image Compression Using Inexact Computing. *IEEE Trans. Comput.* 67, 2 (Feb. 2018), 149–159. https://doi.org/10.1109/TC.2017.2731770

[4] Yukihiro Arai, Takeshi Agui, and Masayuki Nakajima. 1988. A Fast DCT-SQ Scheme for Images. *IEICE TRANSACTIONS (1976-1990)* 71, 11 (1988), 1095–1097.

[5] Mario Barbareschi, Federico Iannucci, and Antonino Mazzeo. 2016. Automatic Design Space Exploration of Approximate Algorithms for Big Data Applications. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 40–45. https://doi.org/10.1109/WAINA.2016.172

[6] Mario Barbareschi, Federico Iannucci, and Antonino Mazzeo. 2016. An Extendible Design Exploration Tool for Supporting Approximate Computing Techniques. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. 1–6. https://doi.org/10.1109/DTIS.2016.7483888

[7] F. M. Bayer and R. J. Cintra. 2012. DCT-like Transform for Image Compression Requires 14 Additions Only. *Electronics Letters* 48, 15 (2012), 919. https://doi.org/10.1049/el.2012.1148 arXiv:1702.00817

[8] Saad Bouguezel, M. Omair Ahmad, and M.N.S. Swamy. 2011. A Low-Complexity Parametric Transform for Image Compression. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. 2145–2148. https://doi.org/10.1109/ISCAS.2011.5938023

[9] Saad Bouguezel, M. Omair Ahmad, and M. N. S. Swamy. 2008. Low-Complexity 8$\times$ 8 Transform for Image Compression. *Electronics Letters* 44, 21 (2008), 1249–1250.

[10] Saad Bouguezel, M. Omair Ahmad, and M. N. S. Swamy. 2009. A Fast 8×8 Transform for Image Compression. In *2009 International Conference on Microelectronics - ICM*. 74–77. https://doi.org/10.1109/ICM.2009.5418584

[11] J. Castro-Godínez, H. Barrantes-García, M. Shafique, and J. Henkel. 2021. AxLS: A Framework for Approximate Logic Synthesis Based on Netlist Transformations. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2021), 1–1. https://doi.org/10.1109/TCSII.2021.3068757

[12] Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. 2013. Analysis and Characterization of Inherent Application Resilience for Approximate Computing. In *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*. ACM Press, Austin, Texas, 1. https://doi.org/10.1145/2463209.2488873

[13] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan. 2014. Scalable Effort Hardware Design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, 9 (Sept. 2014), 2004–2016. https://doi.org/10.1109/TVLSI.2013.2276759

[14] Renato J. Cintra and Fábio M. Bayer. 2011. A DCT Approximation for Image Compression. *IEEE Signal Processing Letters* 18, 10 (Oct. 2011), 579–582. https://doi.org/10.1109/LSP.2011.2163394

[15] I. Das and J. E. Dennis. 1997. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural Optimization* 14, 1 (Aug. 1997), 63–69. https://doi.org/10.1007/BF01197559

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197. https://doi.org/10.1109/4235.996017

[17] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. 2007. Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation - GECCO '07*. ACM Press, London, England, 1187. https://doi.org/10.1145/1276958.1277190

[18] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant Colony Optimization. *IEEE Computational Intelligence Magazine* 1, 4 (Nov. 2006), 28–39. https://doi.org/10.1109/MCI.2006.329691

[19] F. Fang, T. Chen, and R. A. Rutenbar. 2002. Floating-Point Bit-Width Optimization for Low-Power Signal Processing Applications. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3. III–3208–III–3211. https://doi.org/10.1109/ICASSP.2002.5745332

[20] E. Feig and S. Winograd. 1992. On the Multiplicative Complexity of Discrete Cosine Transforms. *IEEE Transactions on Information Theory* 38, 4 (July 1992), 1387–1391. https://doi.org/10.1109/18.144722

[21] Vaibhav Gupta, Debabrata Mohapatra, Anand Raghunathan, and Kaushik Roy. 2013. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 1 (Jan. 2013), 124–137. https://doi.org/10.1109/TCAD.2012.2217962

[22] Xuejue Huang, Wen-chin Lee, Charles Kuo, Digh Hisamoto, Jakub Kedzierski, Erik Anderson, Hideki Takeuchi, Yang-kyu Choi, Kazuya Asano, Vivek Subramanian, Tsu-jae King, Jeffrey Bokor, and Chenming Hu. [n.d.]. Sub 50-Nm FinFET: PMOS (Revised 12/9/1999).

[23] Arnaud Liefooghe, Matthieu Basseur, Laetitia Jourdan, and El-Ghazali Talbi. 2007. ParadisEO-MOEO: A Framework for Evolutionary Multi-Objective Optimization. In *Evolutionary Multi-Criterion Optimization*, Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata (Eds.). Vol. 4403. Springer Berlin Heidelberg, Berlin, Heidelberg, 386–400. https://doi.org/10.1007/978-3-540-70928-2_31

[24] C. Loeffler, A. Ligtenberg, and G.S. Moschytz. 1989. Practical Fast 1-D DCT Algorithms with 11 Multiplications. In *International Conference on Acoustics, Speech, and Signal Processing,*. 988–991 vol.2. https://doi.org/10.1109/ICASSP.1989.266596

[25] Sparsh Mittal. 2016. A Survey of Techniques for Approximate Computing. *Comput. Surveys* 48, 4 (May 2016), 1–33. https://doi.org/10.1145/2893356

[26] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique. 2019. autoAx: An Automatic Design Space Exploration and Circuit Building Methodology Utilizing Libraries of Approximate Components. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[27] K. Nepal, Y. Li, R. I. Bahar, and S. Reda. 2014. ABACUS: A Technique for Automated Behavioral Synthesis of Approximate Computing Circuits. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. https://doi.org/10.7873/DATE.2014.374

[28] Uma Sadhvi Potluri, Arjuna Madanayake, Renato J. Cintra, Fábio M. Bayer, Sunera Kulasekera, and Amila Edirisuriya. 2014. Improved 8-Point Approximate DCT for Image and Video Compression Requiring Only 14 Additions. *IEEE Transactions on Circuits and Systems I: Regular Papers* 61, 6 (June 2014), 1727–1740. https://doi.org/10.1109/TCSI.2013.2295022

[29] U S Potluri, A Madanayake, R J Cintra, F M Bayer, and N Rajapaksha. 2012. Multiplier-Free DCT Approximations for RF Multi-Beam Digital Aperture-Array Space Imaging and Directional Sensing. *Measurement Science and Technology* 23, 11 (Nov. 2012), 114003. https://doi.org/10.1088/0957-0233/23/11/114003

[30] Arnab Raha, Swagath Venkataramani, Vijay Raghunathan, and Anand Raghunathan. 2015. Quality Configurable Reduce-and-Rank for Energy Efficient Approximate Computing. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. 665–670. https://doi.org/10.7873/DATE.2015.0569

[31] Abbas Rahimi, Amirali Ghofrani, Kwang-Ting Cheng, Luca Benini, and Rajesh K. Gupta. 2015. Approximate Associative Memristive Memory for Energy-Efficient GPUs. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1497–1502. https://doi.org/10.7873/DATE.2015.0579

[32] Bhawna Rani, R. K. Bansal, and Savina Bansal. 2009. Comparison of JPEG and SPIHT Image Compression Algorithms Using Objective Quality Measures. In *2009 International Multimedia, Signal Processing and Communication Technologies*. 90–93. https://doi.org/10.1109/MSPCT.2009.5164181

[33] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan. 2014. ASLAN: Synthesis of Approximate Sequential Circuits. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. https://doi.org/10.7873/DATE.2014.377

[34] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. 2013. SAGE: Self-Tuning Approximation for Graphics Engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-46*. ACM Press, Davis, California, 13–24. https://doi.org/10.1145/2540708.2540711

[35] Lukas Sekanina, Zdenek Vasicek, and Vojtech Mrazek. 2019. Automated Search-Based Functional Approximation for Digital Circuits. In *Approximate Circuits*, Sherief Reda and Muhammad Shafique (Eds.). Springer International Publishing, Cham, 175–203. https://doi.org/10.1007/978-3-319-99322-5_9

[36] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. 2011. Managing Performance vs. Accuracy Trade-Offs with Loop Perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering - SIGSOFT/FSE '11*. ACM Press, Szeged, Hungary, 124. https://doi.org/10.1145/2025113.2025133

[37] M. Traiola, A. Savino, M. Barbareschi, S. D. Carlo, and A. Bosio. 2018. Predicting the Impact of Functional Approximation: From Component- to Application-Level. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. 61–64. https://doi.org/10.1109/IOLTS.2018.8474072

[38] Marcello Traiola, Alessandro Savino, and Stefano Di Carlo. 2019. Probabilistic Estimation of the Application-Level Impact of Precision Scaling in Approximate Computing Applications. *Microelectronics Reliability* 102 (Nov. 2019), 113309. https://doi.org/10.1016/j.microrel.2019.06.002

[39] Swagath Venkataramani, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. 2015. Approximate Computing and the Quest for Computing Efficiency. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2744769.2744904

[40] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou. 2017. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36, 3 (March 2017), 513–517. https://doi.org/10.1109/TCAD.2016.2587683

[41] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[42] Zhixi Yang, Ajaypat Jain, Jinghang Liang, Jie Han, and Fabrizio Lombardi. 2013. Approximate XOR/XNOR-Based Adders for Inexact Computing. In *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*. 690–693. https://doi.org/10.1109/NANO.2013.6720793

[43] Georgios Zervakis, Sotirios Xydis, Dimitrios Soudris, and Kiamal Pekmestzi. 2019. Multi-Level Approximate Accelerator Synthesis Under Voltage Island Constraints. *IEEE Transactions on Circuits and Systems II: Express Briefs* 66, 4 (April 2019), 607–611. https://doi.org/10.1109/TCSII.2018.2869025