Approximate Circuits in Low-Power Image and Video Processing: The Approximate Median Filter

Lukas SEKANINA, Zdenek VASICEK, Vojtech MRAZEK

Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Czech Republic

{sekanina, vasicek, imrazek}@fit.vutbr.cz

Submitted July 15, 2017 / Accepted July 15, 2017

Abstract. *Low power image and video processing circuits are crucial in many applications of computer vision. Traditional techniques used to reduce power consumption in these applications have recently been accompanied by circuit approximation methods which exploit the fact that these applications are highly error resilient and, hence, the quality of image processing can be traded for power consumption. On the basis of a literature survey, we identified the components whose implementations are the most frequently approximated and the methods used for obtaining these approximations. One of the components is the median image filter. We propose, evaluate and compare two approximation strategies based on Cartesian genetic programming applied to approximate various common implementations of the median filter. For filters developed using these approximation strategies, trade-offs between the quality of filtering and power consumption are investigated. Under conditions of our experiments we conclude that better trade-offs are achieved when the image filter is evolved from scratch rather than a conventional filter is approximated.*

Keywords

Approximate computing, circuit design, evolutionary computation, image filter

1. Introduction

An efficient implementation of computer vision algorithms is crucial for many smart embedded systems such as traffic control systems, driver assistant systems, production line inspection systems, and robotics. However, providing high-quality outputs in these applications is usually associated with high computation cost and non-trivial requirements on energy. In order to meet real-time constraints and cope with limited power budget, image and video processing algorithms are often accelerated in application-specific integrated circuits (ASIC) or field programmable gate arrays (FPGAs). If additional energy consumption reduction is requested because of, for example, very limited energy available in remote sensors, mobile or wearable devices, the *circuit approximation* is one of the most promising approaches to deliver a suitable solution.

Approximate computing [1] exploits the fact that many applications (image and video processing in particular) are highly error resilient. If occasional errors are acceptable by the users – which is possible because the users as consumers of the outputs of these applications are often unable to recognize small imperfections in images or video sequences – implementations of these applications can be simplified. The goal is to create such an implementation which shows the best trade-off between the error, performance and power consumption. Approximate computing has been progressively developed in recent 5 years and influenced the way how energy efficient computer systems (ranging from tiny battery powered devices via common desktop computers to supercomputers) are now constructed and operated.

In this paper, we focus on approximate circuits that are used in image and video processing applications. On the basis of a literature survey, we identified the components whose implementations are the most frequently approximated and the methods used for obtaining these approximations. One of the components is the median-outputting circuit (median for short) which is typically employed to filter out undesired artefacts (such as shots) in digital images.

As the circuit approximation problem can be formulated as a multi-objective optimization problem (with error, performance and power consumption as objectives), various ad hoc and heuristic methods have been introduced to solve it. In our previous work, we have developed circuit approximation methods [2], [3] based on Cartesian genetic programming (CGP) which is a branch of evolutionary algorithms capable of designing and optimizing digital circuits.

Unfortunately, the quality of approximation methods has been compared in the literature only rarely (see a survey of associated methodological problems in [2]); in most cases only parameters of approximate adders and multipliers were compared [4]. In this paper, we compare two approximation strategies based on CGP applied to approximate various common implementations of the median filter. The first strategy starts with an exact median filter implementation and tries to remove some circuit components (comparators) and re-

connect the remaining ones in such a way that the error of filtering is minimized. The second strategy employs CGP to evolve the image filter from scratch; only on the basis of the training data supplied during the evolution. The goal is to demonstrate how different approximation strategies can influence the trade-offs that are obtained between the quality of filtering and power consumption for target circuits.

The rest of the paper is organized as follows. The research area of approximate computing is introduced in Sec. 2. Section 3 deals with a survey of circuits that were approximated for purposes of power consumption reduction in image and video processing applications. Various aspects of the approximation strategies used to obtain desired approximations have been analyzed. Section 4 is devoted to our case study – approximate circuits for image filtering. We present conventional implementations of image filters, CGP as the method used to perform desired approximations and two different approximation strategies based on CGP. Results are summarized in Sec. 5. Conclusions are given in Sec. 6.

2. Approximate Computing

The concept of approximation has been well established in computer science and engineering for decades. For example, a paper with the title "Approximate signal processing" was published in 1997 [5]. However, new problems emerged in the last decade that stimulated new research in applying approximation techniques, but in a slightly different context than before.

In particular, problems with high power density of integrated circuits led to the end of Dennard scaling, i.e. simultaneous doubling the number of transistors on a chip, increasing operation frequency and reducing Vdd have no longer worked together. The coming era of "dark silicon", when many transistors are available on a chip, but cannot be used at the same time on high operating frequency because of thermal issues, has forced us to rethink the basic design principles of computer-based systems [6]. As conventional power reduction techniques such as dynamic voltage-frequency scaling and power gating do not scale sufficiently and alternative post CMOS technologies are not widely adopted, the only solution seems to be to relax the requirement on precise computing across the computer stack.

In approximate computing, the requirement of exact equivalence between the specification and all implementations levels is relaxed in order to reduce power consumption or improve other system parameters such as performance [1], [7].

The approximation can be conducted at the level of software as well as hardware. Mittal [1] discusses a wide spectrum of approximation techniques which include precision scaling, loop perforation, load value approximation, memorization, task dropping/skipping, memory access skipping, using different SW/HW versions, refresh rate reducing in memory, inexact read/write and relaxed synchronization. In the case of digital circuit approximation, voltage over-scaling and functional approximation are the most popular techniques. In the case of *voltage over-scaling*, the circuit is supplied with lower Vdd than nominal, which reduces power consumption, but introduces errors for those inputs whose processing requires attending the critical path of the design. In the case of *functional approximation*, a slightly different function is implemented with respect to the original one, provided that the error is acceptable and key system parameters are improved. The errors induced by approximation are measured using various error metrics such as the average error, error probability, and worst case error.

The approximate solution is usually obtained by a heuristic procedure that modifies the original implementation. In the case of software approximation, programmers can typically declare which parts of the program can be computed approximately and specialized compiler and optimizer then preform requested approximations (see, e.g., EnerJ [8]). In the case of hardware approximation, either general-purpose or circuit-specific approximation methods have been applied. While the aim of general-purpose approximation methods (e.g. SALSA [9], AXILOG [10], ASLAN [11], ABACUS [12], CGP [2], [3]) is to automatically approximate any circuit regardless of its structure, the circuit-specific methods are focused on a rather specific class of circuits (such as adders or multipliers [4]).

3. Approximate Circuits for Image and Video Processing

Based on the analysis of 12 image and data processing applications, Chippa et al. showed that about 83% runtime is spent in error resilient computation kernels that are suitable for approximation [7]. The most dominant kernels were the dot product computation and distance computation. The fact that image and video processing circuits are good targets for circuit approximation can be documented by dozens of papers dealing with this topic in the literature.

It has to be noticed that elementary arithmetic circuits (such and adders and multipliers) are often approximated independently of a potential application. The objective is to create a general-purpose library of approximate implementations showing different trade-offs between power consumption and error. Jiang et al. [4] provided a detailed survey of approaches developed in this direction. In this paper, we will deal with approximate adders or multipliers only if they have been applied in an approximate implementation of image or video processing system.

Approximate circuits are also crucial in energy efficient implementations of image and video processing systems (image classifiers, object detectors) based on (deep) neural networks (DNN). As this is rather a specific area [13], [14], we will not consider it in our survey table, but provide a brief introduction in this paragraph. In DNNs, approximations were introduced at levels of the data type quanti-

Application Type	Ref.	Module	Approx. Component	Approx. Method	Approx. Level	Platform
Filter	[26]	Median	Comparators/Network	Ad hoc	transistor	ASIC
	[24]	Median	Median	CGP	RTL	ASIC
	[27]	Median	Median	CGP	RTL	FPGA
	[31]	Gaussian	Multiplier	Ad hoc	gate	ASIC
	[27]	Gaussian	Adder, Multiplier	CGP	gate	FPGA
	[27]	Sobel	Adder	CGP	gate	FPGA
	[32]	Sobel	OpenCL code	truncation	RTL	FPGA
	[11]	Sobel	Sobel	ASLAN	gate	ASIC
	[10]	Sobel	Verilog code	AXILOG	RTL	ASIC
Metrics	[16]	SAD	SAD	Logic Isolation	gate	ASIC
	[9]	SAD	SAD	SALSA	gate	ASIC
	[16]	EUD	EUD	LogicIsolation	gate	ASIC
Transforms	[16]	DCT	DCT	Logic Isolation	gate	ASIC
	[16]	FFT	FFT	Logic Isolation	gate	ASIC
	[9]	DCT	DCT	SALSA	gate	ASIC
	[32]	DCT	OpenCL code	truncation	RTL	FPGA
JPEG	[31]	JPEG	Multiplier	Ad hoc	gate	ASIC
	[29]	DCT	Adder	Ad hoc	transistor	ASIC
	[34]	DCT	Adder/Multiplier	truncation/MINPS	full adders	ASIC
MPEG	[12]	Block Matching	Verilog code	ABACUS	RTL	ASIC
	[11]	DCT	DCT	ASLAN	gate	ASIC
	[33]	DCT	Adder/Subtractor	Ad hoc	gate	ASIC
	[33]	ME	Adder/Subtractor	Ad hoc	gate	ASIC
HEVC	[28]	SAD in ME	Adder	Ad hoc	gate	ASIC, FPGA
	[30]	DCT	DCT	Ad hoc	gate	ASIC
	[19]	DCT	Adder/Subtractor	CGP	gate	ASIC

SAD (Sum of Absolute Differences) ME (Motion Estimation) MINPS (Mixed Integer Nonlinear Problem Solver)
DCT (Discrete Cosine Transform) FFT (Fast Fourier Transform) EUD (Euclidean distance)

Tab. 1. Circuits approximated in the area of image and video processing.

zation, microarchitecture (e.g. neurons insignificantly contributing to the quality of outputs can be removed), training algorithm (an iterative process which can be stopped when good enough results are obtained), the multiply-accumulate-transform circuits (where the design of approximate multipliers and adders for DNN applications represents an independent topic [15], [16]), and memory cells and architecture (where, e.g., less significant bits can be stored in energy efficient, but less reliable memory cells [17]). An ultra-low power deep learning ASIC for IoT was implemented on a single chip, capable of performing 374 GOPS/W and consuming less than 300 μW. However, performance of this solution is limited as it operates at 3.9 MHz only [18]. While specific low-power electronic circuits can be developed in ASICs (see, e.g., specialized on-chip memory cells and architecture in [18]) to minimize power consumption of DNN, the optimization of an FPGA solution has to be focused on microarchitecture and memory subsystem organization that are composed of (fixed and pre-defined) FPGA primitives.

In our survey, we will primarily focus on functional approximation which is less technology dependent and provides more predictable errors than voltage over scaling. The survey is based on representative papers published in 2011 – 2017 on key relevant conferences and in journals.

The result of our survey is presented in the form of table: Table 1 shows that the papers included into the survey are organized according to the Application Type, where the following major application types were identified: Filters, Metrics, Transforms, image compression (JPEG), and video (de)coders (according to MPEG and HEVC standards). In these Application Types, we investigated:

- what is approximated, i.e. whether the approximation is performed at the level of components (such as adders, multipliers, comparators) or modules (such as filters, DCT and FFT created using these components),
- how the approximation is conducted, i.e. whether an ad hoc or general purpose method is taken,
- what is the level of abstraction, where an approximation is conducted, i.e. whether circuits are approximated at the transistor, gate, register-transfer (RT) or behavioral source code level, and
- target platform, i.e. an ASIC or FPGA.

It can be seen that less complex applications such as image filters can be holistically approximated as a single system. In the case of more complex applications, the design is firstly decomposed and selected components then undergo the approximation process. Some of them can even be removed to further reduce power consumption. The approximation is predominately conducted at the gate level, but there are tools (such as AXILOG, ABACUS and GRATER) in which requirements on the approximation are specified directly at the source code (RT or behavioral) level. The actual approx-

imations are then performed internally by the tool during the synthesis and netlist optimization.

It remains unclear what is the best performing approximation approach in the area of image and video processing. Unfortunately, approximate solutions have been only compared with exact solutions, but almost never with other competitive approximate solutions.

4. Case Study

The purpose of this case study is to compare the impact of two fundamentally different approximation strategies on the quality and power consumption of a selected module of an image processing system. We decided to approximate the circuits implementing the shot noise image filter. The approximations will be conducted by CGP which proved to be highly competitive with respect to other circuit approximation methods [3], [19].

Section 4.1 provides a brief overview of conventional implementations of median filters and their extensions. CGP is then introduced in Sec. 4.2. Two approximation strategies (AS) are proposed in Sec. 4.3: (AS1) CGP is employed to approximate circuit implementations of the considered filters. (AS2) CGP is used to holistically evolve desired image filters from scratch.

4.1 Median Filters

Conventional implementations of shot noise elimination filters are usually based on calculating the median over the pixels belonging to the filtering window.

The *median filter* (MF) is a special case of order statistic filters which may be implemented in several different ways [20]. In this paper, we will consider a pipelined implementation based on a median network which is suitable for high-performance applications. The median network consists of a sequence of *compare-and-swap* operations (Fig. 1). Each compare-and-swap (CS) operation acts as a small 2-input sorting network which produces a sorted sequence by outputting the minimum and maximum of the input values.

The *weighted median filter* is an extension of the common median filter, which gives more weight to some values within the filtering window. The *center weighted median filter* (CWMF) represents a special case in which only the central value of the window is counted with additional weight [21]. Compared to the median filter, this modification can preserve more details along the horizontal and vertical directions while suppressing additive white and/or impulsive-type noise.

The median filters uniformly replace the value of every pixel of the filtered image by the median of its neighbors. Consequently, in addition to the removal of noisy pixels, these filters also remove desirable details and thus smudge the resulting image. In order to address this problem, more advanced concepts were introduced. The *adaptive median filter* (AMF) represents a multi-level approach which tries to detect and subsequently replace corrupted pixels only [22]. At each level, filtering windows of different sizes are utilized. Usually, two levels working with the 3×3 and 5×5 filtering window, respectively, are sufficient to obtain a very good image quality (Fig. 2). Hardware implementation consists of two median filters, circuitry that determines minimal and maximal values for each filter window, delay buffers to compensate different latency of median filters and simple logic.

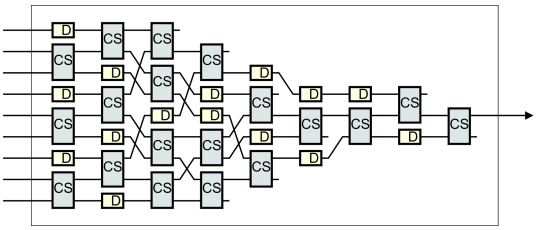

Fig. 1. Pipelined implementation of 9-input median filter consisting of compare-and-swap (CS) blocks and registers (D). All CS blocks contain the output register.

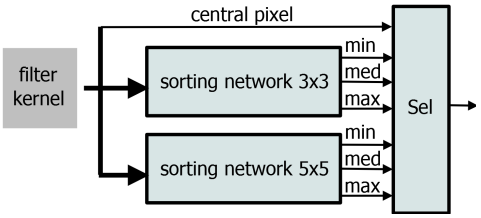

Fig. 2. Adaptive median filter internally computing minimum, maximum and median over kernels with 3×3 and 5×5 pixels and determining the output value using Selector.

4.2 Evolutionary Approximation

CGP [23] is a form of genetic programming in which each candidate solution is modeled using a two-dimensional array of $n_c \times n_r$ programmable n_a-input/n_b-output nodes whose functions are taken from a set G. The circuit utilizes n_i primary inputs and n_o primary outputs. A unique address is assigned to all primary inputs and to the outputs of all nodes to define an addressing system enabling circuit topologies to be specified (Fig. 3). As no feedback connections are allowed in the basic version of CGP, only combinational circuits can be created. Each candidate circuit is represented using the so-called chromosome which contains $n_c \times n_r \times (n_a + n_b) + n_o$ integers. The $(n_a + n_b)$ integers specify one programmable node: n_a integers specify destination addresses for its inputs and n_b integers determine the function codes from G. All possible legal chromosomes constitute the search space.

The search is usually performed using a simple $(1 + \lambda)$ evolutionary algorithm. In this algorithm, every new population consists of the best individual of the previous population and its λ offspring created using a mutation operator. This operator randomly modifies up to h randomly selected genes (integers) of the chromosome. The search is typically terminated after generating a given number of populations.

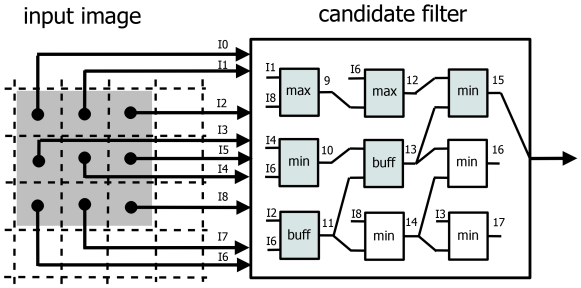

Fig. 3. An example of a simple filtering circuit (with filtering window 3×3 pixels) represented in CGP with parameters: $n_i = 9$, $n_o = 1$, $n_c = 3$, $n_r = 3$, $n_a = 2$, $n_b = 1$, G = {buffer (0), min (1), max(2)}. Nodes 14, 16 and 17 are inactive. Chromosome: 1,8,2; 4,6,1; 2,6,0; 6,9,2; 10,11,0; 8,11,1; 12,13,1; 13,14,1; 13,14,1; 15.

In order to evaluate the population, each candidate solution is evaluated using the so-called fitness function. As the problem is in principle multi-objective (error versus power consumption or area), a suitable multi-objective optimization algorithm has to be taken [2], [3]. While the circuit area on a chip can be easily estimated by summing the areas of components involved in the circuit, the error computation is more time demanding (see next sections).

4.3 Approximation Strategies based on CGP

Two strategies are compared in this case study:

AS1: Since the median filter is implemented as a network of compare-and-swap operations, an obvious approximation strategy is to remove some of them and reconnect the remaining ones in such a way that the error of filtering is minimized. We propose to seed CGP with the best known implementations of median filters and evolve approximate median filters containing fewer comparators than needed in the fully functional implementation. The fitness is constructed according to [24]. AS1, therefore, works at the level of comparators.

AS2: The whole image filtering function is evolved with CGP from scratch. The function set G contains all suitable two-input components, not only the minimum and maximum functions. CGP thus holistically develops a new image filter with the aim to minimize the error of filtering on the training data. Following the approach developed for the evolutionary design of image filters [25], the error is measured by means of the *mean absolute error* (MAE) between the outputs O_f produced by a candidate filter and reference (golden) outputs O_g for a given *training data set*, formally:

$$MAE = \frac{1}{K}\sum_{i=1}^{K} |O_f(i) - O_g(i)| \qquad (1)$$

where K is the number of filtered pixels.

As this approach is not biased by a conventional solution (median filter), there is a chance to discover an implementation showing better filtering properties and lower power consumption.

It has to be noticed that these approximation strategies differ from the approximate median filters proposed in the literature because: paper [26] utilizes approximate transistor-level circuits to implement the comparators (our comparators are always exact) and papers [3], [27] do not initialize CGP with existing median implementations, but rather evolve approximate circuits form scratch using insufficient resources.

5. Results

This section presents the setup used to perform desired approximations, parameters of evolved circuits and a comparison of approximate and original filters in terms of power consumption, area and filtering quality. In order to obtain parameters of evolved filters, we described the filters in VHDL and synthesized them using Synopsys Design compiler with 45 nm PDK. The filters were implemented as pipelined circuits with 8 bit operands. The goal of the synthesis was to produce implementations operating at least at 1 GHz. Section 5.1 deals with the implementation cost of conventional and approximate filters. The filtering quality is compared in Sec. 5.2.

5.1 Implementation Cost

Conventional (Exact) Filters: Table 2 summarizes the synthesis results for various median filters discussed in Sec. 4.1 – median filter operating on 3×3 (5×5) filter window denoted as MF9 (MF25), center weighted median filter operating on 3×3 pixels with the weight equal to 3 (CWMF9), and adaptive median filter (AMF25). While MF9 consists of 19 compare-and-swap operations (ops), AMF25 requires nearly ten times more operations. Each compare-and-swap operation is implemented using an 8-bit magnitude comparator and two 8-bit multiplexers. For each filter, the number of compare-and-swap operations, total power consumption and occupied area are presented. Contributions to power and area are given separately for registers and logic. The delay is intentionally omitted in all tables because timing constrains were met in all cases.

The key observation is that logic consumes less than 20% of the total power consumption. This is due to the pipeline nature of the circuits. The area on a chip increases with the increasing complexity (i.e. with the number of compare-and-swap operations) of the filters. As expected,

		power [mW]			area [$\mu m^2 \times 10^3$]		
filter	**ops**	**total**	**regs**	**logic**	**total**	**regs**	**logic**
MF9	19	6.8	80%	19%	7.8	65%	34%
CWMF9	28	12.1	81%	18%	13.6	64%	35%
MF25	99	45.0	86%	13%	37.9	50%	49%
AMF25	182	58.1	85%	14%	52.8	53%	46%

Tab. 2. Results of synthesis for conventional filters.

the common median filter operating with 3×3 pixels is the cheapest solution. If we extend the filter window to 5×5, the power consumption increases more than 6 times and the area on a chip increases nearly 5 times. The adaptive median filter represents the most complex and power-demanding filter in our study. Its power consumption is more than 8 times higher compared to MF9. The implementation costs of CWMF is between MF9 and MF25 since CWMF9 is, in fact, a median network with 11 inputs whose three inputs are connected to the central pixel of the filter window. The power as well as the area on a chip are doubled compared to MF9.

Filters Approximated with AS1: In order to obtain approximate median filters, CGP was seeded with the known optimal implementations of 9-input, 11-input and 25-input median networks exhibiting the minimal number of compare-and-swap operations. CGP operated with $n_a = n_b = 2$, $\lambda = 20$, $h = 5$, and 10^7 (6.10^5 respectively) generations were produced for 9-input (25-input, respectively) circuits. The function set contained 8-bit compare-and-swap functions and identity function. The error was determined as the position distance with respect to the exact median according to [24]. The goal of CGP was to minimize the position error under constrained area (experimented with max. 20% – 95% area of the exact implementation). As the statistical evaluation of this type of evolutionary design has been performed in the literature [24], we will report just the best evolved solutions.

Several hundreds of approximate implementations were produced by CGP in total. We identified ten Pareto-dominant solutions for each type of filter and synthesized them using Synopsys Design compiler to obtain their electrical parameters. Table 3 summarizes the total number of operations, power consumption and area for selected approximate filters. The obtained reduction with respect to the (exact) original solution is included in the 'red.' columns.

Table 3 shows that pruning of the number of compare-and-swap operations and their rearranging enables to significantly reduce not only the area on a chip but also the power consumption. The filtering quality will be reported in Sec. 5.2. For example, 9-input approximate median filter MF9 #9 exhibits a 75% reduction in power consumption and a 69% reduction in the area compared to the accurate optimal implementation. Overall, more than 75% of power budget is due to switching activity of registers. The majority of the area on a chip is utilized by registers.

Table 3 also includes parameters of approximate adaptive median filters. These approximate filters were obtained by replacing the exact 9-input median and 25-input median with their selected approximate implementations. The rest of the circuitry remained unchanged. Three variants of approximate adaptive median filter are presented – AMF25 #19, AMF25 #79 and AMF25 #99. The first variant consists of the exact 9-input approximate median network MF9, the second of approximate MF9 #7 and third employs MF9 #9. In all cases, approximate MF25#9 is employed. The approximate AMFs occupy nearly half of the area and achieve up to 61% power saving with respect to AMF.

	ops		power [mW]			area [$\mu m^2 \times 10^3$]		
filter	**total**	**red.**	**total**	**red.**	**regs**	**total**	**red.**	**regs**
MF9								
#5	15	21%	4.6	31%	78%	5.6	27%	68%
#7	12	36%	3.0	55%	76%	4.1	48%	71%
#9	8	57%	1.7	75%	75%	2.4	69%	73%
CWMF9								
#5	25	10%	8.7	27%	80%	10.3	24%	66%
#7	19	32%	6.9	43%	82%	7.7	42%	63%
#9	13	53%	3.6	70%	78%	4.5	66%	68%
MF25								
#6	64	35%	32.5	27%	89%	26.4	30%	45%
#8	50	49%	20.1	55%	87%	17.7	53%	51%
#9	42	57%	14.5	67%	81%	16.4	56%	64%
AMF25								
#19	125	31%	27.6	52%	81%	31.4	40%	64%
#79	118	35%	23.9	58%	81%	27.7	47%	64%
#99	114	37%	22.5	61%	81%	26.0	50%	64%

Tab. 3. Results of synthesis for filters approximated in AS1.

Filters Approximated with AS2: CGP started with a randomly generated initial population and used two-input 8-bit functions (such as minimum, maximum, addition, absolute difference, conditional assignment) and other settings ($n_c = 7$, $n_r = 9$, $n_a = 2$, $n_b = 1$, $\lambda = 7$, $h = 15$) according to [25]. All filters were evolved using fitness function (eq. 1) and appropriate training and golden images consisting of 384×256 pixels, i.e. $K = 98,304$.

Parameters of the best performing filters evolved under AS2 are summarized in Tab. 4. Two noise-specific filters are included in our comparison – a salt-and-pepper noise filter (denoted EVO #1) and a random-valued impulse noise filter (denoted EVO #2). Please, refer to Sec. 5.2 for details dealing with noise description. Both filters operate on the filter window consisting of 5×5 pixels. EVO #1 consists of 27 8-bit components (including 17 min/max functions) and occupies approximately the same area as MF9 but consumes about 50% more power. This is an interesting result because it operates on nearly three times higher number of inputs. EVO #2 is a more complex circuit having 33 8-bit components (including 20 min/max functions). Considering the fact that both filters have 25 inputs, they exhibit significantly lower implementation cost and power compared to MF25. Their filtering quality will be discussed in Sec. 5.2.

In order to improve the quality of output images, an ensemble of filters is often employed. In this evaluation, a bank of filters was constructed using 3 best filters evolved for each type of noise [25] (i.e. BNK #1 for salt-and-peper and BNK #2 for random shot noise). As Fig. 4 shows these filters

	power [mW]			area [$\mu m^2 \times 10^3$]		
filter	**total**	**regs**	**logic**	**total**	**regs**	**logic**
EVO #1	10.2	86%	13%	7.5	55%	44%
BNK #1	39.7	90%	9%	25.5	45%	54%
EVO #2	16.1	85%	14%	11.8	55%	44%
BNK #2	52.5	90%	9%	33.3	44%	55%

Tab. 4. Results of synthesis for filters approximated in AS2.

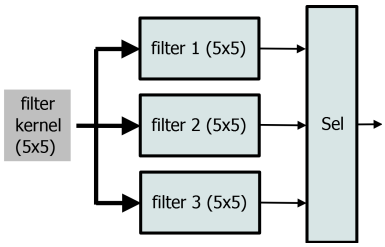

Fig. 4. Bank of filters composed of 3 different evolved filters.

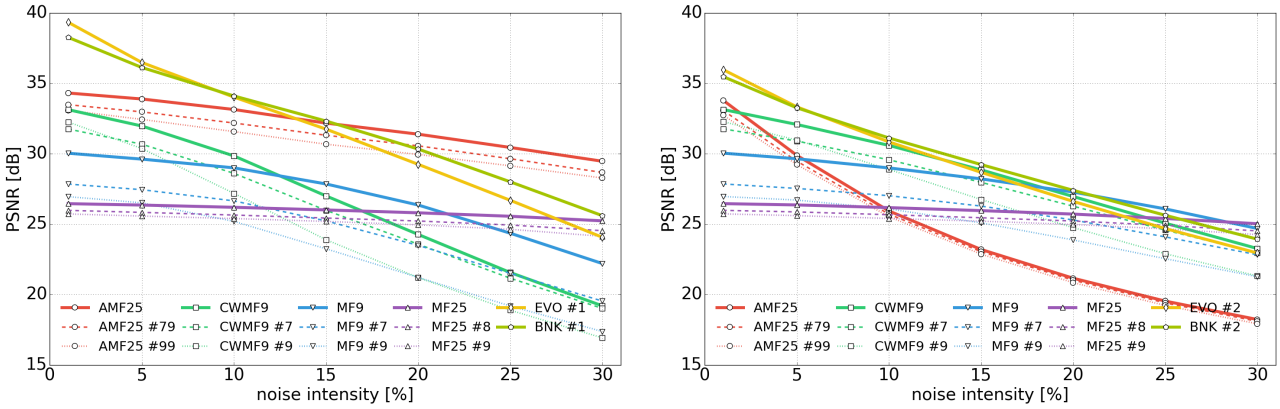

Fig. 5. Mean PSNR on 30 test images and different noise intensities obtained for conventional and approximate filters: salt-and-pepper noise (left) and impulse noise (right).

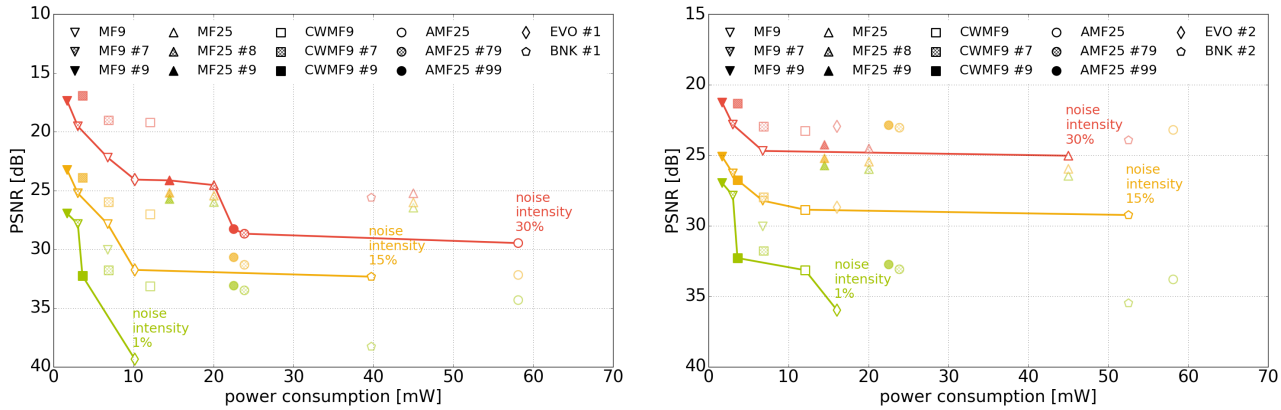

Fig. 6. Mean PSNR and power consumption of selected image filters: salt-and-pepper noise (left) and impulse noise (right).

operate in parallel over the filter window. If the majority of the filters of the bank indicates that the processed pixel is a shot, then the median value is calculated from the outputs of these filters and sent to the primary output of the bank. Otherwise, the original value of the processed pixel is sent to the primary output of the bank. While BNK #1 occupies significantly lower area on a chip compared to MF25 or AMF25, BNK#2 is comparable to AMF25.

5.2 Filtering Quality

The quality of the proposed filters was evaluated using a set of 30 test images corrupted by means of two common types of noise – salt-and-pepper noise and random shot noise. While the salt-and-pepper noise removal represents a typical benchmark problem which can be satisfactory addressed using adaptive median filter, the random shot noise removal is known to be a significantly harder problem. The reason is that the values of noisy pixels for salt-and-pepper noise are equal to either 0 or 255. In the case of the random shot noise, a noisy pixel can gain an arbitrary value from the whole range (i.e. 0 – 255). Therefore, it is more difficult to detect this noise because the deviation of a noise pixel can be very close to its original value.

Figure 5 shows the filtering quality of common and approximate filters in terms of the mean peak signal-to-noise ratio (PSNR). As 7 noise intensities (ranging from 1% to

30%) were considered, every filter was, in fact, applied to 2 (noise type) × 30 (images) × 7 (noise intensity) = 420 images. Resulting trade-offs between power consumption and filtering quality for noise intensity 1%, 15% and 30% are illustrated in Fig. 6.

The most interesting observations are as follows. Regarding the filtering quality, the mean PSNR indicates that filters evolved in AS2 significantly outperform other filters especially if the noise intensity is lower (15-20% depending on the noise type). For highly corrupted images, the bank of evolved filters can be employed to even improve the quality of filtering.

AMF performs well on salt-and-pepper noise, but it is rather poor for random shot noise; however, it is a very expensive solution. When approximate filters are introduced to AMF, the mean PSNR remains practically the same as for AMF25. The output quality depends mainly on the quality of MF9 (see the resulting PSNR for AMF25 #79 and AMF25 #99), but the difference is below 1 dB even when MF9 #9 consisting of eight compare-and-swap operations was employed. Anyway, approximate versions of AMF significantly reduced power consumption of the original AMF. It has to be emphasized that filters evolved in AS2 still consume only around 50% of the power budget of AMF25#19.

CWMF9 and its approximations provide very good results for random shot noise. Hence, CWMF9 (or CWMF#7 having a slightly worse PSNR) seems to be a solution of the first choice for energy constrained applications because it provides 25% benefit in power compared to EVO#2.

If low power consumption is the key design objective then approximate versions of MF9 show the best trade-off.

6. Conclusions

On the basis of the literature survey, we reported approximate circuits and approximation methods that have been applied in the area of image and video processing. We observed that the approximations are conducted at different levels of abstraction (from transistors via gates to RT) and focused either on the whole modules (such as filters or DCT) or elementary components (such as adders and multipliers) of these modules. In addition to ad hoc approximation methods, many general-purpose approximation methods have been used. Only in rare cases the approximation methods were mutually compared in terms of quality of produced approximate circuits.

In order to investigate the impact of approximation methods on the quality of resulting approximate circuits, the median circuit approximation problem was chosen. We compared two CGP-based approximation strategies based on removing and rearranging some components (AS1) and complete redesigning of the circuit (AS2). Three conventional median-based circuits (MF, CWMF, and AMF) were included to our study. The approximations were performed for two types of noise and evaluated for 7 noise intensities.

As all circuits were implemented as pipelined structures operating at least at 1 GHz, the approximation and optimization was focused on obtaining the best trade-offs between power consumption and filtering error. In the case of AS1, approximate circuits consistently show slightly worse filtering quality, but significantly reduced power consumption with respect to their exact counterparts. The best trade-offs were obtained with AS2, i.e. when CGP was not biased by conventional designs and could deliver new well-optimized filtering structures.

We can conclude that complete resynthesizing of the circuit rather than approximating a conventional solution provides better trade-offs, especially if good filtering quality is desired. While this approach (SA2) was applicable to image filtering circuits, it is not currently applicable for complex circuits (such as the whole HEVC coder) because the design process based on CGP is not fully scalable. Improving its scalability will be one of our future research objectives.

Acknowledgments

This work was supported by Czech science foundation project GA16-17538S.

References

[1] MITTAL, S. A survey of techniques for approximate computing. *Journal ACM Computing Surveys (CSUR)*, 2016, vol. 48, no. 4, p. 62:1–62:33. DOI: 10.1145/2893356

[2] MRAZEK, V., HRBACEK, R., VASICEK, Z., et al. Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2017, p. 258–261. DOI: 10.23919/DATE.2017.7926993

[3] VASICEK, Z., SEKANINA, L. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation*, 2015, vol. 19, no. 3, p. 432–444. DOI: 10.1109/TEVC.2014.2336175

[4] JIANG, H., LIU, C., LIU, L., et al. A review, classification and comparative evaluation of approximate arithmetic circuits. *ACM Journal on Emerging Technologies in Computing Systems*, 2017, p. 1–37.

[5] NAWAB, S. H., OPPENHEIM, A. V., CHANDRAKASAN, A. P., et al. Approximate signal processing. *Journal of VLSI signal processing systems for signal, image and video technology*, 1997, vol. 15, no. 1, p. 177–200. DOI: 10.1023/A:1007986707921

[6] MARKOV, I. L. Limits on fundamental limits to computation. *Nature*, 2014, vol. 512, p. 147–154. DOI: 10.1038/nature13570

[7] CHIPPA, V. K., CHAKRADHAR, S. T., ROY, K., et al. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual Design Automation Conference (DAC'13)*. 2013, p. 1–9. DOI: 10.1145/2463209.2488873

[8] SAMPSON, A., DIETL, W., FORTUNA, E., et al. EnerJ: Approximate data types for safe and general low-power computation. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2011, p. 164–174. DOI: 10.1145/1993498.1993518

[9] VENKATARAMANI, S., SABNE, A., KOZHIKKOTTU, V. J., et al. SALSA: Systematic logic synthesis of approximate circuits. In *Proceedings of the 49th Annual Design Automation Conference (DAC'12)*. 2012, p. 796–801. DOI: 10.1145/2228360.2228504

[10] YAZDANBAKHSH, A., MAHAJAN, D., THWAITES, B., et al. Axilog: Language support for approximate hardware design. In *Proceedings of the Design, Automation and Test in Europe (DATE)*. 2015, p. 1–6. DOI: 10.7873/DATE.2015.0513

[11] RANJAN, A., RAHA, A., VENKATARAMANI, S., et al. ASLAN: Synthesis of approximate sequential circuits. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'14)*. 2014, p. 1–6. DOI: 10.7873/DATE.2014.377

[12] NEPAL, K., HASHEMI, S., TANN, H., et al. Automated high-level generation of low-power approximate computing circuits. *IEEE Transactions on Emerging Topics in Computing*, 2016, no. 1, p. 1–13. DOI: 10.1109/TETC.2016.2598283

[13] ESMAEILZADEH, H., SAMPSON, A., CEZE, L., et al. Neural acceleration for general-purpose approximate programs. *Communications of the ACM*, 2015, vol. 58, no. 1, p. 105–115. DOI: 10.1145/2589750

[14] PANDA, P., SENGUPTA, A., SARWAR, S. S., et al. Invited – cross-layer approximations for neuromorphic computing: From devices to circuits and systems. In *Proceedings of the 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2016, p. 1–6. DOI: 10.1145/2897937.2905009

[15] MRAZEK, V., SARWAR, S. S., SEKANINA, L., et al. Design of power-efficient approximate multipliers for approximate artificial neural networks. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 2016, p. 811–817. DOI: 10.1145/2966986.2967021

[16] SHUBHAM JAINA, S. V., RAGHUNATHANC, A. Approximation through logic isolation for the design of quality configurable circuits. In *Proceedings of the Design, Automation & Test in Europe Conference and Exhibition (DATE)*. 2016, p. 1–6. DOI: 10.3850/9783981537079_0416

[17] SRINIVASAN, G., WIJESINGHE, P., SARWAR, S. S., et al. Significance driven hybrid 8t-6t sram for energy-efficient synaptic storage in artificial neural networks. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE)*. 2016, p. 151–156. DOI: 10.3850/9783981537079 0909

[18] BANG, S., WANG, J., LI, Z., et al. 14.7 A 288μw programmable deep-learning processor with 270KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence. In *Proceedings of the IEEE International Solid-State Circuits Conference*. 2017, p. 250–251. DOI: 10.1109/ISSCC.2017.7870355

[19] VASICEK, Z., MRAZEK, V., SEKANINA, L. Towards low power approximate DCT architecture for HEVC standard. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2017, p. 1576–1581. DOI: 10.23919/DATE.2017.7927241

[20] HUANG, T., YANG, G., TANG, G. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1979, vol. 27, no. 1, p. 13–18. DOI: 10.1109/TASSP.1979.1163188

[21] KO, S., LEE, Y. Center weighted median filters and their applications to image enhancement. *IEEE Transactions on Circuits and Systems*, 1991, vol. 15, p. 984–993. DOI: 10.1109/31.83870

[22] HWANG, H., HADDAD, R. Adaptive median filters: New algorithms and results. *IEEE Transactions on Image Processing*, 1995, vol. 4, no. 4, p. 499–502. DOI: 10.1109/83.370679

[23] MILLER, J. F. *Cartesian Genetic Programming*. Springer-Verlag, 2011. DOI: 10.1007/978-3-642-17310-3

[24] VASICEK, Z., MRAZEK, V. Trading between quality and non-functional properties of median filter in embedded systems. *Genetic Programming and Evolvable Machines*, 2017, vol. 18, no. 1, p. 45–82. DOI: 10.1007/s10710-016-9275-7

[25] VASICEK, Z., BIDLO, M., SEKANINA, L. Evolution of efficient real-time non-linear image filters for FPGAs. *Soft Computing*, 2013, vol. 17, no. 11, p. 2163–2180. DOI: 10.1007/s00500-013-1040-8

[26] MONAJATI, M., FAKHRAIE, S. M., KABIR, E. Approximate arithmetic for low-power image median filtering. *Circuits, Systems, and Signal Processing*, 2015, vol. 34, no. 10, p. 3191–3219. DOI: 10.1007/s00034-015-9997-4

[27] VASICEK, Z., MRAZEK, V., SEKANINA, L. Evolutionary functional approximation of circuits implemented into FPGAs. In *Proceedings of the IEEE Symposium Series on Computational Intelligence, Evolvable Systems (SSCI ICES)*. 2016, p. 1–8. DOI: 10.1109/SSCI.2016.7850173

[28] EL-HAROUNI, W., REHMAN, S., PRABAKARAN, B. S., et al. Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2017, p. 1384–1389. DOI: 10.23919/DATE.2017.7927209

[29] GUPTA, V., MOHAPATRA, D., RAGHUNATHAN, A., et al. Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017, vol. 32, no. 1, p. 124–137. DOI: 10.1109/TCAD.2012.2217962

[30] JRIDI, M., MEHER, P. A scalable approximate dct architectures for efficient HEVC compliant video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016, p. 1–10. DOI: 10.1109/TCSVT.2016.2556578

[31] KULKARNI, P., GUPTA, P., ERCEGOVAC, M. D. Trading accuracy for power in a multiplier architecture. *Journal of Low Power Electronics*, 2011, vol. 7, no. 4, p. 490–501. DOI: 10.1166/jolpe.2011.1157

[32] LOTFI, A., RAHIMI, A., YAZDANBAKHSH, A., et al. Grater: An approximation workflow for exploiting data-level parallelism in FPGA acceleration. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2016, p. 1279–1284. DOI: 10.3850/9783981537079_0805

[33] RAHA, A., JAYAKUMAR, H., RAGHUNATHAN, V. A power efficient video encoder using reconfigurable approximate arithmetic units. In *Proceedings of the 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems*. 2014, p. 324–329. DOI: 10.1109/VLSID.2014.62

[34] SNIGDHA, F. S., SENGUPTA, D., HU, J., et al. Optimal design of JPEG hardware under the approximate computing paradigm. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. 2016, p. 106:1–106:6. DOI: 10.1145/2897937.2898057

About the Authors . . .

Lukáš SEKANINA received all his degrees (Ing. in 1999, Ph.D in 2002) from Brno University of Technology, Czech Republic. He was awarded with the Fulbright scholarship to work with NASA Jet Propulsion Laboratory at Caltech in 2004. Prof. Sekanina was a visiting professor with CEI UPM Madrid (2012), Pennsylvania State University, Erie (2001) and visiting researcher with Department of Informatics, University of Oslo (2001). He has served as an associate editor of IEEE Transactions on Evolutionary Computation (2011-2014), Genetic Programming and Evolvable Machines

Journal and International Journal of Innovative Computing and Applications. Prof. Sekanina (co)authored over 150 papers mainly on evolutionary design and evolvable hardware and 1 patent. He is currently a full professor and Head of the Department of Computers Systems at Faculty of Information Technology, Brno University of Technology.

Zdeněk VAŠÍČEK received the Ing. and Ph.D. degrees in electrical engineering and computer science from the Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic, in 2006 and 2012. He is an Associate Professor with the Faculty of Information Technology, Brno University of Technology. His research interests include evolutionary design and optimization of complex digital circuits and systems. He has (co)authored over 40 conference/journal papers focused on evolvable hardware and hardware design. His work was awarded with Silver (2011) and Gold (2015) medal at HUMIES.

Vojtěch MRÁZEK received the Ing. degree in computer science and engineering from the Faculty of Information Technology, Brno University of Technology, Czech Republic in 2014. Currently, he is a PhD student at the Faculty of Information Technology with Evolvable Hardware Group. He has (co)authored over 10 conference/journal papers focused on evolvable hardware and hardware design. His research interests are evolvable hardware, power-aware design, approximate computing and genetic programming.