High level validation of an optimization algorithm for the implementation of adaptive Wavelet Transforms in FPGAs

Rubén Salvador, Félix Moreno, Teresa Riesgo
Centre of Industrial Electronics
Universidad Politécnica de Madrid
José Gutierrez Abascal, 2
28006, Madrid, Spain
Email: ruben.salvador@upm.es

Lukáš Sekanina
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
612 66 Brno, Czech Republic
Email: sekanina@fit.vutbr.cz

Abstract

The work reported in this paper describes the steps given towards an FPGA-based implementation of evolvable wavelet transforms for image compression in embedded systems. An Evolutionary Algorithm (EA) for the design and optimization of the transform coefficients is tailored for a suitable System on Chip implementation. Several cut downs on the computing requirements have been done to the original algorithm, adapting it for the FPGA implementation. What this paper addresses more specifically is the validation of the algorithm using fixed point arithmetic for the whole optimization process. The results show how high quality transforms are evolved from scratch with limited precision arithmetic. Also, preliminary results of the implementation in an FPGA device are included.

1. Introduction

Traditional implementations of image compression algorithms, whether they are conceived for small computing devices or bigger systems such as medical or industrial image vision systems, lack an adaptation capability that is gaining importance.

New compression standards such as JPEG2000 are based on the Discrete Wavelet Transform (DWT) [1], in contrast to the previous JPEG standard based on the Discrete Cosine Transform (DCT). One of the main characteristics of the wavelet transform is its inherent adaptation capability. The transform performance, from the compression point of view, is determined by the wavelet used. This means that it may deteriorate if a different type of image than the one the wavelet is adapted to is used. The accepted standard wavelet considered to be the state of the art in compression of photographic images is the hand-designed D9/7 Cohen-Daubechies-Feauveau (9/7-CDF or also D9/7). Therefore, though it has a better coding performance compared to JPEG, this will hold just for the type of images the wavelet is adapted to.

The line of research open by the authors of this work deals with the implementation of adaptive wavelet transforms in FPGA devices so that embedded systems are provided with adaptation capabilities, which may be applied, for instance, to an artificial vision system able to self calibrate itself when it is deployed on different environments (even to adapt through its operational life) and has to deal with different type of images.

The first step in the work is the design of an algorithm able to find new wavelets adapted to an specific kind of images. An Evolutionary Algorithm (EA), specifically an Evolution Strategy (ES) [2] was chosen as the search algorithm. This approach was already proved useful in previous works reported by other authors, which will be analyzed in Section 4. Since the intended computing platform is an FPGA device, a relatively low computing power will be available, what will, undoubtedly, affect the performance of the evolutionary search.

The structure of the paper is as follows. Section 2 covers an introduction to Evolution Strategies, followed by a brief overview of the wavelet transform in Section 3. An analysis of previously reported works is given in Section 4. Afterwards, the original proposals of this work are presented in Section 5, where the fixed point implementation is tackled, as well as the mapping of the algorithm to the hardware architecture.

The experimental setup developed and a further insight into the details of the implementation, along with the results obtained and some comments on the future work can be found in Section 6. Paper is concluded in Section 7.

2. Design and optimization via Evolution Strategies

An Evolution Strategy (ES) is one of the fundamental algorithms among Evolutionary Algorithms (EA) that utilize a population of candidate solutions and bio-inspired operators to search for a target solution. EAs have been chosen as the optimization algorithm because of their applicability to a wide range of problems and because they do not need much tailoring for specific problems. Still, they deliver good (not necessarily optimal) solutions within acceptable time.

ESs are primarily used for optimization of real-valued vectors. The algorithm operators are iteratively applied within a loop, where each loop run is called a *generation*, until a termination criterion is met. Variation is accomplished by the so-called *mutation* operator. For real-valued search spaces, mutation is normally performed by adding a normally distributed random value to each component under variation. Based on a normal (Gaussian) distribution, with mean ξ and standard deviation σ, mutations are then done by adding some Δx_i to each parameter x_i encoded in the individuals.

One of the particular features of ES is that the individual step sizes of the variation operator for each coordinate (or correlations between coordinates) is governed by self-adaptation (or by covariance matrix adaptation (CMA-ES)). This self-adaptation of the step size, also known as *mutation strength* (i.e. standard deviation of the normal distribution), implies that σ is also included in the chromosomes, undergoing variation and selection itself (coevolving along with the solutions).

The canonical versions of the ES are denoted by $(\mu/\rho, \lambda)$-ES and $(\mu/\rho + \lambda)$-ES, where μ denotes the number of parents, $\rho \leq \mu$ the mixing number (i.e., the number of parents involved in the procreation of an offspring), and λ the number of offspring. The parents are *deterministically selected* from the set of either the offspring, referred to as *comma-selection* ($\mu < \lambda$), or both the parents and offspring, referred to as *plus-selection*. Selection is based on the ranking of the individuals' fitness $F(\mathbf{y})$ taking the μ best individuals. Once selected, ρ out of the μ parents are *recombined* to produce an offspring individual using *intermediate recombination*, where the parameters of the selected parents are averaged, or randomly chosen if *discrete recombination* is used.

Each ES individual $\mathbf{a} := (\mathbf{y}, \mathbf{s})$ comprises the *object parameter vector* $\mathbf{y} \in \mathcal{Y}$ to be optimized and a set of strategy parameters $\mathbf{s}$, needed especially in self-adaptive ESs. For a general algorithmic description of the $(\mu/\rho \stackrel{+}{,} \lambda)$-ES see [2].

3. Overview of the Wavelet Transform

The DWT is a multiresolution analysis (MRA) tool widely used in signal processing due to its joint time-frequency signal analysis characteristics, used for the analysis of the frequency content of a signal at different resolutions. It concentrates the signal energy into fewer coefficients to increase the degree of compression when the data is encoded. The energy of the input signal is redistributed into a low resolution trend sub-signal (scaling coefficients) and high resolution sub-signals (wavelet coefficients; horizontal, vertical and diagonal sub-signals for image transforms). If the wavelet chosen for the transform is suited for the type of image being analyzed, most of the information of the signal will be kept in the trend sub-signal, while the wavelet coefficients (high frequency details) will have a very low value. For this reason, the DWT can reduce the number of bits required to represent the input data.

For a general introduction to wavelet based multiresolution analysis see [3]. The *Fast Wavelet Transform* (FWT) algorithm computes the wavelet representation by means of a *subband filtering* scheme, recursively filtering the input data with a pair of high-pass and low-pass digital filters and downsampling the results by a factor of two [1]. The D9/7 wavelet gets its name because its high-pass and low-pass filters have 9 and 7 coefficients respectively.

The FWT algorithm was improved by the method known as *Lifting scheme* (LS), introduced by Sweldens [4], which reduces the computational cost of the transform. It does not rely on the Fourier Transform for its definition and application, and has given rise to the so called *Second Generation Wavelets* [5]. Besides, the research effort put on LS has made easier the construction of custom wavelets for very specific and different types of data.

The basic LS, shown on Figure 1, consists of three stages: ***Split***, ***Predict*** and ***Update***. The main idea consist on exploiting the correlation structure of the input data to obtain a more compact representation of the signal [6].

Split stage divides the input data into two smaller subsets, s_{j-1} and d_{j-1}, usually, the even and odd samples. It is also called the *Lazy Wavelet*.

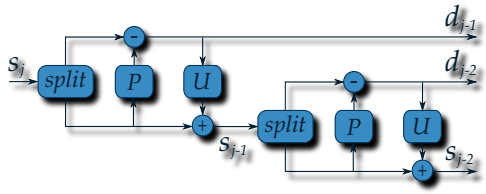

Figure 1. Lifting scheme

To obtain a more compact representation of the input data, the s_{j-1} subset is used to *predict* the d_{j-1} subset, called the wavelet subset, and which is based on the correlation of the original data. The difference between the prediction and the actual samples is stored, also, as d_{j-1}, overwriting its original value. If the prediction operator P is reasonably well designed, the difference will be very close to 0, so the two subsets s_{j-1} and d_{j-1} yield a more compact representation of the original data set s_j.

In most cases, it is interesting to maintain some properties of the original signal after the transform, such as the mean value. For this reason, the LS proposes a third stage that not only reuses the computations already done in the previous stages, but that also defines an easily invertible scheme. This is accomplished by *updating* the s_{j-1} subset with the already computed wavelet set d_{j-1}. The wavelet representation of s_j is therefore given by the set of coefficients $\{s_{j-2}, d_{j-2}, d_{j-1}\}$.

This scheme can be iterated up to n levels, so that an original input data set s_0 will have been replaced with the wavelet representation $\{s_{-n}, d_{-n}, \ldots, d_{-1}\}$. Therefore the algorithm for the LS implementation is:

for $j \leftarrow 1, n$ **do**
  $\{s_j, d_j\} \leftarrow Split(s_{j+1})$
  $d_j = d_j - P(s_j)$
  $s_j = s_j + U(d_j)$
end for

There exists a different notation for the transform coefficients $\{s_{j-i}, d_{j-i}\}$; for a 2 level image decomposition it is $\{LL, LH, HL, HH\}$ where L stands for low pass and H for high pass coefficients respectively.

4. Previous work on evolutionary wavelets design

Research on adaptive wavelets has been active during the last two decades. Basically, dictionary-based methods, in some cases combined with EA, as well as several stochastic optimization techniques like simulated annealing, or even using the LS technique, have been reported.

The work described on this paper gets its original idea of [7] by Grasemann and Miikkulainen. In their work, the authors proposed the original idea of combining the lifting technique with EA for designing wavelets. As it is drawn from [4], [5] the LS is really well suited for the task of using an EA to encode wavelets, since any random combination of lifting steps will encode a valid wavelet, what guarantees perfect reconstruction.

Table 1 shows the most remarkable and up to date reported results in the design of wavelet transforms by means of Evolutionary Computation (EC). It should be noted that in the cases of MRA, the coefficients evolved each at level were different, since the authors reported better results with this scheme.

Two are the most original contributions to the state of the art reported by Grasemann and Miikkulainen. First, the use of a coevolutionary GA (parallel evolving populations) to encode wavelets as a sequence of lifting steps. Second, the proposal of an idealized version of a transform coder to save time in the complex evaluation method they used, which involved computing Peak Signal to Noise Ratio (PSNR) for one individual combined a number of times with other individuals from each subpopulation. They propose using only a certain percentage of the largest coefficients for reconstruction.

The evaluation consisted of 80 runs, each of which took approximately 45 minutes on a 3 GHz Xeon processor (total time $80 * 45$). The results obtained in this work outperformed the considered state-of-the-art wavelet for fingerprint image compression, the FBI standard based on the D9/7 wavelet, in 0.45 dB. The set of 80 images used was the same as the one used in this work, as shown in Section 6.

Works reported by Babb, Moore et. al. can be considered the current state of the art in the use of EC for image transform design. The milestones followed in their research are summarized on the next list.

1) Evolve the inverse transform for digital photographs under conditions subject to quantization.
2) Evolve *matched* forward and inverse transform pairs.
3) Evolve coefficients for three and four level MRA transforms.
4) Evolve a different set of coefficients for each level of MRA transforms.

As stated by the authors, Babb, Moore, et al., their algorithms are highly computationally intensive, so the training runs were done using supercomputing resources, available through the use of the Arctic Region Supercomputer Center (ARSC) in Fairbanks, Alaska.

Table 1. State of the Art in evolutionary wavelets design.

Ref.	EA	Seed	Conditions	Image set	Result (improvement)
[8]	GA	D4 mutations	MRA (3). 64:1 Q[a]	Photographs	0.60 dB (MSE)
[9]	GA	D9/7 mutations	MRA (4). 16:1 T[b]	Fingerprints	0.76 dB (MSE)
				Satellite	**1.79 dB** (MSE)
[10]	CMA-ES[c]	D9/7 mutations	64:1 Q	Fingerprints	**3.00 dB**
				Photographs	**2.39 dB**
[11]	CMA-ES	0.2	MRA (3). 64:1 Q	Fingerprints	0.54 dB (MSE)
[7]	Coevolutionary GA	Random Gaussian	MRA. 16:1 T	Fingerprints	0,45 dB (PSNR)

[a] Quantization [b] Thresholding [c] Covariance Matrix Adaptation-Evolution Strategy

Although the work by Grasemann and Miikkulainen was done on an accessible computer, both training times and computing resources needed in both cases, show the complexity of the algorithms developed. These approaches are highly unfeasible for implementation as a hardware embedded system.

5. Proposed Evolutionary Algorithm

As proposed in the last reports by Babb, Moore, et al. an ES seemed also to be the most suited algorithm to meet the requirements of this work, but, however, a simpler one was chosen so that a viable hardware implementation was possible. By the contrary, this work uses LS to encode the wavelets, as Grasemann and Miikkulainen. Therefore, it is being originally proposed in this work to mix both proposals from the literature; using ES as the search algorithm and encoding the individuals (wavelets) using the LS.

The whole process of evolution has to be downscaled in complexity. This implies changing not only the parameters of evolution, but the EA itself. Although the hardware architecture of an FPGA may be considered up to some extent the paradigm of parallelism, the complexity of such an implementation is much higher than the software choice. A trade-off between scalability, use of hardware resources and difficulty (complexity) of the implementation has to be taken into account when setting system performance requirements. Therefore, the first task accomplished by the authors was centered on building a prototype implementation of the system to validate the proposed resource consumption vs performance trade-off. In [12] the decisions made for designing a simpler algorithm than those previously published in the State of the Art are presented. They are summarized below:

1) *Single evolving population* opposed to the parallel populations of the coevolutionary genetic algorithm reported in [7] .
2) Use of *uncorrelated mutations with one step size* [2] instead of the overcomplex CMA-ES reported in [10], [11].
3) Evolution of *one single set of coefficients for all MRA levels*.
4) *Ideal evaluation of the transform*. Since doing a complete compression would turn to be an unsustainable amount of computing time, the simplified evaluation method reported in [7] was further improved. For this work, all wavelet coefficients d_j are zeroed, keeping only the trend level s_j of the transform from the last iteration of the algorithm. The evaluation of the individuals in the population is done by computing the PSNR after setting entire bands of high-pass coefficients to 0. Two decomposition levels are done and only the LL coefficients (trend image) are kept before reconstructing the image and measuring the resulting error. For 2 levels of decomposition, this is an idealized 16:1 compression ratio.

These simplifications yielded very positive results in previous works by the authors [12]. But, since there were still some complex operations around in the algorithm, the complexity relaxation before doing the hardware implementation was taken even further, so that a trade-off between performance and size is observed. They are summarized below:

1) *Uniform random distribution*. Instead of using Gaussian distributions for the mutation of the object parameters, a Uniform distribution was tested, for being simpler to implement in HW.
2) *Mean Absolute Error (MAE) as evaluation figure*. PSNR is the figure of merit more widely used for image processing tasks. But, as previous works show for image filter design problems [13], using MAE gives almost identical results, because the interest is in relative comparisons among population members.

5.1. Fixed point arithmetic

For the implementation of the algorithm in an FPGA embedded system, special care with binary arithmetic should be taken, since floating point representation is

not hardware(FPGA)-friendly. Thanks to the LS, the Integer Wavelet Transform (IWT) [14] turns up as a good solution for wavelet transforms in embedded systems. But, since floating point arithmetic is still in the system as filter coefficients, a fixed point implementation is needed for hardware.

As shown on [15], [16], for 8 bits per pixel (bpp) integer inputs from an image, a fixed point fractional format of Q2.10 for the lifting coefficients and a bit length in between 10 and 13 bits for a 2 to 5 level MRA transform for the partial results is enough to keep a rate-distortion performance almost equal to what is achieved with floating point arithmetic. This yields Multiply and Accumulate (MAC) units of 20-23 bits (10 bits fractional coefficients part + 10-13 bits for partial transform results). Details on the fixed point implementation of the whole evolutionary process are given in Section 6.

5.2. Algorithm to Hardware architecture mapping

Figure 2 shows the general architecture of the system to be implemented in the FPGA. Critical operations of the algorithm has been identified and optimized so that a hardware implementation is feasible. So far, these modules are the *Wavelet Transform* (the most time consuming operation), the *Fitness Computation*, the *Population Sorting* and the *Evolution Strategy*. All of them, except for the ES will be implemented as hardware peripherals attached to a microprocessor embedded in the FPGA. This is a typical scenario in these kind of systems, where some degree of performance is sacrificed to gain flexibility in the system, so that modifications may be easily done to the software implementation of the EA (which is, of course, much easier than changing its hardware counterpart).

Since the LS was proposed, several hardware implementations have been reported both for ASICs and FPGAs (JPEG2000 adopted LS). This means that good results centered on exploiting LS features to obtain fast implementations have already been done. But the objectives at this stage of the work are just directed to prove and validate the concepts and the feasibility of the system as a whole. Therefore, the implementation of the Wavelet Transform is a direct, algorithmic mapping of the LS to its hardware equivalent VHDL description (i.e., no hardware optimizations at the level of data dependencies are accomplished).

As stated before, the embedded microprocessor is responsible for running the ES (and each one of its associated operators, to know, recombination, mutation and selection). After generating a new offspring

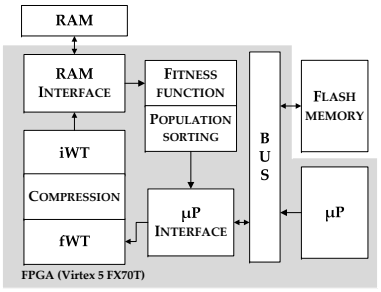

Figure 2. System level architecture

population, each of them are sequentially sent to the peripheral responsible of its evaluation. This comprises the computation of the fitness as the MAE figure. To tackle it, the following sequence of operations (as named on Figure 2) has to be computed: Forward Wavelet Transform (fWT), Compression (C), Inverse Wavelet Transform (iWT) and MAE (Fitness) computation. Besides, when each offspring individual has been evaluated, the population has to be sorted according to it (Population Sorting). At this stage, the microprocessor may close the evolutionary loop by applying *selection* to form the new parent population. After it, *recombination* and *mutation* are applied and a new offspring population will be available to be evaluated.

Taking advantage of the LS features, the fWT and iWT can be computed by just doing a sign flip and a reversal in the order of operations (P and U stages). As a first attempt, they share resources, using the same hardware in the FPGA. Compression block is simple, since it only needs to substitute the fWT result by zeros in the detail bands $\{LH, HL, HH\}$. Therefore, it is working in parallel to the fWT. In a similar manner, the Fitness module computes the difference image as each pixel is produced by the iWT.

The fWT/iWT module is built up by applying the sequence of P, U stages dictated by the LS. To mimic the high level modeling of the algorithm (see Section 6), 6 stages have been implemented (3 P and 3 U), each one containing 4 filter coefficients. The implementation of each P, U stage can be seen in Figure 3. Section 6 show the first preliminary results of the implementation.

6. Experimental setup, results and discussion

6.1. Experimental setup

Before accomplishing the hardware implementation, modeling and simulation of the algorithm was

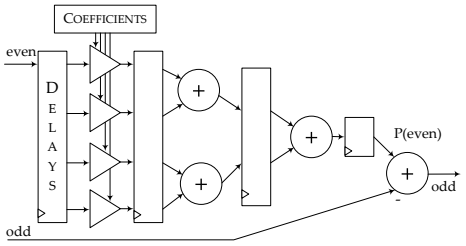

Figure 3. Predict/Update stage implementation

accomplished using Python computing language together with its numerical and scientific extensions, NumPy and Scipy [17], as well as the plotting library, MatPlotlib [18]. Fixed-point arithmetic was modeled with integer types, defining the required quantization/dequantization and bit-alignment routines to mimic hardware behavior. The computer used was a laptop containing an Intel Core™2 Duo processor at 2 GHz running a Debian GNU/Linux 64 bits operating system. Each run of the algorithm took around 40 minutes to evolve for 1000 generations.

Standard ***representation*** of the individuals in ES is composed of a set of *object parameters* to be optimized, and *strategy parameters*, which determines the extent to which the object parameters are perturbed by the mutation operator:

$$\langle x_1, \ldots, x_n, \sigma \rangle$$

with x_i being integer values representing the fixed-point coefficients of the predict and update stages. The individuals were seeded both randomly and with the D9/7 wavelet.

The ***encoding*** of each wavelet individual is of the form:

$$\langle P_1, U_1, P_2, U_2, P_3, U_3, k_1, k_2 \rangle$$

where each P_i, U_i is made up of 4 coefficients and k_i are single coefficients. Therefore, the total length of each chromosome is $n = 26$. D9/7 wavelet uses $\langle P_1, U_1, P_2, U_2, k_1, k_2 \rangle$.

The ***mutation*** operator is an *uncorrelated mutation with one step size*, σ. The formulas for the mutation mechanism are:

$$\sigma' = \sigma \cdot \exp^{\tau \cdot N(0,1)} \quad (1)$$

$$x_i' = x_i + \sigma' \cdot U_i(-\sigma', \sigma') \quad (2)$$

where $N(0,1)$ is a draw from the standard normal distribution and $U_i(-\sigma', \sigma')$ a separate draw from the discrete uniform distribution for each variable i (for each object parameter). The parameter τ resembles the so called *learning rate* of neural networks, and it is proportional to the square root of the object variable length n:

$$\tau \propto 1/\sqrt{n}$$

The ***fitness function*** used to evaluate the offspring individuals, MAE, is defined as:

$$MAE = \frac{1}{RC} \sum_{i=0}^{R-1} \sum_{j=0}^{C-1} |I(i,j) - K(i,j)| \quad (3)$$

where R, C are the rows and columns of the image and I, K the original and transformed images respectively.

For the ***survivor selection***, *comma-selection* has been chosen. It is generally preferred in ES over the *plus-selection* for being, in principle, able to leave (small) local optima and not letting survive misadapted strategy parameters. Therefore, no *elitism* is allowed.

The ***recombination*** scheme chosen is *intermediate recombination*, that averages the parameters (alleles) of the selected parents.

To sum up, for the various tests run, the set of parameters used correspond to a $(10/5, 70)$-ES with a varying $\sigma = \{0.1, \ldots, 1.5\}$. Initial population was seeded both randomly and with the D9/7 wavelet.

The experiments reported in this work have also used, as in [7], the first set of 80 images of the FVC2000 fingerprint verification competition. Images were black and white, sized 300x300 pixels at 500 dpi resolution. One random image was used for training and the other 79 for testing purposes.

6.2. Algorithm optimization results

All the results obtained are compared with the D9/7 transform implemented in fixed point arithmetic and evaluated with the proposed method. Although evolution used MAE as the fitness function, for comparison purposes, part of the results shown in this Section are given as PSNR. As a reference, for the fixed point D9/7 wavelet, an average PSNR of 28.74 dB was measured over the set of 79 test images.

Figure 4 shows how the algorithm behaves during a typical run. As it can be seen, around generation 100 the performance of the evolved wavelet is already similar to the D9/7. In Figure 5 the best evolved wavelet is compared with the D9/7 for the whole image test set, showing how the algorithm was able to evolve a solution averaging 29.80 dB, that outperforms the standard D9/7 by an average of 1,06 dB. Results correspond to $\sigma = 1.0$ and a randomly seeded population.

In Figure 6 a comparison of the best evolved wavelet with a fixed point implementation of D9/7 is shown.

(a) (b) (c)

Figure 6. (a) Original fingerprint image and comparison of the performance obtained with the best evolved individual (b) and the D9/7 Fixed Point implementation (c)

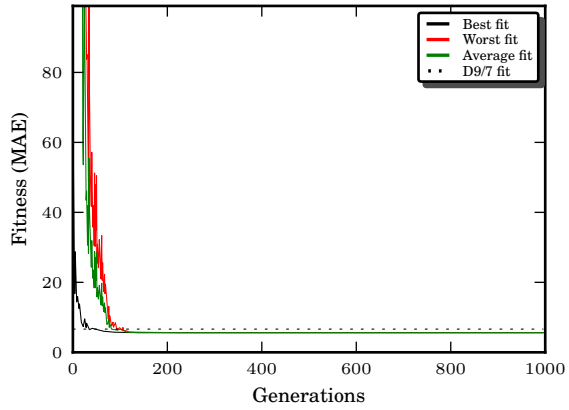

Figure 4. Result of a typical evolution run

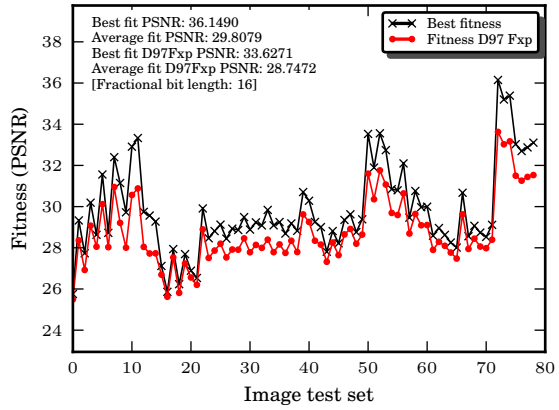

Figure 5. Tests of the best evolved wavelet. The best individual (and D9/7 for comparison) is exercised for each of the 79 images of the test set

6.3. Preliminary Hardware results

An ML507 development board has been chosen as the prototype platform. It contains a Xilinx Virtex 5 XC5VFX70T FPGA device with an embedded PowerPC processor, which runs the ES. Table 2 shows the preliminary implementation results for an over-dimensioned datapath of 32 bits, using 16 bits for the fractional part representation. This implementation is directed to a system level functional validation in the FPGA, yielding higher area results than expected for the final system.

Table 2. Preliminary implementation results for the main modules in the system.

Module	Resources	Frequency (max)
fWT/iWT	2285 / 11200 (20%)	147 MHz
Fitness function	25 / 11200 (<1%)	330 MHz
Population sorting	984 / 11,200 (8%)	265 MHz

6.4. Discussion and future work

The results presented have shown how the algorithm is not only able of evolving good quality wavelets, but also, outperforming the standard D9/7 wavelet. This can been noticed in Figure 6 where some artifacts appear in the D9/7 result. It has to be stated, that, if a real transform coder had been used, these artifacts may have not appeared. They are due to the ideal compression used in this work. Therefore, higher quality results (PSNR/MAE) are to be expected with a standard image coder in both cases (D9/7 and evolved).

The preliminary (and partial) hardware results show the feasibility of the implementation of the complete system. The FPGA chosen as the implementation device is more than enough to host the system, probably leaving extra available resources if a more complex implementation is decided to be carried out.

At present, the rest of the system is being implemented in the FPGA. Once the whole system is

functionally validated, the hardware description will be optimized for a final implementation. For this implementation, further tests for an optimized σ mutation strategy will be done.

7. Conclusion

In this paper, a simplified ES (as compared to the related reported works) has been validated for a fixed point implementation, showing better average performance than the D9/7 wavelet, used in JPEG2000 and in the FBI compression standard. The results of the preliminary FPGA implementation validate its viability from the point of view of, both, an implementation directed to the FPGA acceleration of the evolutionary process and for an adaptive embedded system.

Acknowledgment

The main author would like to thank the support received from the Department of Computer Systems, Brno University of Technology, during his research stay as part of his PhD degree.

Lukas Sekanina has been supported by MSMT under research program MSM0021630528 and by the grant of the Czech Science Foundation GP103/10/1517.

References

[1] S. Mallat, *A Wavelet Tour of Signal Processing, Second Edition*, 2nd ed. Academic Press, Sep. 1999.

[2] H. Beyer and H. Schwefel, "Evolution Strategies. A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, Mar. 2002.

[3] B. Jawerth and W. Sweldens, "An overview of wavelet based multiresolution analyses," *SIAM Review*, vol. 36, no. 3, pp. 377–412, 1994.

[4] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186 – 200, 1996.

[5] ——, "The lifting scheme: a construction of second generation wavelets," *SIAM J. Math. Anal.*, vol. 29, no. 2, pp. 511–546, 1998.

[6] ——, "The lifting scheme: a new philosophy in biorthogonal wavelet constructions," in *Wavelet Applications in Signal and Image Processing III*, A. F. Laine and M. Unser, Eds., vol. 2569, no. 1. SPIE, 1995, pp. 68–79.

[7] U. Grasemann and R. Miikkulainen, "Effective image compression using evolved wavelets," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. Washington DC, USA: ACM, 2005, pp. 1961–1968.

[8] F. Moore and B. Babb, "Revolutionary image compression and reconstruction via evolutionary computation, part 2: multiresolution analysis transforms," in *Proceedings of the 6th WSEAS International Conference on Signal, Speech and Image Processing*. Lisbon, Portugal: World Scientific and Engineering Academy and Society (WSEAS), 2006, pp. 144–149.

[9] B. Babb and F. Moore, "The best fingerprint compression standard yet," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2007, pp. 2911–2916.

[10] B. Babb, F. Moore, M. Peterson, T. H. O'Donnell, M. Blowers, and K. L. Priddy, "Optimized satellite image compression and reconstruction via evolution strategies," in *Evolutionary and Bio-Inspired Computation: Theory and Applications III*, vol. 7347. Orlando, FL, USA: SPIE, May 2009, pp. 73 470O–10.

[11] B. J. Babb, F. W. Moore, and M. R. Peterson, "Improved multiresolution analysis transforms for satellite image compression and reconstruction using evolution strategies," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. Montreal, Qubec, Canada: ACM, 2009, pp. 2547–2552.

[12] R. Salvador, F. Moreno, T. Riesgo, and L. Sekanina, "Evolutionary design and optimization of wavelet transforms for image compression in embedded systems," in *IEEE International Conference on Adaptive Hardware and Systems - To appear*, 2010.

[13] Z. Vasicek and L. Sekanina, "An evolvable hardware system in Xilinx Virtex II Pro FPGA," *Int. J. Innov. Comput. Appl.*, vol. 1, no. 1, pp. 63–73, 2007.

[14] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 332 – 369, 1998.

[15] M. Martina, G. Masera, G. Piccinini, and M. Zamboni, "A VLSI architecture for IWT (integer wavelet transform)," in *Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on*, vol. 3, 2000, pp. 1174–1177 vol.3.

[16] M. Grangetto, E. Magli, M. Martina, and G. Olmo, "Optimization and implementation of the integer wavelet transform for image coding," *Image Processing, IEEE Transactions on*, vol. 11, no. 6, pp. 596–604, 2002.

[17] T. E. Oliphant, "Python for scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 10–20, 2007. [Online]. Available: http://link.aip.org/link/?CSX/9/10/1

[18] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 90–95, 2007.