A Novel 2D Filter Design Methodology For Heterogeneous Devices

Christos-Savvas Bouganis, George A. Constantinides and Peter Y. K. Cheung
Department of Electrical and Electronic Engineering
Imperial College London
London, U.K.
Email: christos-savvas.bouganis@imperial.ac.uk

Abstract

In many image processing applications, fast convolution of an image with a large 2D filter is required. Field Programable Gate Arrays (FPGAs) are often used to achieve this goal due to their fine grain parallelism and reconfigurability. However, the heterogeneous nature of modern reconfigurable devices is not usually considered during design optimization. This paper proposes an algorithm that explores the implementation architecture of 2D filters, targeting the minimization of the required area, by optimizing the usage of the different components in a heterogeneous device. Experiments show that the proposed algorithm can achieve a reduction in the required area in a range of 34% *to* 70% *when compared to current techniques.*

1 Introduction

In recent years, many image processing applications have appeared in the literature that require the use of large 2D filters. Moderate size examples can be found in face detection/recognition applications [4] where kernels with size of 23×23 pixels are used, and some more extreme examples can be found in medical imaging where applications require kernels with size of up to 63×63 pixels. At the same time, real-time implementation is often required, making the use of hardware acceleration a necessity [1].

FPGAs are often used to achieve this goal due to their fine grain parallelism and reconfigurability. Modern FPGAs are heterogeneous devices, often targeting the DSP community, and thus providing a mixture of resources that can be used by DSP applications. The two main silicon cores that are usually included in the recent devices are embedded RAMs [15] and embedded multipliers [5]. The first one provides fast localized memory access, while the second one provides high speed accurate multiplication.

Current techniques for 2D filter optimization for a modern reconfigurable device, such as word-length optimization [3] and Singular Value Decomposition [12], do not take into account the heterogeneity of the device. However, research concerning the exploitation of heterogeneity for a particular application has recently started to appear in the literature. In [9], the authors propose an approach for exchanging embedded RAMs for multipliers, whereas in [14] the author proposes an algorithm that identifies part of the circuit that can be implemented in embedded RAMs.

In line with this direction, the proposed algorithm departs from the current methods of 2D filter implementation by providing an approach that makes explicit use of the heterogeneity of the device, targeting designs that use less area. Furthermore, it provides a framework which allows the designers to move their designs to different points in the three dimensional design space of embedded RAMs, embedded multipliers, and 4-input look-up tables (4-LUTs), keeping the arithmetic error in the filter implementation below a specified level.

The novel contributions of this paper are:

- To use Singular Value Decomposition to approximate a 2D filter with a number of $2\times$1D convolutions and one low complexity 2D convolution. This reduces the number of high precision multipliers required for implementation.
- To develop a resource allocation algorithm that maps the decomposed filter design onto a given set of heterogeneous resources on an FPGA, including dedicated multipliers, LUTs and embedded RAM in order to minimize resource usage. The resulting method shows a significant reduction in used resources.

The paper is organized as follows. Section 2 describes the current related work regarding filter designs for heterogeneous devices. High level and detailed descriptions of the proposed algorithm are given in Section 3. Section 4 is focused on the cost models for the filter components and Section 5 illustrates the speed and the convergence properties of the algorithm. Finally, Section 6 focuses on the evaluation of the algorithm, and Section 7 concludes the paper.

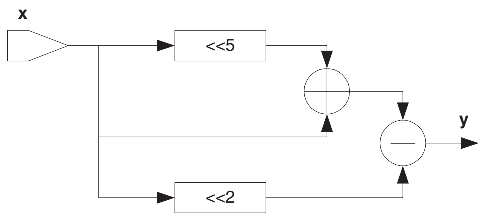

Figure 1. Detailed diagram of a constant coefficient multiplier using canonic signed digit recoding. Coefficient 29 is recoding to [1 0 0 -1 0 1] using CSD, which leads to a reduction in the required adders.

2 Related work

The paper focuses on the case where designs with high throughput are required, but design latency is of secondary importance. For this reason, only pipelined techniques for implementation of a 2D convolution filter are considered. A common technique for implementation of such a filter on an FPGA is to use constant coefficient multipliers and a number of embedded RAMs. The RAMs are used to buffer the incoming data in order to create the necessary 2D window on which the mask is applied. The constant coefficient multipliers are often implemented as shift/add combinations using 4-LUTs and, to further optimize the design, the coefficients are sometimes transformed using *canonic signed digit* recoding, which reduces the required logic [6].

Canonic signed digit recording represents the coefficients in a way such that high-speed low area multiplication can be achieved. It is a radix-2 signed digit system using the set $\{1, 0, -1\}$. Given a number, its canonic signed digit representation has two important properties: (a) the number of non-zero digits is minimum, and (b) no two non-zero digits are adjacent. Due to the first property, canonic signed representation is widely used for implementing constant coefficient multipliers [11]. Figure 1 illustrates an example of CSD, in the case where the coefficient has the value 29. The usual quantization needs three adders, where under CSD recoding only two are required.

Another technique exploits the potential separability of a 2D filter into sets of two 1D filters by using the Singular Value Decomposition (SVD) [12] to express the original filter as a linear combination of separable filters. Using this technique, the initial filter can be implemented as a set of 1D filters where half of them are applied to the rows of the image, and the other half to the columns. By decomposing the 2D filter, the number of necessary multiplications may be reduced. For a 2D filter with size $m \times m$, the number of multiplications in the original form is m^2. By applying the Singular Value Decomposition algorithm, each stage of the decomposition requires $2m$ multiplications. However, the number of levels of decomposition that are required depends on the separability properties of the filter and the arithmetic accuracy for storing intermediate results. Moreover, the decomposition of the filter into separable masks is independent of the quantization process of the coefficients. This technique can be combined with algorithms that minimize the area cost of a filter by representing the coefficients using an appropriate number of bits such that the final error at the output of the filter is bounded by a user defined value such as in [3]. Current algorithms can be classified in three categories. Those that use an analytic approach to scaling and error estimation [10], those that use simulation [7] and finally those that use both techniques [2].

In this paper, we propose a novel algorithm to optimize a 2D convolution filter implementation in a heterogeneous device, given a set of constraints regarding the number of embedded multipliers and 4-LUTs. The algorithm estimates an approximation of the original 2D filter which minimizes the mean square error and at the same time meets the user's constraints on resource usage. The proposed method alters the structure of the original filter in order to find a structure that can be mapped in a more efficient way to the targeted device. It is proposed to perform an exploration of the design space at a higher level than the word-length optimization methods, since they do not consider altering the computational structure of the filter. The proposed technique is thus complementary to these previous approaches.

3 Algorithm description

The basic idea of the algorithm is to explore the redundancy in the filter's impulse response using Singular Value Decomposition. In this way, the coefficients are ordered according to their impact to the overall error, and residual errors are propagated to the next level of decomposition, and so on.

The proposed algorithm takes as input the impulse response of a 2D filter $\mathbf{F}$ with size $m \times m$ and a set of constraints for the available embedded multipliers and slices[1]. It produces as output an approximation of the filter which minimizes the mean square error and at the same time meets the set of resource constraints. It should be noted that the assumption of a square 2D filter is used for reasons of clarity; the algorithm works for any 2D filter of any size.

The main idea behind the algorithm is to explore the separability of the input 2D filter. A 2D filter is called separable if its impulse response $f(n_1, n_2)$ is a separable sequence so $f(n_1, n_2) = f_1(n_1) f_2(n_2)$. The important property is that

[1]A slice is a term used by Xilinx, one of the two major FPGA manufacturers, to denote a combination of two 4-LUTs together with additional circuitry to support efficient arithmetic.

a convolution with a separable filter can be decomposed as follows:

$$y(n_1, n_2) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x(n_1 - i, n_2 - j) f_1(i) f_2(j)$$
$$= \sum_{i=-\infty}^{\infty} f_1(i) \sum_{j=-\infty}^{\infty} x(n_1 - i, n_2 - j) f_2(j)$$

where x and y denote the input and output images respectively.

The algorithm decomposes the input filter to a set of N separable filters ($\mathbf{A}_i$) and a non-separable one ($\mathbf{E}$):

$$\mathbf{F} = \sum_{i=1}^{N} \mathbf{A}_i + \mathbf{E} \tag{1}$$

At the same time, the algorithm quantizes the coefficients of the separable and non-separable filters using different precision levels, in order to achieve a more area efficient filter implementation, while meeting the resource constraints. These two processes are not independent since the error in the approximation of one level of the decomposition affects the remaining levels of decomposition. The algorithm calculates the approximation error at each decomposition level and propagates it to the remaining decomposition levels, in order to produce a better approximation of the original filter.

The addition of the non-separable filter $\mathbf{E}$ is essential in the case where the filter under consideration has poor separability properties. It helps to reduce considerably the required levels of decomposition, permitting the algorithm to achieve the required filter approximation within only a few decomposition levels. The cost related to implement the non-separable mask is kept low, since the mask encodes the residual errors of the approximation by dedicating only a few bits for the representation of the coefficients, as described below.

To illustrate the potential for resource saving of this approach, consider an $m \times m$ 2D filter with 16-bit coefficients. A parallel implementation requires m^2 16×16 multipliers, if the data width is 16 bits. Assuming that this is approximated with three levels of separable filters, only $6m$ high precision multipliers are required. Although the implementation of $\mathbf{E}$ still requires m^2 multiplications, the coefficients have a small number of bits, suitable to be mapped to a LUT-based multiplier implementation.

The separability exploration of the filter is performed using the Singular Value Decomposition algorithm [12] which decomposes a matrix into a linear combination of the fewest possible separable matrices. By applying the Singular Value Decomposition algorithm, the initial filter $\mathbf{F}$ is decomposed into a set of separable filters $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_L$:

$$\mathbf{F} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$$
$$= \sum_{i=1}^{L} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T$$
$$= \sum_{i=1}^{L} \mathbf{A}_i \tag{2}$$

where $\mathbf{U}$ and $\mathbf{V}^T$ are orthogonal matrices and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i. The eigenvalues are sorted in descending order, thus $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_L$. $\mathbf{u}_i$ and $\mathbf{v}_i$ correspond to the i^{th} columns of the $\mathbf{U}$ and $\mathbf{V}$ matrices respectively.

Without taking into account the quantization effects of the coefficients into the filter approximation, the mean squared error that is achieved by including the first D decomposition levels is given by:

$$Err_D = \frac{1}{m^2} ||\mathbf{F} - \hat{\mathbf{F}}||^2$$
$$= \frac{1}{m^2} \left|\left| \sum_{i=D+1}^{L} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \right|\right|^2$$
$$= \frac{1}{m^2}$$
$$\operatorname{tr}\left\{ \left(\sum_{i=D+1}^{L} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \right) \left(\sum_{i=D+1}^{L} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \right)^T \right\}$$
$$= \frac{1}{m^2} \sum_{i=D+1}^{L} {\lambda_i}^2 \tag{3}$$

where $|| \cdot ||$ denotes the Frobenius norm, tr$\{\cdot\}$ denotes the trace of a matrix and λ_i denotes the eigenvalue that corresponds to the i^{th} decomposition level. This provides a lower bound for the mean square error in the approximation when only the first D levels of the decomposition are considered without taking into account the quantization error of the coefficients.

Given an input image $\mathbf{I}$ and a 2D filter $\mathbf{F}$, the resulting image of the convolution is given by $\mathbf{Y} = \mathbf{I} \circledast \mathbf{F}$, where $\circledast$ denotes convolution. Using the filter decomposition in (1) and (2), the resulting image is given by:

$$\mathbf{Y} = \mathbf{I} \circledast \left(\sum_{i=1}^{N} \mathbf{A}_i + \mathbf{E} \right)$$
$$= \mathbf{I} \circledast \sum_{i=1}^{N} \mathbf{A}_i + \mathbf{I} \circledast \mathbf{E}$$
$$= \sum_{i=1}^{N} \mathbf{I} \circledast \mathbf{A}_i + \mathbf{I} \circledast \mathbf{E}$$
$$= \sum_{i=1}^{N} \mathbf{I} \circledast (\lambda_i \mathbf{u}_i {\mathbf{v}_i}^T) + \mathbf{I} \circledast \mathbf{E}$$

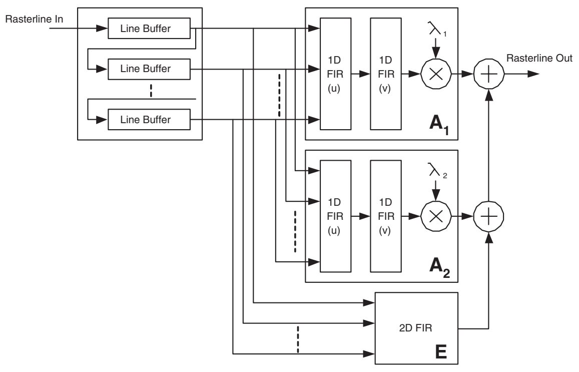

Figure 2. Diagram of the decomposition with $N = 2$

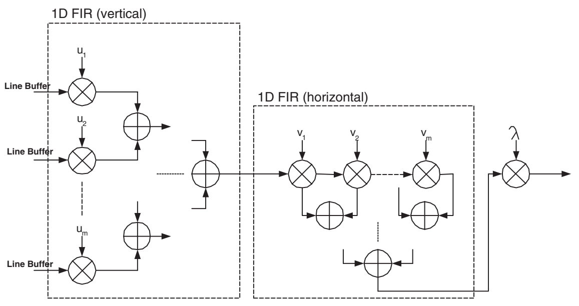

Figure 3. Detailed diagram of a decomposition stage. The necessary registers at each stage are omitted for reasons of clarity.

$$= \sum_{i=1}^{N} \lambda_i (\mathbf{I} \circledast \mathbf{u}_i) \circledast \mathbf{v}_i{}^T + \mathbf{I} \circledast \mathbf{E} \qquad (4)$$

This equation illustrates that the final result can be expressed as the addition of two different types of convolution. The first one concerns a convolution with separable filters and the second one a conventional 2D convolution. The former type of convolution is performed in the columns of the image first, followed by a convolution along the rows of the image. A diagram of the decomposition is illustrated in Figure 2, where the number of decomposition levels is $N = 2$. Figure 3 illustrates the inner structure of each separable decomposition stage. The line buffers are implemented using the embedded RAM blocks. The available embedded multipliers are used for the realization of the multiplications with the eigenvalues λ_i, and the remaining multipliers are used to realize appropriate constant coefficient multiplications in the separable filters. Finally, filter $\mathbf{E}$ is implemented using only LUT-based multipliers since it contains coefficients with low complexity.

The impact of each separable mask onto the approximation of the original filter is determined by the corresponding eigenvalue. Thus, the coefficients in the first levels of decomposition, which correspond to masks with large eigenvalues, should be approximated more accurately than the coefficients that correspond to the masks of the final levels. The algorithm achieves this by allocating the available embedded multipliers in the first stages of the decomposition in order to achieve high accuracy in the coefficient approximation. For these levels, an extra multiplier for the eigenvalue multiplication (see (2)) is not required since it can be accommodated in the corresponding coefficients. Moreover, the algorithm allocates an embedded multiplier for the final addition of the partial results that correspond to the different levels of decomposition. This is performed by multiplying each partial result with the eigenvalue that corresponds to the particular decomposition level. The rest of the coefficients are mapped to multipliers realized using slices, using the appropriate precision level for each one. The non-separable component $\mathbf{E}$ corresponds to the decomposition stages that are not taken into account and is given by:

$$\mathbf{E} = \sum_{i=N+1}^{L} \mathbf{A}_i \qquad (5)$$

More specifically, for a decomposition of an $m \times m$ 2D filter into N decomposition levels, the algorithm implements the first $K = \lfloor \frac{M-N}{2m} \rfloor$ decomposition levels using embedded multipliers, where M is the number of available embedded multipliers and N is the number of decomposition levels. The remaining $N - K$ stages are implemented using a combination of embedded multipliers and slices. The non-separable filter component, $\mathbf{E}$, is always implemented using only slices since it consists of low complexity coefficients. Thus, the final decomposition of the filter can be written as (6).

$$\mathbf{F} = \sum_{i=1}^{K} \mathbf{A}_i + \sum_{i=K+1}^{N} \mathbf{A}_i + \mathbf{E} \qquad (6)$$

For example, consider a filter $\mathbf{F}$ with size 23×23, and assume that 100 available multipliers and $N = 3$. The algorithm determines $K = 2$ and decomposes $\mathbf{F}$ as follows. It implements the masks $\mathbf{A}_1$ and $\mathbf{A}_2$ using only embedded multipliers. As these are separable masks they need only 2×23 multipliers per level of decomposition for a total of 92. These are implemented using the embedded multipliers, while mask $\mathbf{A}_3$ is implemented using a combination of the remaining 8 multipliers and slices. Finally, the non-separable component $\mathbf{E}$ is implemented using only slices.

In the above top level description of the algorithm, the effect of the coefficient quantization process to the final approximation of the filter is not considered. Below, a detailed description of the proposed algorithm is given that addresses this effect. The algorithm can be divided into three stages. The multiplier allocation stage, the decomposition stage and the refinement stage. An overview of the algorithm is given in Figure 4. Each stage is described in detail below.

3.1 Multiplier allocation stage

In this stage, the algorithm determines the number of levels of decomposition K that can be implemented using only embedded multipliers. The algorithm decomposes the input filter $\mathbf{F}$ using the SVD algorithm into a set of separable masks as:

$$\mathbf{F} = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \tag{7}$$

The algorithm reserves the appropriate embedded multipliers for the multiplications by λ_i and implements the first K masks using the remaining embedded multipliers. This is desirable since the coefficients that correspond to the initial decomposition levels have larger impact to the overall filter approximation than the coefficients that correspond to later decomposition levels. Also, the number of coefficients for word-length optimization is reduced leading to faster execution time of the algorithm. In the quantization of the coefficients, the whole available precision that is provided by the embedded multipliers of the available device can be used.

In order to take into account the error inserted due to quantization, the algorithm updates the input filter $\mathbf{F}$ after each level of the decomposition as:

$$\mathbf{F} \leftarrow \mathbf{F} - \mathbf{u}_q {\mathbf{v}_q}^T \tag{8}$$

where $\mathbf{u}_q$ and $\mathbf{v}_q$ correspond to the quantized vector of coefficients $\sqrt{\lambda_i}\mathbf{u_i}$ and $\sqrt{\lambda_i}\mathbf{v_i}$ respectively. The SVD decomposition is repeated to the updated filter $\mathbf{F}$ until the first K masks are estimated. In this way, the decomposition of each filter stage is optimized, given the quantization of the coefficients for the previous levels. Thus, any error that has been injected due to the quantization process, is addressed through the rest of the decomposition levels. The algorithm then proceeds to the decomposition stage.

3.2 Decomposition stage

In the decomposition stage the algorithm further decomposes the remaining input filter into a set of separable masks and a non-separable mask using the SVD algorithm. The rest of the available embedded multipliers are assigned to some of the coefficients, as described below, while the remaining coefficients are represented using only one non-zero signed digit, allowing the associated multiplications to be implemented cost free in bit parallel hardware.

The algorithm first allocates the available embedded multipliers to the coefficients of the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ by taking into account the coefficients from all the remaining levels of the decomposition. The algorithm explores further the decomposition by taking into account the error that is inserted to the final filter approximation, by the quantization of all the coefficients and not only due to the coefficients of the current decomposition level. The actual allocation is performed only for selected coefficients of the first level of the new decomposition. This is done because the approximation of the new decomposition level determines the coefficients of the remaining levels. The rest of the coefficients for that level are quantized as before. Due to the fact that the coefficients are quantized, the eigenvalue of that level of decomposition is then re-evaluated to correct for the quantization effects. The new λ_i is calculated using the following system of linear equations [13]:

$$\begin{aligned} \mathbf{F}\mathbf{v}_i &= \lambda_i \mathbf{u}_i \\ \mathbf{F}^T\mathbf{u}_i &= \lambda_i \mathbf{v}_i \end{aligned} \tag{9}$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ have been quantized.

The filter $\mathbf{F}$ is updated as follows:

$$\mathbf{F} \leftarrow \mathbf{F} - \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \tag{10}$$

which is similar to (8) but has been adapted in order to accommodate the λ_i parameter, and the process is repeated for the remaining separable decomposition levels.

The final level, which is the non-separable mask $\mathbf{E}$, is formed by the resulting filter $\mathbf{F}$ where all the coefficients are quantized as before. This mask is actually the error term of the initial input filter $\mathbf{F}$ and its approximation using a set of separable masks and fixed-point arithmetic.

3.3 Refinement stage

In the refinement stage, the algorithm assigns extra bits to the coefficients that have the largest error due to the quantization process, in order to minimize the error of the approximation. The quantization is performed using canonic signed digit representation [6]. The addition of extra bits is constrained by the number of available slices.

In the case where the algorithm selects to update a coefficient that belongs to the stage J of the decomposition, a new eigenvalue λ_J is estimated according to (9) using the values already estimated for the rest of the coefficients, and the whole algorithm is repeated starting from the *decomposition stage* for the next level of decomposition. This is required because the slice allocation of the later stages of decomposition for minimizing the error in the approximation depends on the coefficients of the previous levels. The algorithm terminates when the number of used slices exceeds the number of available slices. The final approximation to $\mathbf{F}$ is given by:

$$\mathbf{F} \approx \sum_{i=1}^{N} \widehat{\mathbf{A}}_i + \widehat{\mathbf{E}} \tag{11}$$

where $\widehat{\mathbf{A}}_i$ and $\widehat{\mathbf{E}}$ are the quantized $\mathbf{A}_i$ and $\mathbf{E}$ respectively.

4 Cost model

The presented algorithm explores the cost of a 2D filter design in the three dimensional space of embedded RAMs, embedded multipliers, and slices, while simultaneously minimizing the error in the approximation of the original filter. The cost model for the embedded multipliers and embedded RAMs is straightforward. For the number of slices for the required multiplications and the adder-trees that are required by the design, an upper bound estimate is derived from the number of non-zero bits in the canonic signed digit encoding. This provides a fast and a reliable estimate of the required number of slices. Moreover, in the cost model of the adder trees, the worst case scenario has been taken into account keeping all the available precision from the results of the multipliers.

5 Properties of the algorithm

The execution time of the proposed algorithm depends on the separability properties of the filter, the size of the 2D filter, and also on the number and type of the available resources. The number of available embedded multipliers reduces the number of the coefficients under optimization. The main property of the algorithm that could lead to potentially long execution times is its ability to backtrack to a previous decomposition stage and discard any optimization that was performed in the latter stages. This is a necessity, if the optimum area usage for a given approximation error is required, since the error in the approximation can be propagated to the remaining decomposition levels and minimized. In the case where the filter is highly separable, only a few decomposition levels are needed to be explored leading to short execution times. In the worst case, the execution time in our experiments was up to two hours for a 21×21 filter and $N = 3$ on a Pentium 4 at 3.2GHz.

Due to its iterative nature, when the algorithm backtracks to a previous decomposition level discarding any optimization of the remaining stages, the error in the approximation jumps to a higher value before it is further minimized. For this reason, the algorithm keeps track of the best decomposition during its execution presenting it at the end to the user. If all the initial decomposition levels have been approximated without any error, these are not altered again by the algorithm. Thus, the algorithm will always converge to

```
Algorithm: Optimized $m \times m$ 2D filter design for $N$ levels
of decomposition, using $M$ embedded multipliers and $S$
slices
Set $\mathbf{F_{original}} \leftarrow \mathbf{F}$
Calculate levels with only multipliers $K = \lfloor \frac{M-N}{2m} \rfloor$
Multiplier allocation stage
    FOR $j = 1 : K$
        Using SVD estimate: $\mathbf{F} = \sum_i \lambda_i u_i v_i^T$
        Quantize $u_q = \sqrt{\lambda_1} u_1$ and $v_q^T = \sqrt{\lambda_1} v_1^T$
        Set $\widehat{\mathbf{A}}_j \leftarrow u_q v_q^T$
        Update $\mathbf{F}$ as: $\mathbf{F} \leftarrow \mathbf{F} - \widehat{\mathbf{A}}_j$
    END
Set $d_s = K + 1$
Decomposition stage
    FOR $j = d_s : N$
        Using SVD estimate: $\mathbf{F} = \sum_i \lambda_i u_i v_i^T$
        Determine coeff. to allocate embedded muls
        Quantize the coeff. of $u_q \leftarrow u_1$ and $v_q \leftarrow v_1$
        Estimate and quantize new $\lambda_q \leftarrow \lambda_1$ using (9)
        Set $\widehat{\mathbf{A}}_j \leftarrow \lambda_q u_q v_q^T$
        Update $\mathbf{F}$ as: $\mathbf{F} \leftarrow \mathbf{F} - \widehat{\mathbf{A}}_j$
    END
    Quantize the coefficients of $\mathbf{F}$ and set $\widehat{\mathbf{E}} \leftarrow \mathbf{F}$
IF slice constraints $S$ are violated THEN EXIT
Refinement stage
    Find the coefficient $c$ that is not allocated to an
    embedded multiplier and has the largest
    approximation error
    Assign extra bit for its representation
    Let $c$ belong to the $J^{th}$ level of the decomposition
    IF $J == N$ THEN update $\widehat{\mathbf{E}}$
    ELSE update $\widehat{\mathbf{A}}_J$
IF slice constraints $S$ are violated THEN EXIT
IF $J == N$ THEN
    GOTO Refinement stage
ELSE
    Estimate $\mathbf{F} \leftarrow \mathbf{F_{original}} - \sum_{i=1}^{J} \widehat{\mathbf{A}}_i$
    Set $d_s = J + 1$
    GOTO Decomposition stage
```

Figure 4. Outline of the algorithm

a solution and terminate. The number of required iterations depends on the number of decomposition stages N, the size of the mask m and the number of available slices S.

6 Performance Evaluation

For the evaluation of the proposed algorithm, we focus on the ability of the algorithm to find a design that fits in a given area of the reconfigurable device. Such devices are manufactured by regular repetition of a silicon tile, thus the relation between the number of slices, the number of embedded RAMs and embedded multipliers gives an indication for the relative number of resources that are found in the device in an area of a certain size. The device that is used for the evaluation of the algorithm is the XC2V8000 high-end FPGA from Xilinx. It contains 168 embedded 18×18 multipliers, 168 embedded RAMs and 46,592 slices. Since we are interested in finding a design that fits in a given area of the device, the $\max$ operator is used to map the cost from the three dimensional space onto one dimension, i.e.

$$\text{Total cost} = \max\left(\frac{\#RAMs}{168}, \frac{\#MULs}{168}, \frac{\#slices}{46592}\right)$$

For comparison, we test the proposed algorithm against a direct implementation of the 2D filters using constant coefficient multipliers. The comparative design is also optimized by using canonic signed digit representation for the coefficients. It should be noted that in the experiments, the coefficients that are dedicated to embedded multipliers are quantized to 10 bits using two's complement encoding.

6.1 Filter approximation for generated filters

The performance of the proposed algorithm depends on the separability properties of the filter under investigation. Thus, two 2D filters of size 21×21 pixels with different separability properties are considered in order to assess the performance of the algorithm. The two masks are illustrated in Figures 5 and 6. Figure 7 illustrates the separability properties of the two filters by plotting their first five eigenvalues as a function of levels of decomposition. According to the figure, MaskA is less separable than MaskB. Figure 8 shows the total cost as a percentage of the area of the device, using the proposed method and the direct implementation using canonic signed representation, versus the minimum square error that can be achieved for the filter approximation. It can be concluded that the proposed algorithm reduces the total cost by between 34% and 70% (mean 55%). The proposed algorithm has less effect on MaskA than on MaskB since the former is less separable. However, an improvement between 34% and 55% (mean 47%) is still obtained. Also, for the MaskB curve, clear jumps can be seen in the

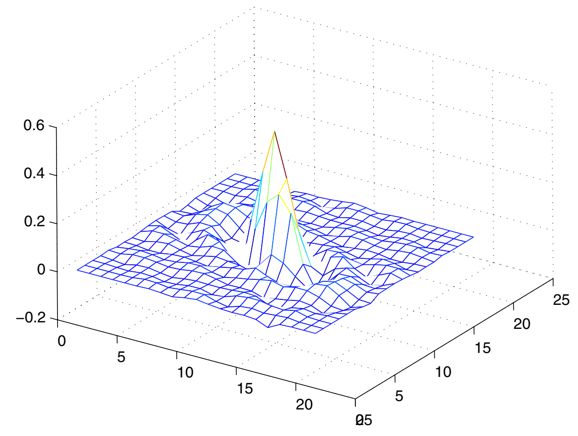

Figure 5. Impulse response of MaskA

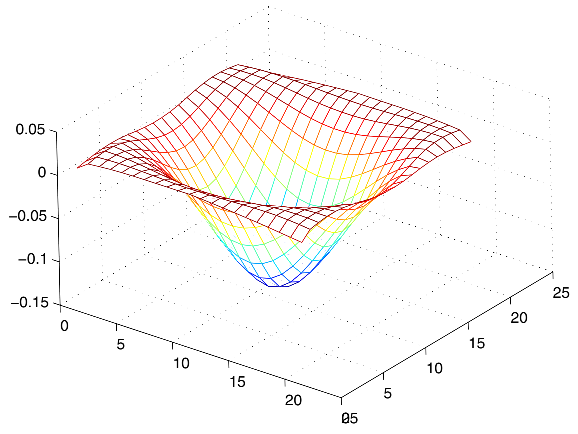

Figure 6. Impulse response of MaskB

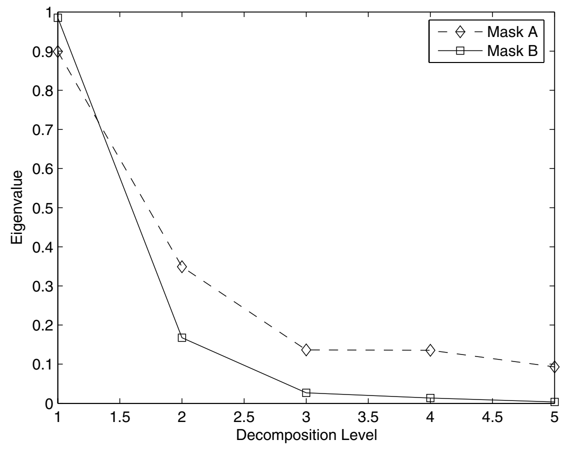

Figure 7. Plot of the eigenvalues of the masks A and B

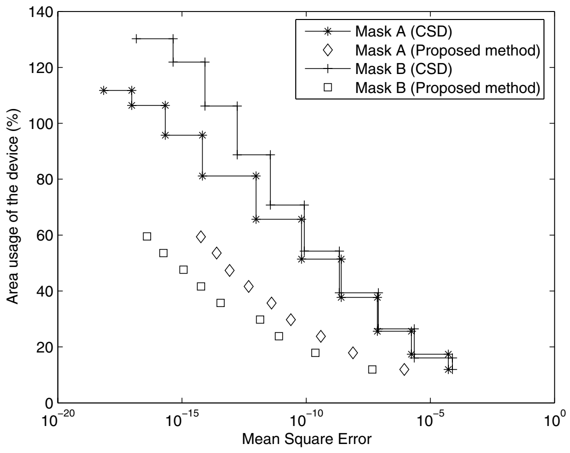

Figure 8. Area usage of the design versus achieved mean square error for a 21×21 mask

mean square error wherever the algorithm has inserted an extra decomposition level. This is because its eigenvalues are reduced substantially for up to the third level of the decomposition, whereas in the case of MaskA they do not decrease as much after the first level.

6.2 Filter approximation for a real application

The proposed algorithm was also applied to optimize filters that are used in a real application [1]. The authors use a set of 15×15 bandpass filters to decompose an image into scale and orientation selective channels for scene analysis. The size of the proposed filters and the limited resources of the available target device lead the authors to truncate the filters to smaller size and to process each frame three times in order to calculate all the filter responses. Figure 9 illustrates one filter of the set. Applying the proposed algorithm to this filter results in a reduction of between 43% and 76% (mean 59%) in the required area. We also mapped the same filter under the constraint that only two multipliers are available. This forces the algorithm to use only slices except for the eigenvalue scaling. Figure 10 illustrates the percentage of the device that is required for the implementation of the filter versus the mean square error of the approximation, under the two different scenarios. From the graph it can be concluded that the extra multipliers that are used by the algorithm influence the results in the case where a low mean square error has to be achieved. Moreover, both proposed designs out-perform the canonic signed digit methodology that is currently used. The average gain in the design when only two multipliers are used is 55%, which is very close to the case where the algorithm uses all the available resources (59%). This demonstrates the ability of the algorithm to efficiently implement a 2D filter convolution in the absence of available multipliers by allocating appropriately the available slices.

6.3 Word-length optimization performance

An experiment is performed to assess the performance of the algorithm regarding the optimum allocation of the available slices of the device. The idea is to investigate the ability of the algorithm to reconstruct a filter design given the filter impulse response and implementation cost. This provides a measure of the algorithm's ability to converge to a known feasible design. A set of 200 2D filters with size 7×7 are randomly generated having the following properties:

- The filters are designed using the SVD algorithm
- The first two levels are kept separable, where the rest form a 2D filter ($\mathbf{E}$).
- The eigenvalues of the two separable levels and the coefficients are quantized using canonic signed digit

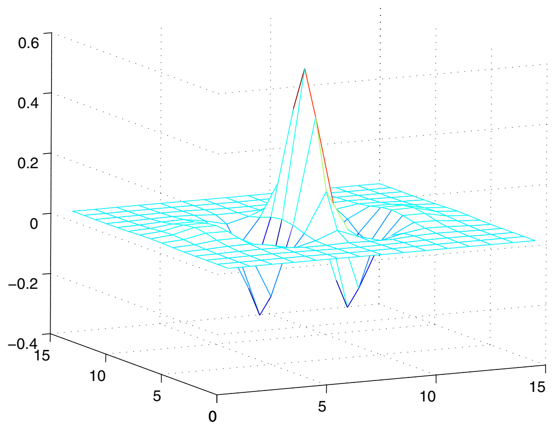

Figure 9. Impulse response of MaskC

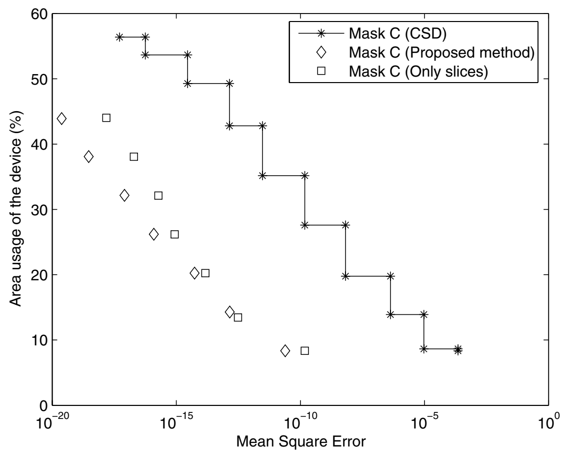

Figure 10. Area usage of the design versus achieved mean square error for a 15×15 mask

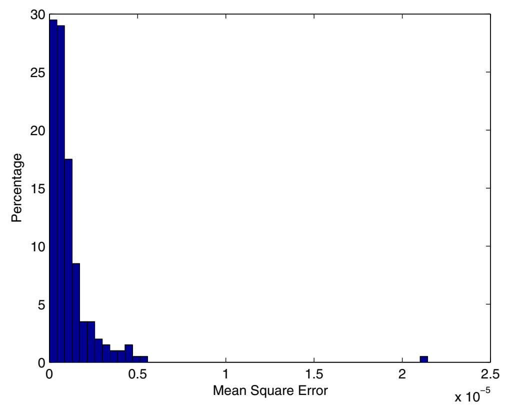

Figure 11. Histogram of the mean square error in the approximation of 200 separable 7x7 masks

recoding with randomly selected number of non-zero bits between one and three.

- The $\mathbf{E}$ part of the filter is quantized using canonic signed digit recoding with a randomly selected number of non-zero bits between one and three.

For each 2D filter, the associated cost in multipliers and slices is calculated. The impulse responses of the 2D filters are given as input to the algorithm in order to calculate an approximation of the filters having as constraints their actual implementation cost. Moreover, the algorithm is forced to decompose the filters into the same number of separable masks, i.e. two.

The above design of 2D filters allows us to explore the ability of the algorithm to allocate the available slices for the quantization of the coefficients in an efficient way regardless of the separability properties of the filter under investigation. Figure 11 shows the histogram of the mean square error of the approximation for the case of 7×7 filters. The graph shows that the majority of the filters are approximated with mean square error around $4 * 10^{-7}$. A limited number of filters are approximated with less accuracy, with maximum error in the approximation $2.1 * 10^{-5}$. It should be noted that due to the quantization process in the original design of the 2D filters, the decomposition of the filter alters. This further restricts the algorithm from approximating the original filter.

The same experiment is performed for 2D filters of size 15×15. The histogram of the mean square error in the approximation is illustrated in Figure 12. It can be concluded that the algorithm approximates the 15×15 filters better than the 7×7 filters.

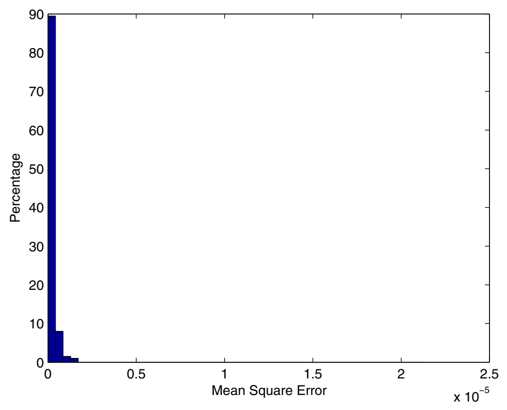

Figure 12. Histogram of the mean square error in the approximation of 200 separable 15x15 masks

It should be mentioned that in the worst case, where the separability property of the mask is poor, the algorithm will behave as well as the current techniques.

7 Conclusion

This paper presents a novel 2D filter design methodology for heterogeneous devices. The main point of departure from the current algorithms for efficient 2D filter implementation is that it explores the computational structure of the filter according to the different types of available resources in the device. Experiments with filters with size up to 21×21 and various separability properties have been performed. The results indicate a reduction in the total cost by a factor between 34% and 70% (mean 55%). Future work will involve the use of word-length optimization techniques [3] and the use of current techniques for high-speed multiplication [8] to further enhance the performance of the algorithm. Moreover, due to the mean square error criterion that is used, the proposed method is more appropriate for channel equalization applications. However, a different criterion like a min-max approximation of a filter frequency response can be accommodated by slightly modifying the proposed algorithm.

Acknowledgement

This work was funded by the UK Research Council under the Basic Technology Research Programme "Reverse Engineering Human Visual Processes" GR/R87642/02.

References

[1] C.-S. Bouganis, P. Y. K. Cheung, J. Ng, and A. A. Bharath. A Steerable Complex Wavelet Construction and its Implementation on FPGA. *Field-Programmable Logic and its applications*, September 2004.

[2] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens. A methodology and design environment for dsp asic fixed point refinement. *Design, Automation, and Test in Europe, Munich, Germany*, pages 271–276, 1999.

[3] G. A. Constantinides, P. Y. K. Cheung, and W. Luk. Wordlength Optimization for Linear Digital Signal Processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10), October 2003.

[4] S. Gong, S. McKenna, and A. Psarrou. *Dynamic Vision: From Images to Face Recognition*. Imperial College Press, 1st ed. edition, 2000.

[5] S. Haynes and P.Y.K.Cheung. Configurable multiplier blocks for embedding in FPGAs. *Electronics Letters*, 34(7):638–639, 1998.

[6] I. Koren. *Computer Arithmetic Algorithms*. New Jersey: Prentice-Hall Inc., 2nd ed edition, 2002.

[7] K.-I. Kum and W. Sung. Combined word-length optimization and high-level synthesis of digital signal processing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(8):921–930, August 2001.

[8] M. Martinez-Peiro, E. I. Boemmo, and L. Wanhammar. Design of high-speed multipliersless filters using a nonrecursive signed common subexpression algorithm. *IEEE Transactions on Circuits and Systems-II: Anslog and Digital Signal Processing*, 49(3):196–203, March 2002.

[9] G. Morris, G. A. Constantinides, and P. Y. K. Cheung. Migrating Functionality from ROMs to Embedded Multipliers. *IEEE International Symposium on Field-Programmable Custom Computing Machines*, 2004.

[10] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee. Precision and error analysis of matlab applications during automated hardware synthesis for fpgas. *Design, Automation, and Test in Europe, Munich, Germany*, pages 722–728, 2001.

[11] I.-C. Park and H.-J. Kang. Digital filter synthesis based on minimal signed digit representation. *Annual ACM IEEE Design Automation Conference*, pages 468–473, 2001.

[12] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[13] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 3rd edition edition, 1998.

[14] S. Wilton. SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays. *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, February 1998.

[15] S. Wilton, J. Rose, and Z. Vranesic. The Memory/Logic Interface in FPGA's with Large Embedded Memory Arrays. *IEEE Transactions on Very-Large Scale Integration Systems*, 7(1), March 1999.