A Novel 2D Filter Design Methodology

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 the heterogenous device. Experiments show that the proposed algorithm can achieve on average 55% reduction in the required area when compared to current techniques.**

I. 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 [1] 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 [2]. At the same time, real-time implementation is often required, making the use of hardware acceleration a necessity [3].

FPGAs are often used to achieve this goal due to their fine grain parallelism and reconfigurability. Modern FPGAs are heterogenous 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 [4] and embedded multipliers [5]. The first one provides fast distributed 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 [6] and singular value decomposition [7], 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 [8], the authors propose an approach for exchanging embedded RAMs for multipliers, whereas in [9] the authors propose 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 to 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 approximation at the same level.

II. 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 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 [10].

Another technique exploits potential separability of a 2D filter into two 1D filters by using the Singular Value Decomposition (SVD) [7] 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, at the expense of using more embedded RAMs. 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.

In this paper, we propose a novel algorithm which optimizes a 2D convolution filter implementation in a heterogenous 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.

Researchers have explored 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 [6]. These methods perform an exploration of the design space at a lower level than the proposed approach, since they do not consider altering the computational structure of the filter. The proposed technique is thus complementary to these previous approaches.

0-7803-8834-8/05/$20.00 ©2005 IEEE.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 17, 2009 at 11:21 from IEEE Xplore. Restrictions apply.

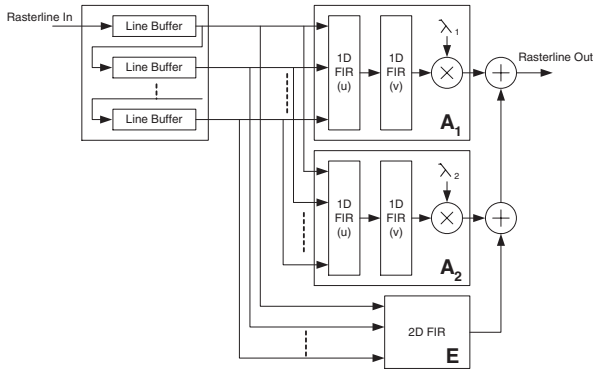

Fig. 1. Diagram of the decomposition with $N = 2$

III. Algorithm description

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 M 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 constraints. The main idea behind the algorithm is to explore the separability of the input 2D filter and decompose it to a set of separable filters ($\mathbf{A}_i$) and a non-separable one ($\mathbf{E}$):

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

A diagram of the decomposition is illustrated in Figure 1, where the number of decomposition levels is $N = 2$. The separability exploration of the filter is performed using the singular value decomposition algorithm [7] which decomposes a matrix into a linear combination of the fewest possible separable matrices. The algorithm implements the first $K = \lfloor \frac{M-N}{2m} \rfloor$ decomposition levels using embedded multipliers, while 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. Thus, the decomposition of the filter can be written as:

$$\mathbf{F} = \sum_{i=1}^{K} \mathbf{A}_i + \sum_{i=K+1}^{N} \mathbf{A}_i + \mathbf{E} \tag{2}$$

For example, let us assume a filter $\mathbf{F}$ with size 23×23, 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 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 multipliers and slices. Finally, the non-separable component $\mathbf{E}$ is implemented using only slices.

[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.

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 2. Each stage is described in detail below.

A. 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:

$$\begin{aligned} \mathbf{F} &= \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \\ &= \sum_{i=1}^{N} \lambda_i \mathbf{u}_i {\mathbf{v}_i}^T \end{aligned} \tag{3}$$

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 $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$. $\mathbf{u}_i$ and $\mathbf{v}_i$ correspond to the i^{th} columns of the $\mathbf{U}$ and $\mathbf{V}$ matrices respectively. The algorithm reserves the appropriate embedded multipliers for the multiplications by λ_i and implements the first K masks using the remaining embedded multipliers.

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

$$\mathbf{F} \leftarrow \mathbf{F} - \mathbf{u}_q {\mathbf{v}_q}^T \tag{4}$$

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. The algorithm then proceeds to the decomposition stage.

B. 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 actual allocation is performed only for selected coefficients of the first level of the new decomposition. 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

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 17, 2009 at 11:21 from IEEE Xplore. Restrictions apply.

λ is calculated using the following system of linear equations [11]:

$$\begin{aligned} \mathbf{F}\mathbf{v}_i &= \lambda_i \mathbf{u}_i \\ \mathbf{F}^T\mathbf{u}_i &= \lambda_i \mathbf{v}_i \end{aligned} \quad (5)$$

The filter $\mathbf{F}$ is updated using (4), which has been adapted in order to accommodate the λ 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.

C. 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 [10]. 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, then a new eigenvalue λ_J is estimated according to (5) 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 values of the previous levels. The algorithm terminates when one of the constraints is violated. Figure 2 summarizes the steps of the algorithm. The final approximation to $\mathbf{F}$ is given by:

$$\mathbf{F} \approx \sum_{i=1}^{N} \widehat{\mathbf{A}}_i + \widehat{\mathbf{E}} \quad (6)$$

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

Algorithm: Optimized 2D filter design for N levels of decomposition
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 (5)
    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 constraints 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 belongs to the J^{th} level of the decomposition
  IF $J == N$ THEN update $\widehat{\mathbf{E}}$
  ELSE update $\widehat{\mathbf{A}}_J$
IF constraints 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

Fig. 2. Outline of the algorithm

IV. 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, since it provides a fast but a reliable estimate of the required number of slices.

V. 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.

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

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 17, 2009 at 11:21 from IEEE Xplore. Restrictions apply.

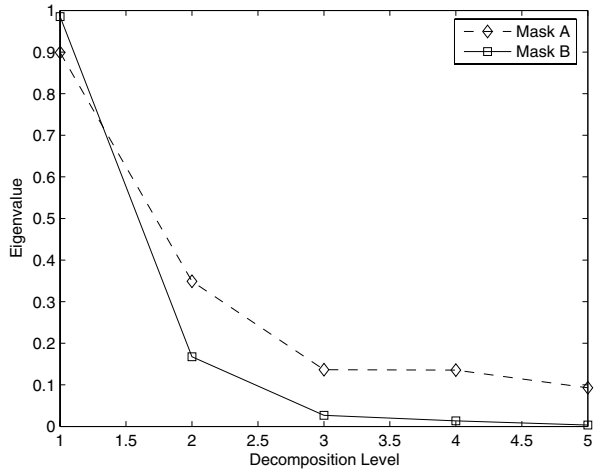

Fig. 3. Plot of the eigenvalues of the masks

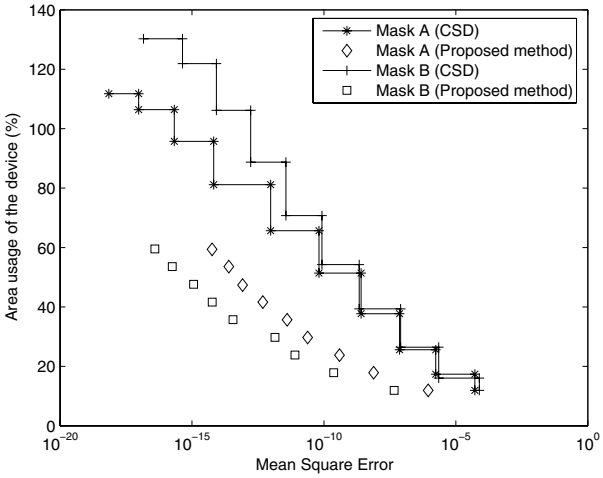

Fig. 4. Area usage of the design versus achieved mean square error

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.

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. Figure 3 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 4 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 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. Moreover, the proposed algorithm was applied to optimize the filters that are used in [3]. An improvement by a factor of 30% on average was obtained. In the worst case, where the separability property of the mask is poor, the algorithm will behave as well as the current techniques.

VI. Conclusion

This paper presents a novel 2D filter methodology for heterogenous 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 [6] to further enhance the performance of the 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] S. Gong, S. McKenna, and A. Psarrou, *Dynamic Vision: From Images to Face Recognition*, 1st ed. Imperial College Press, 2000.

[2] (2004) The xilinx website. [Online]. Available: http://direct.xilinx.com/bvdocs/appnotes/xapp241.pdf

[3] 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.

[4] 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*, vol. 7, no. 1, March 1999.

[5] S. Haynes and P.Y.K.Cheung, "Configurable multiplier blocks for embedding in FPGAs," *Electronics Letters*, vol. 34, no. 7, pp. 638–639, 1998.

[6] 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*, vol. 22, no. 10, October 2003.

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

[8] 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.

[9] 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.

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

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

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 17, 2009 at 11:21 from IEEE Xplore. Restrictions apply.