A LOW-POWER 490 MPIXELS/S HARDWARE ACCELERATOR FOR PYRAMIDAL DECOMPOSITION OF IMAGES

Vladan Popovic and Yusuf Leblebici

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Microelectronic Systems Laboratory
Lausanne, Switzerland

ABSTRACT

This paper introduces a pyramidal decomposition system suitable for high frame rate and real-time applications. The presented system's architecture omits the image transpose block used in standard separable filters, and implements internal downsampling to reduce number of computations. The decomposition is implemented in form of a field programmable gate array (FPGA) hardware accelerator and the presented results show the low resource utilization of the design. The internal downsampling reduces the power consumption by an order of magnitude compared to state-of-the-art, which makes this accelerator an excellent addition to co-processors on mobile platforms.

1. INTRODUCTION AND RELATED WORK

Pyramidal decomposition of images has been extensively used since the work of Burt and Adelson [1]. The pyramid is a data structure representing band-pass filtered images, where each level of the pyramid provides information on a different scale. Apart from the original idea of using it for image compression, the multiscale representations are also used in image fusion [2, 3, 4], image mosaicing [5], and tone mapping of high dynamic range images [6].

The advancements in areas such as machine vision, aerial surveillance and object tracking emphasized the importance of real-time image and video processing. Timing constraints in real-time processing are significantly tighter than in offline processing, and speed optimization of filters becomes necessary. This is especially important for algorithms that are highly dependent on multiresolution analysis [7, 8].

Changes in the processing platforms followed development of computationally demanding algorithms. Dedicated application-specific integrated circuits (ASIC) and field-programmable-gate-arrays (FPGA) are often used, in addition to personal computers (PC) and digital signal processors (DSP). FPGAs [9] and ASICs [10, 11] became popular tools for image processing, thanks to superior performance and shorter execution time compared to standard PCs and DSPs. The FPGA design offers more flexibility, which is demonstrated in designs of dedicated filtering blocks [8, 12] or multiprocessor implementations [13].

Two-dimensional (2-D) filters are the fundamental tool for creating image pyramids. Traditional hardware implementation of filtering in spatial domain involves efficient implementation of 2-D convolution. Direct 2-D convolution is simple for hardware implementation [8, 14, 15]. However, these designs can be inefficient depending on several circumstances.

This work has been funded by the Science and Technology Division of the Swiss Federal Competence Center Armasuisse.

First, high filter order requires simultaneous access to a large number of pixels. For a $K \times K$ filter, K pixels from K different rows should be loaded for each calculation. This is incompatible with most raw images acquired by the modern cameras, which store images in the raster scan format, *i.e.* line-by-line. Thus, a $K \times K$ filter requires at least K accesses to the storage medium, depending on the processor bus width. Furthermore, storage mediums such as external memory or hard disk add additional latency to the processing system, due to longer access time. Problem with random access is emphasized even further when images are stored in DRAM, since random memory access to DRAM can reduce data rates by an order of magnitude. Apart from the speed issues, number of multipliers in a direct 2-D filter implementation is K^2, which further increases the design area and required hardware blocks.

Separable filtering is often used to avoid the presented problems [16]. Separable filters first process rows and then columns, hence there is no need for random memory access. Furthermore, the number of multipliers is reduced to $2K$. However, use of separable filters introduces two new problems: (1) separable filtering requires an image transpose after the row filtering, and (2) large overhead is created when pyramidal decomposition is used, due to downsampling between pyramid levels.

State-of-the-art solutions [16] resolve image transposition by placing K large line buffers before the row filter, storing enough data for both row and column filters to operate. However, these implementations still produce large overhead, as the optional downsampling is performed after the filtering.

In this paper we present a hardware accelerator for low-power high-performance pyramidal decomposition, based on systolic separable filters. The designed filtering scheme lowers the computational overhead by supporting internal downsampling, hence reducing the size of line buffers. Pyramid levels are processed in a single clock domain, with implemented clock-gating. This method reduces the dynamic power consumption and simplifies the clock tree routing compared to multi-clock-domain designs [3]. Furthermore, the high-throughput of the system is achieved using a fully systolic (pipelined) architecture.

The design is demonstrated on a Laplacian pyramid (LP) example, using a $K = 5$ filter. Section 2 introduces fundamentals of LP decomposition. The proposed accelerator design and its FPGA implementation are presented in Section 3. Performance results of the implementation and comparisons of frame rate and power consumption are shown in Section 4.

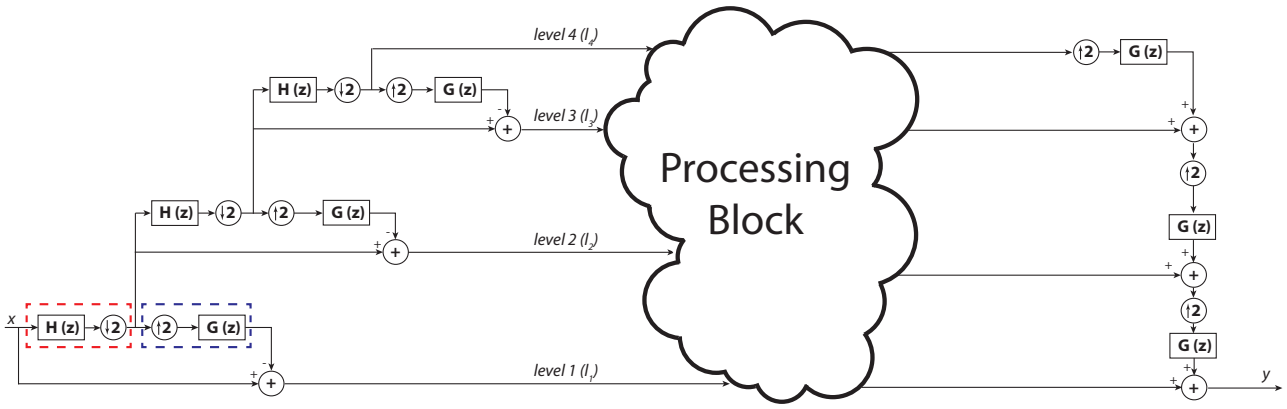

Fig. 1. Laplacian pyramid decomposition and reconstruction with four levels. The top level (*level 4*) represents the coarse approximation, whereas the bottom level represents the details. The new analysis and synthesis filters with internal downsampling are marked with red and blue dashed rectangles. They are marked only in the first level for clarity reasons. The processing block denotes operations on the decomposed pyramid, and it does not change the inter-pixel timing within LP levels. The LP reconstruction is performed on the processed pixels, on the right side of the figure.

2. LAPLACIAN PYRAMID

Laplacian pyramid is a multispectral and multiscale representation of an image, where each pyramid level contains one frequency band of the image. LP transform is illustrated in Fig. 1. Let x be the source image. The image is filtered using the analysis low-pass filter **H(z)** and downsampled by two, in both horizontal and vertical directions. The decimated image is then upsampled and filtered with the synthesis filter **G(z)**. The first level (l_1 in Fig. 1) of the LP is obtained by subtraction of the interpolated image from the source x, and it represents the high-frequency content of the image, *i.e.* the details. The decimated image is also used as the source for the second level of decomposition. An L-level LP is created using $L-1$ repetitions of the mentioned principle.

Both image quality and timing performance are dependent on the filters **H(z)** and **G(z)**. The 2-D finite impulse response (FIR) filters are often used in FPGA and ASIC designs due to their inherent stability and simplicity of design in digital systems. The implementation of 2-D FIR filters can be either separable or non-separable. The non-separable (direct) implementation consists of 2-D convolution of the filter matrix with the image. For $N \times M$ image resolution and $K \times K$ filter matrix size, the computational complexity of such filtering is $\mathcal{O}(MNK^2)$. The hardware design requires K^2 multipliers and complex input buffer structure for larger filter sizes.

Oppositely, separable filters require less multipliers and adders compared to the direct implementation. However, traditional separable implementation based on software algorithms is very resource-demanding and quite inefficient. Such computation is mathematically expressed as:

$$x' = (x * h_r)^T * h_c \tag{1}$$

where x and x' are the original and the filtered image, respectively, and h_r and h_c are row and column 1-D filters. Operation denoted with $*$ represents a 1-D convolution. Without loss of generality, in the rest of the paper we will consider symmetric 2-D filters, *i.e.* $h = h_r = h_c$.

The main issue of this implementation is the transposition block. Even though the complexity of $\mathcal{O}(MNK)$ is lower compared to direct filtering, it requires more memory, as the whole intermediate image result is buffered. The buffering is obligatory due to reordering (transposing) of the pixels before the second 1-D filter is applied, which increases the system latency by $N \times M$.

3. FPGA IMPLEMENTATION

The main goals of our design are real-time performance, reduction of required hardware in the transposition block, and low-power operation. Real-time performance is achieved by reducing the critical path delay using pipeline architecture, *i.e.* result of each arithmetic operation in the algorithm is followed by a register. Hence, the critical path is reduced to the length of the longest path in a single arithmetic block. Furthermore, the proposed design includes data sharing, which reduces number of memory read requests, increases performance and reduces hardware complexity. The low-power operation is achieved by switching reduction, hence lowering dynamic power dissipation.

3.1. Analysis Filter Architecture

The analysis filter architecture is shown in Fig. 2. The filter operates as follows. The pixels are read from the memory row-wise. The row-wise streaming was chosen due to the common raster scan output of the camera. Therefore, streaming row-by-row is a reasonable choice as it reduces the access time to memory thanks to the sequential address access. K pixels are buffered in a shift register, where K is the length of the used 1-D filter. The pixels in the register are shifted with the arrival of each new pixel. All K pixels are available at the output and they are used by 1-D row filter.

The row filter provides horizontally filtered pixels at its output. In standard separable filter implementations, these filtered pixels are stored in memory, transposed and filtered again. However, using the proposed architecture, we avoid storing and transposing the full frame. The intermediate memory is replaced by a chain of $K-1$ line buffers, which are implemented as BlockRAMs in FPGA.

Furthermore, not all filtered pixels are needed in the subsequent stages. Hence, we introduce two new blocks, named *Column control logic* and *Row control logic* in Fig. 2. Since the filtered image will be downsampled, we distribute the downsampling operation into

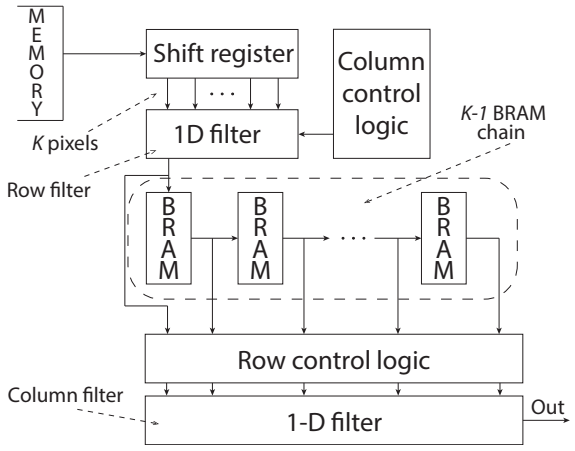

Fig. 2. Internal architecture of the 2-D separable filter. Both analysis and synthesis filters are implemented using the same architecture, with difference in control logic blocks. The difference is explained in Sections 3.1 and 3.2.

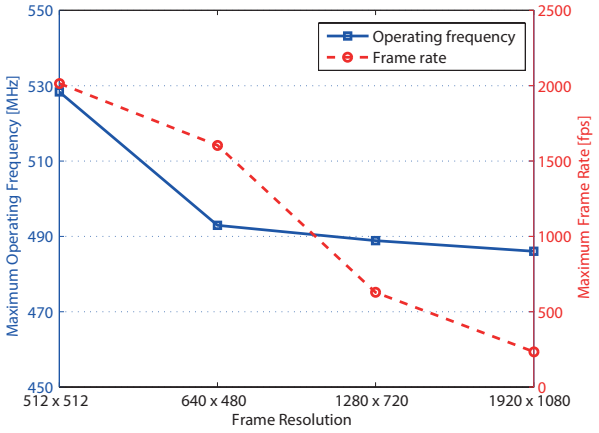

Fig. 3. System performance of the LP decomposition and reconstruction for different frame sizes. The full blue line represents operating frequencies and the dashed red line represents the achieved frame rates.

row and column procedures, and embed it in the hardware filter. After one pixel is filtered by the row filter, the *Column control logic* disables filtering for the next pixel, *i.e.* pixel positioned in the next column. After skipping one pixel, the control logic again enables the filter. This principle is repeated for all pixels in the image, and it corresponds to horizontal downsampling by two.

The pixels belonging to the same row are buffered in the same BlockRAM, and only $K-1$ half-rows are stored in this chain thanks to the control logic. Whenever a new filtered pixel arrives, it is stored in the first BlockRAM at the location addressed by the pixel's column in the frame. Since the utilized BlockRAMs behave as a dual-port memory, the second port is used for reading the pixel from the same memory location, *i.e.* the pixel in the same column from the previous row. The read pixel is then stored in the following BlockRAM in the chain. Hence, this BlockRAM chain can also be regarded as a set of stacked shift registers.

The outputs of $K-1$ BlockRAMs and the output of the row filter form a set of K vertically neighbouring pixels. Hence, the transposition is no longer required, as the pixels are available in the appropriate order. Similar to *Column control logic*, *Row control logic* block disables filtering of every second row in the column filter. It is important to note that even when column filtering is disabled, shifting of pixels between BlockRAMs is enabled. This is obligatory due to the fact that one source pixel contributes to $(K+1)/2$ filtered pixels in a single column.

The pixels allowed through the *Row control logic* are filtered using the second 1-D filter (column filter in Fig. 2) and streamed out to the rest of the processing system. The outputs are sorted in the same order as the original input, in the row-wise order.

3.2. Synthesis Filter Architecture

Opposite to the analysis filter **H(z)** that downsamples the image, the synthesis filter **G(z)** upsamples it. A property of the upsampling operation is that output data rate of the filter is higher than at the input. We implement a time-multiplex system to resolve this issue, under the safe assumption that the processing block in Fig. 1 does not increase the data rate.

The synthesis filter is implemented using the same top-level architecture as the analysis filter. The main difference is in the control logic blocks of the synthesis filter, since it multiplexes the input pixels with the upsampled zero-valued pixels. When the filter receives a pixel from i^{th} column, the *Column control logic* allows the row filter to output pixel from $i-(K-1)/2$ column. In the following clock cycle, the logic will enable the filter to output the $i-(K+1)/2$ column. The insertion is allowed because of two reasons: (1) the corresponding input pixel for the second output pixel is zero, and (2) the assumption that input data rate is not faster than the output rate of the LP decomposition. Levels l_i, for $i=\{2,...,L\}$ cannot provide pixels in each clock cycle, hence upsampling of the pyramid levels can be embedded in the filtering operation. Level l_1 is the only level that can theoretically provide pixels every cycle, but its pixels are not being filtered by **G(z)** during the reconstruction (see Fig. 1).

The line buffers store the upsampled rows. The *Row control logic* operates on the same principle as *Column control logic* with the exception that it inserts the row pixels. When column filter provides a pixel from i^{th} row, the *Row control logic* enables the column filter to output pixel from $i-(K-1)/2$ row. In the following clock cycle, a pixel from $i-(K+1)/2$ row will be calculated. Hence, reconstruction of each pyramid level provides interpolated image at four times higher frame rate, which makes the real-time LP reconstruction possible.

4. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed system was implemented on the Xilinx VC-707 development board, with XC7VX485T-2FFG1761 Virtex 7 FPGA and 1 GB of DDR3 memory operating at 800 MHz. The system is designed in Xilinx ISE 14.7 and synthesized using Xilinx XST. To identify the correct performance and resource utilization of the accelerator, the experimental results will correspond only to the presented hardware, *i.e.* excluding the DDR3 controller and μBlaze microcontroller used for setting the filter parameters.

The design was synthesized for four different frame resolutions: 512×512, 640×480 (VGA), 1280×720 (720p) and 1920×1080 (1080p). The obtained operational frequencies are shown in Fig. 3 in blue color. The frequencies vary around 490 MHz, with 486 MHz for a 1080p frame. Critical path of the system is in the control logic,

(a) Original image (b) Software result (c) Our result

Fig. 4. An example image used for image quality benchmark. The decomposed and reconstructed image using the accelerator presented in this work (c) provides image with PSNR = 65.34 dB compared to the double-precision result from Matlab (b).

and not in the computational part, thanks to the pipelined architecture. The XST confirms it by reporting the critical path in the part of control block dedicated to edge extension for correct filtering. Thus, a slight decrease in frequency observed in Fig. 3 is expected and relates to a 1-bit increase in size of the counters counting up to the maximum resolution. A significant increase of frequency for 512 $\times$ 512 frame is thanks to the frame size which is a power of two. Hence, the counters do not require a comparator at the output, and counter reset logic, since the counter cycles to zero without reset.

A change of filter order affects only the system latency. Frequency change is negligible and not noticeable in operation. The critical path remains the same, as it is not affected by the number of coefficients, because the FIR filtering is implemented as a fully pipelined block.

Fig. 3 also shows the achieved frame rate, in red color. The frame rate is 234 frames per second (fps) for the largest tested frame size (1080p), and 1604 fps for VGA frame. The achieved frame rates are suitable for any state-of-the-art real-time application, even for high frame resolutions. For comparison, the non-separable filtering implementation [8] achieves 94 fps for 1080p frame. The design in [17] reports around 45 fps for VGA frame, whereas its related GPU implementation [16] achieves 233 fps. Finally, frame rate of the multiprocessor implementation [13] is calculated to be 977 fps for VGA frame, and 145 fps for 1080p.

The resource utilization summary is given in Table 1. The system is synthesized for 1080p frame resolution and $K = 5$ filter size. The presented work uses approximately 0.1% of the resources available on the selected Virtex 7 FPGA. Table 1 shows that the resource utilization of the proposed design is lower than utilization of the direct 2-D filter implementation from [8]. The pyramidal decomposition is isolated from the full system in [8] and synthesized separately, to provide a fair comparison. Furthermore, resource utilization of our system is comparable to other FPGA implementations [4, 17], which have significantly lower frame rates. The comparison numbers are taken from the original publications. Utilization data for [17] relates only to the reconfigurable filter design, and not the full system. The mark "–" is used for the numbers which are not available.

Table 1. FPGA device utilization summary

Resource	This work	[8]	[4]	[17]
Slice LUT	2015	5891	2730	1852
Slice Register	3307	7612	2812	1759
BlockRAM	24	14	38	–
DSP	30	0	–	–

The proposed design utilizes the clock gating technique to disable filtering of unnecessary pixels. Hence, a reduction in dynamic power consumption is reported by Xilinx Power Estimator (XPE) tool. The chosen figure of merit (FOM) for comparison with the related systems is energy-per-processed-pixel. FOM of the proposed architecture is $15.05\,pJ/pixel$, compared to $546\,pJ/pixel$ of [8], $1\,nJ/pixel$ of [13], and $26\,nJ/pixel$ of [16].

Finally, Fig. 4 illustrates image results of the proposed hardware. The original image (Fig. 4(a)) is decomposed and reconstructed using the proposed system with 16-bit fixed-point precision. The image in Fig. 4(b) is obtained in Matlab R2014a and the default double-precision. Fig. 4(c) is the simulation result of our system. The filter coefficients and pixel luminance have 16-bit and 8-bit precision, respectively. The filtered image from our system has PSNR = 65.34 dB compared to the floating-point double-precision of Matlab, which shows that there is no loss compared to the original 8-bit image.

5. CONCLUSION

In this paper, we proposed a high frame rate hardware accelerator for pyramidal decomposition based on separable filtering. The accelerator is based on a pipeline architecture, without the image transposition between row and column filters. Instead, a chain of line buffers is used for partial frame storage. Furthermore, the architecture supports internal downsampling needed for pyramidal decomposition, which significantly lowers the power consumption. The system offers almost constant operating frequency, irrespective of the frame resolution and the filter size. Finally, the low resource utilization and compactness of the proposed implementation make this accelerator an excellent choice for image decomposition on the low-power mobile systems.

6. REFERENCES

[1] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532 – 540, April 1983.

[2] D.C. Zhang, Sek Chai, and G. Van der Wal, "Method of Image Fusion and Enhancement Using Mask Pyramid," in *Information Fusion (FUSION)*, July 2011, pp. 1–8.

[3] O. Sims and J. Irvine, "An FPGA Implementation of Pattern-Selective Pyramidal Image Fusion," in *International Conference on Field Programmable Logic and Applications (FPL)*, August 2006, pp. 1–4.

[4] Y. Song, K. Gao, G. Ni, and R. Lu, "Implementation of real-time Laplacian pyramid image fusion processing based on FPGA," *Proceedings of SPIE*, vol. 6833, 2007.

[5] M. Brown and D. Lowe, "Automatic Panoramic Image Stitching Using Invariant Features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, August 2007.

[6] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic Tone Reproduction for Digital Images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 267–276, July 2002.

[7] C. Zhang, C. Wang, and M. O. Ahmad, "A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform," *IEEE Trans. Circuits Syst. I*, vol. 59, no. 8, pp. 1775–1785, 2012.

[8] V. Popovic, K. Seyid, A Schmid, and Y. Leblebici, "Real-time Hardware Implementation of Multi-resolution Image Blending," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 2741–2745.

[9] S. Kestur, D. Dantara, and V. Narayanan, "SHARC: A Streaming Model for FPGA Accelerators and its Application to Saliency," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2011.

[10] G.S. van der Wal, "Technical Overview of the Sarnoff Acadia II Vision Processor," in *SPIE Defense, Security, and Sensing Conference, Subconference 7710: Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, April 2010.

[11] Y-C. Tseng, P-H. Hsu, and T-S. Chang, "A 124 Mpixels/s VLSI Design for Histogram-Based Joint Bilateral Filtering," *IEEE Transactions on Image Processing*, vol. 20, no. 11, pp. 3231–3241, Nov 2011.

[12] C. Bouganis, G.A. Constantinides, and P.Y.K. Cheung, "A Novel 2D Filter Design Methodology For Heterogeneous Devices," in *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2005, pp. 13–22.

[13] D. T-H. Tsuei, M-Y. Dabbagh, and M. Sachdev, "Multiprocessor FPGA Implementation of a 2D Digital Filter," in *24th Canadian Conference on Electrical and Computer Engineering*, 2011, pp. 630–633.

[14] L.-D. Van, "A New 2-D Systolic Digital Filter Architecture Without Global Broadcast," *IEEE Trans. VLSI Syst.*, vol. 10, no. 4, pp. 477–486, 2002.

[15] I-H. Khoo, H. C. Reddy, L.-D. Van, and C.-T. Lin, "2-D Digital Filter Architectures without Global Broadcast and Some Symmetry Applications," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2009, pp. 952–955.

[16] D. Llamocca, C. Carranza, and M. Pattichis, "Separable FIR Filtering in FPGA and GPU Implementations: Energy, Performance, and Accuracy Considerations," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2011, pp. 363–368.

[17] D. Llamocca and M. Pattichis, "Real-time Dynamically Reconfigurable 2-D Filterbanks," in *IEEE Southwest Symposium on Image Analysis Interpretation*, 2010, pp. 181–184.