Real Time Image Rotation and Resizing, Algorithms and Implementations

Robert D. Turney and Chris H. Dick
CORE SOLUTIONS GROUP, XILINX, INC.
2100 LOGIC DRIVE SAN JOSE, CA 95124-3450

ABSTRACT

Recent growth in the area of digital communications has been fueled by new and exciting communications algorithms to satisfy the ever increasing bandwidth needs. While the majority of the bandwidth requirements are driven by multi-channel voice or data needs, video and imaging communications is becoming both realizable and practical. Additional progress in the area of video and imaging compression has reduced the bandwidth requirements for video communications thus enabling video conferencing, video broadcast, and video wireless applications. With the continued growth in the video communication industry different size and orientation of image data sets is a noteworthy problem to address. In this work we will explore implementation of image rotation and resizing for both low and high quality imaging systems. Implementations are proposed for real time processing using field programmable gate arrays (FPGAs).

1. Introduction

When addressing the problem of image resizing one needs to satisfy the two dimensional sampling theorem [1]. Aliasing and "imaging" found in one dimensional signal processing are extended to two dimensional processing with spatial distribution rather than time dependency. In a majority of one dimensional multi-rate systems integer decimation or interpolation resampling are often the case. Image resizing typically involves fractional resampling and can lead to prohibitively large implementations resulting in comprimises in range and resolution. Additional requirements due the the human visual system (HVS), such as constant aspect ratio, edge sensitivity, and noise uniformity add constraints to the image resize algorithm. In addition, frequently one must also deal with interlaced video data as an input to the imaging system. In this work we will discuss traditional bilinear and bicubic interpolation methods from an algorithm and real time implementation perspective.

The problem of image rotation is becoming more prevalent due to the emergence of flat panel and CCD acquisition devices. While commercial applications for rotation may be of a limited nature, medical, military and industrial uses for image rotation abound. Similar to image resizing, interpolation algorithms must be utilized to achieve image quality acceptable for the particular application. Since, in general, image rotation involves a rectangular shaped image, image cropping and padding needs to be incorporated into the algorithm. One of the more challenging problems in image resizing and rotation is dealing with edge preservation in the presence of noise with a real time implementation. In this work we deal with the computational aspects of implementing image resize and rotation by efficient co-design of algorithmic aspects and implementation aspects so that memory bandwidth and frame latency considerations are included.

2. 1-D Resampling

Resampling by decimation of digital data requires one to pay attention to the Nyquist rate to eliminate aliasing. For example if a 512 pixel line is decimated by 4 to produce 128 pixels a quarter-band filter would be employed to eliminate the high frequecy from folding into the low frequecy terms. This structure is usually implemented as a polyphase decimator shown in Figure 1. The primary advantage of this structure is the filters run at ¼ the input sample rate.

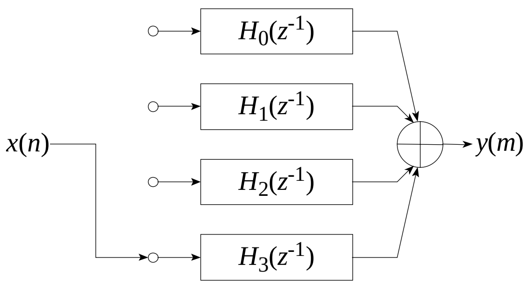

Figure 1 Polyphase Decimator

Up-sampling requires interpolation of digital data and requires one to pay attention to "images" of the original spectra. For example, if we were interpolating by 3 a 256 pixel line to 768 pixels one would utilize a filter to reject the "image" spectra. This structure is usually implemented as a polyphase interpolator shown in Figure 2. Note that the three filters in this structure run at the low input sample rate in contrast to the high sample rate of the output stream.

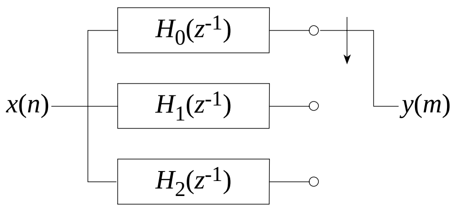

Figure 2 Polyphase Interpolator

To perform up-sampling at various interpolation rates a fractional filter needs to be designed. Consider the case of 512 input samples with desired up sample by 4 (2048) and resolution of 16 pixels. A polyphase fractional upsampling architecture is presented in Figure 3 that allows P/Q resampling [5]. In this structure P polyphase filters are used with a Q sequencer to determine filter utilization. In the present example Q is 512/16 or 32. The number of filter banks (P) is defined by 2048/16 (128). During the actual operation of the P/Q resampling the number of P filters included in the sequencing of Q is determined by the up-sampling rate. The design of the actual filters is application dependent. For image processing linear and cubic have been widely accepted in order to keep the filter lengths to a minimum. Implementation of this polyphase up-sampling filter is usually carried out with a single filter with a coefficient bank.

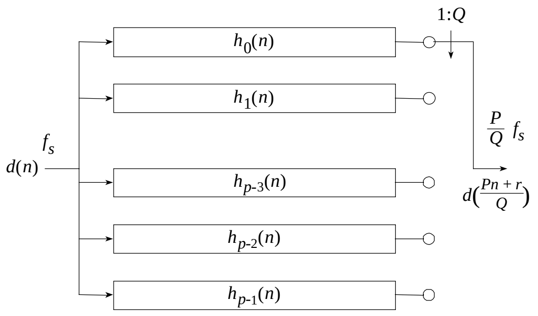

Figure 3 Polyphase Resampling

To design a system which can both down sample by 4 and upsample by 2 with resultion of 16 pixels we can utilize the polyphase resampling structure. In this case the sequencer would require a skip condition where a sample has been input but no sample is extracted from the polyphase structure. It can be shown that the overall filter response during decimation is providing the necessary aliasing protection. For this resampling case Q is 32, P is 64, and the range of filters included in the computation is 8 to 64 to support 128 to 1024 pixel sizes.

3. 2-D Image Resizing

Extending the resampling to two dimensions for the purpose of imaging or video applications involves considerations for the filter design and the impact on the Human Visual System (HVS). Preservation of edges and noise characteristics play a major role in this area of research. In this paper we will not address this issue but rather explore current algorithm and architecture tradeoffs. We have considered the design parameters (range and resolution) as they relate to the one dimensional polyphase resampling. Extending our current example to two dimensions consider a 512x512 image which we would like to resize from 128x128 to 1024x1024. Depending on the real time constraints of the imaging application and human interaction with the imaging system two dimensional architectures can be formed with the polyphase resampler. The driving imaging parameters are usually specified as frame rate and frame latency.

The natural way to extend to two dimensions is to incorporate an intermediate buffer between row and column operations as shown in Figure 4. This structure has considerable memory overhead since the intermediate buffer would have to be double buffered to allow for continuous operation. The imaging system must also allow for a frame latency in the resizing structure. Complicated systems in the medical imaging area generally have a number of frame latencies and thus any processing you can perform without using a frame latency is desirable. To remove the frame latency and excessive memory requirements Figure 5 shows an architecture that performs vertical resampling cascaded with horizontal resampling. This structure inputs a number of rows simultaneously and performs the vertical filtering with the Q sequencer only changing on a row basis. Depending on your vertical down sample rate throw away out lines are possible just as in the one dimensional case. Similarly repeat lines on the input are needed when up-sampling.

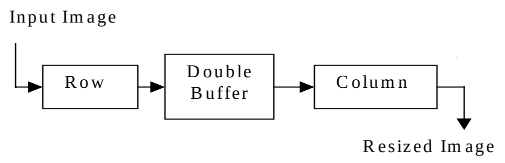

Figure 4 Resize Architecture

We would ideally like to have an implementation with a raster interface and provide the x and y resize parameters on a frame basis. Accomplishing these goals requires integration of line buffer memory and pushes the limits of present silicon technology.

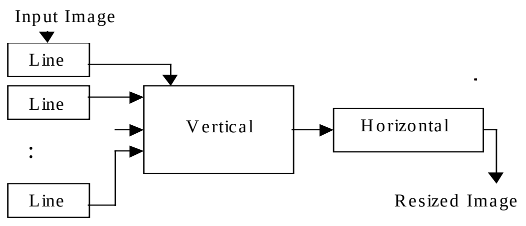

Figure 5 Zero Frame Latency Resizing

4. 2-D Image Rotation

Image rotation in the digital domain is a form of resampling but is performed on non-integer points. A typical system specification might include 1024x1024x12 30 frames/s with 40 MHz pixel clock parameters. With the center point defined, a single parameter θ specifies the transformation. The derivation utilizes the general form of the Hotelling transform with basis vectors cos(θ) and sin(θ). Equation (1) gives the coordinate transformation in terms of rotation of the coordinate axis.

$$\begin{aligned} S_x &= D_x\cos(\theta) + D_y\sin(\theta) \\ S_y &= -D_x\sin(\theta) + D_y\cos(\theta) \end{aligned} \qquad (1)$$

where S and D represent source and destination coordinates. Separable rotation algorithms and architectures require intermediate memory and will not be included in this discussion.

Given the requirement of linear addressing through the destination image the first step in the rotation algorithm is computation of the source values S_x and S_y. From these values the neighborhood of pixels are known for the filter operation. The location of the destination pixel in the source pixel matrix also gives the weighting factors for bilinear or bicubic interpolation to be performed. The pixel value is then calculated with the weighting factors and pixel values in the neighborhood. The algorithm repeats by incrementing the D_x value and continuing in a raster out format. One drawback of this process is the non uniform addressing of the source pixels. Essentially the input memory design must have four times the bandwidth (sixteen for bicubic) because there is no sharing of source pixels between destination pixel operations.

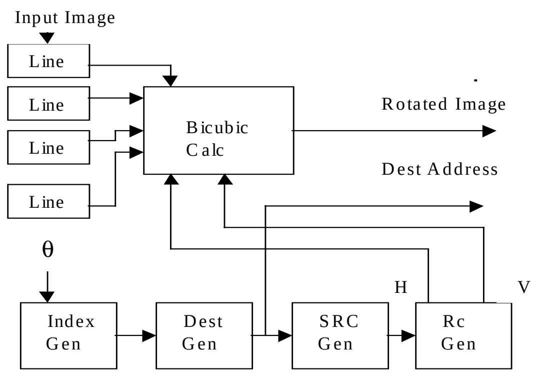

Figure 6 Rotation Architecture

To address the short commings of source address generation consider raster scan through the source image and construct the rotated image. By inverting (1) we get destination coordinates which can be truncated to find integer D_x and D_y values. These values are reiterated into (1) and used to calculate weighting factors for bilinear or bicubic interpolation in the neighborhood of source pixels available. Rastering through the input image with this procedure fills in the resultant rotated image. A block diagram of the architecture is shown in Figure 6.

5. FPGA Implementation

Real time implementation of image resizing and rotation systems involves evaluating FPGA vendor silicon functionality and software support including predefined parameterized cores available for use. Most of the devices are basically organized as an array of logic elements and programmable routing resources used to provide the connectivity between the logic elements, FPGA I/O pins and other resources such as on-chip memory. Device features such as block memory and delay locked loop (DLL) technology are also significant factors that influence the complexity and performance of the implementation.

The architecture of the Xilinx 4th generation FPGA Virtex^{TM} [7] device is shown in Figure 7.

The objective of the FPGA image processing designer is to efficiently map a imaging system to the FPGA hardware, using the logic array, block memory and DLL in the most efficient and economic way possible. This may include both algorithm and architecture analysis to achieve the optimal implementation. The most important advance in this technology with respect to the image processing community is the Virtex E technology which has double the number of block rams available compared to Virtex technology. These block rams can be configured as 1024x4, 512x8, or 256x16 and can also be combined for line buffering applications. The number of block rams available in the largest VirtexE device at present is 208.

The logic fabric can be employed to implement various types of arithmetic functional units, for example high-speed (150 MHz) adders, subtracters, multipliers and dividers to name a few. One design option for realization single and multi-rate filters in FPGAs is to simply schedule a MAC unit to perform the inner-product calculation. In the case of a single rate filter data management and the coefficient addressing is simple. For polyphase decimators and interpolators it is still possible to employ a single MAC data path in conjunction with a slightly more sophisticated control unit to sequence the filter coefficient set in the correct manner. Performance can easily be purchased by employing a parallel data path based on multi-MAC processing units. The filter coefficients can be stored in block RAM/ROM or in the logic array itself. The same is true of the sample history buffer.

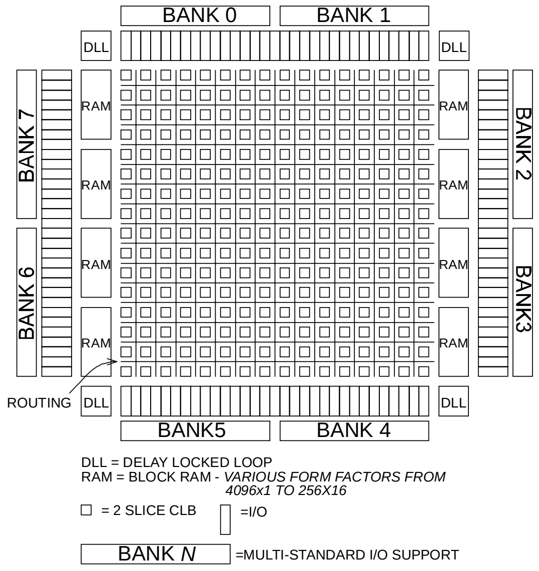

Figure 7 Xilinx Virtex FGPA Architecture

Alternative algorithms are also available to construct the filtering engine. For example, distributed arithmetic (DA) [9] has been demonstrated to be an extremely useful re-organization of the inner-product equation for FPGA systems. It exploits the look-up table and distributed memory [7] features of the FPGA. Filters and computational units are available from FPGA Vendors (Xilinx CoreGen™) as part of the software tool suite for implementation.

For the resizing architecture Cores are available for vertical and horizontal functions of Figure 5. This algorithm can be implemented in a single chip device. Depending on the image quality required for the application Xilinx Virtex E part types can range from a XCV300 to XCV1000. Similarly analysis of the rotation architecture of Figure 6 yields Xilinx Virtex E part types ranging from XCV600 to XCV1000.

6. Conclusion

In this paper we have discussed the algorithms, architectures, and implementations for image rotation and resizing. Single chip FPGA implementations have been proposed which enable real time processing. Technology advances in the area of embedded memory on FGPAs is particularly attractive to the video and image processing community. Applications in the video communications and medical imaging area can benefit from the real time processing capabilities. Future work in this area will be directed toward image quality issues such as edge preservation and noise characteristics.

References

[1] Bracewell, R., *Two-Dimensional Imaging*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Inc., 1995.

[2] Pratt, W., *Digital Image Processing,* John Wiley & Sons, Inc., 1991.

[3] Gonzalez, R., Wintz, P., *Digital Image Processing 2nd Ed., Addison-Wesle, 1987.*

[4] Gonzalez, R., Woods, R., *Digital Image Processing, Addison-Wesley, 1992.*

[5] Dick, C., harris, f., "FPGA Interpolators Using Polynomial Filters", *The 8th International Conference on Signal Processing Applications and Technology*, September 1998.

[6] Rasche, V., et al. "Resampling of Data Between Arbitrary Grids Using Convolution Interpolation", *IEEE Transactions on Medical Imaging, Vol. 18, NO. 5, May1999.*

[7] Xilinx Inc., *The Programmable Logic Data Book*, 1999.

[8] Carey, W., Chuang, D., Hemami, S., "Regular-Preserving Image Interpolation", *IEEE Transactions on Image Processing, Vol. 8, NO. 9, Setpember 1999.*

[9] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing", *IEEE ASSP Magazine*, Vol. 6(3), pp. 4-19, July 1989.