Indian J.Sci.Res. 17(2): 198-203, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

FPGA IMPLEMENTATION OF AN IMAGE COMPRESSION USING VERILOG

[1]Ch.Swathi, [2]B.Indira Priyadarshini ,[3]S.Upender
[1]Electronics and Communication Engineering, NRI Institute of Technology,Vijayawada
[2]Electronics and Communication Engineering, Matrusri Engineering College, Hyderabad.
[2]Electronics and Communication Engineering, Vidya Jyothi Institute of Technology, Hyderabad.

Abstract -Recently, Field Programmable Gate Array (FPGA) technology has become a viable target for the implementation of algorithms suited to video image processing applications. The unique architecture of the FPGA has allowed the technology to be used in many such applications encompassing all aspects of video image processing. The goal of this paper is to develop FPGA realizations of three such algorithms on two FPGA architectures.As image sizes and bit depths grow larger, software has become less useful in the video processing realm. Real-time systems such as those that are the target of this project are required for the high speeds needed in processing video. In addition, a common problem is dealing with the large amount of data captured using satellites and ground-based detection systems. DSP systems are being employed to selectively reduce the amount of data to process, ensuring that only relevant data is passed on to a human analyst. Eventually, it is expected that most video processing can and will take place in DSP systems, with little human interaction. This is obviously advantageous, since human data analysts are expensive and perhaps not entirely accurate.

Keywords -Image, VHDL, DSPSYSTEM,FPGA, Algorithm.

I. Introduction

There are several different choices a designer has when implementing a DSP system of any sort.Hardware, of course, offers much greater speed than a software implementation, but one must consider the increase in development time inherent in creating a hardware design. Most software designers are familiar with C, but in order to develop a hardware system, one must either learn a hardware design language such as VHDL or Verilog, or use a software-to-hardware conversion scheme, such as Streams-C , which converts C code to VHDL, or MATCH, which converts MATLAB code to VHDL. While the goals of such conversion schemes are admirable, they are currently in development and surely not suited to high speed applications such as video processing. Ptolemy is a system that allows modeling, design, and simulation of embedded systems. Ptolemy provides software synthesis from models.) .While this type of system may be a dominant design platform in the future, it is still under much development, meaning that it may not be a viable design choice for some time. A discussion on the various viable options for DSP system design is found below.

II. Literature review

Signal processing programs used on a PC allow for rapid development of algorithms, as well as equally rapid debug and test capabilities. It is common for many hardware designers to use some sort of PC programming environment to implement a design to verify functionality prior to a lengthy hardware design. MATLAB [6] is such an environment. Although it was created for manipulating matrices in general, it is well suited to some image processing applications. MATLAB treats an image as a matrix, allowing a designer to develop optimized matrix operations implementing an algorithm. However, if the eventual goal is a hardware device, the algorithms are instead often written to operate similarly to the proposed hardware system, which results in an even slower algorithm. Systems such as IDL andits graphical component ENVI are more specifically geared to image processing applications, and include many pre-written algorithms commonly used to process images. However, even specialized image processing programs running on PCs cannot adequately process largeamounts of high-resolution streaming data, since PC processors are made to be for general use. Further optimization must take place on a hardware device.

Application Specific Integrated Circuits (ASICs) represent a technology in which engineers createa fixed hardware design using a variety of tools. Once a design has been programmed onto an ASIC, it cannot be changed. Since these chips represent true, custom hardware, highly optimized, parallelalgorithms are possible. However, except in high-volume commercial applications, ASICs are oftenconsidered too costly for many designs. In addition, if an error exists in the hardware design and is notdiscovered before product shipment, it cannot be corrected without a very costly product recall.

III. Design Approach

Prior to any hardware design, the author chose to create software versions of the algorithms inMATLAB. Using MATLAB procedural routines to operate on images represented as matrix data, thesesoftware algorithms were designed to resemble the hardware algorithms as closely as possible. While ahardware system and a matrix-manipulating software program are fundamentally different, they can produce identical results, provided that

[1]Corresponding Author

care is taken in development. This approach was taken because it speeds understanding of the algorithm design. In addition, this approach facilitates comparison of the software and synthesized hardware algorithm outputs, allowing detailed error calculations. This project was targeted for FPGA systems for two reasons. One, the author had some previous experience in FPGA implementations of video processing algorithms . Two, FPGAs represent a VHDL design environment and the FPGA-specific tools. In the first state, a design is created in VHDL. Next, the code's syntax is verified and the design is synthesized, or compiled, into a library. The design is next simulated to check its functionality. Stimulating the signals in the design and viewing the outputwaveforms in the VHDL simulator allows the designer to determine proper functionality of the design. Next, the design is processed with vendor-specific place-and-route tools and mapped onto a specific FPGA in software. This allows the engineer to view a floorplan and hierarchical view of the design, which can help verifying a proper mapping procedure. Next, the design is verified for proper functionality once again. This step is important because it assures that the design is correct in its translation from Verilog to gatelevel.

If this is found to be correct, the design can then be programmed onto the specified FPGA. For this project, the author had access to two FPGAs, each from a different company and each with different design tools: the Altera FLEX 10K100 and the Xilinx Virtex XCV300.: patterns and classes are characterized by formal structures. In these methods, the basic units are defined as Primitive. All models are expressed in terms of the inter relationships between Primitive Grammar. In most cases, these methods are applied to certain structural pattern.

ANN:Artificial neural networks, each pattern are described in terms of several characteristics. Point's attributes are considered in a multidimensional space. Feature space is divided into several regions corresponding to each class. Pre specified classes in supervised classification by training data determine the boundaries of different classes, there confirm the marked areas. Template feature vector obtained through measurement or observation.

Due to architecture differences, the Altera FLEX 10K series is termed a Programmable Logic Device (PLD) and is not officially considered to be an FPGA. However for the purpose of simplicity it is commonly referred to as an FPGA, and will be so named in this document. The FLEX 10K100 is a CMOS SRAM-based device, consisting of an embedded array for memory and certain logic functions and a logic array for general logic implementation. The embedded array is constructed of Embedded Array Blocks (EABs). The EABs can be used to implement limited memories such as First In First Out (FIFO) or RAM units. The FLEX 10K100 has 12 EABs, each with 2048 bits for use in a design.The logic array in the FLEX 10K series is built from Logic Array Blocks (LABs). Each LAB consists of 8 Logic Elements (LEs), each of which is constructed of a 4-input Look Up Table (LUT) and a flip-flop. Each LAB can be considered to represent 96 logic gates. The FLEX 10K100 has 624 LABs, accounting for most of its 100,000 gates (the rest are accounted for in memory). Figure 2 shows the basic units in a FLEX 10K LE. Input/output functionality on the FLEX 10K series is handled in the Input/output Blocks (IOBs).

Each IOB has one flip -flop to register either input or output data. However for bi-directional signals, this is an inefficient design, since two flip -flops are needed and only one is available in the IO B. The second flip-flop must be implemented in the logic array, resulting in an overall slower design .a floor plan view of the Altera FLEX 10K architecture, highlighting the elements discussed.

IV. Methodology

This project was focused on developing hardware implementations of three popular image processing algorithms for use in an FPGA -based video processing system. This chapter discusses thesealgorithms and their software implementations in MATLAB.In image processing, several algorithms belong to a category called windowing operators.

Windowing operators use a window (shown in Fig 1), or neighborhood of pixels, to calculate their output. For example,windowing operator may perform an operation like finding the average of all pixels in the neighborhood ofa pixel. The pixel around which the window is found is called the *origin*.

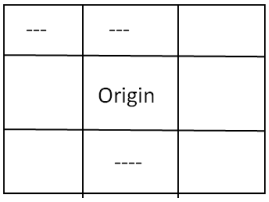

Fig:1 Pixel window and origin

The work for this project is based on the usage of image processing algorithms using these pixelwindows to calculate their output. Although a pixel window may be of any size and shape, a square 3x3size was chosen for this application because it is large enough to work properly and small enough toimplement efficiently on hardware. This filter works by analyzing a neighborhood of pixels around an origin pixel, for every valid pixel in an image. Often, a 3x3 area, or window, of pixels is used to calculate its output. For every pixel inan image, the window of neighboring pixels is found. Then the pixel values are

sorted in ascending, or rank, order. Next, the pixel in the output image corresponding to the origin pixel in the input image is replaced with the value specified by the filter order. The rank order filter can be represented by the following lines of pseudo-code:

Order = 5 (this can be any number from 1 -> # pixels in the window)

for loop x –> number of rows

for loop y –> number of columns

window_vector = vector consisting of current window pixels

sorted_list = sort (window_vector)

output_image(x,y) = sorted_list(order)

end

end.

Fig 2shows an example of this algorithm for a median filter (order 5), a filter that is quite useful in salt -and-pepper noise filtering . Since the rank order filter uses no arithmetic, a mathematical

description is difficult to represent efficiently.

Low 10

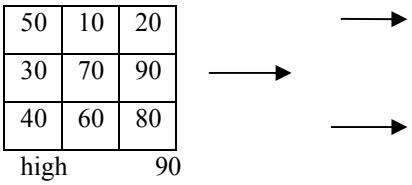

Fig: 2Graphic depiction of rank order of filter

As is evident in the above fig 2, it is possible to use any or der up to the number of pixels in the window. Therefore a rank order filter using a 3x3 window has 9 possible orders and a rank order filter using a 5x5 window has 25 possible orders. No matter what the window size used in a particular rank order filter, using the middle value in the sorted list will always result in a median filter. Similarly, using the maximum and minimum values in the sorted list always results in the flat dilation and erosion of the image, respectively. These two operations are considered part of the morphological operations,.

V. MATLAB Implementation

The PC software program MATLAB was used to develop an initial version of the rank order filter,so that its operation could be verified and its results could be compared to the hardware version. WhileMATLAB offers features that speed up operations on matrices like images, custom operations were used so that the software would closely mimic the functionality of the proposed hardware implementation n. The MATLAB implementation of the rank order filter is called ro_filt.m. It works by using *for* loops to simulate a moving window of pixel neighborhoods. For every movement of the window, the algorithm creates a list of the pixel values in ascending order. From this list, the algorithm picks a specific pixel. The pixel that is chosen from the list is specified in the order input. The output of the program is an image consisting of the output pixels of the algorithm. Since a full 3x3neighborhood is used in this implementation, the window must have gotten to the second line of the input image in order to create an output. The result of this is that some ‘edge effects’ occur in the output image, meaning that there is always an invalid strip along the borders of the output image. This is true for all algorithms using the windowing approach to image processing. Figure shows some example output images for a given input image using ro_filt.m. From this figure it is easy to observe the effect that the rank order filter has on an image, given the various algorithm orders use.

Fig:3Input image

Fig : 4 Filtered image order-2

Fig : 5 Filtered image order-5

The focus of this project is the actual implementation of the proposed algorithms on target FPGA hardware. As discussed in previous chapters, this was accomplished by composing the algorithms in the Verilog HDL language and synthesizing the algorithms for the FPGAs. This chapter discusses the hardware design specifics for each algorithm. The rank order filter was the first algorithm to use the window_3x3 pixel window generator. Since its operation is fairly simple, it was an ideal choice. As discussed above, the rank order filter must first sort the pixel values in a window in ascending (or rank) order. The most efficient method accomplishing this is with a system of hardware compare/sort units, which allow for sorting a window of nine pixels into an ordered list for use in the rank order filter.This system results in a sorted list after a latency of 14 clock cycles. Since the design is pipelined, after the initial latency the system produces a valid sorted list on every clock cycle. The Verilog HDL algorithm which implements this design, sort_3x3.v, is really just a series of registers and compares, as is shown if Fig6. Not all levels of the sorting algorithm are shown to conserve space.

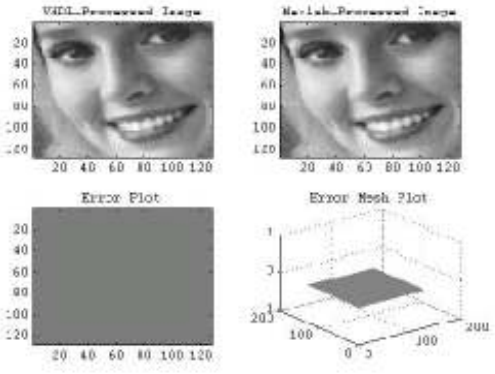

Fig : 6 VHDL and MATLAB Comparison Plots for ro_filt_3x3 with Order = 4

The VHDL rank order filter design has been synthesized for both the Altera and Xilinx architectures. Since the Xilinx Virtex is a newer generation FPGA, it was expected that it would provide superior performance over the Altera FLEX 10K F PGA. This surmise was true, and was a constant throughout the design.

VI. Convolution

The design of the convolution algorithm in VHDL was a much more difficult problem than the rank order filter design. This was due to its use of more complex mathematics. For example, the rankorder filter really just sorts the pixels in a window and outputs one of them, while the convolutionalgorithm uses adders, multipliers, and dividers to calculate its output. On FPGAs, use of mathematicstends to slow down performance. Many designers favor techniques that reduce the algorithm's dependency on complex mathematics. Still, since the mathematics used in convolution are simple, implementation of a convolution algorithm was an achievable goal.Yet another obstacle in this algorithm's design was implementing the capability to handle negative numbers. In a proper convolution algorithm, the mask can (and often does) consist of negative numbers. Effectively, the Verilog HDL had to be written to handle these numbers by using *signed* data types. Signed data simply means t hat a negative number is interpreted into the 2's complement of its non -negative dual. This means that all vectors within the design must use an extra bit as compared to unsigned numbers. The extra bit always carries the sign of the number – 0 for a positive number, 1 for a negative number.

Because of this, the output of the convolution algorithm is a number in 2's complement. In orderfor another unit to interface data from this algorithm, the unit must be able to understand or convert 2's complement data. Fortunately, this is a simple matter in the ACS system, Addition and multiplication were instantiated using simple + and * signs in the VHDL code. The VHDL synthesis tool provides mapping to efficient hardware mathematics designs for each of these, sodevice-specific parameterized modules were not necessary. Since a proper convolution involves a division by the number of pixels in the window, some thought had to be put into this part of the algorithm's hardware implementation. Hardware dividers on FPGAs are quite large and slow. In addition they must be tied directly to the FPGA's architecture, meaning that one divider would not work for both architectures pursued. It was deemed necessary to instead use the bit shifting method of division. Since this is only possible with powers of two, a divide by 8 was implemented instead of a divide by 9, as was planned in the algorithm's design.Optimization of the convolution algorithm can be easily achieved if one has limited kernelspecifications. For example, if all coefficients in the kernel are powers of two, the VHDL synthesizer is able to result in a design that uses fewer resources. This is due, of course, to the way numbers are represented in digital systems, where a number that is a power of two is represented with only one bit. Further optimization is possible by reducing the bit widths of the kernel constants. This is result in a smaller coefficient data range, but this compromise may be acceptable in certain cases.

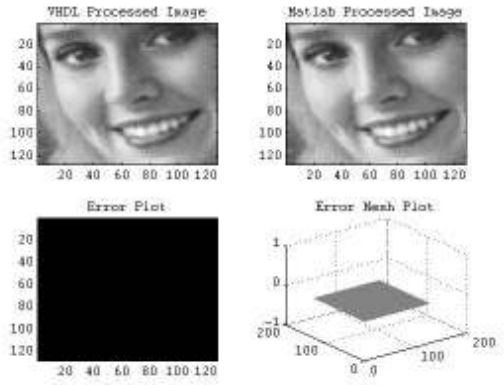

Fig : 7 Verilog HDL and MATLAB Comparison Plots for conv_3x3 with K1 Kernel

VII. Conclusions

The development of FPGA image processing algorithms can at times be quite tedious, but theresults speak for themselves. If high-speed, windowing algorithms are desired, this paper shows that FPGA technology is ideally suited to the task. In fact, with the aid of the window generator, a whole series of image processing techniques is available to the designer, many of which can be synthesized for high–speed applications.

One of the drawbacks of the techniques presented in the paper is the large size of the algorithms,used in the design. If off -chip RAM is used for FIFO operations, the designs' synthesized size can be greatly reduced.Also, the stack filter method of image processing can greatly reduce the size of algorithms using a window generator. Still, this method achieves a more serial method of processing, which is not entirely efficient with FPGA systems. The design presented here is quite capable, and it tries to take advantage of the parallelism possible with FPGA devices.

A great deal of knowledge was gained from the completion of this project. While FPGAs areexcellent for some uses, such as a large number of image processing applications, difficulties in using more complex mathematics speak volumes towards the argument of using dedicated DSP chips for some applications. Indeed, it is expected that a designer who desires the best combination of speed and flexibility should look toward a system consisting of both FPGAs and DSPs. Such a system can take advantage of the positive aspects of each architecture, and can allow the designer to create an algorithm on a system that is best suited for it. That said, it should also be noted that this project's algorithms were excellent choices for FPGA implementation. This is because they don't use floating -point mathematics and they include no complex mathematics.

VIII. Future Work

The interchangeable nature of the VerilogHDL components of this design allow for its components to be used in different designs quite easily. For example, the window_3x3 architecture allows it to be used in any algorithm that uses a pixel window to compute its output. Since VerilogHDL components can easily be instantiated in any design, using the pixel window generator is as simple as dropping component and portmap statements into another VerilogHDL design. Because of this, the applications for the code created for this project can be used in many different image processing algorithms. With the window generator and row/column counter code complete, about fifty percent of the work is done and the designer simply has to use t heir outputs to generate a desired result. It could be said that the real result of this project is not simply a few algorithms, but instead a system of VerilogHDL code which allows for efficient implementations of many algorithms. Still, these VerilogHDL designs should be made to operate more generically, so that modification of hard-coded values is not necessary. A large part of the improvement possible in this design lies in the algorithms themselves. For the rank order filter, changing the order to be an input vector would allow on-the-fly switching of algorithm properties. While this does increase the synthesized size of the design, it also maximizes its on–chip capability. Similarly, if the kernel for the convolution design were to be changed to inputs instead of constants in a package, the convolution algorithm would also have increased functionality.

References

[1] Chou, C., Mohanakrishnan, S., Evans, J.: "FPGA Implementation of Digital Filters," Proc. ICSPAT,1993.

[2] Benedetti, A., Perona, P.: "Real-time 2-D Feature Detection on a Reconfigurable Computer,"Proceedings of the 1998 IEEE Conference on Computer Vision and Pattern Recognition, 1998.

[3] Gokhale, M., et. al.: "Stream -Oriented FPGA Computing in the Streams -C High Level Language,"unpublished paper, 2000.

[4] Banerjee, N., et. al.: "MATCH: A MATLAB Compiler for Configurable Computing Systems,"Technical Report, Center for Parallel and Distributed Computing, Northwestern University, August1999.

[5] Lee, E., et. al.: "Overview of the Ptolemy Project," Department of Electrical Engineering andComputer Science, University of California, Berkeley, July 1999.

[6] Mathworks, Inc.: "MATLAB 5.3 Fact Sheet," Natick, MA, 1999.

[7] Research Systems, Inc.: "Getting Started with IDL," Boulder, CO, September 1999.

[8] Research Systems, Inc.: "ENVI User's Guide," Boulder, CO, July 1999.

[9] Texas Instruments, Inc.: "TMS320C4X User's Guide," Houston, TX, May 1999.

[10] Moore, M.: "A DSP-Based Real-Time Image Processing System," Proceeding of the 6 th InternationalConference on Signal Processing Applications and Technology, Boston, MA, August 1995.

[11] Texas Instruments, Inc.: "C67x Floating -Point Benchmarks," Houston, TX, 2000.

[12] Virtual Computer Corporation: "What is Reconfigurable Computing," Reseda, CA, 2000.

[13] Nelson, A.: "An Implementation of t he Optical Flow Algorithm on FPGA Hardware," IndependentStudy Paper, December 1998.

[14] Nelson, A.: "Further Study of Image Processing Techniques on FPGA Hardware," Independent StudyPaper, May 1999.

[15] Altera, Inc.: "Altera FLEX 10K Embedded Program mable Logic Family Data Sheet," San Jose, CA,1999.

[16] Xilinx, Inc.: " XilinxVirtex 2.5V Field Programmable Gate Array Specification," San Jose, CA, 2000.

[17] Andraka Consulting Group, Inc,: "Digital Signal Processing for FPGAs," Seminar Notes, 1999.

[18] Russ, J.: "The Image Processing Handbook," CRC Press, Boca Raton, FL, 1992.

[19] Hussain, Z.: "Digital Image Processing – Practical Applications of Parallel Processing Techniques,"Ellis Horwood, West Sussex, UK, 1991.