Techniques for FPGA Implementation of Video Compression Systems

Brian Schoner, John Villasenor, Steve Molloy and Rajeev Jain, Department of Electrical Engineering, University of California, Los Angeles, California

Abstract

Real-time video compression is a challenging subject for FPGA implementation because it typically has a large computational complexity and requires high data throughput. Previous implementations have used parallel banks of FPGAs or DSPs [1,2,3] to meet these requirements. Using design techniques that maximize FPGA utilization, we have implemented two video compression systems, each of which uses a single FPGA. In the first system, algorithmic optimizations are made to create a low-complexity implementation that exploits the in-system programmability of the FPGA. This low-complexity implementation performs well, but is limited to a single compression algorithm. In the second system, the FPGA is augmented with an external, low-complexity, video signal processor (VSP [4,5,6].) This combination of ASIC and FPGA is flexible enough to implement four common compression algorithms, and powerful enough to execute them in real time.

1.0 Introduction

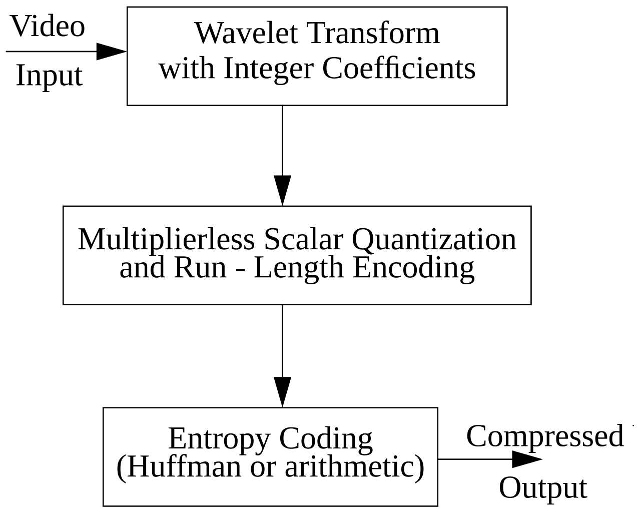
Video compression is used in many applications to reduce the amount of information required to represent a sequence of images. There is a tremendous variety of applications for video compression ranging from high-definition television, with a compressed data rate of several Mbits/sec., to low-power wireless video transmission at several tens of kbits/sec. Many compression algorithms are available depending on application requirements such as the amount of compression required, the required resiliency to errors, and the type of video source. This wide range of compression techniques makes programmable implementations especially attractive.

Video processing typically requires high data throughput and computational complexity. For example, the discrete cosine transform (DCT) [7] is the basis of many compression systems. An efficient algorithm for computing the DCT on 8 x 8 blocks of a 15 frames / sec. video sequence with 256 x256 byte frames requires a 24MHz multiplier and a 55MHz adder. Since the DCT is usually followed by other algorithms such as run-length encoding (RLE) and Huffman coding, the DCT may require multiplications faster than 50MHz. These data rates are beyond those achievable by most DSP chips, and are challenging for FPGAs.

Parallel banks of FPGAs and DSPs have been used to prototype some video processing routines successfully [1,2,3], but implementation on a single FPGA might lead to a more cost-effective system. In this paper, we describe two implementations utilizing a single FPGA. In the first system, the video compression algorithm is designed for a low-complexity implementation using a single in-system reprogrammable FPGA. Optimizing the algorithm to fit the system results in an efficient implementation, but the system is limited to the single algorithm. In a second implementation, the FPGA is augmented with an external processor. While the first system demonstrates techniques to reduce the algorithm complexity, the second system describes techniques to increase the computational power of the system.

2.0 FPGA Implementation Using Low-Complexity Video Algorithms

We have identified low-complexity video compression algorithms based upon wavelet transforms for use in low-power wireless communications [8]. A search of wavelet transform filters has identified filters that have integer coefficients and can be implemented with shift-and-add operations rather than multiplications [9]. An adaptive scalar quatization algorithm has also been developed for implementation without multipliers. In some of the steps of the video coding algorithm, we choose approaches that are slightly suboptimal for compression performance, but which allow a large (at least a factor of two) reduction in complexity over a theoretically optimal implementation. For example, the requirements of integer coefficient filters for the wavelet transform and the use of scalar quantization instead of more efficient (and complex) vector quantization results in lower image quality at a given bit rate. Comparisons with other compression algorithms [8] show the low complexity system performing within 1dB PSNR of high complexity systems. A block diagram of the low complexity algorithm we implemented is shown in figure 1.


Video
Input
Wavelet Transform
with Integer Coefficients
Multiplierless Scalar Quantization
and Run - Length Encoding
Entropy Coding
(Huffman or arithmetic)
Compressed
Output

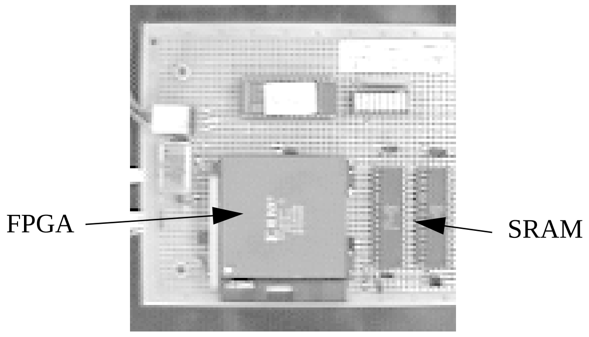


Figure 1: Low-Complexity Video Compression Algorithm and the Current Video System

2.1 Commonality and Complexity

The complete compression algorithm requires 15000 gates if placed on a single FPGA. But, because the compression routines are run sequentially, it is possible to sequentially reprogram a smaller FPGA in real-time with each individual routine. Assuming the smaller FPGA can be reprogrammed in 1ms [10], the three reprogramming periods are only a 10% overhead at 30 frames / sec.

Commonality is exploited wherever possible to reduce the amount of hardware. For example, the high- and low-pass filters are merged into a single, combined filter. The wavelet transform, adaptive quantization, and arithmetic coding designs each require addressing logic to read and write data by subbands. An FPGA that allows partial reconfigurability could exploit this commonality by only configuring the portions of the FPGA that differ.

The techniques to reduce algorithm complexity can (and will) be used to develop low-complexity ASICs. By comparing implementations on ASICs and reconfigurable FPGAs, we hope to identify areas in which FPGAs can and cannot compete with ASICs.

2.2 Architecture and Results

The current design contains the hardware required for the wavelet transform, a simplified quantizer, and a run-length encoder. This design runs at 20 frames per second with a frame size of 256 x 256 x 8 bits and consumes less than 1 watt of power. It has been used very successfully as a prototype within a UCLA wireless computing demonstrator. Figure 2 shows an original image, and the same image after being compressed at a ratio of 15:1 and decompressed with the single FPGA video compression system.

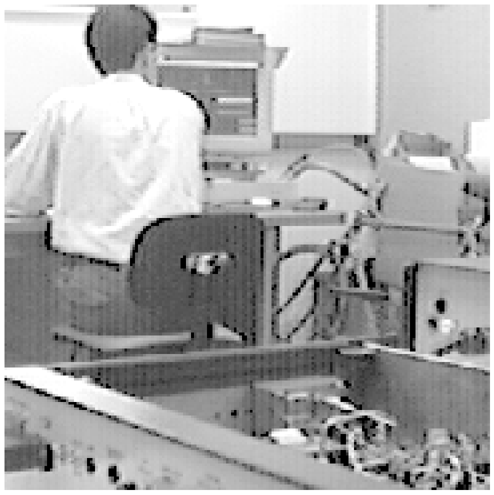


Figure 2a: Original Image

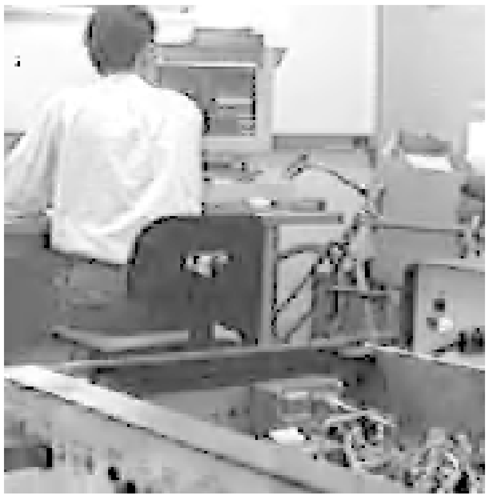


Figure 2b: Image after compression and decompression

Currently, reconfiguration is only used to switch between the compression and decompression circuits. Each of these designs fits on a Xilinx XC4008 using only automatic place and route utilities. By the end of '95, we hope to have the adaptive quantizer and arithmetic coder modules also working. We are also working to implement a version on the National Semiconductor CLAy FPGA [10] architecture which supports partial reconfiguration.

3.0 FPGA With External Processor (VSP)

The single FPGA implementation of a low-complexity algorithm works, but it is not a general solution to the implementation problems of complex video compression algorithms. To overcome the high computational complexity and high data throughput of video compression algorithms, we have identified the hardware that is common, and moved it to an external ASIC called the video signal processor (VSP [4,5,6].). The VSP consists of a parallel multiply and accumulate data path and small local memories. The combination of a reprogrammable FPGA with the VSP ASIC has both the flexibility to implement four common video compression algorithms and the computational power to execute them in real time.

3.1 Algorithms

The system is designed to implement the discrete cosine transform (DCT,) two dimensional filtering (2DFIR,) vector quantization (VQ,) and the wavelet transform. Each of the computations for the four algorithms can be computed from the general equation:

$$Z_k = W_k - \sum_i \left(\sum_j XY \right) \qquad (1)$$

For example, the DCT can be written in matrix form as:

$$Y = AXA^T \qquad (2)$$

Where A is the N x N basis matrix defined as:

$$a(n, m) = \sqrt{\frac{2}{N}} a(n) \frac{\cos(2m+1)\pi n}{2N} \qquad (3)$$

n = (0...N-1), m = (0...N-1), a(n) = sqrt(1/2) for n = 0 and 1 otherwise.

To implement the DCT in the form of equation 1, X represents a matrix row element, Y represents a matrix column element, and W is set to zero. Similarly, the algorithms for 2DFIR, VQ, and the wavelet transform can be performed in a manner consistent with equation 1. The VSP is designed to perform the parallel multiply and accumulate operation that is central to equation 1 and common to all four algorithms. The tasks of updating coefficients, organizing the data, and performing any additional processing is done by the FPGA.

3.2 Architecture and Results

The division of operations between the FPGA and VSP is shown in figure 3. Operations that require flexibility fit naturally onto the reconfigurable FPGA. Placing the operations of addressing and pixel processing on the FPGA also allow for easy adaptation to different frame sizes and pixel formats (i.e. 2's complement and grey-scale.) Since the algorithm state machine is on the FPGA, it is also easy to make algorithm modifications (i.e. change the depth of the VQ search, or the size of the filter mask.) The parallel multiply, accumulate, and memory operations performed by the VSP are common to all algorithms, and are efficiently implemented as an ASIC.

Implementing both the operations of the VSP and FPGA in a single FPGA would have required approximately 20k gates. Alternatively, there are comparable commercial video signal processors with no FPGA, but requiring > 900k transistors [11,12]. The combination of the 80k transistor VSP and 3000 gate FPGA seems to be the most efficient method to implement a processor that requires both flexible and static operations.

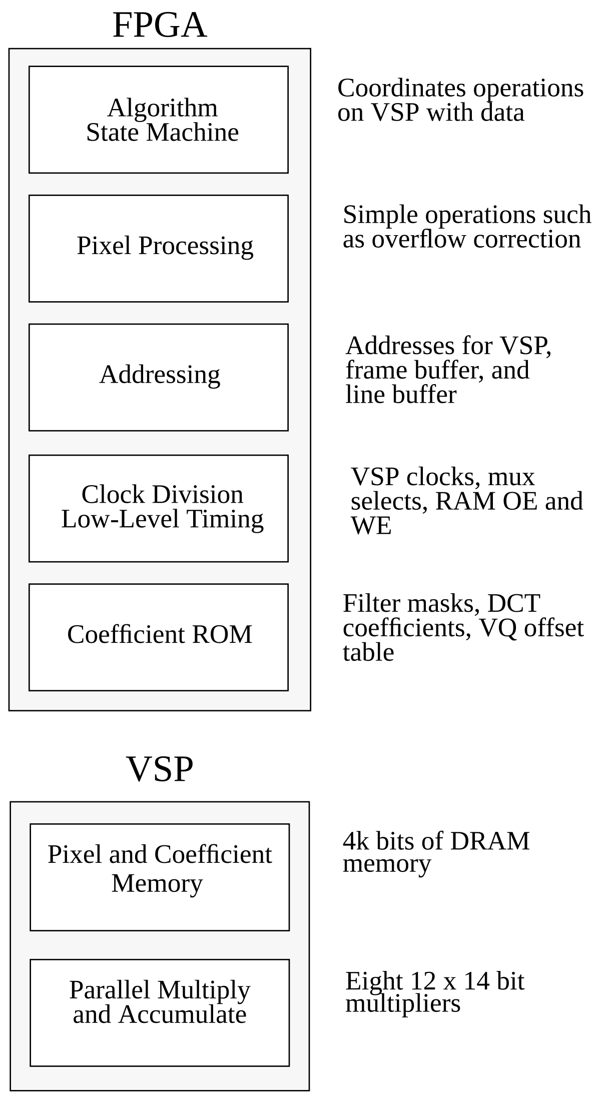

FPGA
Algorithm State Machine
Coordinates operations on VSP with data
Pixel Processing
Simple operations such as overflow correction
Addressing
Addresses for VSP, frame buffer, and line buffer
Clock Division Low-Level Timing
VSP clocks, mux selects, RAM OE and WE
Coefficient ROM
Filter masks, DCT coefficients, VQ offset table
VSP
Pixel and Coefficient Memory
4k bits of DRAM memory
Parallel Multiply and Accumulate
Eight 12 x 14 bit multipliers

Figure 3: Division of Operations Among FPGA and VSP

The VSP and FPGA are the basis for a prototyping system operating on a Sun workstation shown in figure 4. The FPGA used is a Xilinx XC4008 [13] (8000 gate equivalent.) A 3000 gate FPGA could have been used, but there is a problem with the minimizer on the VSP used for VQ. Consequently, the VQ minimizer and VQ offset table (6k bits of ROM) are placed on the FPGA until the problem can be fixed. Some statistics for the working system are shown in figure 5.

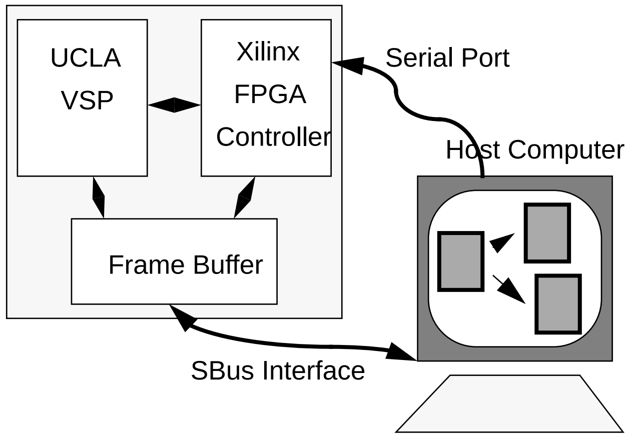


Figure 4a: Conceptual View of Video Prototyping System

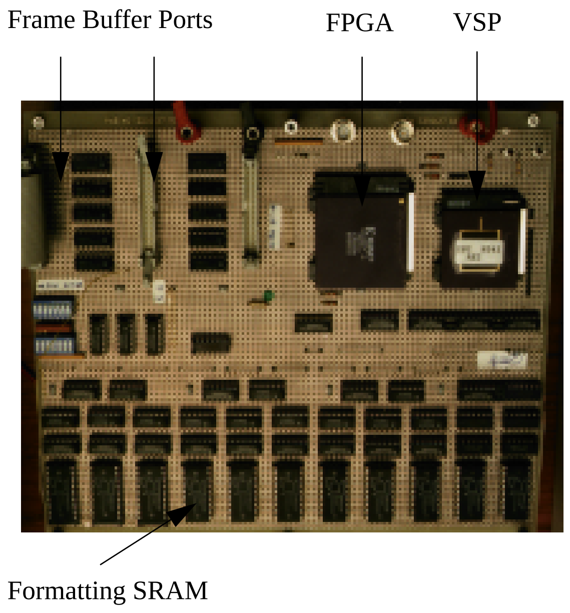


Figure 4b: Photograph of Video Prototyping System

Frame Rates for FPGA-VSP Processor:	
(All operations on 256 x 256 x 8 bit pictures)	
7 x 7 Mask 2D Filter	13.3 frames / sec.
8 x 8 Block DCT	55 frames / sec.
4 x 4 Block VQ at 1/2 bpp	7.4 frames / sec.
One level wavelet transform	35.7 frames / sec.
Max FPGA Clock Rate	20MHz*
(*Except VQ with minimizer on FPGA at 5MHz)	
Max VSP Clock	50MHz (400M Multiply/sec.)
Other System Parameters:	
Total time from disk to processor to disk	3.5s
Time to reconfigure FPGA via serial port	7.5s
Approximate total system power	13.0W

Figure 5: Prototyping System Statistics

For this system, each FPGA design is entered into ViewLogic's ViewDraw where it can be simulated with a VHDL description of the VSP. Ideally, we would like to synthesize FPGAs for the system in a more automated fashion, but the problem of creating one chip to use another chip to perform an operation is challenging. (More precisely, the existing VSP hardware places heavy constraints on the methods the FPGA may use to implement a given algorithm. At this time, we do not know how to automatically map a given algorithm onto the FPGA-VSP system.)

When designing the VSP we were attempting to move the common hardware (multipliers, memory) to an ASIC. In retrospect, it is easy to see that some low-level operations (clock division, DRAM refresh) are also common to all algorithms and would have resulted in a simpler and more user-friendly system if placed on the ASIC as well.

Perhaps the next step for the FPGA-VSP processor would be the creation of a single, general purpose FPGA-DSP chip. The work on the VSP project suggests that other DSP applications might benefit from the combination of user-programmable logic and dedicated memory and multiplier elements. Of course, memories and multipliers can be synthesized on FPGAs, but dedicated multipliers and SRAMs would consume less silicon area. (As a rough estimate of 1 bit SRAM = 1 gate and 1 full adder = 11 gates, the simple VSP chip would have consumed 18,784 gates if implemented on an FPGA.)

A conceptual view of such an FPGA-DSP chip is shown in figure 6. A major design challenge would be to keep the SRAM and multiplier design flexible so that the chip could be used for many applications. The Texas Instruments MVP [2] uses multipliers that can perform one 16 by 16 bit multiplication or two 8 by 8 bit multiplications. Similar design techniques might be used to create the configurable multipliers and SRAMs depicted in figure 6. The FPGA-VSP system is certainly not the first semi-programmable [14] architecture, but it is further testimony to the potential for processors to be designed in multiple technologies.

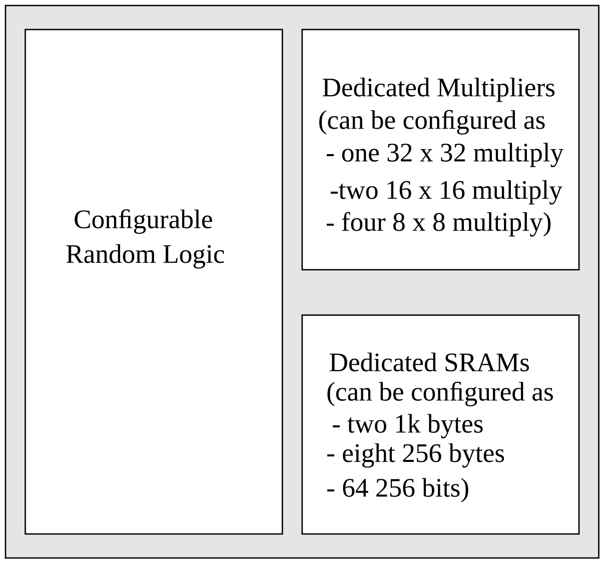


Figure 6: Conceptual Architecture for FPGA-DSP Hybrid

4.0 Conclusions

We have demonstrated two video compression systems that make efficient use of FPGAs. In the first system, the algorithm is low-complexity and uses in-system reprogrammability to fit on a single FPGA. In the second system, operations that do not need in-system reprogrammability are moved to an ASIC where they will not consume valuable FPGA space. The FPGA-VSP system demonstrates how implementations that combine ASIC and FPGA technology can be more efficient than either technology alone. This suggests that future processors can achieve high efficiency by selectively applying reconfigurable and dedicated hardware to the tasks for which they are best suited.

ACKNOWLEDGMENTS

This work was supported by Hughes Aircraft Co., ARPA/CSTO under contract J-FBI-93-112, and by ARPA/CSTO under contract number DABT63-94-C-0085 under a subcontract to National Semiconductor. Software was provided by Xilinx. We are grateful for the assistance of Avi Madisetti, Judy Liao, Chris Jones, Ben Belzer, and Sean Nazareth.

REFERENCES

[1] Shoup, R. G., “Real-Time Image Manipulation Using Soft Hardware,” IEEE SMC 1993, Le Toquet, France, Oct 17-20, 1993.

[2] Beinart, J. H., “MVP: A Multimedia Video Processor Architecture,” IEEE Workshop on Visual Signal Processing and Communications, Rutgers, New Jersey, 1994.

[3] Athanas, P.M., Abbott, A. Lynn “Image Processing on a Custom Computing Platform,” Field Programmable Logic, Lecture Notes in Computer Science, Springer-Verlag, 1994.

[4] Molloy, S. A., “Architecture and Implementation of a High Performance Video Compression Processor,” Master’s thesis, University of California at Los Angeles, 1993.

[5] Schoner, B. F., “Prototyping of Video Compression Algorithms Using Reconfigurable Hardware,” Master’s thesis, University of California at Los Angeles, 1994.

[6] Molloy, S. A., “An 80k-Transistor Configurable 25MPixels/s Video-Compression Processor Unit,” *ISSCC Digest of Technical Papers*, Feb., 1994.

[7] Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[8] Villasenor, J, et al., “Algorithms and System Prototype for Low-Power, Low-bit-rate Wireless Video Coding,” Submitted to IEEE Transactions on Circuits and Systems for Video Technology.

[9] Villasenor, J., Belzer, B., Liao, J., “Wavelet Filter Evaluation for Efficient Image Compression,” accepted for publication in IEEE Transactions on Image Processing.

[10] Configurable Logic Array (CLAy) Configuration Datasheet, National Semiconductor Corp., Santa Clara, California, 1993.

[11] H. Yamauchi, et al., “Architecture and Implementation of a Highly Parallel Single-Chip Video DSP”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, No. 2, June 1992.

[12] K.Aono, et al., “A Video Digital Signal Processor with a Vector-Pipeline Architecture”, *IEEE Journal of Solid-State CIrcuits*, Vol. 27, No. 12, December 1992.

[13] The Programmable Logic Data Book, Xilinx Inc., San Jose, California, 1994.

[14] Chen, D. C., Guerra, E. H., et. al. “An Integrated System for Rapid Prototyping of High Performance Algorithm Specific Data Paths,” IEEE Conference on Application Specific Array Processors, Berkeley, California, 1992.