2A4 Gallium Arsenide Application Symposium GAAS®96 Paris CNAM
5-7 June 1996

A GaAs video rate morphological processor for image coding and compression

Andrew Mehnert[1], Neil Burgess[2], Kamran Eshraghian[1], Wojciech Kuczborski[1] and Roberto Sarmiento[3]

[1] Edith Cowan University
Perth, Western Australia

[2] University of Adelaide
Adelaide, South Australia

[3] University of Las Palmas
de Gran Canaria, Spain

Abstract

Several image coding and compression schemes based on Mathematical Morphology have been documented in the literature including: morphological skeleton coding [1], multiresolution analysis [2], morphological pyramid coding [3,4], morphological subband coding [5,6], and segmentation-based coding using the watershed transform [7,8]. The advantages of the morphological approach over more traditional linear methods are: low computational complexity, inherent parallelism, and elimination of the ringing and blurring associated with linear techniques such as linear subband coding and transform coding using the discrete cosine transform (the latter is used in JPEG and MPEG). Morphological compression algorithms rely on two basic transformations: the dilation and the erosion. Hardware implementation of these operations is needed to realise the computational speed required for the real time processing of video. This paper proffers specifications for a video rate VLSI processor and examines the throughput and organisation for an implementation in 0.5 μm GaAs technology.

Introduction

JPEG and MPEG have emerged as international compression standards for still image and video compression respectively. Both standards employ transform coding, based on the discrete cosine transform (DCT), to remove spatial redundancy. This has two major drawbacks. Firstly, the DCT turns out to be the most demanding part of the coding/decoding process in terms of the number of arithmetic operations required [9]. Secondly, transform coding introduces a distortion known as *blocking* that is particularly evident in homogeneous regions, especially for high compression ratios [6]. The DCT also induces *blurring* and *ringing* (Gibbs phenomenon). However, it is possible to augment JPEG and MPEG syntax within the imposed block-based processing framework. For instance, subband methods, such as wavelets, combined with improved quantisation and entropy coding could offer coding gains on the order of 2-4 *db* over MPEG Intra pictures [10]. Unfortunately linear subband coders also suffer from the ringing distortion. This has led several researchers to propose subband decompositions based on Mathematical Morphology. Morphological filters not only eliminate ringing but also the blurring associated with linear filtering. Moreover they are attractive because of their low computational complexity (no multiplications are required, only additions and comparisons) and inherent parallelism. The future of the MPEG standard is still not entirely certain. Multimedia title developers have complained about the incompatibilities between various MPEG coders and decoders. Consequently a window of opportunity still exists for developers of other standards. Mathematical morphology offers several approaches to image coding and compression including: morphological skeleton coding, multiresolution analysis, pyramid coding, subband coding, and segmentation-based coding using the watershed transform. All of these methods rely on two basic transformations: the dilation and the erosion. Modern microprocessors are unable to meet the computational requirements for a single such transformation in real time, let alone the 25 or 30 per second required for video. This motivates the design and implementation of a dedicated VLSI morphological processor.

Mathematical Morphology

Mathematical Morphology (MM) is a theory for the analysis of spatial structures [11]. In the context of image processing and analysis it represents a dramatic departure from classical linear processing. MM has its roots in several mathematical disciplines including: set theory, topology, non-linear statistics, geometrical probability, and algebraic systems such as lattice theory and group theory. Although these disciplines have been used previously in image processing they most naturally arise in the context of MM. Moreover they are paramount to the application and theoretical development of MM. For example, the recent generalisation of MM to *complete lattices* provided new understanding and insights that facilitated the study of new classes of filters such as openings constructed from inf-overfilters, and the morphological centre [12].

Video Rate VLSI Morphological Processor

Edith Cowan University (Western Australia), the University of Adelaide (South Australia), and the University of Las Palmas (Spain) are jointly developing a VLSI implementation of a video rate morphological processor using GaAs technology.

A. Specifications

Processor specifications include:

- Real time processing of CCIR-601 video: 720 pixels × 480 lines × 30 frames per second (NTSC) or 720 pixels × 576 lines × 25 frames per second (PAL).
- A dynamic range of 14-bits per pixel for image (luminance) data. Symmetry about a middle grey-level is required [13] and therefore only the range [–8191,8191] is used.
- A structuring element (SE) size of at least 3 × 3 and possibly 5 × 5 or larger (origin at the centre). A SE size of 3 × 3 is sufficient to generate any required SE on the square grid and is thus suitable for applications including: thinning using the Golay alphabet; estimation of the Minkowski functionals; generation of polyhedra and line segments; calculation of city-block, chessboard and chamfer distance transforms.
- A dynamic range of 9-bits per pixel for SE data. Symmetry about zero is required and therefore only the range [–255,255] is used. The value –256 is used as an *undefined* value sentinel.
- Programmable to perform either a dilation or an erosion of an entire video frame in real time.

More detail concerning this choice of specifications is given in [13].

B. Computational requirements

For a grey-scale image $f(x,y)$ and a SE $g(x,y)$ the dilation and erosion transformations, respectively, are defined as follows:

$$D(f,g)(x,y) = \max_{(x',y')\in \text{Domain}(g)} \big(f(x-x',y-y') + g(x',y')\big)$$

$$E(f,g)(x,y) = \min_{(x',y')\in \text{Domain}(g)} \big(f(x+x',y+y') - g(x',y')\big).$$

These transformations require two basic types of operation: addition and comparison. In VLSI terms these operations are essentially identical. For an $n \times n$ SE, with n odd and the origin at the centre, a dilation or an erosion equates to moving a $n \times n$ template over the image. The origin of the template is placed over each frame pixel in turn, the corresponding template and frame pixels are summed, and the maximum or minimum of these sums is assigned to the corresponding output pixel. Thus n^2 additions and $n^2 - 1$ comparisons are required for each pixel of each frame. Moreover these additions/comparisons must be done within $1/(720 \times 480 \times 30) \approx 96$ *ns*. For a 3 × 3 SE this implies that a single addition/comparison must be done every 5.7 *ns* (see Tab. 1). Although this rate is achievable in CMOS, for 14-bit integers, using conventional 2's complement arithmetic, the associated rate of memory accesses needed to keep an adder fully occupied is not.

Table 1: Computational requirements.

SE size	No. of adds/compares per template	No. of adds/compares per second	Time per add/compare (*ns*)
3	17	176×10^6	5.7
5	49	508×10^6	2.0
7	97	101×10^7	1.0

C. Optimising memory accesses

Suppose that a single frame of video (image) is w pixels wide and h pixels high, and that the SE is n pixels square. In addition, assume that the processor reads the frame store memory by stepping down a column (or a part of a column) one pixel at a time and then moves on to the next column (see Fig. 1). Imagine that the processor can accommodate n columns of the image data at any one time (i.e. an $n \times h$ block of pixels). Then for each subsequent pixel read from the frame store only one $n \times n$ pixel block is completed. However, each pixel read actually contributes to $n^2 - 1$ other $n \times n$ pixel blocks. This suggests that the processor should comprise n^2 *template processors* so that each pixel read from the frame store is fully operated on thus widening the memory *bottleneck*.

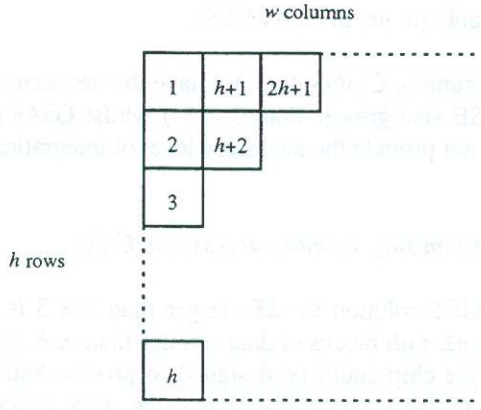

Figure 1: Sequence in which the processor accesses the frame store memory.

A single template processor would comprise a ROM containing the n^2 SE values, an address generator for the ROM, a 14-bit adder, a 14-bit comparator, and a pair of

registers: one to hold the value read from the frame store and the other to hold the partial result. This approach would permit the placement of many template processors on a single chip; but how many?

As before we assume that the processor can accommodate n columns of image data at a time. Now at any given time there will be $n.h$ partially calculated $n \times n$ templates; i.e. $n.h$ values have to be stored. Suppose that the processor comprises only a single template processor (rather than n^2), a (≤)11-bit[1] address / 14-bit data RAM, and an I/O interface. For this implementation, the processor must read one image pixel, update n^2 templates, and output one processed pixel every 96 *ns*. Each template update requires a memory read, an addition, a comparison, and a memory write. For 0.5 μ*m* CMOS VLSI technology the memory read step, the memory write step, and the add/compare step would each take about 4 *ns* (assuming 2's complement arithmetic). Consequently in 96 *ns* only 8 pixels (templates) could be updated irrespective of the size of the SE. With a dual-port memory approximately 1.5 times as many templates could be updated, thus permitting the use of 3×3 SEs. Increasing the number of template processors on the morphology processor leads to an increase in the amount of time available to perform the individual add/compare operations but does not impact at all on the time required for the memory accesses. Consequently, even if the add/compare operations take no time at all only 12 templates can be updated per frame store fetch. It is clear that memory access time is the limiting factor.

In comparison, for 0.5 μ*m* GaAs technology, the memory read step, the memory write step, and the add/compare step would each take about 1 *ns* and so in 96 *ns*, 48 templates could be updated (assuming that a dual-port RAM is available). This permits the use of 5×5 SEs. Unfortunately, though, the level of integration required is on the order of 250,000 transistors[2] which is far in excess of that currently available (to us) in GaAs VLSI.

In summary, CMOS does not have the necessary speed (for any SE size greater than 3×3) whilst GaAs technology does not provide the necessary level of integration.

D. Minimising memory accesses in GaAs

A CMOS solution for SEs larger than 3×3 is possible if we work with blocks of data smaller than $n.h$. For instance a single chip could be designed to process half an image. Two of these chips would then be used concurrently to process the entire image. Repeating this idea, four quarter image chips could be designed and so on. However, with GaAs technology a single chip solution is possible.

A crucial observation is that each time the morphological processor reads a new pixel from the frame store only n new templates are initiated. This is because $n(n-1)$ of the templates activated by the previous read are required for the new data. Consequently the number of fast memory accesses is dramatically reduced from n^2 to n. This indicates that a GaAs VLSI chip should comprise a single template processor and an $n^2 \times$ 14-bit on-chip RAM and should operate in conjunction with a dedicated $n.h \times$ 14-bit CMOS static RAM chip (to hold all the partially-computed template results). In each cycle the GaAs chip would need to fetch 1 data item, read and write n items to the CMOS RAM, and update n^2 templates. If the CMOS RAM operates in parallel with the template processing then the full 96 *ns* cycle time can be used to perform the n^2 on-chip memory reads/writes and n^2 adds/compares concurrently with n off-chip memory reads/writes. This architecture easily permits 5×5 SEs, and at a slightly lower frame rate 7×7 SEs.

E. Summary of processor options

Tab. 2 lists the possible implementation options along with the corresponding performance levels.

Table 2: Summary of morphological processor implementation options.

		Possible frame rates (frames per second)		
Option	#FETs	3×3 SE	5×5 SE	7×7 SE
CMOS - 1 chip	250,000	42	14	7
CMOS - 2 chips	125,000	85	29	14
CMOS - 4 chips	60,000	170	59	29
GaAs - 1 chip*	5,000	170	59	29

* in conjunction with a dedicated CMOS 11-bit address / 14-bit data SRAM

F. Grey-level under- and overflow

Given that the processor only supports a finite number of grey-levels it is clear that after successive dilations/erosions with non-flat (i.e. containing other than zero values) SEs, grey-level under- or overflow can occur. Depending on the particular application/algorithm to be implemented, using one or more morphological processors, this may or may not be an issue. If under- and overflow is an issue then it would seem that the obvious solution is to simply truncate all values above the maximum grey-level and all those below the minimum grey-level. Unfortunately this leads to incorrect results.

---

[1] For a SE size of 3×3.

[2] Using 6 FETs per SRAM cell → $2^{11} \times 14 \times 6 +$ 15% for decoders etc. + registers + arithmetic cell.

An erosion followed by a dilation, using the same SE, is called an *opening*. A dilation followed by an erosion, using the same SE, is called a *closing*. The opening and closing are two elementary morphological filters. The opening is antiextensive; i.e. it is a subimage of the original image (both in spatial and grey-level extent). The closing is extensive; i.e. the original image is a subimage of the closing. Unfortunately simple truncation can lead to a situation where a truncated erosion followed by a truncated dilation produces a result that is not antiextensive (and therefore not an opening). Similarly the reverse composition can produce a result that is not extensive (and therefore not a closing) [13]. In practice this affects the integrity of morphological algorithms. For example the residue of an image minus its opening (or closing minus image) must be non-negative and yet because of truncation error this may not always be the case.

The solution is to replace the grey-level addition and grey-level subtraction operators in the dilation and erosion definitions, with $\dot{+}$ and $\dot{-}$ respectively. For $t \in [-8191,8191]$ and $z \in [-255,255]$ the operators $\dot{+}$ and $\dot{-}$ are defined as follows:

$$t \dot{+} z = \begin{cases} -8191 & \text{if } t = -8191 \\ -8191 & \text{if } t > -8191 \text{ and } t + z < -8191 \\ t + z & \text{if } t > -8191 \text{ and } -8191 \le t + z \le 8191 \\ 8191 & \text{if } t > -8191 \text{ and } t + z > 8191 \end{cases}$$

and

$$t \dot{-} z = \begin{cases} -8191 & \text{if } t < 8191 \text{ and } t - z < -8191 \\ t - z & \text{if } t < 8191 \text{ and } -8191 \le t - z \le 8191 \\ 8191 & \text{if } t < 8191 \text{ and } t - z > 8191 \\ 8191 & \text{if } t = 8191. \end{cases}$$

This leads to the following definitions for dilation and erosion:

$$D(f,g)(x,y) = \max_{(x',y') \in \text{Domain}(g)} \left(f(x-x', y-y') \dot{+} g(x',y') \right)$$
$$E(f,g)(x,y) = \min_{(x',y') \in \text{Domain}(g)} \left(f(x+x', y+y') \dot{-} g(x',y') \right),$$

and to a marginal increase in complexity in terms of hardware implementation.

Conclusion

Using CMOS VLSI technology a single chip processor, limited to 3 × 3 structuring elements, is realisable for CCIR-601 video. However for larger SEs, higher resolution frames, and higher frame rates GaAs VLSI technology is mandatory. Moreover with GaAs technology a single chip solution for HDTV image sequences is feasible[3]. A GaAs video rate morphological processor offers the possibility to implement video rate morphological (and hybrid) subband coding as a replacement for transform coding using the DCT. In addition the processor can be used for more novel compression schemes including segmentation-based coding using the watershed transform.

References

[1] G. Sapiro and D. Malah, "Morphological image coding based on a geometric sampling theorem and a modified skeleton representation", *Journal of Visual Communication and Image Representation*, vol. 5, no. 1, pp. 29-40, 1994.

[2] Z. Zhou and A.N. Venetsanopoulos, "Morphological methods in image coding", in *ICASSP-92: 1992 IEEE International Conference on Acoustics Speech and Signal Processing*, IEEE, New York, 1992.

[3] X. Kong and J. Goutsias, "A study of pyramidal techniques for image representation and compression", *Journal of Visual Representation*, vol. 5, no. 2, pp. 190-203, 1994.

[4] L.A. Overturf, M.L. Comer and E.J. Delp, "Color image coding using morphological pyramid decomposition", *IEEE Transactions on Image Processing*, vol. 4, no. 2, pp. 177-185, 1995.

[5] S. Pei and F. Chen, "Subband decomposition of monochrome and color images by mathematical morphology", *Optical Engineering*, vol. 30, no. 7, pp. 921-933, 1991.

[6] O. Egger, W. Li and M. Kunt, "High image compression using an adaptive morphological subband decomposition", *Proceedings of the IEEE*, vol. 83, no. 2, pp. 272-287, 1995.

[7] J.R. Casas and L. Torres, "Coding of details in very low bit-rate video systems", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 317-327, 1994.

[8] P. Salembier and M. Pardás, "Hierarchical morphological segmentation for image sequence coding", *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 639-651, 1994.

[9] B. Ackland, "The role of VLSI in multimedia", *IEEE Journal of Solid-State Circuits*, vol. 29, no. 4, pp. 381-388, 1994.

[10] C. Fogg, *MPEG-2 FAQ* (version 3.7) [HTML document], available http://www-plateau.cs.berkeley.edu/mpegfaq/MPEG-2-FAQ.html, 1995.

[11] J. Serra and P. Soille (Eds.), *Mathematical morphology and its applications to image processing*, Kluwer Academic Publishers, The Netherlands, 1994.

[12] J. Serra (Ed.), *Image analysis and mathematical morphology. Volume 2: Theoretical advances*, Academic Press, London, 1988.

[13] A.J. Mehnert, J.M. Cross, W. Kuczborski and K. Eshraghian, *A VLSI processor for the real time implementation of mathematical morphology operations: specifications* (Research Report, ISBN: 0-7298-0233-7), Edith Cowan University, 1995.

[3] 1280H × 720V × 60 frames per second → 18 *ns* cycle time and so 9 templates can be updated thus permitting 3 × 3 SEs.