Hardware Implementation Perspectives of Digital Video Watermarking Algorithms

Nebu John Mathai, *Student Member, IEEE*, Deepa Kundur, *Member, IEEE*, and Ali Sheikholeslami, *Member, IEEE*

***Abstract*—We consider hardware implementation aspects of the digital watermarking problem through the implementation of a well-known video watermarking algorithm called Just Another Watermarking System (JAWS); we discuss the time and area constraints that must be satisfied by a successful hardware implementation. A hardware architecture that implements the algorithm under the constraints is then proposed. The architecture is analyzed to gain an understanding of the relationships between algorithmic features and implementation cost. Some general findings of this work that can be applied toward making algorithmic developments more amenable to hardware implementation are finally presented.**

***Index Terms*—Digital video watermarking, hardware implememtation, JAWS, real-time robust data hiding, VLSI.**

I. Introduction

DIGITAL watermarking is the process of embedding a message within the content of another message. The initial focus of digital watermarking research was on the development of robust methodologies for copyright protection applications; several algorithmic and performance-enhancing approaches were proposed. More recently, a theory of watermarking has emerged in which analytic tool-sets are borrowed from areas including data communications, statistical signal processing, information theory, and cryptographic protocols. The multidisciplinary nature and underlying themes of the research area have unfolded, as witnessed in many recent books addressing the topic [1]–[4]. There is now a trend toward application of the technology to novel problems such as "self-healing" media in which data that has been tampered can later correct itself, signal tagging in which region-of-interest coordinates or value-added information is embedded in hypermedia content, among others.

The current focus in algorithm development has involved improving robustness primarily through the use of sophisticated perceptual models [5]–[7], interference and attack modeling for advanced detector design [8]–[10], appropriate transform domains for superior modulation [11]–[13], and powerful error-correction codes [14].

Manuscript received February 4, 2002; revised November 8, 2002. The associate editor coordinating the review of this paper and approving it for publication was Dr. Ton A. C. M. Kalker.

N. J. Mathai and A. Sheikholeslami are with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4 Canada (e-mail: mathai@ecf.toronto.edu; ali@eecg.toronto.edu).

D. Kundur is with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843-3128 USA (e-mail: deepa@ee.tamu.edu).

Digital Object Identifier 10.1109/TSP.2003.809382

Most measures used to evaluate performance involve robustness and imperceptibility. To characterize robustness, we often use the bit error rate of the extracted watermark, similarity measures between the embedded and extracted watermark such as the correlation coefficient, the theoretically maximum amount of information that can be reliably hidden (called the watermark capacity), and the probabilities of false positive and false negative detection. To evaluate imperceptibility, measures of mean squared error (MSE) between the original and watermarked image, or peak signal-to-noise ratio (PSNR), as well as qualitative assessments are employed.

In this paper, we add a third dimension—*implementation cost*—to this measure of performance. In particular, we focus on hardware complexity. Our overall objective is to identify those algorithmic performance improvement approaches that are disproportionate to the design effort and cost of implementation and, hence, exhibit a poor performance compromise when cost is taken into account.

A. Hardware versus Software

A watermarking system can be implemented with either *software* or *hardware*. In a *software* implementation, the algorithm's operations are performed as code running on a microprocessor. For example, high-level scripts written for a symbolic interpreter running on a workstation or machine code software running on an embedded processor are both classified as software implementations. Software-based watermarking also provides the following:

- *Abstraction of the implementation from any hardware details.* Thus, instead of being concerned with elements such as flip-flops, RAMS, and gates, the designer focuses on implementation of the algorithm at a much "higher" level.
- *Availability of software tools to aid in realizing various data operations.* For instance, software designers have libraries of common processing functions so that they may borrow, to a large extent, from past implementations.
- *Limited means of improving area and improving time complexity (speed) of the implementation.* The software designer does not have direct control over the way RAM and processor interact, posing a limit on speed. To reduce area, (s)he must try to limit the total amount of RAM required. This is in contrast with hardware where there is full control over timing of operations into the RAM and direct control over the usage of expensive hardware resources.

Conversely, a *hardware*-based implementation is one where the algorithm's operations are fully implemented in custom-designed circuitry. The overall advantage is that hardware consumes less area and less power.

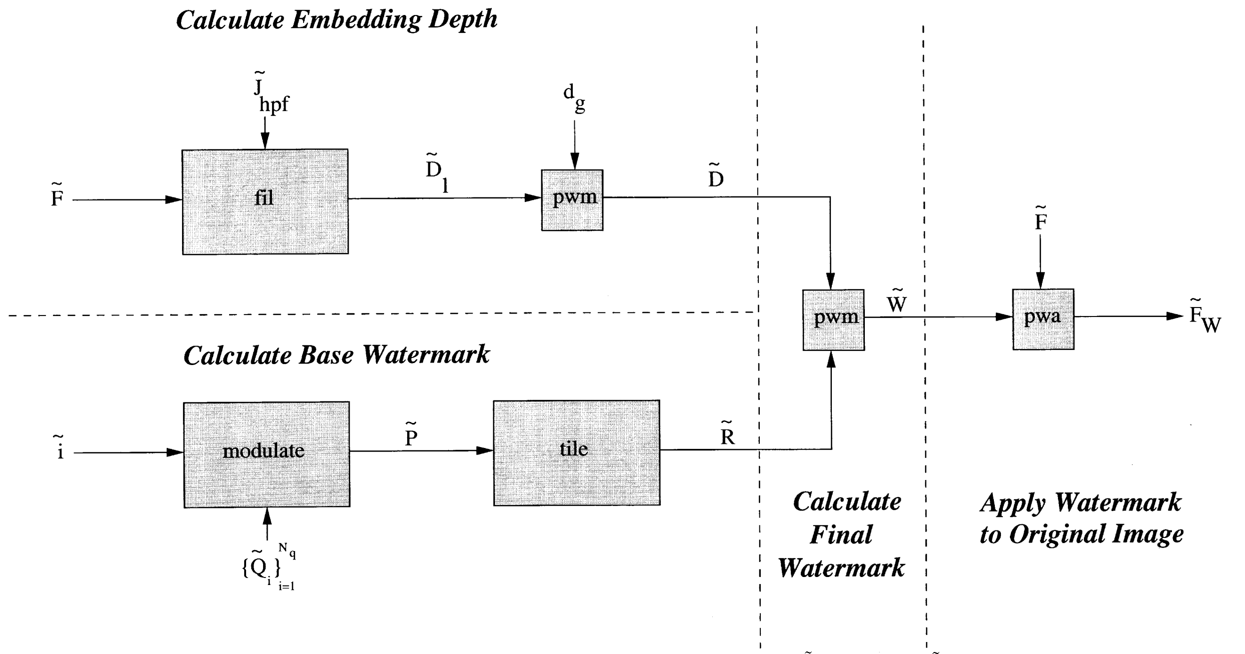


Fig. 1. Overview of the JAWS embedder. The algorithm takes three inputs: the video stream frame $\tilde{F}$, the payload $\tilde{i}$, and the global embedding depth d_g. The video frame matrix is highpass filtered and then multiplied by the global embedding depth to create the embedding depth matrix $\tilde{D}$. $\tilde{D}$ is pixel-wise multiplied with $\tilde{R}$, which is the payload-specific watermark, to create the final watermark $\tilde{W}$. $\tilde{W}$ is pixel-wise added to $\tilde{F}$ to produce the watermarked frame $\tilde{F}_W$.

Although it might be faster to implement an algorithm in software, there are a few compelling reasons for a move toward hardware implementation. In consumer electronics devices, a hardware watermarking solution is often more economical because adding the watermarking component takes up a small dedicated area of silicon. In software, implementation requires the addition of a dedicated processor such as a DSP core that occupies considerably more area, consumes significantly more power, and may still not perform adequately fast. In this paper, our hardware-level design offers many more options to reduce area and improve speed than software-level design.

B. Contributions of This Paper

In this paper, we mainly focus on hardware-design aspects of Just Another Watermarking System (JAWS) [15]. Through this case study, we illustrate implementation challenges, costs, and tradeoffs of different components of a video watermarking scheme. Due to similar structure, the design insights provided in this paper are applicable to other watermarking algorithms. In JAWS, the embedding occurs on the raw data and not on compressed data so that there is no sharing of components with a codec that would complicate our cost analysis. Furthermore, JAWS is an established algorithm that targets real-time applications where frame-rate is an issue. A software-solution for JAWS using a Trimedia DSP platform has been already implemented and tested [16]. In this work, we go beyond this implementation and propose a fully integrated hardware solution.

The objectives of this paper are two-fold:

1) *To identify hardware-friendly strategies that improve the performance of video watermarking algorithms.* Performance tradeoffs with complexity are investigated through simulations. Watermarking design is studied from the perspective of hardware figures-of-merit and tool-sets.
2) *To present a state-of-the-art VLSI architecture for the JAWS algorithm in order to illustrate the design issues at the hardware level.* To the best knowledge of the authors, this is the first standalone video watermarking chip developed in a 0.18-μm CMOS technology.

It should be mentioned that for simplicity, this paper focuses on the nonideal effects on the performance of video watermarking algorithms due to practical implementation constraints. We do not include, in this work, the additional effects of attacks on the watermarked video.

The remainder of this paper is structured as follows. Section II provides a primer on JAWS and a description of basic hardware implementation constraints. Section III proposes a novel architecture illustrating hardware design issues related to video watermarking applications. Section IV deals with the issues of complexity and cost. The relationship between various algorithmic parameters and features to the tradeoff between robustness and cost is investigated. The paper concludes with final remarks and ideas for future directions.

II. JAWS and Video Watermarking Constraints

This section provides a primer on the JAWS embedding and detection algorithms [15]–[17] and specifies constraints on the implementation.

A. Embedder: Definition and Constraints

The JAWS embedding algorithm, which is illustrated in Fig. 1, covertly embeds payload data $\tilde{i}$ into individual frames of video $\tilde{F}$, where $\tilde{i}$ represents a bit string (word) of length N, and the f_x by f_y matrix $\tilde{F}$ represents a video frame. Each element of $\tilde{F}$ corresponds to the color value of a pixel in the frame, represented by a B-bit integer. The process of creating a

TABLE I
LIST OF VARIABLES AND FUNCTION DEFINITIONS FOR THE JAWS EMBEDDING ALGORITHM

symbol	meaning
B	color depth of pixel
N	payload bit string length
G	detector folding depth
f_x	video frame x-dimension
f_y	video frame y-dimension
N_q	number of primitive PN patterns
q_x	primitive pattern x-dimension, $q_x < f_x$
q_y	primitive pattern y-dimension, $q_y < f_y$
r	number of pixels per modulation grid increment
a_x	grid x-dimension for modulation of watermark by payload, $\frac{q_x}{r}$
a_y	grid y-dimension for modulation of watermark by payload, $\frac{q_y}{r}$
$\tilde{J}_{hpf}$	embedder high-pass filter $dim = 3 \cdot 3$
$\tilde{F}$	video stream frame $dim(\tilde{F}) = f_x \cdot f_y, \tilde{F}_{[x,y]} \in [0, 2^B]$
$\tilde{i}$	payload bit string a binary string (word) of length N
d_g	global embedding depth
$\tilde{F_W}$	watermarked video stream frame (input to the encoder) $dim(\tilde{F_W}) = f_x \cdot f_y, \tilde{F_W}[x, y] \in [0, 2^B]$

function	meaning
$fil(\tilde{J}, \tilde{F})$	apply filter matrix $\tilde{J}$ to signal matrix $\tilde{F}$
$a \cdot b$	scalar multiply of a with b
$modulate(\{\tilde{Q}_i\}_1^{N_q}, \tilde{i})$	modulate $\{\tilde{Q}_i\}_1^{N_q}$ by $\tilde{i}$ to create a primitive pattern
$tile(\tilde{P})$	replicate $\tilde{P}$ to create a new matrix of size $f_x \cdot f_y$
$dim_r(\tilde{A})$	the number of rows in $\tilde{A}$
$dim_c(\tilde{A})$	the number of columns in $\tilde{A}$
$dim(\tilde{A})$	the number of elements in $\tilde{A}$, $dim_r(\tilde{A}) \cdot dim_c(\tilde{A})$
$pwm(\tilde{A}, \tilde{B})$	point-wise-multiply the elements of $\tilde{A}$ with those of $\tilde{B}$ $\tilde{A}$ and $\tilde{B}$ have identical dimensions
$pwa(\tilde{A}, \tilde{B})$	point-wise-add the elements of $\tilde{A}$ with those of $\tilde{B}$ $\tilde{A}$ and $\tilde{B}$ have identical dimensions

*In this work, letters without an overhead ˜ denote scalars, uppercase letters with a ˜ denote general matrices, lowercase letters with a ˜ specifically denote matrices with a single row, an overhead ^ denotes the transform-domain of the specified quantity.

payload- and frame- specific watermark is described in the following.

The creation of the payload-specific base watermark $\tilde{R}$ is shown in Fig. 1 under *Calculate Base Watermark.* First, a set of N_q sequences of pseudo-random noise (the primitive patterns $\{\tilde{Q}_i\}_{i=1}^{N_q}$) are modulated by $\tilde{i}$. This modulation is accomplished by superimposing shifted versions of the primitive patterns, where the shifts are a function of the payload. [1] The result of modulation $\tilde{P}$ (called a *tile*) is then replicated (*tiled*) over the same dimensions as the input frame to generate $\tilde{R}$.

If the base watermark were directly superimposed on the video frame, the watermark may be perceptible. To correct this, local embedding depths $\tilde{D_l}$ are calculated for each pixel by highpass filtering[2] the video frame, as shown in Fig. 1 under *Calculate Embedding Depth.* To control the overall strength of the embedded watermark, the local embedding depths are scaled by the global embedding depth d_g to create the embedding depth of the watermark $\tilde{D}$. The base watermark is then pixel-wise multiplied with $\tilde{D}$ to create the final watermark $\tilde{W}$, which is pixel-wise added with the original video frame to create the watermarked frame $\tilde{F_W}$.

TABLE II
SUMMARY OF THE JAWS EMBEDDER ALGORITHM

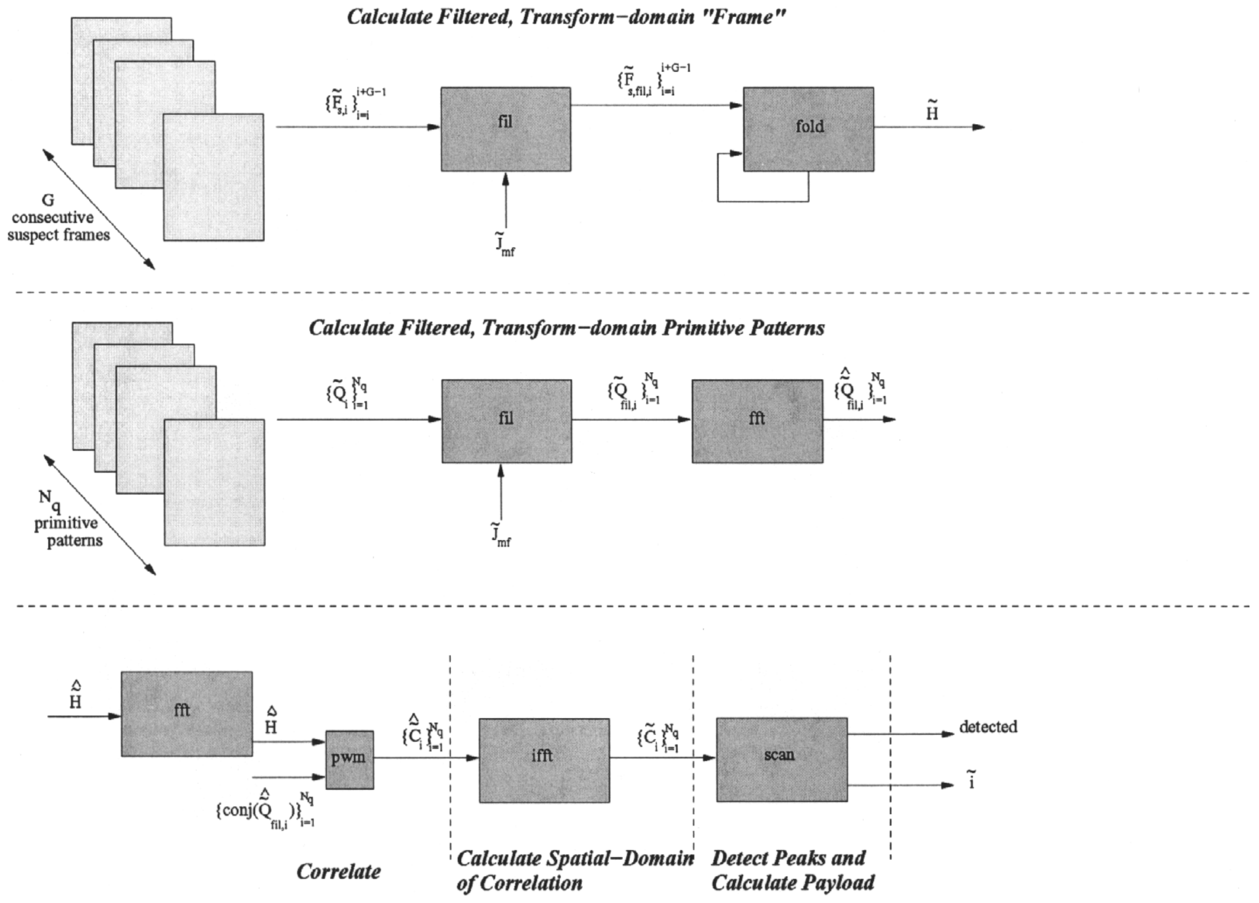
Step	Equation
Obtain Embedding Depth	$\tilde{D_l} = fil(\tilde{J}_{hpf}, \tilde{F})$ $\tilde{D} = d_g \cdot \tilde{D_l}$
Obtain Base Watermark	$\tilde{P} = modulate(\{\tilde{Q}_1, \ldots, \tilde{Q}_{N_q}\}, \tilde{i})$ $\tilde{R} = tile(\tilde{P}), dim(\tilde{R}) = f_x \cdot f_y$
Obtain Final Watermark	$\tilde{W} = pwm(\tilde{D}, \tilde{R})$
Apply Watermark	$\tilde{F_W} = pwa(\tilde{F}, \tilde{W})$

The algorithm variables, functions, and basic steps are summarized in Tables I and II.

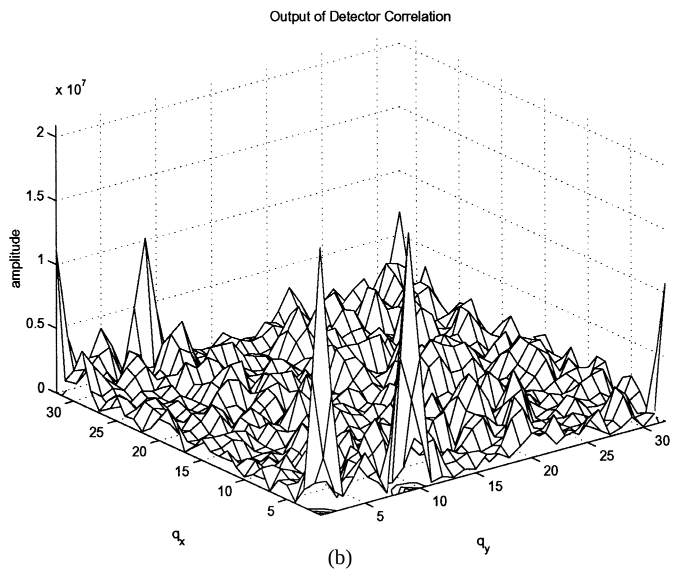
Constraints: Two emerging applications of digital video watermarking are broadcast monitoring and copy control; a practical implementation of a watermarking system must conform

[1]The shifts are further constrained to be over a grid of dimension $a_x \cdot a_y$ imposed over the span of $q_x \cdot q_y$. The motivations for this are from practical considerations at the watermarking system level and are explained in [15].

[2]The highpass filter serves as a crude model of the human visual system's masking characteristics.


Calculate Filtered, Transform−domain "Frame"
G consecutive suspect frames
$\{\tilde{F}_{s,i}\}_{i=i}^{i+G-1}$
fil
$\tilde{J}_{mf}$
$\{\tilde{F}_{s,fil,i}\}_{i=i}^{i+G-1}$
fold
$\tilde{H}$
Calculate Filtered, Transform−domain Primitive Patterns
N_q primitive patterns
$\{\tilde{Q}_i\}_{i=1}^{N_q}$
fil
$\tilde{J}_{mf}$
$\{\tilde{Q}_{fil,i}\}_{i=1}^{N_q}$
fft
$\{\hat{Q}_{fil,i}\}_{i=1}^{N_q}$
$\hat{H}$
fft
$\hat{H}$
pwm
$\{conj(\hat{Q}_{fil,i})\}_{i=1}^{N_q}$
$\{\hat{C}_i\}_{i=1}^{N_q}$
ifft
$\{\tilde{C}_i\}_{i=1}^{N_q}$
scan
detected
$\tilde{i}$
Correlate
Calculate Spatial−Domain of Correlation
Detect Peaks and Calculate Payload

(a)



(b)

Fig. 2. (a) Overview of the JAWS detector. The algorithm takes in the possibly tampered watermarked video stream frame $\tilde{F}_S$ extracts an estimate of the payload bit string $\tilde{i}$ and makes a decision on the status of whether or not the watermark was detected. (b) Output of the *ifft* calculation for watermark detection where the payload was "1."

to the constraints on time (performance) and space (area) demanded by these applications.

In broadcast monitoring, the watermark can either be embedded into the media long before transmission or at the time of transmission (e.g., in a live video feed). It is in the latter case that a strict real-time requirement must be met: The watermark embedder must function as fast as the video frame rate. Since the high cost of broadcast equipment can absorb costs associated with the embedder hardware, no strict area constraints are posed by broadcast monitoring.

In copy control, embedding is usually done in the recording device prior to storage onto the media [18] in a process called remarking. There may also be applications where remarking occurs in the playback device. Since video data at these points will

TABLE III
LIST OF VARIABLES AND FUNCTION DEFINITIONS FOR THE JAWS DETECTOR ALGORITHM

symbol	meaning
$\tilde{J}_{mf}$	detector whitening filter $dim = 3 \cdot 3$
$\{\tilde{F}_{s,i}, \cdots, \tilde{F}_{s,i+k-1}\}$	k consecutive suspect video frames (input to the decoder) $dim(\tilde{F}_s) = f_x \cdot f_y, \tilde{F}_s[x,y] \in [0, 2^B]$
$detected$	status on whether a watermark was detected
function	**meaning**
$fold(\{\tilde{A}_i\}_1^N, b_x, b_y)$	$\forall i \in [1, N]$, over the complete span of $\tilde{A}_i$, accumulate (using the $pwa()$ operator) sub-matrices (of dimension $b_x \cdot b_y$) of $\tilde{A}_i$ so that the result of the accumulation is a matrix (of dimension $b_x \cdot b_y$)
$fft(\tilde{A})$	fast-Fourier transform of $\tilde{A}$
$ifft(\tilde{A})$	inverse fast-Fourier transform of $\tilde{A}$
$conj(\tilde{A})$	complex conjugate of each element of $\tilde{A}$
$scan(\{\tilde{A}_i\}_1^N)$	scan the set of matrices $\tilde{A}_i$ for spikes, from which to derive what payload was modulated

TABLE IV
SUMMARY OF THE JAWS DETECTOR ALGORITHM

Step	Equation
Calculate Filtered, Transform-Domain Frame	$\{\tilde{F}_{fil,i}, \cdots, \tilde{F}_{fil,i+G-1}\} = \{fil(\tilde{J}_{mf}, \tilde{F}_i), \cdots, fil(\tilde{J}_{mf}, \tilde{F}_{i+G-1})\}$ $\tilde{H} = fold(\{\tilde{F}_{fil,i}, \cdots, \tilde{F}_{fil,i+G-1}\}, q_x, q_y)$ $\widehat{\tilde{H}} = fft(\tilde{H})$
Obtain Filtered, Transform-domain Primitive Patterns	$\{\tilde{Q}_{1,fil}, \cdots, \tilde{Q}_{N_q,fil}\} = \{fil(\tilde{J}_{mf}, \tilde{Q}_1), \cdots, fil(\tilde{J}_{mf}, \tilde{Q}_{N_q})\}$ $\{\widehat{\tilde{Q}}_{1,fil}, \cdots, \widehat{\tilde{Q}}_{N_q,fil}\} = \{fft(\tilde{Q}_{1,fil}), \cdots, fft(\tilde{Q}_{N_q,fil})\}$
Correlate and detect	$\{\widehat{\tilde{C}}_1, \cdots, \widehat{\tilde{C}}_{N_q}\} = \{pwm(conj(\widehat{\tilde{Q}}_{1,fil}), \widehat{\tilde{H}}_{fil})), \cdots, pwm(conj(\widehat{\tilde{Q}}_{N_q,fil}), \widehat{\tilde{H}}_{fil}))\}$ $\{\tilde{C}_1, \cdots, \tilde{C}_{N_q}\} = \{ifft(\widehat{\tilde{C}}_1), \cdots, ifft(\widehat{\tilde{C}}_{N_q})\}$ $\{detected, \tilde{i}\} = scan(\{\tilde{C}_1, \cdots, \tilde{C}_{N_q}\})$

likely be streaming at the real-time frame rate, there is a performance requirement that the embedder operate as fast as the frame rate. As copy control schemes will be used in consumer electronics where low cost is essential, minimizing space complexity is also very important.

We constrain the embedder architecture to satisfy the aggressive requirements that the embedder handle real-time video frame rates and minimize implementation area.

B. Detector: Definition and Constraints

Fig. 2(a) illustrates the flow of data in the JAWS detection algorithm where the presence of an embedded watermark in a suspect frame of video is detected and the payload extracted.

The detection algorithm first filters G consecutive frames $\tilde{F}_s$ by a whitening filter $\tilde{J}_{mf}$ to create G filtered frames $\tilde{F}_{s,fil}$. These filtered frames are then folded to form $\tilde{H}$. Next, the correlation between $\tilde{H}$ and the N_q primitive patterns (used in the embedding process) is computed. The result of this correlation indicates the presence of any primitive pattern or its shifted versions in $\tilde{H}$. Finally, the payload is extracted from the distances between adjacent instances of each primitive pattern.

For computational efficiency, the correlation used for payload detection is computed as a convolution in the frequency domain. To do this, the fast Fourier transforms of $\tilde{H}$ and each of the primitive patterns (which have also been filtered with $\tilde{J}_{mf}$) are calculated.[3] The complex conjugate of each frequency-domain primitive pattern is then pixel-wise multiplied with the frequency-domain folded image. Then, the inverse fast Fourier transform of the product is applied. The result will be a series of spikes indicating the presence of primitive patterns from which the payload can be derived; this is illustrated in Fig. 2(b) for $N_q = 1$, $q_x = q_y = 32$.

The algorithm variables, functions and basic steps are summarized in Tables III and IV.

Constraints: As with the embedder, the applications of broadcast monitoring and copy control are used to obtain constraints for architecting the detector.

In broadcast monitoring, several channels of video data are monitored simultaneously and checked for the presence of watermarks. Due to the large amount of real-time video data that must be processed (continuous streams over many channels),

[3]Since the computation of the fast Fourier transforms of the primitive patterns is independent of any input to the detector, they can be precomputed and considered constants to the process.

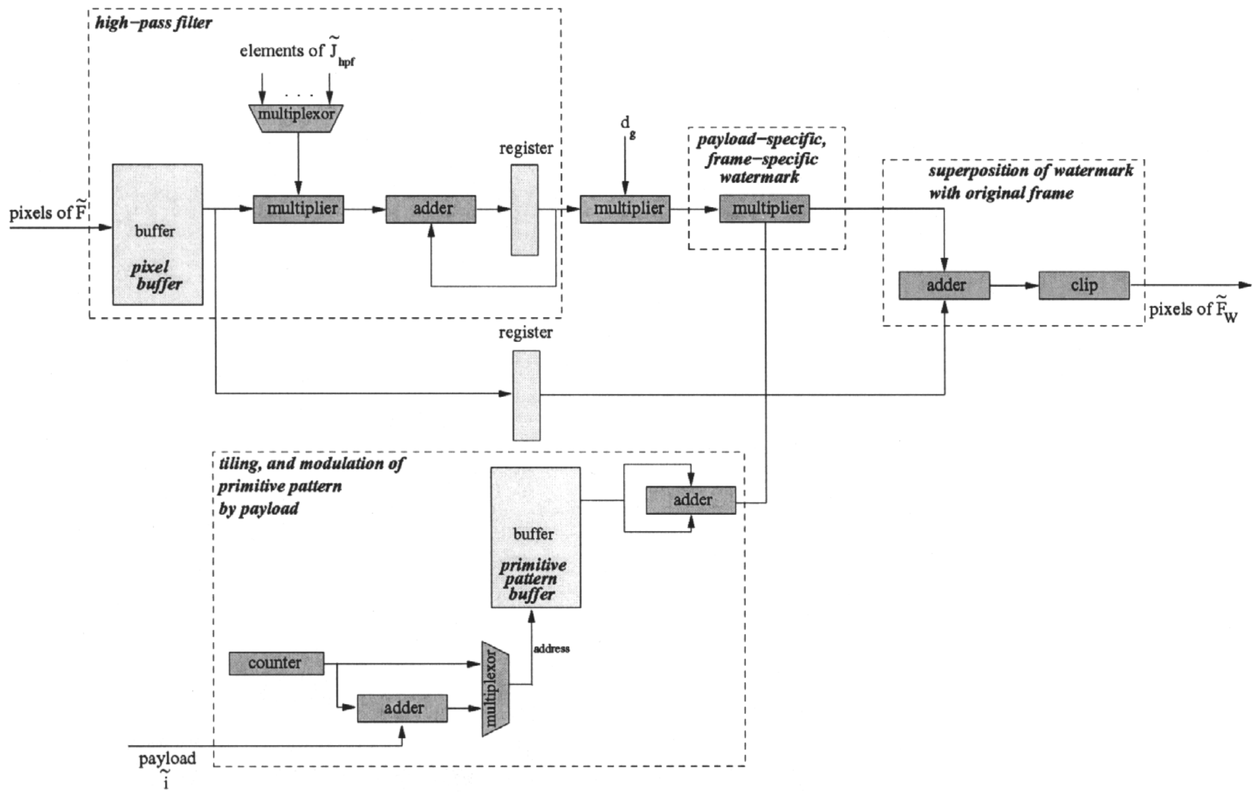


Fig. 3. Proposed hardware architecture for the JAWS embedder.

detection at the frame-rate of video is required to ensure thorough monitoring; at a slower rate, portions of the video streams would have to be dropped. For copy control, as with the embedder, detection can be done at the frame rate of the video as video data is being streamed. Hence, both applications pose the requirement for detection at the frame rate of video.

As with the embedder, the detector space complexity is of no consequence in broadcast monitoring; however, with copy control, due to the need to minimize economic cost, a small implementation must be targeted.

Hence, the detector's architecture will be under the constraints of maintaining real-time performance and minimizing area.

III. Hardware Architecture

This section details architectures that implement the embedder and detector algorithms under the constraints defined in Section II. Unlike the previous implementation of JAWS [16], this work implements JAWS in custom hardware.

A. Watermark Embedder

We assume in our implementation that the input video to the embedder is provided as a stream of pixel data. That is, the elements of $\tilde{F}$ are provided one row at a time, beginning with the first row and ending with the last. The data rate is assumed to be consistent with the frame rate of real-time video. Referring to Fig. 1, there is a unidirectional flow of data from the input to the output. Each block simply processes a few elements of data (pixels) and forward the results to the next block—there are no iterative operations. This makes the algorithm very amenable to an implementation known as *pipelining*, where each block (*stage*) completes one operation and submits its output to the next stage in the pipeline. Pipelines allow for high data throughput as long as each stage only adds some delay (*latency*) to the data stream but does not block it.

Fig. 3 illustrates the embedder architecture. The hardware to perform the two main operations of calculating the embedding depths and calculating the base watermark and the final generation of the watermarked frame are described in the following.

Calculating the Embedding Depths: The main computation in calculating the embedding depths is the highpass filtering of the input frame. Due to the 3×3 dimensions of the filter ($\tilde{J}_{hpf}$) and the row-by-row format of the input stream, filtering requires at least three consecutive rows of frame data. Therefore, incoming pixels are stored into a pixel buffer until three rows are available. At that point, one row of filter data will be computed. For each element of the row, nine pixels are read from the buffer, multiplied with the appropriate filter matrix element (from $\tilde{J}_{hpf}$) and accumulated. It is apparent that this part of the embedding algorithm impacts the pipeline in two ways: First, it adds some delay to the data stream exiting the embedder, and second, it may block the pipeline for some duration. Latency does not disturb the overall throughput of the pipeline; however, blockage may disrupt real-time operation. To prevent blockage, the rate of processing of the embedder (*clock frequency*, f_{clock}) must satisfy a constraint defined by the following equation taking into account the frame size ($f_x \cdot f_y$), frame rate (R_{frame}), and the number of memory accesses per pixel (N_{access}) for the filtering process:

$$f_{\text{clock}} \geq f_x \cdot f_y \cdot R_{\text{frame}} \cdot N_{\text{access}} \tag{1}$$

where N_{access} is an implementation-specific constant. As an example, if a single-port RAM[4] is used to implement the pixel buffer, then ten memory accesses (there is one clock cycle per access) are required per pixel computation: one to write the pixel to the buffer and nine to read pixel values from the buffer to calculate the filtered value.

Calculating the Base Watermark: To calculate the base watermark, a payload-specific tile is created and then tiled over the span of the image frame. Creating the primitive tile requires a means to generate the noise sequences that will serve as the primitive patterns and a means to modulate the primitive patterns by payload data.

A variety of methods are available to generate the noise sequence, such as linear feedback shift registers (LFSRs) or table lookup where precomputed noise sequences are stored in a buffer. The former method is considerably cheaper than the latter, as table lookup requires large silicon to implement. Lookup tables, on the other hand, provide a trivial solution to modulation, where indexing into arbitrary points of the noise sequence is needed; LFSRs do not offer such solutions. Another advantage of table lookup is that each element of the noise sequence can be directly set so that the statistics of the sequence can be contrived in any manner. For these reasons, in our architecture, a lookup table is used to store the noise sequences for the primitive pattern. Both modulation of the noise sequences by payload data and tiling are accomplished at negligible cost (in area and time) by the use of a counter and adder to generate the addresses to the noise buffer. The counter, by nature, implements the tiling function as it counts modulo q_x and q_y over the range of f_x and f_y. At each step in the count, two reads from the noise buffer occur: one at the count and another offset (by a function of $\tilde{i}$) from the count; the offsetting of two identical primitive patterns by $\tilde{i}$ accomplishes modulation.

Generating the Final Watermarked Frame: Having the two major embedder functions architected, the remainder of the architecture consists of relatively small hardware to multiply the watermark by the depths and add the result to the original pixels.

Real-Time Performance: Only the constraint defined in (1) must be satisfied to maintain real-time performance as the filtering hardware is the only place in the embedder architecture where data flow is blocked.

B. Watermark Detector

As with the embedder, the detector is architected to function as a stage in a system-level pipeline designed to process real-time video data, with the same input data stream characteristics. However, in contrast to the embedder, the detection algorithm is considerably more complex. As Fig. 4 illustrates, detection consists of a series of computations that operate on an entire matrix, as opposed to elements of a data stream. Moreover, the computations themselves are iterative. Both of these aspects will result in prolonged calculations and potential blockage; the architecture must compensate to prevent the blockage from adversely affecting the real-time performance.

[4]A single-port buffer only allows *either* a single write *or* a single read to be performed on the RAM at a time.

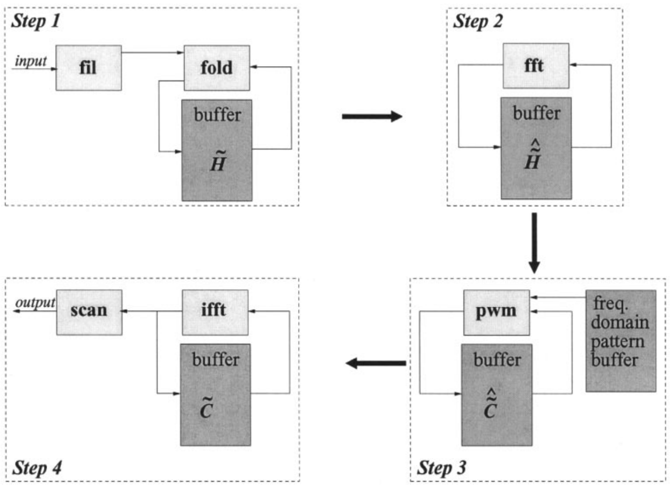


Fig. 4. Sequence of processes and flow of data involved in detection.

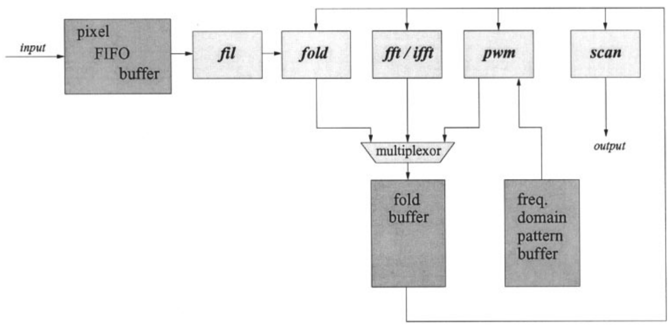


Fig. 5. Proposed hardware architecture for the JAWS detector.

Fig. 5 illustrates the detector architecture. A first-in first-out (FIFO) buffer at the input port prevents blockage in the system-level pipeline by buffering pixels at the input port while the detection process is occurring. The whitening filter at the output of the FIFO drains the pixels of the FIFO, filters them, and forwards the filtered result to the fold hardware. The fold hardware accumulates the filtered results into a matrix (the fold buffer).[5] Then, the *fft*, *pwm*, *ifft*, and *scan* blocks sequentially operate on the fold buffer to detect and extract any payload. At any one step in the detection process, only one of the computational blocks (*fft*, *pwm*, *ifft*, or *scan*) is actively operating on the contents of the shared fold buffer; this is enforced by the *multiplexor* at the input of the fold buffer that selects the currently active path into the buffer.

Filter and Fold Process: The topology of the detector's whitening filter is identical to the highpass filter of the embedder (shown in the *highpass filter* section of Fig. 3), the only difference being that the elements of $\tilde{J}_{mf}$ are used in the detector instead of $\tilde{J}_{hpf}$. Fig. 6 illustrates the fold datapath; *input operand 1* is obtained from the filter, *input operand 2* is obtained from the previously folded value in the fold buffer (it is 0 for the initial iteration), and *op* is addition. From the detailed discussion on the filter (Section III-A) and the unidirectional

[5]Due to the linearity of the filtering process, another possibility would be to first fold the frames and then filter the result. This would save area, at the expense of increased detection time.

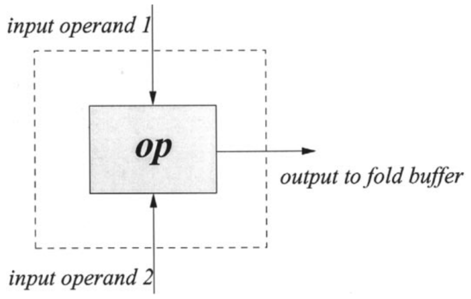


Fig. 6. Basic topology of fold and pwm data path.

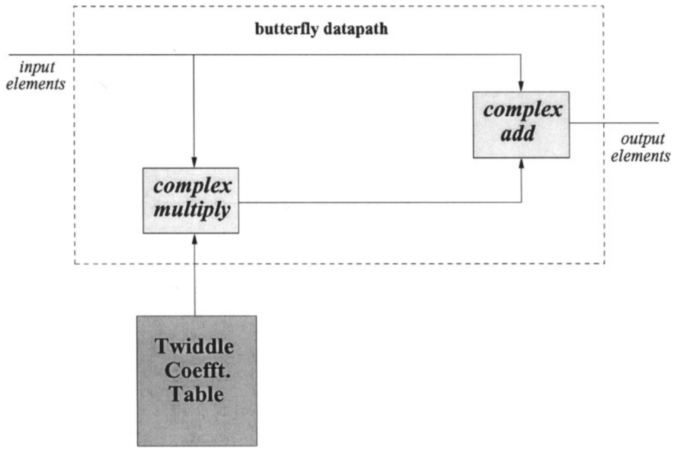


Fig. 7. Basic topology of fft data path.

flow of folding, no blockage to the pipeline occurs due to this step.

Fast Fourier Transform Hardware: Fig. 7 illustrates the topology of the radix-2 butterfly that implements the in-place decimation-in-time q-point fast Fourier transform for both the *fft* and *ifft* operations[6] ($q = q_x = q_y$ for the remainder of this paper). The same butterfly is used for all iterations of the transform. The duration of the overall calculation is defined in units of clock cycles by the following equation in terms of q and the number of cycles to compute one butterfly $T_{\text{path,butterfly}}$ (an implementation-specific constant):

$$T_{\text{fft}} = 2 \cdot q \cdot \log_2(q) \cdot \frac{q}{2} \cdot T_{\text{path,butterfly}} \tag{2}$$

Point-Wise Multiplication: Fig. 6 illustrates the topology of the *pwm* computation, where *input operand 1* is obtained from the fold buffer, *input operand 2* is the corresponding value in the frequency domain pattern buffer, and *op* is a complex multiplication. The operation is iterated over the entire fold buffer. The duration of the calculation is defined in units of clock cycles by the following equation in terms of q and the number of cycles required to compute one point-wise multiplication $T_{\text{path,pwm}}$ (an implementation-specific constant):

$$T_{\text{pwm}} = q \cdot q \cdot T_{\text{path,pwm}}. \tag{3}$$

[6]Optimizations that were possible with the *fft* due to the real-valued data on which it operates were not considered since they only contribute a constant reduction in the latency and no reduction in the area cost.

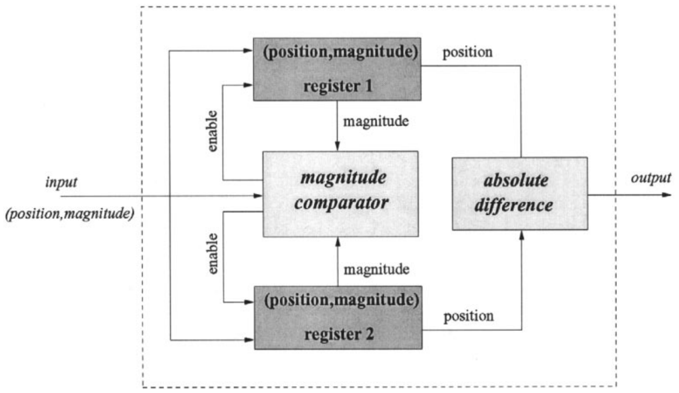


Fig. 8. Basic topology of scan data path.

TABLE V
TIME AND SPACE COMPLEXITY OF THE EMBEDDER

feature	area complexity	time complexity
high-pass filter	filter buffer (RAM) $\mathcal{O}(3 \cdot f_x \cdot B\text{-bits})$	$\mathcal{O}(3 \cdot f_x)$ pixel-time latency
payload modulation	pattern buffer (ROM) $\mathcal{O}(N_q \cdot q^2 \cdot W_e)$	negligable
other datapath	negligable	negligable
overall	RAM: $\mathcal{O}(B \cdot f_x)$ ROM: $\mathcal{O}(N_q \cdot q^2 \cdot W_e)$	$\mathcal{O}(3 \cdot f_x)$ pixel-time latency

Scan: The *scan* operation, shown in Fig. 8, scans the buffer and stores the magnitudes and positions of the two largest distinct values in the fold buffer. At the end, it computes the absolute distance between the two positions and returns this as the detected payload. The scan operation works in parallel with the last iteration of *ifft* and does not contribute to the duration of detection.

Real-Time Performance: Real-time performance is maintained by ensuring that the detector does not block the stream of pixels. Due to the shared fold buffer in the architecture, pixels cannot be processed by *fold* while the fold buffer is being used by the *fft*, *ifft*, and *pwm* operations; this blockage must be compensated for by properly sizing the input FIFO. The total duration during which pixels cannot be processed is

$$T_{\text{detect}} = 2 \cdot T_{\text{fft}} + T_{\text{pwm}} \tag{4}$$

and the incoming pixel rate is

$$R_{\text{pixel}} = \frac{f_x \cdot f_y \cdot R_{\text{frame}}}{f_{\text{clock}}}. \tag{5}$$

Hence, to ensure real-time performance, the input FIFO must be large enough to buffer $T_{\text{detect}} \cdot R_{\text{pixel}}$ pixels. Alternatively, rather than buffering the pixels, they could simply be dropped.

IV. ARCHITECTURE COMPLEXITY

Our perspective in this paper, in part, highlights the practical application-specific requirements of a video watermarking system. Many tradeoffs may be needed to create a watermarking system that is constrained by the often conflicting

TABLE VI
TIME AND SPACE COMPLEXITY OF THE DETECTOR

feature	area complexity	time complexity
whitening filter	filter buffer (RAM) $\mathcal{O}(3 \cdot f_x \cdot W_d)$	$\mathcal{O}(3 \cdot f_x)$ pixel-time latency
fold	fold buffer (RAM) $\mathcal{O}(q^2 \cdot W_d)$	$\mathcal{O}(G \cdot f_x \cdot f_y)$ pixel-time latency
overall *filter and fold pipeline*	RAM: $\mathcal{O}(W_d \cdot 3 \cdot f_x + q^2)$	$\mathcal{O}(3 \cdot f_x + G \cdot f_x \cdot f_y)$ pixel-time latency
fft / ifft	1 to 4 multipliers 2 to 4 adders twiddle table (ROM) $\mathcal{O}(q/2)$	$\mathcal{O}(q^2 \cdot \log_2 q)$ clock-cycle latency
pwm	primitive pattern buffers (ROM) $\mathcal{O}(N_q \cdot q^2 \cdot W_d)$	latency $\mathcal{O}(N_q \cdot q^2)$ cycles
scan	negligable	none; overlapped with last pass of *ifft*
overall *detection process*	ROM: $\mathcal{O}(N_q \cdot q^2 \cdot W_d)$	latency $\mathcal{O}(q \cdot \log_2 q + q^2)$
overall	RAM: $\mathcal{O}(W_d \cdot 3 \cdot f_x + q^2)$ ROM: $\mathcal{O}(N_q \cdot q^2 \cdot W_d)$	latency $\mathcal{O}(3 \cdot f_x + G \cdot f_x \cdot f_y + q^2 \cdot \log_2 q + q^2)$

requirements of the application and economic cost. To create an optimal solution, analysis must be performed to balance the benefits of algorithmic features with the implementation costs involved. This section creates a basis for cost-benefit analysis of watermarking schemes implemented in hardware. It presents the hardware costs associated with various algorithmic features, with some assessment of the "algorithmic benefit" obtained from each feature. It concludes with a discussion of strategies to improve general watermarking schemes and the associated implementation perspectives.

A. Complexity of Algorithmic Features

From a signal processing perspective, the objective of a watermarking scheme is to achieve high data rate of the embedded payload such that perceptual distortion is minimized and the detectability of the watermark ("robustness") is maximized. When implementation is considered, real-time performance and reduction of area also become important.

To measure the benefit of algorithmic features, a software model (in C) of the watermarking system was created to allow quick simulation of different cases; this model is completely identical, bit-for-bit, to the hardware description code (RTL) that implements the architecture. The simulations were run on a 350-MHz Pentium II system running Linux with 256 MB of RAM. The metric we use to measure perceptibility of the embedded watermark is the root mean square error (RMSE) between the watermarked frame $\tilde{F}_W$ and the original unmarked frame $\tilde{F}$, as defined by

$$\text{RMSE} = \sqrt{\frac{1}{f_x \cdot f_y} \sum_{x=1}^{f_x} \sum_{y=1}^{f_y} \left(\tilde{F}_W[x,y] - \tilde{F}[x,y] \right)^2}. \quad (6)$$

The metric we have used to measure robustness of detection is the signal-to-noise ratio (SNR) of the correlated output of the detector (i.e., the output of the *ifft*), where we considered the "signal" to be the correlation peaks that the detector used to extract payload information and "noise" to be the remaining correlation values that the detector ignored. The following equation defines the calculation of the metric:

$$\text{SNR} = 10 \log \left(\frac{\sum\limits_{\forall i} \text{signal}^2(i)}{\sum\limits_{\forall i} \text{noise}^2(i)} \right). \quad (7)$$

To arrive at some conclusions on the cost of the algorithmic features of JAWS, we treat the hardware components of the embedder and detector separately. The following discussion reports the results of our complexity analysis and is summarized in Tables V and VI as well.

1) Achieving High Data Rate: JAWS achieves high data rate by modulating a set of primitive patterns by payload data [17]. The number of bits that can be embedded per primitive pattern (N_1), the number of bits that can be embedded in a frame using N_q primitive patterns (N), and the time rate of data of the video (R_{data}) are defined in the following equations in terms of the design parameters N_q, q, and G and the application-derived frame rate R_{frame} (note that r is a constant defined in Table I):

$$N_1 = \log_2 \left(\frac{\frac{q^2}{r^2} + 2}{2} \right) \quad (8)$$

$$N = N_q \cdot N_1 \quad (9)$$

$$R_{\text{data}} = R_{\text{frame}} \cdot \frac{1}{G} \cdot N. \quad (10)$$

(We will not consider G for now since from Tables V and VI it is clear that G does not impact area.) Equations (8)–(10) show that the data rate varies with $N_q \cdot \log_2(q)$. From Tables V and VI, we see that the implementation cost in terms of area varies with q^2 (for expensive RAM) and $N_q \cdot q^2$ (for the relatively inexpensive ROM to store primitive patterns). The cost in terms of detection

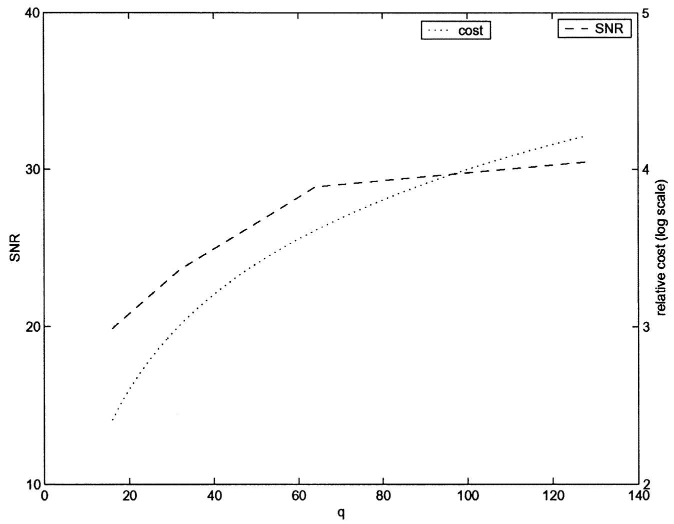


Fig. 9. Improvement in detection performance and its relative cost with increasing q.

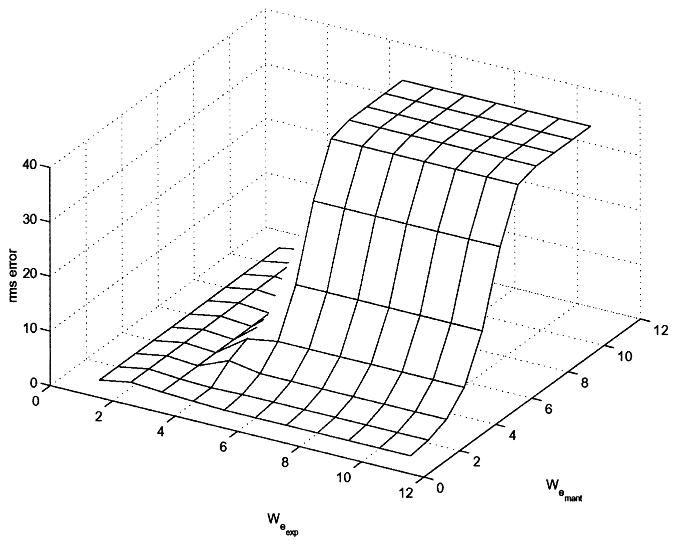


Fig. 10. Influence of floating-point precision on the perceptibility of the embedded watermark; the highpass filter of the embedder is disabled.

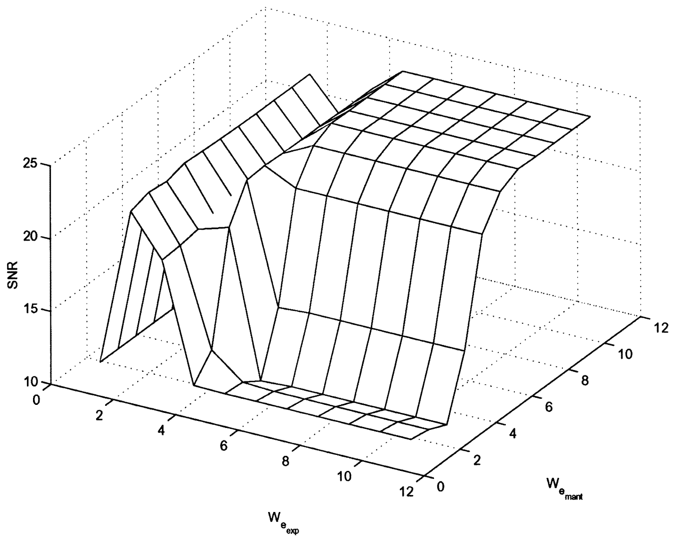


Fig. 11. Influence of floating-point precision on the detectability of the embedded watermark; the highpass filter of the embedder is disabled.

time varies with $q^2 \cdot \log_2(q) + N_q \cdot q^2$. Hence, N_q can be increased at less cost and provide more benefit than q.

To gain some more perspective on the limits to which q can be varied, we measure the performance of the detector with varying q. Fig. 9 shows the results of this simulation along with the associated cost due to q; we see that reducing q tends to impair detector performance. In addition to this limit on the degree to which q can be reduced, [17] reports that the degree that N_q can be increased is limited due to increased perceptual distortion.

2) Minimizing Perceptual Distortion: The JAWS embedder employs a highpass filter to allow the imperceptible addition of relatively strong watermarks; however, the filter is quite expensive. To gain a quantitative measure of the importance of the highpass filter, we simulate the watermarking system with and without the filter with varying global embedding depth. Without the filter, only very weak watermarks can be added; however, the detection of these watermarks is quite robust as they are present with constant strength throughout the entire image. With the filter, very strong watermarks can be imperceptibly added to portions of the image selected by the filter; the high strength of the watermark allows very robust detection as well. Considering the results of this simulation, the use of perceptual models does not seem to offer a significant benefit in the cost-benefit compromise; however, this simulation only illustrates the case for passive watermarking applications where a degradation attack is not a serious threat.

We can view practical floating-point precision effects as an "attack" on the watermarked signal. We simulate this by varying the floating-point parameters W_e[7] of the embedder (Section IV-A4 will discuss this in detail) and measuring the perceptibility of the embedded watermark and detector performance. Figs. 10 and 11 are plots of the perceptibility of the embedded watermark with the corresponding detector performance, with the high-pass filter disabled. Figs. 12 and 13 are plots of the perceptibility of the embedded watermark with the corresponding detector performance, with the highpass filter enabled. Considering the size of the "flat" region of the plot that includes the high-precision floating-point result (the high-precision result represents ideal performance), we can gain an appreciation of how resilient the system is against the attack. It is clear that with the filter enabled, the embedder is able to generate imperceptible and detectable watermarks over a much wider range of the "attack" than with the filter disabled.

3) Maximizing Robustness of Detection: In terms of implementation, the JAWS detector uses a fairly complex process to detect watermarks. Although the area complexity (due to buffers) is directly proportional to $N_q \cdot q^2$ as discussed above, the very requirement for the costly buffers is posed by the iterative and block-based computations in detection. Iterative computations require storage of intermediate values, and block-based computations require large amounts of storage to represent the

[7]In this paper, $W_{x,\mathrm{mantissa}}$ denotes the precision of the mantissa and $W_{x,\mathrm{exponent}}$ denotes the precision of the exponent, such that $W_x = W_{x,\mathrm{mantissa}} + W_{x,\mathrm{exponent}}$. The x subscript specifies whether the parameter applies to the *e*mbedder or *d*etector.

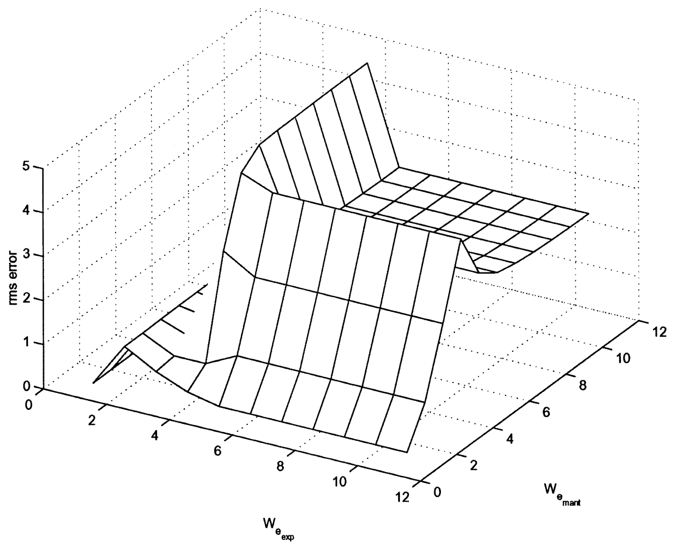


Fig. 12. Influence of floating-point precision on the perceptibility of the embedded watermark; the highpass filter of the embedder is enabled.

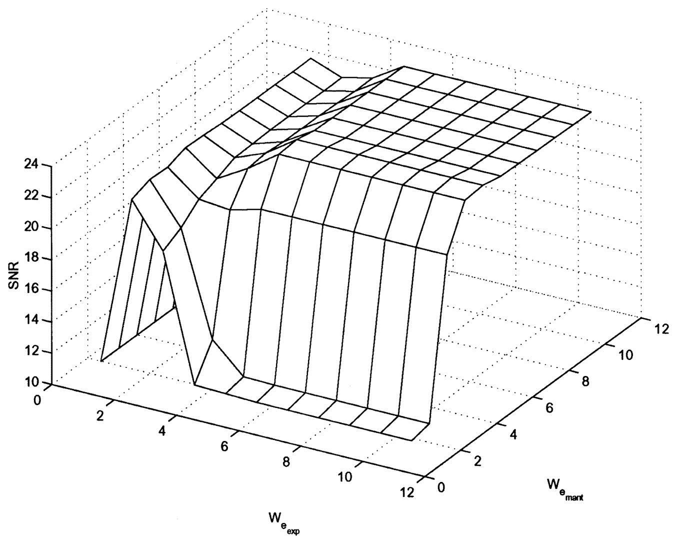


Fig. 13. Influence of floating point precision on the detectability of the embedded watermark; the highpass filter of the embedder is enabled.

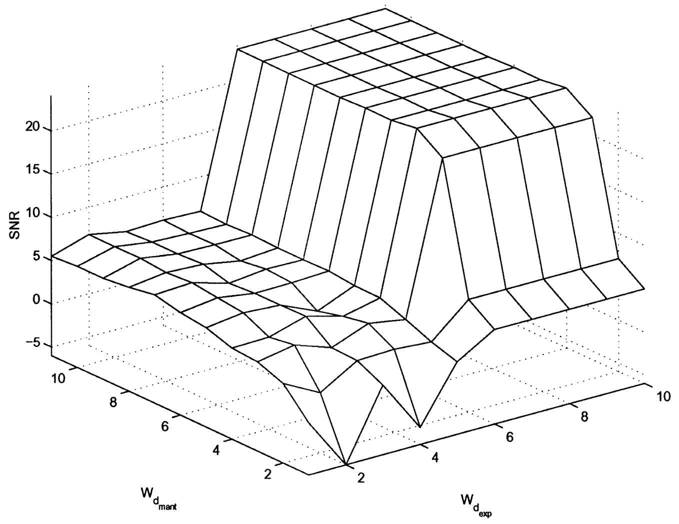


Fig. 14. Influence of floating-point precision on detection.

values (which are matrices, rather than elements). In addition to requiring blocks of storage for the computations, detection also incurs long latencies; as discussed in Section III-B, the addition of a buffer is required to maintain real-time performance in the presence of long latencies. Given that such large time and space complexity is inevitable with a JAWS-like detection sequence, [8] we can attempt to ensure that hardware utilization is maximized. From Table VI, the fold process latency is $\mathcal{O}(G\cdot f_x\cdot f_y)$; the constant of proportionality is a multiple of the frame rate R_{frame}. If the detection processes were to occur in parallel with the fold process, the detection process would complete faster than fold as it runs at the rate of the system clock f_{clock}, which is many times faster than R_{frame}. Hence, this *slack* in time can be utilized to increase the complexity of detection with no additional time complexity penalties; in light of the enormous sizes of the buffers (which are required), any additional data path and control hardware would certainly be minimal. In Section IV-B, possible improvements to exploit slack are discussed.

4) Floating-Point Representation: An important parameter that impacts the efficacy of the aforementioned algorithmic features and the area complexity is the floating-point representation, as represented by W_e and W_d.

In the embedder, W_e impacts the area of the pattern buffer, as well as the sizes of datapath elements such as multipliers and adders. To evaluate the influence of W_e on the embedder, we independently vary $W_{e,\text{mantissa}}$ and $W_{e,\text{exponent}}$ and measure the perceptibility of the watermark (see Fig. 12) and the detectability of the embedded watermark (see Fig. 13). For these measurements, we keep all other parameters in the embedder and detector constant and set the detector to use high-precision floating-point parameters. These plots show three regions with common characteristics. First, there is a region where the watermark is most imperceptible—and undetectable. This corresponds to where W_e does not have enough dynamic range to represent a strong enough watermark. Second, there is a region where detection is much better, but the watermark is very perceptible. This corresponds to where floating-point numbers behave like fixed-point numbers; the very poor granularity in the represented embedding depths causes stronger watermarks to be embedded. Finally, there is a third region where perceptibility is minimized, and robustness of detection is maximized. Moreover, these performance metrics are roughly constant throughout the region; the border of this region and the second define the point beyond which increasing W_e ceases to increase algorithmic performance.

In the detector, W_d also plays an important role since the choice of W_d will restrict the dynamic range available to the detection computations, where, due to the iterative nature, there is a possibility that the numbers will grow in size. To understand the influence of W_d on the detector, we vary $W_{d,\text{mantissa}}$ and $W_{d,\text{exponent}}$ and measure the performance of the detector on a frame where the watermark is imperceptible. In Fig. 14, two regions can be seen: one where the detector fails because W_d

[8]It is out of the scope of this work to modify the core algorithmic nature of a JAWS-like scheme.

TABLE VII
SUMMARY OF COST-BENEFIT RELATIONSHIPS

feature	influence on implementation cost	influence on algorithmic performance
$\tilde{J}_{hpf}$	(embedder) cost: one large buffer with $dim_r(\tilde{J}_{hpf})$ rows of length f_x (embedder) cost: increases latency	allows the use of stronger embedding depths while maintaining imperceptability; however, weak global embedding depths with no filtering may be acceptable
$\tilde{J}_{mf}$	(detector) cost: one large buffer with $dim_r(\tilde{J}_{mf})$ rows of length f_x	strongly improves detection performance
N_q	cost: increases number of large pattern buffers (detector) cost: latency $\propto N_q$	increases the number of bits that can be embedded in a frame ($N \propto N_q$); however, if N_q is too high, the watermark may be perceptible
q	cost: pattern buffer size $\propto q^2$ (detector) cost: fold buffer size $\propto q^2$ (detector) cost: latency $\propto q^2 \cdot log_2(q)$	increases the number of bits that can be embedded in a frame ($N \propto log_2(q^2)$); larger values of q improve detection perfomance
G		larger G improves detection performance but reduces data-rate
W_e	cost: increases buffer dataword sizes cost: increases size of data computation elements	proper selection will result in improved imperceptability and detectability of watermarks
W_d	cost: increases buffer dataword sizes cost: increases size of data computation elements	proper selection will result in better dynamic range for computations so that more robust detection can be achieved

had insufficient dynamic range and another where the detector succeeds and performs the same with further increases in W_d.

Although two separate binary representations were used for the detector and embedder of this work, all hardware internal to the detector used a uniform representation, as did all hardware internal to the embedder. However, to aggressively maximize the dynamic range in each computation, a wide variety of number representations such as various forms of fixed-point, floating-point, and block floating-point or number systems such as logarithmic, residue, and anti-tetrational, might be used to optimize the dynamic range of *each* computational element with implementation cost. Future work can undertake a more comprehensive analysis of a watermarking system with locally optimized binary number representations. In addition, investigations into more suitable number systems for watermarking algorithms can be done.

B. More General Implications

Throughout this paper, we have employed JAWS as a case-study to gain insight into "hardware-friendly" approaches for watermark performance improvement. Table VII summarizes the cost-benefit relationship among various features of the system. In this section, we attempt to generalize this cost analysis to assess the potential of various popular signal processing performance-enhancing strategies in terms of hardware implementation constraints.

1) Perceptual Models: Masking characteristics of human perception are often employed during the watermark embedding phase to increase the energy of the mark and, hence, improve robustness during detection. Although sophisticated perceptual model are available in the human factors literature, it has been demonstrated that in practice, more *ad hoc* measures may outperform in terms of watermarking reliability [6]. For video watermarking, it follows that perceptual models used for image watermarking could be combined with temporal masking measures to make more efficient use of the perceptual room available for data hiding.

From an implementation perspective, the ideal perceptual model involves the computation and buffering of few elements to obtain a metric used to adapt the watermark for imperceptible embedding. Employing sophisticated models, in which spatially global or temporal masking measures are employed, could often require buffering the video frames, which is impractical. This occurs if the energy of the watermark in a pixel of a frame at time i is dependent on the characteristics of the video at other time instants, say $i - x$ to $i + x$ for $x > 0$. A buffer is required to store all intermediate elements from $i - x$ to $i + x$ so that these elements are available together for watermark adaptation for time i. Furthermore, there is an increase in the latency of the calculation since the result for the element of time i is only available after the last element at time $i + x$ has been processed.

Buffering is also required to employ spatial masking if the entire frame must be first processed before the watermark can be adapted for embedding into that frame. A way in which buffering requirements can be reduced for this case is to assume a *temporal perceptual invariance*. That is, the perceptual characteristics of a past frame can be used to adapt the watermark for embedding in the next one. In such a scenario, the previous frame can be processed in real time to generate a perceptually adapted watermark that is embedded into the subsequent one. If the perceptual characteristics do not significantly change from frame to frame, the watermark should remain invisible. This procedure is only efficient in terms of implementation if the perceptual characteristics in a frame can be represented concisely. The perceptual parameters dictate the size of the buffer required to carry forward the values required to adapt the watermark for embedding in the next one.

2) Attack Modeling: Two main approaches have been proposed in the watermarking literature to address the issue of robustness to specific attacks. In the first case, the watermark is embedded in an attack-invariant domain so that the mark can still be reliably extracted in the face of the specified degradation [19]. In the second, a reference or template watermark is embedded along with the payload watermark to help characterize and undo the attack at the detector [9], [20].

In the former approach, transforms such as the Fourier–Mellin have been proposed [19] to make the watermark invariant to a certain class of geometric attacks. Such transforms and watermarking domains have been found to be numerically sensitive. A hardware solution to this approach could be more fruitful than software because it is possible in hardware to increase the precision of the computations at an incremental cost.[9]

The latter technique involving the embedding of a secondary watermark for attack characterization involves using the additional mark to estimate the attack during the detection phase. The estimated parameters of the attack can be used to partially undo the degradation for more reliable payload detection. Alternatively, the parameters could provide insight into ways to process the watermarked signal for more optimal detection. The advantage of using such an approach in terms of hardware implementation is that many of the components used for the payload watermark could be borrowed for this stage of the attack characterization. As long as the attack characterization is performed on spatially local parts of the video frame, excessive buffering is not required.

3) Transforms: Some early work in the field of watermarking involved selection of appropriate domains to embed the watermark. For implementation in a video watermarking application, care must be taken to guarantee the real-time performance of the system. Some techniques such as JAWS embed the watermark in the spatial domain to avoid the transform processing. Others borrow the transforms used for different processing stages such as lossy compression [21] to keep costs down. Recent work [22] has demonstrated that use of the same domain for both watermarking and lossy compression results in good performance. Thus, this solution of "borrowing" components of the codec for data hiding has potential in terms of both cost and performance.

[9]The precision can be scaled with more granularity in a hardware solution compared with software. In addition, one can be selective in varying the precision in order to optimize the cost for a given overall accuracy.

V. Conclusion

In this work, we consider hardware implementation aspects of the digital watermarking problem; at this moment, this is still largely undiscovered territory. Our aim is to bridge the gap between watermarking algorithm design and hardware implementation. For watermarking technology to gain popularity in commercial applications, it is necessary to characterize the feasibility and cost of implementation. The degree of success of digital watermarking in emerging applications is somewhat in question so it is of value to provide a realistic assessment of the practical potential of the area.

Three general directions for "hardware-friendly" development are seen, which are the following:

- *Eliminating costly hardware elements by investigating alternatives to current methods.* For example, methods of payload modulation that do not involve arbitrarily indexing into a pseudo-random sequence would greatly reduce the need for expensive hardware in the JAWS embedder.
- *Reducing the amount of expensive hardware resources by balancing the algorithmic performance obtained from a watermarking design parameter with the associated implementation costs.* For example, varying the number of primitive patterns and the primitive pattern dimensions in accordance with their cost and benefit functions can maximize overall performance and minimize cost.
- *Shifting the algorithmic "burden" to minimize cost.* For example, with JAWS, there is a very robust but expensive detection process. However, the incremental cost of adding more processing to the detector to further increase robustness is low since the overhead costs have been absorbed by the existing detection hardware. If the embedder is made less robust to reduce cost, additional robustness can be added to the detector to compensate—at low cost. Hence, in this case, shifting the burden from the embedder to the detector can minimize overall cost without degrading performance.

We hope that through effective communication between the signal processing and hardware implementation communities, more effective and practical video watermarking algorithms can be developed.

Acknowledgment

The authors wish to thank Guest Editor Dr. T. Kalker, as well as the anonymous reviewers for their very helpful comments.

References

[1] P. Wayner, *Disappearing Cryptography: Being and Nothingness on the Net*. Toronto, ON, Canada: Academic, 1996.

[2] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, S. Katzenbeisser and F. A. P. Petitcolas, Eds. Boston, MA: Artech, 2000.

[3] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures*. Boston, MA: Kluwer, 2001.

[4] I. Cox, J. Bloom, and M. Miller, *Digital Watermarking*. San Francisco, CA: Morgan Kaufmann, 2001.

[5] J. F. Delaigle, C. De Vleeschouwer, F. Goffin, B. Macq, and J. Quisquater, "Low cost watermarking based on a human visual model," in *Proc. Eur. Conf. Multimedia Applications, Services Techniques*, 1997, pp. 153–167.

[6] F. Bartolini, M. Barni, V. Cappellini, and A. Piva, "Mask building for perceptually hiding frequency embedded watermarks," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1, 1998.

[7] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual watermarks for digital images and video," *Proc. IEEE—Special Issue on Identification and Protection of Multimedia*, vol. 88, pp. 1108–1126, July 1999.

[8] G. Depovere, T. Kalker, and J.-P. Linnartz, "Improved watermark detection reliability using filtering before correlation ," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1, 1998.

[9] S. Voloshynovskiy, F. Deguillaume, and T. Pun, "Optimal adaptive diversity watermarking with channel state estimation," *Proc. SPIE, Security Watermarking Multimedia Contents III*, vol. 4314, Jan. 2001.

[10] D. Kundur and D. Hatzinakos, "Diversity and attack characterization for improved robust watermarking," *IEEE Trans. Signal Processing*, vol. 49, pp. 2383–2396, Oct. 2001.

[11] M. Ramkumar and A. N. Akansu, "Theoretical capacity measures for data hiding in compressed images," *Proc. SPIE, Voice, Video Data Commun.*, vol. 3528, pp. 482–492, Nov. 1998.

[12] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "The effect of matching watermark and compression transforms in compressed color images," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1, Oct. 1998.

[13] C. Fei, D. Kundur, and R. Kwong, "Transform-based hybrid data hiding for improved robustness in the presence of perceptual coding," *Proc. SPIE, Math. Data/Image Coding, Compression Encryption IV*, vol. 4475, July 2001.

[14] S. Baudry, J. F. Delaigle, B. Sankur, B. Macq, and H. Maitre, "Analyzes of error correction strategies for typical communication channels in watermarking," *Signal Process.*, vol. 81, pp. 1239–1250, June 2001.

[15] T. Kalker, G. Depovere, J. Haitsma, and M. Maes, "A video watermarking system for broadcast monitoring," in *Proc. SPIE, Security Watermarking Multimedia Contents*, vol. 3657, P. W. Wong and E. J. Delp, Eds., Jan. 1999, pp. 103–112.

[16] L. De Strycker, P. Termont, J. Vandewege, J. Haitsma, T. Kalker, M. Maes, and G. Depovere, "Implementation of a real-time digial watermarking process for broadcast monitoring on a TriMedia VLIW processor," *Proc. Inst. Elect. Eng. Vision, Image Signal Process.*, vol. 147, Aug. 2000.

[17] M. Maes, T. Kalker, J. Haitsma, and G. Depovere, "Exploiting shift invariance to obtain a high payload in digital image watermarking," in *Proc. Int. Conf. Image Process.*, vol. 2, Oct. 1999, pp. 7–12.

[18] J.-P. Linnartz, J. Talstra, T. Kalker, and M. Maes, "System aspects of copy management for digital video," in *Proc. IEEE Int. Conf. Multimedia Expo*, Aug. 2000, pp. 203–206.

[19] J. J. K.ÓRuanaidh and T. Pun, "Rotation, scale and translation invariant digital image watermarking," in *Proc. Int. Conf. Image Process.*, vol. 1, 1997, pp. 536–539.

[20] D. Kundur and D. Hatzinakos, "Diversity and attack characterization for improved robust watermarking," *IEEE Trans. Signal Processing*, vol. 49, pp. 2383–2396, Oct. 2001.

[21] L. Xie and G. R. Arce, "Joint wavelet compression and authentication watermarking," in *Proc. IEEE Int. Conf. Image Process.*, vol. 2, 1998.

[22] C. Fei, D. Kundur, and R. H. Kwong, "Transform-based hybrid data hiding for improved robustness in the presence of perceptual coding," *Proc. SPIE, Math. Data/Image Coding, Compression Encryption IV, With Applications*, vol. 4475, July 2001.

Nebu John Mathai (S'00) received the B.A.Sc. degree from the Division of Engineering Science (Computer Option), Faculty of Applied Science and Engineering, University of Toronto, Toronto, ON, Canada, in 2000. He is currently pursuing the M.Eng. degree with the Department of Electrical and Computer Engineering, University of Toronto, where he is working on implementing digital video watermarking algorithms.

Since May 2000, he has been a full-time digital ASIC engineer with Cogency Semiconductor, Toronto, where he is involved in the design of devices for powerline networking. His research interests include ASIC design for the acceleration of computationally expensive algorithms.

Deepa Kundur (M'99) received the B.A.Sc. degree in electrical engineering in 1993 and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering in 1995 and 1999, respectively, all from the University of Toronto, Toronto, ON, Canada.

From September 1999 to December 2002, she was an Assistant Professor with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering, University of Toronto, where she held the title of Bell Canada Junior Chair-holder in Multimedia. From 1999 to 2001, she was an Associate of the Nortel Institute in Telecommunications, Toronto, where she conducted research in the area of multimedia security. She is currently a member of Bell University Labs, Toronto, where she is involved in research focusing on enabling technologies for digital rights management (DRM). As of January 2003, she is with the Electrical Engineering Department, Texas A&M University, College Station, where she an Assistant Professor. Her research interests span the areas of multimedia security for digital rights management, covert channel analysis in communication networks, steganography, and nonlinear and adaptive communication algorithm design.

Prof. Kundur has been on the technical program committees for numerous conferences in the area of multimedia security, including the International Conference on Acoustics, Speech and Signal Processing 2002, the International Conference on Information Technology: Coding and Computing 2001, the International Federation for Information Processing and Communications and Multimedia Security Joint Working Conference 2001, and the World Digital Watermarking OlymFair. She was the 2002 recipient of the Gordon Slemon Teaching of Design Award from the ECE Department of the University of Toronto.

Ali Sheikholeslami (S'98–M'99) received the B.Sc. degree in 1990 from Shiraz University, Teheran, Iran, and the M.A.Sc. and the Ph.D. degrees in 1994 and 1999, respectively, from the University of Toronto, Toronto, ON, Canada, all in electrical and computer engineering.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, University of Toronto, and holds the L. Lau Junior Chair in Electrical and Computer Engineering. His research interests are in the areas of analog and digital integrated circuits, high-speed signaling, VLSI memory design (including SRAM, DRAM and content- addressable memories), ferroelectric memories, and VLSI for digital signal processing. He has worked with industry on various VLSI design projects in the past few years, including work with Nortel Networks, Canada, in 1994, with Mosaid Technologies in 1996, and Fujitsu Labs, Kawasaki, Japan, in 1998. He is currently supervising three active research groups in the areas of VLSI memories, high-speed signaling, and VLSI for digital signal processing. He has co-authored several journal and conference papers and received two US patents on content-addressable memories in 1998 and 1999.

Dr. Sheikholeslami has served on the Memory Subcommittee of the International Solid-State Circuits Conference since 2001.