Research Article

A Self-Reconfigurable Platform for the Implementation of 2D Filterbanks with Real and Complex-Valued Inputs, Outputs, and Filter Coefficients

Daniel Llamocca and Marios Pattichis

Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA

Correspondence should be addressed to Daniel Llamocca; dllamocca@ieee.org

Received 18 October 2013; Revised 18 March 2014; Accepted 19 March 2014; Published 4 May 2014

Academic Editor: Wieslaw Kuzmich

Copyright © 2014 D. Llamocca and M. Pattichis. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We introduce a dynamically reconfigurable 2D filterbank that supports both real and complex-valued inputs, outputs, and filter coefficients. This general purpose filterbank allows for the efficient implementation of 2D filterbanks based on separable 2D FIR filters that support all possible combinations of input and output signals. The system relies on the use of dynamic reconfiguration of real/complex one-dimensional filters to minimize the required hardware resources. The system is demonstrated using an equiripple and a Gabor filterbank and the results using both real and complex-valued input images. We summarize the performance of the system in terms of the required processing times, energy, and accuracy.

1. Introduction

Filterbank implementations in hardware require significant resources. Each filter in the filterbank is often implemented using multiply-and-add circuitry (modifiable coefficients) or by a fixed-coefficient circuitry. In what follows, we will use the term *static implementations* to describe filterbank implementations that are based on fixed hardware. A clear limitation of static implementations comes from the fact that hardware resources cannot be adjusted based on the number of coefficients or the number of filters. Furthermore, energy consumption is often a function of the implemented static hardware regardless of whether the hardware resources are actually used.

Dynamic partial reconfiguration (DPR) technology is becoming increasingly popular for addressing the aforementioned problems by time-multiplexing FPGA resources [1]. For filterbanks, DPR lets us implement only one filter at a time, which can potentially result in significant power savings over static implementation approaches.

In this paper, we introduce a self-reconfigurable implementation for a 2D complex filterbank that only requires one 1D filter to be present on the FPGA at a time. We study all the possible cases as well as providing results in terms of accuracy,

energy, and performance. Applications of complex filterbanks can be found in [2, 3].

For efficient hardware realizations, we focus on 2D FIR separable filtering that allows implementations by means of two 1D FIR filters. This separability property allows us to consider a DPR approach that keeps only one filter (row or column) at a time.

We presented some related earlier work in [4–7]. In [4], we described an efficient 1D FIR filtering system that combined the distributed arithmetic (DA) technique with DPR. An early 2D real filterbank implementation was presented in [5]. We presented a dynamic real 2D FIR filter and a dynamic complex 2D FIR filter in [6, 7], respectively. Extending this prior work, the main contributions of the current work include (i) a novel implementation for 2D separable complex filterbanks, (ii) a platform to generate 2D filter coefficients for the filterbanks, (iii) a thorough description of all the possible filterbank instances: real/complex input images, real/complex coefficients, (iv) a characterization of the filterbank hardware implementation in terms of energy, performance, accuracy, and memory requirements, and (v) a comprehensive comparison with a static implementation.

Since our 2D filterbank is implemented via dynamic reconfiguration of 1D FIR filters, our approach depends on

the use of an efficient DPR controller. Refer to [8–10] for research related to efficient DPR controller development.

The rest of the paper is organized as follows. Section 2 presents background and related work. Section 3 details the dynamic 2D complex filterbank implementation. Section 4 presents the experimental setup (specific filterbanks employed). Section 5 presents the results, and Section 6 lists the conclusions and further work.

2. Background and Related Work

Most image processing systems rely on static architectures that cannot be modified at run-time. There has been, however, a number of works that make use of DPR in their image processing systems: a design that dynamically reconfigures discrete cosine transform (DCT) modules of various sizes [9], a dynamic systolic array accelerator for Kalman and wavelet filters [11], a fingerprint image processing hardware whose stages (e.g., segmentation, normalization, and smoothing) are time-multiplexed via DPR [12], a 3D Haar wavelet transform (HWT) implemented by dynamically reconfiguring a 1D HWT core thrice [13]. The work in [14] presents a pixel processor that reconfigures input/output widths, number of pixel processed in parallel, and the single-pixel operation.

Our specific focus is on filter/filterbank implementations. We start by listing the static implementations. Multiply-and-add approaches for 2D separable filters are summarized in [15]. The work in [16] presented a design methodology that decomposes a 2D filter into 2D separable and non-separable filters, and efficiently allocates the heterogeneous resources (embedded multipliers, LUTs, FFs) on an FPGA. A filterbank design for 2D discrete wavelet transforms was presented in [17]. Gabor filterbank implementations appear in [18, 19]. The work in [19] details the implementation of a separable/nonseparable Gabor filterbank (using DSP slices) for modeling simple cells. The authors in [20] described an interesting static architecture for particle filters that, by tuning buffer parameters and interconnection switches, allowed run-time modification of the number of particles (trading off accuracy and performance) and the degree of parallelism (trading off power and performance).

As for filter/filterbank implementations using DPR, we note a 2D real filterbank based on the coefficient-only reconfiguration of a single 1D FIR Filter [5], a system that switches between a mean and a median filter via DPR [22], and a 3×3 2D FIR filter that can dynamically reconfigure its coefficients [23]. In [6], we implemented a real input, real coefficients 2D FIR filter by dynamically reconfiguring between the row and column filter. Similarly, a complex input, complex coefficients 2D FIR filter was presented in [7]. This prior research was fundamentally limited in that it lacked the generality for implementing arbitrary filterbanks, it had not been applied to both images and videos, and it did not include comparison with static architectures.

The current paper significantly extends prior work by including (i) a scalable architecture (fully parameterized VHDL core) for implementing a separable digital filter that

considers arbitrary combinations of both real and complex inputs, outputs, and coefficients, (ii) a parameterized 2D filterbank implementation that uses separable filters to support multiple scales and directions (see examples in Section 5.1), (iii) architecture validation that includes full implementations of the ubiquitous Gabor filterbanks using a number of scales and orientations to both digital images and videos of different sizes (e.g., QCIF, VGA, 720 p), (iv) an extensive comparison to a static approach and a discussion that supports the use of the proposed approach for larger image sizes.

The reconfiguration time overhead is a limiting factor in the use of DPR. Techniques to reduce this overhead include improving the access speed to the configuration memory [8], reducing the size of the reconfigurable area [24], efficient run-time task scheduling [25], and compressing the bitstreams while they are moved through the slow parts of the system [9].

3. Implementation of a Run-Time Reconfigurable 2D Filterbank for Video Processing Applications

Here, we describe the dynamic implementation of a 2D separable real/complex input, real/complex coefficients filterbank. We first describe the stand-alone real/complex input, real/complex coefficients 1D FIR filter IP. Then, we explain how to use DPR to implement a 2D filter with only one 1D filter at a time. Finally, we describe the 2D filterbank implementation.

3.1. Distributed Arithmetic Stand-Alone 1D Real/Complex Input, Real/Complex Coefficients FIR Filter. This 1D real/complex FIR core, depicted in Figure 1(a) is implemented with distributed arithmetic and it is fully parameterized. This constant-coefficient filter supports symmetric, antisymmetric, and nonsymmetric coefficients. It has a parameter “mode,” where four cases are supported:

- (i) real input, real coefficients (rIrH): this single-filter hardware was presented in [6, 21]. See Figure 1(a);
- (ii) complex input, complex coefficients (cIcH): this hardware was presented as a core in [7]. See Figure 1(d);
- (iii) real input, complex coefficients (rIcH): this hardware allows the implementation of phase-based methods that produce a complex-valued image from a real-input image [2]. See Figure 1(e);
- (iv) complex input, real coefficients (cIrH): This is the most general case that supports the implementation of Gabor filterbanks that dominate image analysis applications (see numerous applications in [2]). See Figure 1(b);

Figure 1 depicts the 1D real/complex FIR core with its I/Os and parameters for four different cases of parameter “mode.” The parameters N , B , NH , and L (shared by all the individual FIR DA cores) denote the number of coefficients, the input bitwidth, the coefficients’ bitwidth, and the LUT input size,

FIGURE 1: (a) Real/complex FIR filter intellectual property (IP) core. This is a constant-coefficient filter implemented with distributed arithmetic. The parameter “mode” allows for four implementations: (b) rlrH: real input, real coefficients, (c) clrH: complex input, real coefficients, (d) clcH: complex input, complex coefficients, and (e) rlcH: real input, complex coefficients. Refer to [21] for the inner architecture of the real input, real coefficients FIR filter.

respectively. Symmetry can be NO (nonsymmetric), YES (symmetric), and AYES (antisymmetric). The parameter OP controls the truncation/saturation scheme of the real and imaginary outputs, each with NO bits. The input (B) and output (NO) bitwidth refer to the bitwidth of each part (real and imaginary).

We let NO represent the number of output bits with NQ fractional bits. The output format is then expressed as [NO NQ]. Then, we let the fixed-point input format be [B B-1] and the coefficients’ format [NH NH-1], with normalized inputs/coefficients to be in the interval of [-1, 1]. Table 1 lists the required output format (for both real/imaginary outputs) for each “mode.” In cases where we need smaller output formats, overflow is avoided by performing LSB truncation and saturation (controlled by the OP parameter). Table 1 also lists the I/O latency (register levels between input and

TABLE 1: Largest output format and delay for the 1D filters.

Mode	Largest output format
rlrH, rlcH, clrH	$[NH + B - 1 + \lceil \log_2(N + 1) \rceil \quad NH + B - 2]$
clcH	$[NH + B + \lceil \log_2(N + 1) \rceil \quad NH + B - 2]$
Mode	REG_LEVELS: I/O delay (in cycles)
rlrH, rlcH, clrH	$\lceil \log_2(sizeI) \rceil + \lceil \log_2(M/L) \rceil + 2$
clcH	$\lceil \log_2(sizeI) \rceil + \lceil \log_2(M/L) \rceil + 3$

output) where (i) $M = \lceil N/2 \rceil$ and $sizeI = B + 1$ for symmetric/antisymmetric filters, and (ii) $M = N$ and $sizeI = B$ for nonsymmetric filters. The VHDL code of this IP core is available online at <http://www.ipvcl.org/>.

FIGURE 2: Embedded system that allows for DPR and dynamic frequency control. (a) Block diagram. The uBlaze and peripherals run at 100 MHz. (b) Frequency and PR control. The Dynamic Reconfiguration Port (DRP) is shown in blue. The external system clock is the reference clock. (c) Real/complex FIR filter processor as a PLB peripheral. Four *modes* supported: (i) real input, real coefficients, (ii) real input, complex coefficients, (iii) complex input, real coefficients, and (iv) complex input, complex coefficients. Three I/O cases supported: $B = NO = 8$, $B = NO = 16$, and $B = 8, NO = 16$. The PRR includes the filter IP and the I/O interface to the FIFOs. Two extra parameters can be controlled at run-time: input stream length (NX) and *style* (basic, centered, full, and streaming).

3.2. A Dynamically Reconfigurable System for the Real/Complex 1D FIR Filter. The constant-coefficients 1D real/complex FIR filter is turned into a flexible FIR filter via DPR. Here, we allow for the entire filter to be reconfigured, so that we

can modify every parameter and thus explore many different realizations.

Figure 2(a) shows an embedded system that allows for full reconfiguration of the real/complex FIR filter parameters.

FIGURE 3: Input/output interface to the FIFOs. Depending on the bitwidth of the I/Os (three cases supported: $B = NO = 8$, $B = NO = 16$, and $B = 8$, $NO = 16$), and depending on the *mode* (rIrH, rIcH, cIrH, and cIcH), the I/O Interface to FIFOs varies. (a) Real input, real coefficients: the three supported cases of B and NO are shown. (b) Real input, complex coefficients: the three supported cases of B and NO are shown. (c) Complex input, real/complex coefficients: the three supported cases are shown.

The logic outside the PRR is called the static region. The embedded interface containing the 1D real/complex FIR core is called “real/complex FIR filter processor.” This filter processor contains the FIR core along with the glue logic for PLB interfacing.

The real/complex FIR filter processor and the processor interact via a 32-bit PLB bus. The bitstreams (filter realization data) and input data are initially stored in the compact flash (CF) card. At run-time the main memory holds input and output data and the bitstreams. To perform DPR, the bitstreams in memory are streamed to the ICAP. The filter processor supports burst transfers, hence the use of the Xilinx Central DMA core. The ethernet link allows for data communication, throughput measurements, and system status report. Some interfacing is included in the PRR so as to allow for changes to our filter core I/O bitwidth during DPR.

The real/complex FIR filter processor is a hardware IP. It includes the parameters of the real/complex FIR filter IP along with two new parameters: input stream length (NX) and filter output choice (*style*). As for the I/O bitwidth, the 32-bit PLB interface of this FIR filter processor only supports three cases: $B = NO = 8$, $B = NO = 16$, and $B = 8$, $NO = 16$.

Figure 2(c) depicts the real/complex FIR filter processor in detail. We clock the logic inside the PRR at $clkfx$, while the rest is clocked at PLB_clk . The FIFOs (iFIFO, oFIFO) keep the PLB_clk and $clkfx$ clock regions separated.

With NX as the input stream length, the N -coefficient filter peripheral generates $NX + N - 1$. The filter processor offers four filter output choices (*style*): (i) basic: first NX output samples, (ii) centered: NX samples in the range $[N/2] + 1 : NX + [N/2]$, (iii) full: all of the $NX + N - 1$ samples, and (iv) streaming: $NX = \infty$, infinite output samples.

Figure 3 depicts the input/output interface between the filter IP and the FIFOs. The hardware implementation of this interface depends on (i) the parameter *mode* and (ii) the I/O allowed cases ($B = NO = 8$, $B = NO = 16$, or $B = 8$, $NO = 16$).

As for dynamic frequency control, the dynamic reconfiguration port (DRP) of the multimode clock manager (MMCM) inside Virtex-6 FPGAs can adjust the frequency at run-time without loading a new bitstream. Figure 2(b) depicts the frequency and partial reconfiguration (PR) core. The parameter O_0 controls frequency at run-time. Also, the “*PR_done*” signal is asserted after a DPR process is completed, so that the PRR flip flops are cleared [21]. The PRR outputs toggle during DPR, but it does not matter as the FIFO is reset as soon as “*PR_done*” is asserted.

This embedded interface lets us dynamically switch from a real/complex filter to another real/complex filter. We next explain how to implement a 2D real/complex separable FIR filter based on this feature.

3.3. Dynamic Implementation of a 2D Real/Complex Separable FIR Filter. In the embedded system of Figure 2, a 2D real/complex separable FIR filter (represented by 2 bitstreams: row and column filter) is implemented by cyclically swapping the row filter with the column filter via DPR [6]. This approach only requires resources for one 1D FIR filter at a time and can yield significant savings over static implementations of 2D filters. In addition, the following needs to be considered.

- (1) Usually, the output image size has to match the input image size. The real/complex FIR filter processor sets its parameter *style* = *centered* (center of the convolution output).
- (2) The column filter and row filter differ not only in the coefficients, but also on other parameters (I/O format, number of coefficients, etc.), requiring a reconfiguration of the entire 1D filter.
- (3) Unless the image is square, the length of the input signal in the row filtering case is different than in the column case. This means that we have to change NX to match the input size.
- (4) Two reconfigurations are performed per frame. The row filter processes and stores the result in a sequential row-by-row fashion, but the column filter does so in a sequential column-by-column fashion. As a result, the row-filtered output images need to be transposed prior to column filtering.
- (5) This cyclic swapping between the row and column filter usually implies a reconfiguration rate of two reconfigurations per frame. Besides input and output frames, we require memory to store the intermediate row-filtered frame.

We now have to deal with 2D real/complex input, 2D real/complex coefficients, and a 2D real/complex output. Thus, we must establish what 1D row and column filters (*mode*) we should be using in each case.

We represent the complex input image by: $I = a + bj$. We denote the separable complex coefficients by $W = h * v$, $h = c + dj$, $v = f + gj$. The complex output image then results are $Y = I * W = (I * h) * v$. Figure 4 presents a conceptual implementation of the four cases we might encounter, each with a unique set of applications.

- (i) *Real image, real 2D coefficients* (RIRH): in this simple case, the row filter and column filter are of type rIrH (real input, real coefficients).
- (ii) *Complex image, real 2D coefficients* (CIRH): here the row and column filters are of type cIrH (complex input, real coefficients).
- (iii) *Real image, complex 2D coefficients* (RICH): here the row filter must be of type rIcH (real input, complex coefficients), while the column filter must be of type cIcH (complex input, complex coefficients).
- (iv) *Complex image, complex 2D coefficients* (CICH): here, both the row and column filter must be of type cIcH (complex input, complex coefficients).

The input bitwidth of the 2D filter matches the input bitwidth of the row filter. The output bitwidth of the 2D filter matches the output bitwidth of the column filter.

Based on the three allowed I/O cases for B and NO , a common approach is to match the input bitwidth of the column filter with the output bitwidth of the row filter and to set NO to be the same for the row and column filters. Example: $B = 8$, $NO = 16$: row filter ($B = 8$, $NO = 16$), column filter ($B = 16$, $NO = 16$).

3.4. 2D Real/Complex Filterbank Implementation. To implement filterbanks, we extend the cyclic approach by sequentially applying DPR to move along each 2D filter's row and column bitstreams. No extra reconfiguration penalty exists since we are always reconfiguring twice per frame. The execution time grows linearly with the number of 2D filters.

Figure 5 depicts the 2D separable filterbank implementation on an FPGA. Figure 5(a) shows how a set of filterbanks can be stored in memory. Figure 5(b) provides a conceptual implementation of the filterbank using DPR. Figure 5(c) shows how the 1D filters are loaded cyclically on the FPGA. We refer to [5] for early work on a 2D real input, real coefficients filterbank example (coefficient-only reconfiguration).

4. Energy, Performance, and Accuracy Measurements

In this section, we describe how we compute and or measure these quantities that will let us evaluate the filterbank hardware implementation.

4.1. Performance Calculations. We report the performance of the 2D filterbank from the IP angle, that is, assuming a continuous streaming of the input signal and maximum reconfiguration speed. We consider the embedded system as a generic test-bed. In general, the embedded system performance depends on many factors that are specific to each particular system (e.g., cache size, processor type, bus usage, etc.).

4.1.1. Filter Processing Time. A 2D filter operates on a stream-by-stream basis. After each stream (a single row or column) is processed, the register chain in the FIR filter is cleared, ready for a new stream. Let the lengths of the input streams be defined as NXr and NXc for the row filter and column filter, respectively (input video frame of size NXc rows by NXr columns). The processing time is then given by

$$t_{\text{rows}} = \left(NXr + \left\lceil \frac{Nr}{2} \right\rceil + \text{REG_LEVELS}_r \right) \times NXc \times T_{\text{cycle}},$$

$$t_{\text{cols}} = \left(NXc + \left\lceil \frac{Nc}{2} \right\rceil + \text{REG_LEVELS}_c \right) \times NXr \times T_{\text{cycle}}, \quad (1)$$

where Nr and Nc represent the number of row and column filter coefficients, respectively. REG_LEVELS_r and REG_LEVELS_c denote an initial latency (see Table 1) and T_{cycle}

FIGURE 4: 2D separable complex convolution. (a) Real image, real coefficients. (b) Complex image, real coefficients. (c) Real image, complex coefficients. (d) Complex image, complex coefficients. In most cases, the row filter is of similar type as the column filter, except for the case of real image, complex coefficients.

is the clock period. Here, note that $\lfloor Nr/2 \rfloor$, $\lfloor Nc/2 \rfloor$ cycles are needed to provide centered row/column convolution outputs.

4.1.2. Reconfiguration Time. Based on the PRR bitstream size and the reconfiguration speed, the reconfiguration time results are

$$t_{\text{reconfig}} = \frac{\text{PRR size (bytes)}}{\text{Rec.speed (bytes per sec)}}. \quad (2)$$

The maximum reconfiguration speed (400 MB/s in Virtex-4) is achieved with a direct link between the ICAP port and the BRAMs (local memory inside the FPGA) that holds the bitstreams [8, 24]. The approach is limited by the available BRAMs. While BRAMs are scarce in Virtex-4 devices, there are far more BRAM resources in the (newer) Virtex-6 devices [26], making the approach more practical. We will use the maximum reconfiguration speed in our performance results.

4.1.3. 2D Filterbank Performance. Filtering a frame of $NXr \times NXc$ pixels entails row, column filtering, and two reconfigurations. The processing time of a filterbank with “ n ” 2D filters is given by

$$t_{\text{fbank}} = (t_{\text{rows}} + t_{\text{cols}} + 2 \times t_{\text{reconfig}}) \times n. \quad (3)$$

The filterbank performance (in frames per second) results are

$$fps_{\text{fbank}} = \frac{1}{(t_{\text{rows}} + t_{\text{cols}} + 2 \times t_{\text{reconfig}}) \times n}. \quad (4)$$

In (3), the transposition time for the row-filtered image is not considered. The transposition operation is a software routine in the embedded platform (e.g., see [6]). For completeness, we report the transposition time in the embedded results section.

4.2. Energy Measurement Estimation. Here, we report the energy spent by the 2D filterbank and the reconfiguration process for processing a single frame. Energy provides more information than power since 2D filtering involves several stages (row filtering, column filtering, and reconfiguration) that draw different amounts of power.

4.2.1. Power Measurements. Power inside the FPGA is drawn by the following power supply rails: (i) internal supply voltage VCCINT with current ICCINT and (ii) auxiliary supply voltage VCCAUX with current ICCAUX. The output supply

FIGURE 5: 2D complex filterbank implementation based on a single 1D Filter. (a) The memory can hold a group of distinct 2D filterbanks (each 2D filterbank might have a different number of 2D filters). (b) Conceptual implementation of a 2D Filterbank: either the row or column filter are active in the FPGA, but once the 2D filtering is complete, the dynamic manager loads the next row, column filter pair. (c) Example of operation for a 2D filterbank with three 2D filters. Each frame is filtered by each of the 2D filters in the 2D filterbank, before moving on to the next frame.

power is not considered since it is only associated with the power drawn by the external pins.

Power at each supply rail is divided into static and dynamic power. The *static power* is divided into *device static power* and *design static power*. These quantities are thoroughly summarized in [14].

We report the energy spent by the real/complex FIR peripheral and the reconfiguration process for processing one frame. For power consumption of the PLB peripheral ($P_{\text{row}}, P_{\text{col}}$), we use Xilinx Power Analyzer (XPA) at 25°C for $V_{\text{CCINT}} = 1.0\text{ v}$ and $V_{\text{CCAUX}} = 2.5\text{ v}$ rails. XPA provides an accurate estimate based on simulated switching activity of the place-and-routed circuit and exact utilization statistics [27]. We report the power as the sum of the dynamic power and design static power, ignoring the device static power (fixed quantity that depends on the operating conditions and device size: 1.5 W for the XC6VLX240T FPGA at 25°C):

$$P_{\text{row/col}} = V_{\text{CCINT}} \times (\text{ICCINT} - \text{ICCINTQ}) + V_{\text{CCAUX}} \times (\text{ICCAUX} - \text{ICCAUXQ}). \quad (5)$$

The TI power adaptor inside the ML605 board can monitor the current of the FPGA voltage rails (via I^2C). In [7], we found that, during DPR, only the V_{CCINT} supply current increased by 50 mA. The reconfiguration power increase is then $V_{\text{CCINT}} \times 50\text{ mA}$.

4.2.2. Energy per Frame. The total energy per frame is the sum of the energy spent by the following processes: (i) row filtering, (ii) turning a row filter into a column filter via DPR, (iii) column filtering, and (iv) turning a column filter into a row filter via DPR. We apply this process for each 2D filter in the filterbank. Using the power and the processing times of each process, the energy per frame (epf) is given by

$$\text{epf} = [P_{\text{rows}}t_{\text{rows}} + P_{\text{cols}}t_{\text{cols}} + (P_{\text{reconfig-row}} + P_{\text{reconfig-col}}) \times t_{\text{reconfig}}] \times n. \quad (6)$$

4.3. Accuracy Measurements. We measure the accuracy of the filtered images (magnitude response) using the peak signal-to-noise ratio (PSNR) between the fixed-point filterbank

FIGURE 6: 2D Separable real-coefficients equiripple filterbank. The filterbank is composed of four 2D separable real-coefficient Filters. Note that filter #3 appears darker because its band-energy is distributed over four quadrants, as opposed to the spread over two quadrants for filters #2 and #4. We have also kept the maximum gain for the lowpass filter (#1) to be at the same levels as #2 and #4.

output from the FPGA and the 2D filterbank output using double floating point precision. Here, we compare the output of each 2D filter.

5. Experimental Setup

This section describes the filterbanks selected as well as the test images. A 2D FIR filterbank is represented by its associated parameters (for each 2D filter), $2 \times n$ bitstreams (row and column filters), and its energy-performance-accuracy values.

5.1. Generation of the Set of 2D Filterbanks. We test our system by using the four cases our system can support: (i) real image, real coefficients, (ii) real image, complex coefficients, (iii) complex image, real coefficients, and (iv) complex image, complex coefficients. To this end, we selected two filterbank types.

Table 2 summarizes the characteristics of filterbanks we selected: equiripple (32×32 real coefficients) and Gabor (16×16 complex coefficients). Figure 6 depicts the frequency response of the equiripple Filterbank, while Figure 7 depicts that of the Gabor Filterbank. The Gabor filterbank is defined in terms of Gabor functions given by

$$h(x, y) = g(x, y) \times \exp[2\pi j(Ux + Vy)], \quad (7)$$

$$g(x, y) = \frac{1}{2\pi\sigma^2} \times \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

with U, V denoting the central frequency of the Gabor filter, and $g(x, y)$ is the Gaussian function.

The filters $h(k_1, k_2)$ are sampled versions of $h(x, y)$. The coefficients' real part is symmetric, while the imaginary part is antisymmetric. The 2D Gabor and equiripple filterbank

FIGURE 7: 2D Separable complex-coefficients Gabor filterbank. The filterbank is composed of eighteen 2D separable complex-coefficient filters. The filterbank uses 6 orientations with 3 Gabor filters along each orientation (ignoring the low-pass filter).

TABLE 2: Characteristics of the 2D separable filterbanks.

Characteristic	Filterbank	
	Equiripple	Gabor
Number of filters	4	18
Filters' coefficients	Real	Complex
Coefficients per filter	32	16
Coefficients symmetry	Nonsymmetric	Symmetric (real part), antisymmetric (imaginary part)
Bits per coefficient	16	16 (real and imaginary)
Notes		3 scales, 6 directions, $\sigma_x = \sigma_y = 2$

examples considered here is motivated from the wide range of applications [2, 28, 29].

The coefficients are generated in a text file and loaded into the real/complex FIR filter core as a parameter.

We report results based on 8-bit input images. For complex images, the real and imaginary parts are assumed to have 8-bits each.

The input bitwidth of the 2D filters is set to be 8 and the output bitwidth to be 16. In the case of complex filter, these bitwidths apply to both the real and imaginary parts. In the context of the 32-bit PLB peripheral, this means that row filters have $B = 8$, $NO = 16$, and column filters have $B = NO = 16$. We also constrain the output to be in the same range as the input: $[-1, +1]$ ($NQ = NO - 1$). For the outputs, we use an arithmetic mode that uses truncation of the LSB followed by saturation. We also set $L = 4$, $NH = 16$, and N to be the same for both row and column filters.

5.2. Generation of Real/Complex Images and Video. We selected "lena" as the real image and "foreman" as the real

FIGURE 8: Test images (*lena*: 8-bit, CIF resolution) and test filterbanks (equiripple, Gabor). (a) Real image and its spectrum. (b) Complex analytical image and its spectrum. (c) 2D separable Gabor filterbank. (d) 2D separable equiripple filterbank.

video. Both use 8-bit pixels at CIF (352×288) resolution. Since we deal with CIF-sized frames, we need $NXc = 352$, $NXr = 288$ (PLB peripheral parameters).

For a complex image, we consider a 2D Hilbert extended image that is to be processed through a bank of bandpass filters: this is called the analytical image. The 2D Hilbert extended image is computed using [29]

$$I_C(k_1, k_2) = I(k_1, k_2) + jH(I(k_1, k_2)), \quad (8)$$

where $H(I(k_1, k_2))$ is a 2D Hilbert transform applied along each row of the real-valued image $I(k_1, k_2)$, and I_C is a complex-valued image. Figures 8(a) and 8(b) show the real image *lena* and the analytical (complex-valued) image *lena*. Figures 8(c) and 8(d) depict the Gabor and equiripple filterbanks.

The following is the list of cases that we will test:

- (i) RIRH: real image (*lena*), real coefficients (equiripple filterbank);
- (ii) CIRH: complex image (analytical *lena*), real coefficients (equiripple filterbank);
- (iii) RICH: real image (*lena*), complex coefficients (Gabor filterbank);

- (iv) CICH: complex image (analytical *lena*), complex coefficients (Gabor filterbank).

Note that the 300-frames “*foreman*” video undergoes the same process as the “*lena*” image. We generate an analytical image out of each “*foreman*” frame.

We also control the frequency of operation of the real/complex FIR filter PLB peripheral. Figure 2(b) shows how the frequency of “*clkfx*” can be modified at run time. We selected $M = 8$, $D = 2$, and five different frequencies (MHz): 100 ($O_0 = 8$), 66.66 ($O_0 = 12$), 50 ($O_0 = 16$), 40 ($O_0 = 20$), and 33.33 ($O_0 = 24$).

5.3. Platform Testing Scheme. The system is implemented in the ML605 Xilinx Dev. Board that houses a XC6VLX240T Virtex-6 FPGA. It also comes with 512 MB of DDR3RAM. The soft-core MicroBlaze and peripherals run at 100 MHz.

6. Results and Analysis

6.1. Hardware Resource Utilization

6.1.1. 1D Real/Complex FIR Filter Processor IP Utilization. Table 3 presents the resource breakdown as required by

TABLE 3: 1D real/complex FIR filter resource utilization (Virtex-6 XC6VLX240T). Equiripple Case: 32 real-valued nonsymmetric coefficients. Gabor case: 16 complex-valued symmetric/antisymmetric coefficients.

2D filter	1D filter	Slice	(%)	Regs	(%)	LUT	(%)
RIRH	Row	715	2%	2146	1%	2123	1%
	Column	1288	3%	4282	1%	4230	2%
CIRH	Row	1288	3%	4199	1%	4175	2%
	Column	2420	6%	8461	3%	8395	5%
RICH	Row	263	1%	666	1%	730	1%
	Column	1373	3%	4263	1%	4830	3%
CICH	Row	478	1%	1302	1%	1421	1%
	Column	1373	3%	4263	1%	4830	3%

TABLE 4: 1D real/complex FIR filter—PRR sizes.

2D filterbank	PRR size (# of slices)	Bistream size (row/col)
Equiripple, real image (RIRH)	$41 \times 38 = 1558$	290 KB/290 KB
Equiripple, complex image (CIRH)	$60 \times 38 = 2280$	415 KB/415 KB
Gabor, real image (RICH)	$40 \times 38 = 1520$	219 KB/219 KB
Gabor, complex image (CICH)	$40 \times 38 = 1520$	219 KB/219 KB

the row and column filters (this is the complete PLB peripheral, PRR included) for each of the four considered cases.

6.1.2. Size of Reconfigurable Area. The PRR size is set to the largest possible filter realization, which in this case is given by the column filter. Hence, the PRR size is the same for the cases RICH and CICH (the column filter is always cICH), while the PRR size is different for the cases RIRH and CIRH (the column filters are rIRH and cIRH, resp.). Table 4 shows the PRR sizes in number of slices and bitstream size.

6.2. Embedded System Results

6.2.1. Embedded System Resource Utilization. Table 5 shows the hardware resource utilization of the embedded FIR filtering system that can implement all the cases. The PRR includes a 1D real/complex FIR filter and the PLB interface. The largest realization occupies 2160 slices (98% of the allocated space for the PRR). This is slightly lower than what Table 4 reports since the results are obtained by compiling the embedded system (also we only include the part inside the PRR). For transposition of the row-filtered image, we have 9.53 ms for a CIF frame (output bit-width = 16).

For reconfiguration, we used the Xilinx ICAP core, obtaining average reconfiguration speed of 16.28 MB/s. Significant improvements can be obtained with the use of custom-built controllers as reported in [18].

6.2.2. Embedded System Performance. Performance metrics are listed in Table 6. The performance is heavily affected by the transposition time and the reconfiguration time (using the 16.28 MB/s speed). Results are acceptable in the sense that

TABLE 5: Embedded FIR filtering system resource utilization (Virtex-6 XC6VLX240T). Case: equiripple 2D filterbank: Column filter: $N = 32$ (nonsymmetric), $NH = 16$, $NO = 16$.

Module	Slice	(%)	Regs	(%)	LUT	(%)
PRR (column filter)	2160	6%	8461	3%	8398	5%
Static region	4701	12%	9696	3%	8054	5%
Overall	6861	18%	18157	6%	16452	10%

TABLE 6: Embedded system performance of the dynamic system at 100 MHz. Frame size: CIF. t_{rows} , t_{cols} , t_{trans} , t_{fbank} , and t_{reconf} are given in ms. fps: frames per second.

2D Filterbank	Embedded system performance					fps
	t_{rows}	t_{cols}	t_{trans}	t_{reconf}	t_{fbank}	
RIRH	2.11	2.72	9.53	17.81	199.92	5.00
CIRH	2.98	3.53	13.47	25.49	283.84	3.52
RICH	2.65	3.75	13.47	13.45	841.86	1.18
CICH	3.55	3.76	13.47	13.45	858.24	1.16

TABLE 7: 2D real/complex FIR filter—memory requirements.

2D filterbank type	Bitstream size	# of filters (# of bitstreams)	Required memory
Equiripple, real image	290 KB	4 (8)	2.32 MB
Equiripple, complex image	415 KB	4 (8)	3.32 MB
Gabor, real image	219 KB	18 (36)	7.88 MB
Gabor, complex image	219 KB	18 (36)	7.88 MB

TABLE 8: Processing time (ms) as a function of the frequency for the four 2D filterbank types. Frame size: CIF (352×288).

Filterbank	Frequency (MHz)				
	33.33	40	50	66.66	100
RIRH	30.5534	26.1912	21.8290	17.4667	13.1045
CIRH	30.5534	26.1912	21.8290	17.4667	13.1045
RICH	140.9062	121.8168	102.6734	83.5301	64.3867
CICH	152.3657	133.1964	114.0271	94.8578	75.6886

TABLE 9: Frames per second (fps) as a function of the frequency for the four 2D filterbank types. Frame size: CIF (352×288).

Filterbank	Frequency (MHz)				
	33.33	40	50	66.66	100
RIRH	32.7295	38.1808	45.8107	57.2517	76.3098
CIRH	32.7295	38.1808	45.8107	57.2517	76.3098
RICH	7.0942	8.2090	9.7396	11.9717	15.5312
CICH	6.5632	7.5077	8.7698	10.5421	13.2120

we are processing video images using a filterbank, not a single filter. Among the suggested improvements to the embedded system, we note (i) reducing the reconfiguration time by using a custom-built ICAP controller (mentioned in Section 2), (ii) improving the algorithm for fast transposition in memory, and (iii) increasing the DMA burst length (currently 16) and the FIFOs' depth (FIFOs are shown in Figure 2).

FIGURE 9: Energy per frame (mJ) for processing a CIF-sized frame as a function of the frequency and the 2D filterbank type.

6.2.3. Memory Requirements. The 2D filters of a filterbank are represented by a set of bitstreams. The larger the filterbank is, the larger the memory overhead is. Table 7 lists the memory requirements for the filterbank implementations.

6.3. Energy-Performance-Accuracy Results. Tables 8 and 9 list the processing time and frames per second of the filterbanks, respectively (using a CIF frame size). These values are listed as a function of the frequency.

Figure 9 shows the energy consumed by the PLB peripheral and the reconfiguration process for processing a single CIF frame. The energy per frame is shown as a function of the frequency for every type of 2D filterbank. We note that there is a decrease in processing time as the frequency operation increases. As a result, overall, we observe a slight decrease in the total amount of required energy to process each frame. The energy savings occur despite the fact that the increased operating frequency results in higher dynamic power.

Table 10 lists the accuracy (PSNR) values for the four considered cases for *lena* image (CIF). Notice that the accuracy values are very high (~ 90 dB); this is due to the fact that we are working with 16-bit output images. In addition, we streamed the *foreman* video sequence (CIF) through all the filterbanks; the accuracy results are shown in Figure 10. We can see that the (high) values are very close to those of *lena*'s.

Finally, Figure 11 shows the actual FPGA outputs (magnitude, magnitude spectra, and real component) of the equiripple filterbank for an input real image (*lena*). Figure 12 shows the same equiripple filterbank outputs for an input complex image (*analytical lena*). Figures 13, 14, and 15 show the actual FPGA outputs (magnitude, magnitude spectra, and real component) of the Gabor filterbank for an input real

TABLE 10: Accuracy Values (db) for the 2D filterbanks. For the filter number, refer to Figure 6 and Figure 7. Test Image: *lena* (CIF).

Filter number	Gabor filterbank— complex coefficients		Equiripple filterbank— real coefficients	
	Real input image (RICH)	Complex input image (CICH)	Real input image (RIRH)	Complex input Image (CIRH)
1	97.8303	97.8109	95.0615	100.7943
2	98.0088	97.9902	73.9881	97.5084
3	95.8487	96.2240	94.8124	97.8973
4	94.4681	94.3506	95.8114	101.3000
5	97.8652	97.8665		
6	96.8144	96.9237		
7	92.4580	94.3768		
8	96.7251	96.7780		
9	92.5409	93.3594		
10	91.2001	92.3059		
11	95.7933	96.0248		
12	98.6405	98.1494		
13	97.1755	96.3996		
14	96.6219	95.7369		
15	94.6048	94.0747		
16	94.3707	96.2460		
17	91.6033	92.4280		
18	92.0106	92.7980		

image (*lena*). Figures 16, 17, and 18 show the same Gabor filterbank results for an input complex image (*analytical lena*).

6.4. Visual Assessment of the Results. We present a visual assessment of the results to demonstrate the effectiveness of the proposed approach. In Figures 13–18, we present extensive results on the *lena* image based on the use of a Gabor filterbank. The examples allow the readers to assess the performance of our methods by using a popular image with a widely used filterbank as documented in [2]. Furthermore, to allow for comparisons to more recent methods, we also include results from the use of equiripple filterbanks as documented in [28].

To analyze the results, we note that the magnitude outputs of each Gabor filter represents an estimate of the instantaneous amplitude (IA) as documented in [2, 28].

Large values of the IA indicate the presence of significant frequency components that fall within the passband of the corresponding filter. The IA outputs are given in the left column of each example in Figures 13–18. For these figures, in the images presented in the right column we have the real part of the output. These images represent the frequency-modulation (FM) components that capture the spatial frequency variations in the image [2, 28]. The passband of each filter can be visualized in the middle columns of Figures 13–18. Here, we note that the image spectrum that falls within the passband is extracted from the rest of the spectrum.

FIGURE 10: Accuracy values for the filterbanks. (a) Input real-valued video (CIF). (b) Accuracy per frame: complex-valued video, real-valued coefficients (equiripple filterbank). (c) Accuracy per frame: real-valued video, complex-valued coefficients (Gabor filterbank). (d) Accuracy per frame: complex-valued video, complex-valued coefficients (Gabor filterbank).

We present two sets of Gabor filterbank results on the *lena* image. The first set is based on the use of a real-valued *lena* image without prefiltering with the directional Hilbert transform as shown in Figure 8(a). These results are shown in Figures 13–15. The second set of results come from the use of a complex *lena* input image that is formed from the output of the directional Hilbert transform as shown in Figure 8(b). This set of results is shown in Figures 16–18.

A visual evaluation of the IA for each filter in Figures 16–18 clearly indicates that both the direction and magnitudes of each frequency component are well represented in the IA outputs. For the lower frequency components, the real-valued output images presented in the right-hand column

reconstruct the overall image structure with strong emphasis on the image structure that exhibit the frequency components that fall within the passband of each filter. The results indicate that Gabor filters with higher frequency magnitude components provide very good spatial localization of the underlying frequency components. Similar observations apply to the outputs from the real-valued *lena* image as shown in Figures 13–15. However, it is clear that the preapplication of the directional Hilbert transform will provide better results, as indicated by the finer localization of the frequency components.

We also present results based on the equiripple filter design in Figure 11 (real input) and Figure 12 (complex input).

FIGURE 11: 2D separable real-valued coefficients equiripple filterbank. Results with a real-valued input image (RIRH). (a) Output image. (b) Magnitude spectrum of output image.

FIGURE 12: 2D separable real-valued coefficients equiripple filterbank. Results with a complex-valued input image (CIRH). (a) Output image (Magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

The results indicate strong spatial localization of a wide range of frequency components (e.g., see first row for both figures).

We also present filterbank results on the *foreman* video in Figure 10. From the results, it is clear that we have very high accuracy since PSNR remains above 85 dB for all 300 video frames. Furthermore, it is clear that accuracy is a function of the frequency band. For example, the highest, diagonal frequency components captured in Filter 3 of the equiripple design (Figure 10(b)) maintain much higher accuracy than the vertical frequency components of Filter 2. A comparison

of the Gabor filterbank results between the real and complex-valued input video frames clearly demonstrate the consistency of prefiltering with the directional Hilbert transform.

6.5. Comparison with a Static Implementation. Here, we provide a comparison of our dynamic filterbank implementation with an efficient static implementation (no DPR). The static implementation is based on a multiply-and-add 1D FIR core that implements the row and column filters by loading new

FIGURE 13: 2D separable complex-valued coefficients Gabor filterbank. Results with a real-valued input image (RICH). Filters: 1 to 6. (a) Output image (Magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 14: 2D separable complex-valued coefficients Gabor filterbank. Results with a real-valued input image (RICH). Filters: 7 to 12. (a) Output image (Magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 15: 2D separable complex-valued coefficients Gabor filterbank. Results with a real-valued input image (RICH). Filters: 13 to 18. (a) Output image (Magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 16: 2D separable complex-valued coefficients Gabor filterbank. Results with a complex-valued input image (CICH). Filters: 1 to 6. (a) Output image (magnitude), (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 17: 2D separable complex-valued coefficients Gabor filterbank. Results with a complex-valued input image (CICH). Filters: 7 to 12. (a) Output image (magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 18: 2D separable complex-valued coefficients Gabor filterbank. Results with a complex-valued input image (CICH). Filters: 13 to 18. (a) Output image (magnitude). (b) Magnitude spectrum of output image. (c) Output image (real part).

FIGURE 19: Multiply-and-add FIR filter implementations for real input, real coefficients: nonsymmetric filter (a) and symmetric and antisymmetric filter (b). The adder tree is a pipelined architecture with an I/O latency of $\lceil \log_2 M \rceil$ cycles. Refer to [4] for the adder tree details.

TABLE 11: 1D real/complex multiply-and-add FIR filter resource utilization (Virtex-6 XC6VLX240T). Equiripple Case: 32 real-valued nonsymmetric coefficients. Gabor case: 16 complex-valued symmetric/antisymmetric coefficients.

2D filter	1D filter	Slice	(%)	Regs	(%)	LUT	(%)
RIRH	Row	1735	4%	2398	1%	6196	4%
	Column	3345	8%	3718	1%	11935	7%
CIRH	Row	3350	8%	4338	1%	12138	8%
	Column	6375	16%	6064	2%	23581	15%
RICH	Row	1013	2%	709	1%	3666	2%
	Column	1646	4%	1298	1%	6406	4%
CICH	Row	2007	5%	1579	1%	7266	4%
	Column	3273	8%	2414	1%	12765	8%

coefficients into registers (instead of performing DPR). The 1D multiply-and-add FIR core is shown in Figure 19. Note that we show the case rIrH; the other cases (cIrH, rIcH, cIcH) are generated as in Figure 1. The I/O latency is given by $\text{REG_LEVELS} = \lceil \log_2 M \rceil + 3$ for the cIcH mode and $\text{REG_LEVELS} = \lceil \log_2 M \rceil + 2$ for the cIrH, rIcH, and rIrH modes. Prior to each 1D filtering, we need to feed the M coefficients into a register chain. As in the dynamic filterbank implementation, we have row filtering, transposition, and column filtering.

Table 11 shows the resource consumption of the static implementation using the same parameters as in the case of the dynamic system: row filters with $B = 8$, $NO = 16$, and column filters with $B = NO = 16$. Also, $NH = 16$ and N are set to be the same for both row and column filters. The static system uses far more resources than our dynamic system (Table 3). In the static system, since the hardware implementation is always that of the largest column filter (since we cannot reconfigure dynamically), the power consumption of the row and column filters is about the same

(the row filter spends slightly less power since some input ports are driven to zero).

Performance and energy formulas (4) and (6) apply for the static system (omitting the reconfiguration time). In the static system, computing t_{rows} , t_{cols} requires M additional cycles in the inner summation of (1), since we need to feed the M coefficients. Table 12 shows performance comparison for the dynamic and static implementations at 100 MHz. As the frame size increases, the performance of the dynamic system gets very close to that of the static system. Table 13 shows the energy per frame (mJ) for the given filterbanks and for the dynamic and static cases at 100 MHz. Results are given for different frame sizes. For most cases, the static system spends more energy than the dynamic one, except for the small QCIF and CIF frame sizes. This is because the dynamic system incurs a reconfiguration time overhead, which is offset as the frame size increases. Also, it is important to point out that the video frame size has a negligible effect on the hardware resources.

7. Conclusion

The paper presents an efficient implementation of a general-purpose 2D filterbank that supports any possible combination of real/complex inputs, filter coefficients, and outputs. The efficiency of the approach is based on the use of separable implementations and dynamic partial reconfiguration so that the hardware requirements are reduced to a single, general-purpose 1D FIR filter.

The system has been demonstrated using both real and complex-valued images and video frames along with real and complex 2D separable filterbanks. The dynamic implementation was evaluated in terms of performance, energy per frame, and accuracy values for a CIF frame and five different frequencies of operation. Also, we included a comprehensive comparison with a static implementation that demonstrates that the dynamic approach is promising for large image sizes.

TABLE 12: Performance comparison for different filterbanks and implementation types (dynamic and static). Frame sizes used: QCIF, VGA, 720 p, and 1080 p. Note that dynamic implementations incur in reconfiguration time overhead (3). Times obtained at 100 MHz.

2D filterbank		Performance: t_{rows} (ms), t_{cols} (ms), t_{fbank} (ms) fps					
		Dynamic			Static		
		$t_{\text{rows}} + t_{\text{cols}}$	t_{fbank}	fps	$t_{\text{rows}} + t_{\text{cols}}$	t_{fbank}	fps
QCIF	RIRH	0.28 + 0.30	6.71	149	0.33 + 0.35	2.73	366
	CIRH	0.28 + 0.30	6.71	149	0.33 + 0.35	2.73	366
	RICH	0.27 + 0.28	36.11	27.7	0.28 + 0.29	10.33	97
	CICH	0.27 + 0.28	47.39	21.1	0.28 + 0.29	10.33	97
VGA	RIRH	3.18 + 3.23	30.03	33.3	3.33 + 3.42	27.04	37
	CIRH	3.18 + 3.23	30.03	33.3	3.33 + 3.42	27.04	37
	RICH	3.14 + 3.17	139.8	7.15	3.17 + 3.20	114.8	8.7
	CICH	3.14 + 3.17	151.1	6.61	3.17 + 3.20	114.8	8.7
720 p	RIRH	9.38 + 9.52	80.02	12.5	9.61 + 9.92	78.12	13
	CIRH	9.38 + 9.52	80.02	12.5	9.61 + 9.92	78.12	13
	RICH	9.32 + 9.42	363.5	2.75	9.36 + 9.48	339.3	2.9
	CICH	9.33 + 9.42	374.8	2.66	9.36 + 9.48	339.3	2.9

TABLE 13: Energy per frame (mJ) for various Filterbanks and implementation types (dynamic and static) at 100 MHz. Several frame sizes are used. QCIF: 176×14 , CIF: 352×288 , VGA: 640×480 , 4CIF: 704×576 , 720 p: 1280×720 , 1080 p: 1920×1080 , 2000 p: 2048×1536 , 4000 p: 4096×3072 .

Energy per frame (mJ)								
	Dynamic				Static			
	RIRH	CIRH	RICH	CICH	RIRH	CIRH	RICH	CICH
QCIF	0.32	0.39	1.68	2.29	0.17	0.24	0.63	0.68
CIF	0.57	0.84	2.67	3.38	0.57	0.85	2.35	2.55
VGA	1.25	2.01	5.30	6.28	1.65	2.43	6.92	7.53
4CIF	1.57	2.57	6.54	7.65	2.14	3.17	9.10	9.89
720 p	3.27	5.50	13.1	14.8	4.76	7.03	20.4	22.3
1080 p	7.01	12.0	27.6	30.9	10.5	15.5	45.7	49.7
2000 p	10.5	18.0	41.1	45.8	15.8	23.3	69.2	75.2
4000 p	40.9	70.8	159	176	62.2	91.9	275	299

The results show that this implementation is very efficient in terms of resources used while delivering high accuracy results using fixed-point hardware.

Further work can focus on the implementation of image analysis systems. For example, once the filterbank outputs are obtained we can implement AM-FM feature extraction and classification [28].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] J. Becker, M. Hübner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, "Dynamic and partial FPGA exploitation," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 438–452, 2007.
- [2] A. Bovik, *The Essential Guide to Image Processing*, Elsevier/Academic Press, 2nd edition, 2009.
- [3] A. Bovik, *The Essential Guide to Video Processing*, Elsevier/Academic Press, 2nd edition, 2009.
- [4] D. Llamocca, M. Pattichis, and G. A. Vera, "Partial reconfigurable FIR filtering system using distributed arithmetic," *International Journal of Reconfigurable Computing*, vol. 2010, Article ID 357978, 14 pages, 2010.
- [5] D. Llamocca and M. Pattichis, "Real-time dynamically reconfigurable 2-D filterbanks," in *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI '10)*, pp. 181–184, Austin, Tex, USA, May 2010.
- [6] D. Llamocca, C. Carranza, and M. Pattichis, "Separable FIR filtering in FPGA and GPU implementations: energy, performance, and accuracy considerations," in *Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL '11)*, pp. 363–368, Chania, Greece, September 2011.
- [7] D. Llamocca, C. Carranza, and M. Pattichis, "Dynamic multiobjective optimization management of the Energy-Performance-Accuracy space for Separable 2-D complex filters," in *Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL 2012)*, pp. 579–582, Oslo, Norway, August 2012.
- [8] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," in *Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09)*, pp. 498–502, Prague, Czech Republic, September 2009.
- [9] J. Huang and J. Lee, "A self-reconfigurable platform for scalable dct computation using compressed partial bitstreams and blockram prefetching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 11, pp. 1623–1632, 2009.
- [10] M. S. Pattichis and J. C. Hoffman, "A high-speed dynamic partial reconfiguration controller using direct memory access through a multiport memory controller and overclocking with active feedback," *International Journal of Reconfigurable Computing*, vol. 2011, Article ID 439072, 10 pages, 2011.
- [11] A. Sudarsanam, R. Barnes, J. Carver, R. Kallam, and A. Dasu, "Dynamically reconfigurable systolic array accelerators: a case

- study with extended Kalman filter and discrete wavelet transform algorithms,” *IET Computers and Digital Techniques*, vol. 4, no. 2, Article ID ICDTA6000004000002000126000001, pp. 126–142, 2010.
- [12] M. Fons, F. Fons, and E. Cantó, “Fingerprint image processing acceleration through run-time reconfigurable hardware,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 12, pp. 991–995, 2010.
 - [13] A. Ahmad and A. Amira, “Efficient reconfigurable architectures for 3D medical image compression,” in *Proceedings of the International Conference on Field-Programmable Technology (FPT ’09)*, pp. 472–474, Sydney, Australia, December 2009.
 - [14] D. Llamocca and M. Pattichis, “A dynamically reconfigurable pixel processor based on power/energy-performance-accuracy optimization,” *IEEE Transactions on Circuits and Systems For Video Technology*, vol. 23, no. 3, pp. 488–502, 2013.
 - [15] H. Neoh and A. Hazanchuk, “Adaptive edge detection for real-time video processing using FPGAs,” in *Proceedings of the Global Signal Processing Expo and Conference (GSP ’04)*, 2004.
 - [16] C. Bouganis, S. Park, G. Constantinides, and P. Cheung, “Synthesis and optimization of 2D filter designs for heterogeneous FPGAs,” *ACM Transactions on Reconfigurable Technology Systems*, vol. 1, no. 4, 2009.
 - [17] P. Longa, A. Miri, and M. Bolic, “A flexible design of filterbank architectures for discrete wavelet transforms,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’07)*, pp. III1441–III1444, April 2007.
 - [18] N. VoB and B. Mertsching, “Design and Implementation of an accelerated gabor filter bank using parallel hardware,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, 2001.
 - [19] Y. C. P. Cho, S. Bae, Y. Jin, K. M. Irick, and V. Narayanan, “Exploring Gabor filter implementations for visual cortex modeling on FPGA,” in *Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL ’11)*, pp. 311–316, Chania, Greece, September 2011.
 - [20] S. Hong, J. Lee, A. Athalye, P. M. Djurić, and W.-D. Cho, “Design methodology for domain specific parameterizable particle filter realizations,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 9, pp. 1987–2000, 2007.
 - [21] D. Llamocca, *Dynamically Reconfigurable Management of Energy, Performance, and Accuracy applied to Digital Signal, Image, and Video Processing Applications [Ph.D. thesis]*, University of New Mexico, Albuquerque, New Mexico, 2012.
 - [22] S. U. Bhandari, S. Subbaraman, S. S. Pujari, and R. Mahajan, “Real time video processing on FPGA using on the fly partial reconfiguration,” in *Proceedings of the International Conference on Signal Processing Systems (ICSPS ’09)*, pp. 244–247, Singapore, May 2009.
 - [23] T. Raikovitch and B. Fehér, “Application of partial reconfiguration of FPGAs in image processing,” in *Proceedings of the 6th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME ’10)*, deu, Berlin, Germany.
 - [24] Y. Hori, A. Satoh, H. Sakane, and K. Toda, “Bitstream encryption and authentication with AES-GCM in dynamically reconfigurable systems,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’08)*, pp. 23–28, Heidelberg, Germany, September 2008.
 - [25] J. A. Clemente, J. Resano, C. González, and D. Mozos, “A Hardware implementation of a run-time scheduler for reconfigurable systems,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1263–1276, 2011.
 - [26] Xilinx, *Virtex-6 Family Overview (DS150)*, Xilinx, San Jose, Calif, USA, 2010.
 - [27] Xilinx, *Power Methodology Guide (UG786)*, vol. 13. 1, Xilinx, San Jose, Calif, USA, 2011.
 - [28] V. Murray, P. Rodríguez, and M. S. Pattichis, “Multiscale AM-FM demodulation and image reconstruction methods with improved accuracy,” *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1138–1152, 2010.
 - [29] A. C. Bovik, M. Clark, and W. S. Geisler, “Multichannel texture analysis using localized spatial filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, 1990.