Regular Article

Low-complexity chroma down/up sampling conversion using decimation/interpolation FIR filter and FPGA IP-core design

Tran Van Nghia[1], **Dang Trung Hieu**[2], **Pham Viet Dung**[1]

[1] Air Force – Air Defense Technical Institute, Hanoi, Vietnam
[2] Electric Power University, Hanoi, Vietnam

Correspondence: Dang Trung Hieu, hieudt@epu.edu.vn
Communication: received 10 October 2025, revised 30 December 2025, accepted 15 February 2026
Online publication: 18 February 2026, Digital Object Identifier: 10.21553/rev-jec.418

Abstract– **This paper proposes a low-complexity solution for chroma downsampling and upsampling, applicable to video compression. The proposed method uses decimation and interpolation FIR filters, which are known to be effective in signal processing. This approach enhances the quality of chroma recovery and minimizes computational complexity, making it suitable for hardware platforms with limited resources. An experimental design was implemented on an FPGA using a parameterized IP core that supports YCbCr and YUV color spaces, enabling flexible conversion between 4:4:4, 4:2:2, and 4:2:0 formats. This implementation contributes to improved performance, enhanced flexibility, and reduced computational complexity—an important requirement for modern video applications. The IP core was successfully verified using the Xilinx Vitis Model Composer tool, with a focus on visually inspecting the output and comparing it against reference models developed in the MATLAB/Simulink environment.**

Keywords– **Decimation FIR filter, interpolation FIR filter, chroma downsampling and upsampling, YCbCr, YUV.**

1 Introduction

Chroma down/up sampling conversion is an important step in digital video/image processing and compression. To minimize the bit rate required for storage and transmission, RGB data components are typically converted to YCbCr/YUV color spaces [1–4], where the luminance (Y) and two chroma components (Cb and Cr or U and V) are separated. Since the human visual system is more sensitive to brightness changes than to color changes, the chroma components are typically sampled at a lower resolution than the luminance component, a process known as chroma subsampling or downsampling. The 4:2:0 format is one of the most common formats for MPEG-2, MPEG-4, H.264, and H.265 encoding, where the chroma components have 1/4 the resolution of the luminance component. Although chroma downsampling significantly reduces the bit rate, it can also introduce color errors and reduce the color accuracy in the decompressed image/video. This is especially noticeable with the growth of high-definition resolution (HDR) video content, where higher color accuracy is required than standard definition resolutions (SDR).

To achieve a balance between compression efficiency and image quality, research efforts have focused on developing more effective color downsampling and upsampling techniques [3–8]. Several studies have proposed novel chroma processing schemes that leverage the characteristics of the human visual system to redistribute color codes in favor of hues to which the human eye is more sensitive [5]. Other methods have concentrated on optimizing the color downsampling process by considering image content features or incorporating luminance component modifications [3, 8]. The study [6] introduces a chroma subsampling technique that derives downsampled Cb and Cr components based on their contribution to reconstructing the B and R data. Meanwhile, [7] presents an iterative bilinear interpolation-based approach that optimizes the quality-bitrate trade-off in the reconstructed images. Deep learning networks are also utilized for chroma subsampling based on features from different levels and transposed convolution [9, 10]. Chroma processing schemes [3–10] have demonstrated significant improvements in reconstructed image quality, but often at the cost of increased computational complexity.

A noteworthy issue is that, to date, no scientific publications have reported simultaneous chroma interpolation and reduction for both the YCrCb and YUV color spaces, which are commonly used in image and video compression systems. Several FPGA technology vendors, such as AMD, Xilinx, and Intel Altera, provide chroma processing IP cores as built-in components within their design tools, allowing users to configure parameters through the development environment. However, after the design has been synthesized and implemented on the target device, the FPGA system operates with a fixed set of parameters, and flexible run-time adjustment is not supported. Furthermore, vendor-developed IP cores generally support only a single color space and are constrained in hardware compatibility, being limited to particular FPGA families. For instance, Xilinx IP cores mainly support device families such as UltraScale Architecture, Zynq™-7000, and 7-Series.

To address the aforementioned issues, this study introduces a low-complexity chroma downsampling/upsampling approach based on FIR decimation and interpolation filters, along with an FPGA-based IP core implementation. The proposed approach utilizes FIR filters to achieve high-quality chroma subsampling and upsampling while minimizing computational overhead. By optimizing the filter design and targeting FPGA hardware, the proposed IP core achieves real-time performance and reduces hardware resource utilization, making it suitable for high-performance video processing systems and modern video compression standards, such as H.264/AVC and HEVC. Furthermore, this design supports both YCrCb and YUV color spaces and matches with various FPGA chip families.

While closed-source commercial IPs provide pre-assembled components with fixed parameters that are suitable for rapid deployment in practical systems, the design proposed in this study is developed with a transparent chroma processing microarchitecture that exposes internal structure and behavior. This transparency enables users and designers to analyze, understand, and customize the FIR filter's internal operation for specific requirements (e.g., video type, resolution, or quality), which can facilitate targeted optimization, verification, and reuse in research. At the same time, the design maintains practical flexibility and efficient FPGA implementation. By presenting architectural details in an analyzable form, the proposed design serves as a reusable artifact that complements existing IP options and supports future exploration and development in hardware research.

Key contributions of this work include:

1. Efficient chroma conversion: A decimation/interpolation FIR filter architecture tailored for chroma subsampling and upsampling, ensuring minimal distortion and high color fidelity.

2. Hardware optimization: Implementation of an FPGA-based IP core that maximizes throughput and minimizes resource utilization, allowing real-time processing.

3. Low-complexity design: A streamlined algorithm that reduces computational demands compared to state-of-the-art methods while maintaining competitive performance in terms of peak color signal-to-noise ratio (CPSNR) and visual quality.

4. Flexibility: The proposed design supports both YCrCb and YUV color spaces and provides configurable parameters that can be adjusted after programming, without requiring hardware re-synthesis.

The remainder of this paper is organized as follows: Section 2 details the proposed FIR filter for video processing and its FPGA implementation. Section 3 presents the experimental results, and Section 4 concludes the paper.

2 Proposed FIR Filter for Video Processing and Its FPGA Design

Human vision is more sensitive to luminance (brightness) details than to chrominance (color) details. To capitalize on this perceptual feature in image and video processing and compression systems, RGB signals are commonly transformed into YCbCr or YUV representations. In these formats, the Y channel represents luminance, whereas the Cb/U and Cr/V channels encode chrominance, containing color difference information obtained from the initial RGB values. The definitions of these color space conversions are formalized in the ITU-R recommendations [11, 12].

The next step in video processing and compression is chroma subsampling, which reduces the amount of data required for the color components while maintaining the perceived image quality. Figure 1 illustrates the typical chroma subsampling formats used in video coding.

The 4:4:4 format represents the YCbCr/YUV color space at the same sampling rate as the original RGB signal. In this configuration, the chrominance components are sampled at the same spatial resolution as the full-resolution luminance component, ensuring that both luma and chroma information are collocated at each pixel position. Each of the Y, Cb/U, and Cr/V components is transmitted over a separate bus.

In the 4:2:2 chroma subsampling scheme, only horizontal downsampling is applied. This configuration maintains a 2:1 ratio between the luma and chroma samples. The subsampled chroma locations are co-sited with alternating luminance samples. For transmission, the format multiplexes Cr/V and Cb/U components onto a single bus running at full frequency, with Cr/V occupying the first position in the interleaved sequence.

In the 4:2:0 format used in video coding standards such as MPEG-2, MPEG-4 Part 2, H.264, and H.265, the chroma components are subsampled horizontally and vertically relative to the luma, as commonly specified in standards such as [11, 12]. The chroma sample locations do not coincide with the luma pixel positions. In this version, the vertical interpolation method is used to generate chroma values, effectively positioning them between successive pairs of original scan lines. Horizontally, the chroma samples are aligned with alternate luma pixels.

A total of six format conversions are used to support the three chroma subsampling schemes: 4:4:4, 4:2:2, and 4:2:0, as detailed in Table I. To implement these transformations, a dedicated IP core is introduced to provide three alternative methods for video pixel interpolation and decimation: (1) a configurable filter with programmable coefficients for high-performance applications; (2) a static filter with power-of-two coefficients for memory-constrained applications; and (3) duplicating or dropping pixels. The transformation is performed using FIR filtering. Depending on specific requirements, certain transformations involve filtering in only the horizontal dimension, only the vertical dimension, or both dimensions simultaneously (i.e., two-

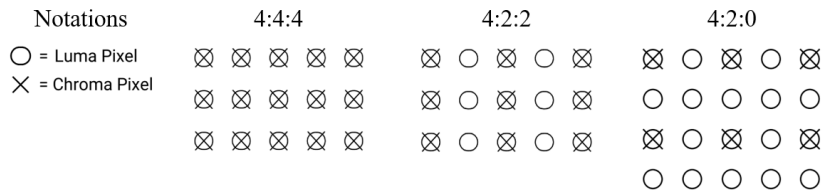

Figure 1. The subsampled video formats.

Table I
FILTER SETUP

Converter	Filter Orientation	Default filter size
4:4:4 to 4:2:2	Horizontal decimation and anti-aliasing	3 H taps
4:4:4 to 4:2:0	Separable 2D anti-aliasing and H & V decimation	2 V taps by 3 H taps
4:2:2 to 4:4:4	Horizontal interpolation	2 H taps
4:2:2 to 4:2:0	Vertical decimation and anti-aliasing	2 V taps
4:2:0 to 4:4:4	Separable 2D interpolation	2 H taps by 2 V taps
4:2:0 to 4:2:2	Vertical interpolation	2 H taps by 2 V taps

dimensional—2D). Interpolation operations are implemented using a two-phase polyphase FIR filter, whereas decimation operations utilize a low-pass FIR filter to mitigate chroma aliasing. The 2D FIR filtering operation is expressed as follows

$$P_{out}(n,m) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} P_{in}(n-N+i, m-M+j) W(i,j). \tag{1}$$

In Equation (1), W denotes the 2D FIR filter kernel, which is a coefficient matrix specifically configured for each transformation; M indicates the number of vertical filter taps, while N specifies the number of horizontal taps; $P_{out}(n,m)$ represents the output pixel value produced at spatial coordinates (n,m); and $P_{in}(n,m)$ refers to the corresponding input pixel at the same location.

When the kernel W can be decomposed as the outer product of two one-dimensional vectors, the filter is said to be separable. By exploiting separability, the 2D convolution can be performed using two sequential 1D convolutions—first horizontally, then vertically—reducing the number of multiplications from $N \times M$ to $N + M$. If the filter coefficients exhibit symmetry, the multiplication count can be further reduced: $[(N+M)/2+1]$ for odd-length filters and $(N+M)/2$ for even-length filters.

The general architecture of the proposed 2D FIR filter implemented in an FPGA is illustrated in Figure 2. This architecture utilizes line buffers (LBs) to store enough lines equal to the vertical size of the kernel before performing vertical filtering, followed by horizontal filtering. The detailed structure of the filter depends on the specific transformation configuration, which is described below.

- **Convert 4:2:2 to 4:4:4**

This transformation involves a 1:2 horizontal interpolation performed using a two-phase polyphase FIR filter. In this configuration, one of the two pixels in the output coincides with the input pixel and can be reused directly to produce the ideal result. Therefore, in phase 0, no filtering coefficient is required, as the output

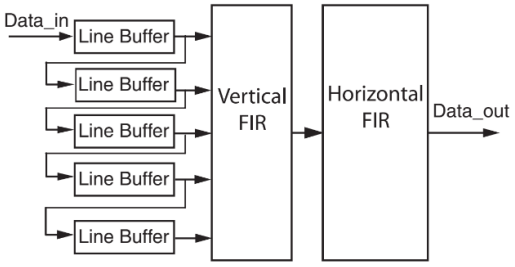

Figure 2. The 2D FIR filter separable with five vertical taps.

is generated by simply replicating the input sample. To achieve the output $P_{out}(n,m)$, the FIR filter convolves the kernel W with pixels from the input image $P_{in}(n,m)$

$$P_{out}(n,m) = \sum_{i=0}^{N-1} P_{in}(n-i,m) W_p(i), \tag{2}$$

where p is the interpolation phase (0 or 1, depending on n).

During phase 1, the coefficient $W_1(0)$ is applied to the latest input sample within the filter window. Figure 3 illustrates the use of coefficients in an example of a four-tap filter, using simplified notation $W_1 = [a,b,c,d]$. For the default two-tap polyphase filter, the phase 2 coefficients are [0.5 0.5], represented in 2-bit form with the corresponding integer values [1 1].

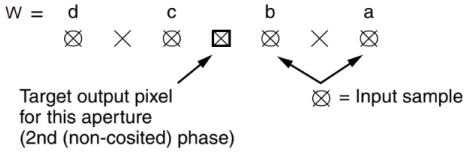

Figure 3. The 4:2:2 to 4:4:4 coefficient configuration.

The inherent latency of the default filter corresponds to eight clock cycles. For alternative filter configurations, the latency can be determined as follows

$$Latency = 2N - 1 \quad \text{(Clock cycles)}. \tag{3}$$

For the replicate configuration, the resulting latency is reduced to seven clock cycles.

- **Convert 4:4:4 to 4:2:2**

This process performs a 2:1 horizontal downsampling using a low-pass FIR filter to prevent chroma aliasing. The operation of the FIR filter is outlined below

$$P_{out}(n,m) = \sum_{i=0}^{N-1} P_{in}(n-i,m)W_0(i). \tag{4}$$

During phase 0 of the filtering process, the coefficient $W_0(0)$ is multiplied by the most recent input pixel. Figure 4 presents a schematic of this operation within a five-tap filter architecture, employing the simplified notation $W_0 = [a,b,c,d,e]$ for the coefficients.

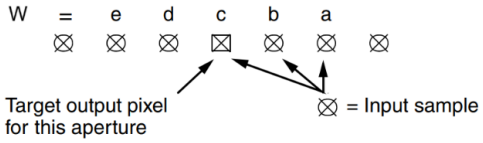

Figure 4. The 4:4:4 to 4:2:2 coefficient configuration.

Typically, a three-tap filter with ideal coefficients [0.25 0.5 0.25] is used, and the fixed-point (3-bit) implementation represents these as an integer set [1 2 1].

This default filter exhibits eight-cycle latency, while non-standard filters require latency determination via the formula below

$$Latency = N \qquad \text{(clock cycles)}. \tag{5}$$

When the pixel drop option is enabled, the filter's latency reduces to just two clock cycles.

- **Convert 4:2:0 to 4:2:2**

This transformation performs 1:2 vertical interpolation through a two-phase polyphase FIR filter. In this configuration, one of the two pixels in the output coincides with the input pixel, and the other is generated by replication of that sample to achieve an ideal output. To compute each output pixel $P_{out}(n,m)$, the FIR filter performs a convolution of the coefficient set $W_p(k)$ (where k indexes the coefficients and p denotes the interpolation phase) with the input pixel values $P_{in}(n,m)$

$$P_{out}(n,m) = \sum_{i=0}^{N-1} P_{in}(n,m-i)W_p(i). \tag{6}$$

In phase 0, the filter coefficient $W_0(0)$ is multiplied by the most recent input pixel. Figure 5 presents a schematic of this operation within the five-tap filter architecture, employing the simplified notation $W_0 = [a,b,c,d]$ for the coefficients.

By default, this process employs a two-phase, two-tap structure with ideal coefficients of [0.25 0.75]. In fixed-point (3-bit) hardware, these values are quantized to the integer set [1 3]. The second phase is generated by reversing the order of these coefficients.

In interlaced video, the default filter coefficients for each field are defined as follows:

- For odd field: phase 0 employs coefficients [3/8 5/8], whereas phase 1 uses [7/8 1/8];

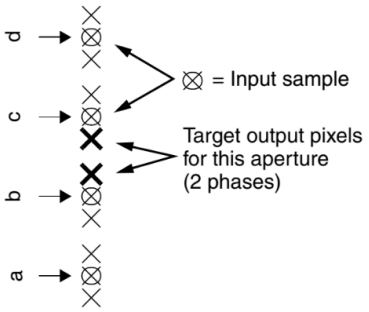

Figure 5. The 4:2:0 to 4:2:2 coefficient configuration.

- For even field: phase 0 uses coefficients [1/8 7/8] and phase 1 applies [5/8 3/8].

In even fields of interlaced video, the phase 0 and phase 1 coefficients are swapped, and the coefficient order within each filter is reversed.

The filter's latency for the default configuration is one line plus nine clock cycles. For other configurations, the latency should be determined as follows

$$\begin{cases} V\_latency = 2(M-1) & \text{(lines)} \\ H\_latency = 1 & \text{(cycles)}. \end{cases} \tag{7}$$

For the replicate option, the system latency is reduced to five clock cycles.

- **Convert 4:2:2 to 4:2:0**

This transformation performs 1:2 vertical interpolations using a low-pass FIR filter to suppress chroma aliasing. The FIR filter's behavior is mathematically defined as follows

$$P_{out}(n,m) = \sum_{i=0}^{M-1} P_{in}(n,m-i)W_0(i). \tag{8}$$

In phase 0, the filter coefficient $W_0(0)$ is multiplied by the most recent input pixel. Figure 6 illustrates a schematic of this operation within the four-tap filter architecture, employing the simplified notation $W_0 = [a,b,c,d]$ for the coefficients.

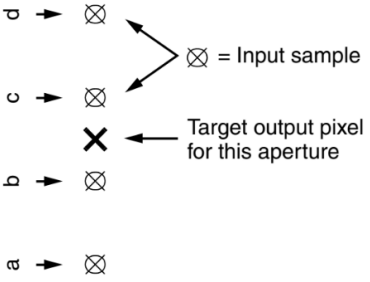

Figure 6. The 4:2:2 to 4:2:0 coefficient configuration.

For progressive video, the default coefficients are [0.5 0.5]. For interlaced video, the default coefficients are [0.25 0.75] for the odd field. For the even field, the default coefficients are reversed [0.75 0.25]. The filter's latency for the default configuration is one line plus eight clock cycles. For other configurations, the latency

should be determined as follows

$$\begin{cases} V\_latency = M-1 & \text{(lines)} \\ H\_atency = 1 & \text{(cycles).} \end{cases} \quad (9)$$

For the drop option, the system latency is three clock cycles.

- **Convert 4:2:0 to 4:4:4**

This conversion procedure performs interpolation in both the vertical and horizontal directions. Functionally, it is equivalent to a 2D separable filter, implemented by cascading two processing blocks: conversion from 4:2:0 to 4:2:2, followed by conversion from 4:2:2 to 4:4:4. The output of the vertical filter is subsequently processed by the horizontal filter (see Figure 1). The intermediate chroma values in the 4:2:2 formats are calculated using Equation (6). The corresponding computation formula is presented in the following equation:

$$T_{out}(n,m) = \sum_{i=0}^{M-1} P_{in}(n,m-i)W_p(i). \quad (10)$$

Next, the values are filtered according to Equation (2). The resulting computation is shown in the following equation

$$P_{out}(n,m) = \sum_{i=0}^{N-1} T_{in}(n-i,m)W_0(i). \quad (11)$$

All generic filters must contain exact $M \times N$ coefficients. The first N coefficients for each $H-$phase serve to process the bottom line of input samples in $V-$phase 0 and the top line of input samples in $V-$ phase 1. For $V-$ phase 0, the coefficient a ($H-$phase 0) operates on the newest input sample (bottom-right position in filter window, following raster-scan order) as shown in Figure 7, whereas the coefficient e ($H-$phase 1) applies to the identical spatial location (refer to Figure 8). For $V-$phase 1, all sample positions are vertically inverted from $V-$ phase 0.

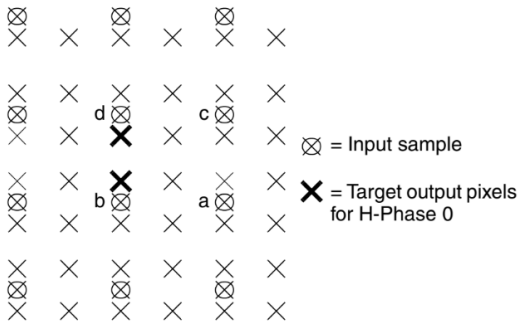

Figure 7. Filtering operation in 4:2:0 to 4:4:4 conversion (H-phase 0).

The default coefficients are retained as originally defined during the conversions from 4:2:0 to 4:2:2 and from 4:2:2 to 4:4:4. For the two-tap polyphase filter, the horizontal coefficients in phase 1 in the second stage are [0.5 0.5]. For progressive video, the default vertical coefficients for phase 0 and phase 1 are defined as [0.25 0.75] and [0.75 0.25], respectively. For interlaced video, the default vertical coefficients are defined separately for each field:

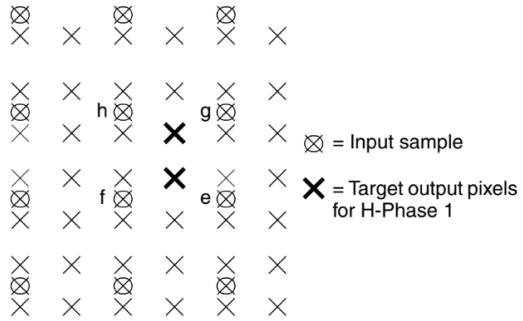

Figure 8. Filtering operation in 4:2:0 to 4:4:4 conversion ($H-$phase 1).

- For odd field: phase 0 employs coefficients of [3/8 5/8], while phase 1 uses [7/8; 1/8];
- For even field: phase 0 uses [1/8 7/8] and phase 1 applies [5/8 3/8].

For the even field of interlaced data, the coefficients for phase 0 and phase 1 are swapped, and the filter coefficients for each filter are reversed.

In the default configuration, the filter introduces a latency of one line and fourteen clock cycles. For other configurations, the latency can be determined using the formula below

$$\begin{cases} V\_latency = 2(M-1) & \text{(lines)} \\ H\_atency = 2N & \text{(cycles)} \end{cases}. \quad (12)$$

When the replicate feature is enabled, the filter's latency is reduced to 10 clock cycles.

- **Convert 4:4:4 to 4:2:0**

This conversion procedure performs decimation in both the vertical and horizontal directions. Functionally, it operates as a 2D separable filter implemented through cascading two processing stages: first converting from 4:4:4 to 4:2:2, and then from 4:2:2 to 4:2:0. The horizontally filtered and quantized output is then passed through a vertical filter (see Figure 1).

Intermediate 4:2:2 chroma values are calculated using Equation 4. The resulting computation is shown in the following equation

$$T_{out}(n,m) = \sum_{i=0}^{N-1} P_{in}(n-i,m)W_0(i). \quad (13)$$

Next, the values are filtered according to Equation (8) and the resulting computation is presented below

$$P_{out}(n,m) = \sum_{i=0}^{M-1} T_{in}(n,m-1)W_0(i). \quad (14)$$

In this conversion, the coefficient a applies to the newest input sample within the filter window (i.e., the bottom-right position in raster-scan order), as shown in Figure 9.

The default coefficients specified for the 4:4:4 to 4:2:2 and 4:2:2 to 4:2:0 conversions are retained without modification. The default horizontal coefficients are [0.25 0.5 0.25]. For progressive video, the default vertical coefficients are [0.5 0.5]. In interlaced mode, the odd field uses vertical coefficients [0.25 0.75], while the even field uses the reversed set [0.75 0.25].

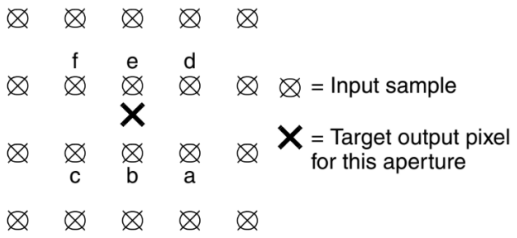

Figure 9. Coefficient configuration for converting 4:4:4 to 4:2:0.

In the default configuration, the filter adds a latency of one line plus nine clock cycles. For alternative configurations, the latency should be calculated as follows:

$$\begin{cases} V\_latency = M - 1 & \text{(lines)} \\ H\_atency = N & \text{(cycles)} \end{cases}. \tag{15}$$

Selecting the drop option reduces the filter's latency to just five clock cycles.

3 Computational Complexity Analysis

In this section, we analyze the computational complexity of three chroma down/up-sampling methods, including the method in [7, 8], and the proposed FIR-based decimation/interpolation method, in terms of the total number of basic operations required to process an entire image. The method in [7] determines the optimal chroma configuration for each $N \times N$ block by computing the sum of squared errors (SSE) and comparing multiple neighboring candidates over L optimization iterations. Consequently, for an image with N_p pixels, the computational complexity of this method can be approximately expressed as $O(N_p \cdot L)$. Although L is a small constant in practical applications, the repeated evaluation of the SSE and the consideration of multiple neighbors in each iteration lead to a relatively large constant factor due to the significant number of multiplications, additions, and comparisons.

The method in [8] processes each 2×2 block independently using comparisons, conditional checks, and linear interpolation with a fixed structure; therefore, its computational complexity increases linearly with the number of pixels N_p and can be expressed as $O(N_p)$. Compared with the method in [7], this approach has a smaller hidden constant, because it does not involve iterative procedures or require neighbor searching.

For the proposed FIR-based decimation/interpolation method, the use of two sequential 1-D convolutions (horizontal followed by vertical) results in an effective total of $(N + M)$ filter taps. Moreover, since the proposed FIR filter is symmetric, the computational complexity of the image filtering operation is reduced to $(N + M)/2$ multiplications and $(N + M)/2$ additions if $(N + M)$ is even or $(N + M + 1)/2$ multiplications and $(N + M + 1)/2$ additions if $(N + M)$ is odd, as given in (1), as a result $\lceil (N + M)/2 \rceil$ multiplications and additions, where $\lceil \bullet \rceil$ is an operator to round toward positive infinity. However, the complexity of the FIR filter is mostly reflected in its mathematical representation. As analyzed above, when the filter weights are chosen as 0.5, 0.25, 0.125, 0.875, and 0.75, the corresponding multiplications are not performed normally during FPGA implementation. Instead, these multiplications are replaced by bit shifts: multiplications by 0.5, 0.25, and 0.125 are equivalent to a 1/2/3-bit right shift. For the 0.75 weight, since it can be represented as the sum of 0.5 and 0.25, the corresponding multiplication is performed by combining a 1-bit right shift and a 2-bit right shift, followed by an addition. Similarly, the weight 0.875 is equal to $(1 - 0.125)$. Hence, in the case where the weights are only 0.5, 0.25, and 0.125, the proposed FIR filter requires $\lceil (N + M)/2 \rceil$ additions. If any weight is 0.75 or 0.875, the number of additions increases by the number of such weights.

The proposed FIR filter also demonstrates a significant advantage when implemented on an FPGA, because it uses only additions and does not require the complex conditional branches or iterative refinement found in other methods. This allows for more effective pipeline optimization, fewer stalls, and improved memory utilization. In contrast, although the method in [8] has a lower theoretical operation count, it involves numerous conditional checks and branching logic, which can lead to pipeline stalls and hinder hardware optimization, ultimately reducing actual throughput. Table II compares the computational complexity of these methods, where N and M are indicated in Table I.

4 Testing the proposed design on FPGA hardware

The FPGA implementation of the proposed chroma-resampling IP core is shown in Figure 10. The IP core design is based on the Vitis Model Composer tool, which enables flexibility across different configurations. For each configuration, the IP core uses the following interfacing signals for input and output definitions:

- *Y_in/out*: *Y_in* is the luma signal fed into the IP-core input to synchronize delays between the luma and chroma components, and *Y_out* is its delayed version.
- *chr_in/out* denotes full-bandwidth CrCb/VU interleaved signal used for 4:2:0 and 4:2:2 formats.
- *cr_in/out*, *cb_in/out*, *U_in/out*, and *V_in/out* are color difference signals. They are only used for the 4:4:4 formats.
- *vs_in*, *hs_in* are the vertical and horizontal sync input signals, respectively.
- *vs_out* and *hs_out* are the vertical and horizontal sync outputs, delayed to match the vertical and horizontal latencies of the filter.
- *din_valid* is a signal used to suppress edge-filtering artifacts when the input data is mirrored or repeated at image edges.
- *dout_valid* is the delayed version of *din_valid*, indicating valid data on the luma and chroma channels.

The IP core is designed to support 8, 10, 12, and

Table II
Computational Complexity Comparison of Methods

Method	Number of Operations	Hidden Constant	Primary Operations	Hardware Implementation Capability
Method [7]	$O(N_p \cdot L)$	Medium	SSE and many neighbor comparisons	Difficult to pipeline
Method [8]	$O(N_p)$	Low	Comparisons, condition checks, fixed interpolation	Pipeline stalls
Proposed	$O(\lceil (N+M)/2 \rceil)$	Lowest	Fixed multiplications and additions per taps	Easy to pipeline, highest efficiency

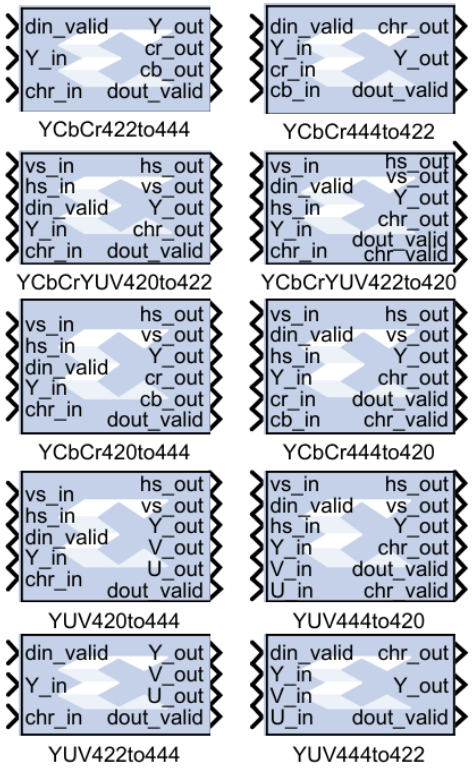

Figure 10. Interface signals in IP-core configurations.

16 bits per color component data, spatial resolutions ranging from 32×32 to 7680×7680, and offers three flexible implementation options as follows:

- A DSP48-based filter, allowing flexible configuration of both filter coefficients and the number of taps. The maximum number of taps is 8 in the vertical direction and 24 in the horizontal direction.
- A fixed-coefficient filter that employs predefined, non-programmable coefficients, with all coefficients being powers of two. The filtering operation relies only on bit shifts and additions, eliminating the need for DSP48 blocks.
- The simplest and most hardware-efficient solution to perform sample dropping (decimation) or sample replication (interpolation). During downsampling, certain input samples are passed directly to the output, while others are completely discarded. In upsampling for an up-converter, the previous input sample is replicated to generate the intermediate output samples.

Figures 11–14 illustrate the circuit-level RTL diagrams of the proposed IPs, excluding the two converters 4:2:0 to 4:4:4 and 4:4:4 to 4:2:0. These two converters are implemented by cascading the corresponding RTL diagrams shown in Figures 11–14. In these diagrams, the orange blocks represent one-pixel-clock registers/delays. Registers with solid borders are mandatory in the architecture, whereas registers with dashed borders are optional and used only when necessary, depending on the type of chroma data being processed in each filter diagram. Additionally, the adders in the design are implemented using a pipelined architecture, so each addition introduces a one-clock-cycle delay. The timing diagram is illustrated for the CbCr data case; this representation is entirely equivalent when applied to UV data.

For Figure 11, the input chroma data are delayed by 2, 1, and 0 pixel clocks to obtain three parallel samples. These three samples are then shifted right by 2, 1, and 2 bits, respectively, before being fed into the adders. This combination of delay, shift, and adder blocks performs the equivalent function of convolution in a FIR filter with weighted coefficients [0.25 0.5 0.25]. In video codecs, the 4:2:2 configuration is represented by interleaving Cb/U and Cr/V data. Therefore, the register/delay block at the adder output appears only on the Cr/V data processing path. To perform sample reduction, a state machine is designed based on a pixel-clock counter. This counter begins operating when *din_valid* = '1', and after N clock cycles, the state machine interleaves Cb/U and Cr/V data by connecting Cb/U to the output when the counter has an odd value and Cr/V to the output when the counter has an even value.

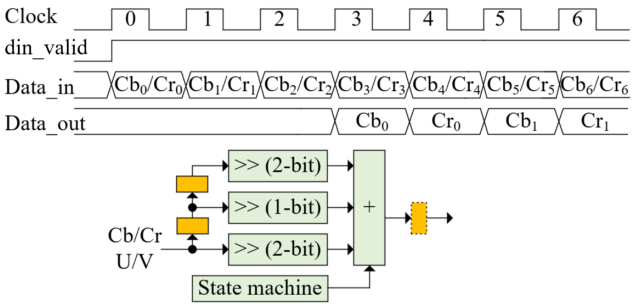

Figure 11. Circuit-level RTL diagram of the 4:4:4 to 4:2:2 converter.

In the 4:2:2 to 4:4:4 converter (Figure 12), because the input uses an interleaved processing structure, the

input chroma data are split into two separate paths, one of which is delayed by two clock cycles to collect two parallel samples. Before being fed into the adder, these samples are right-shifted by one bit. In phase 0, the chroma data are copied verbatim to the output with a delay of $2(N-1)$ clock cycles at the positions marked in red. The interpolated chroma pattern then appears immediately after a clock pulse at the positions marked in black. Therefore, in this IP, the multiplexer is designed to interleave between the original chroma data (at input '0') and the interpolated chroma data (at input '1'). Furthermore, because the Cr/V line is delayed by one clock cycle compared to the Cb/U line, the multiplexer's output for the Cb/U data requires an additional delay register for synchronization. This delay register is absent in the Cr/V processing path, allowing the Cb/U and Cr/V chroma data to reach the output in the same clock cycle.

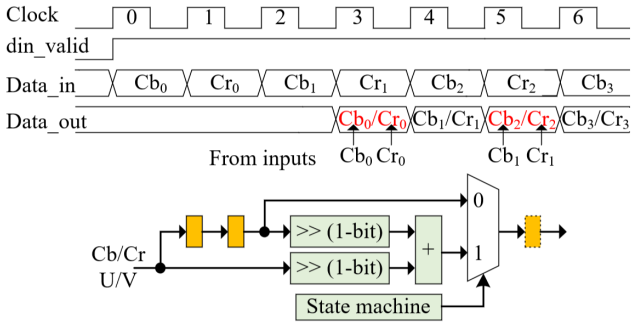

Figure 12. Circuit-level RTL diagram of the 4:2:2 to 4:4:4 converter.

For the 4:2:2 to 4:2:0 converter (Figure 13), the IP collects chroma data from two consecutive lines to create a single output chroma line. The chroma data from the even line is delayed by a line-equivalent time, ensuring that the even and odd lines appear simultaneously at the filter input. Processing of the chroma data from the two lines is performed simultaneously via three different branches, as described below:

- A 1-bit right shift is applied to both the even and odd line data before the adder, whose output is connected to input 0 of the multiplexer. The 1-bit shift and adder together are equivalent to a convolution with the FIR filter weights [0.5 0.5].
- The even line is right-shifted by 2 bits, while the odd line is right-shifted by 1 bit and 2 bits; both are fed into the second adder, whose output is connected to input 1 of the multiplexer. The shifters and adder together are equivalent to a convolution with the FIR filter weights [0.25 0.75].
- The even line is right-shifted by 1 bit and 2 bits, and the odd line is right-shifted by 2 bits; these are fed into the third adder, whose output is connected to input 2 of the multiplexer. The shifters and adder together are equivalent to a convolution with the FIR filter weights [0.75 0.25].

The state machine manages the extraction and writing/reading of even-line data to/from the buffer on the first branch and the extraction of odd-line data to the second branch. For progressive video, the state machine pushes multiplexer input 0 to the output. For interlaced video, it switches to input 1 of the multiplexer when the video stream is in the odd field and to input 2 when the video stream is in the even field.

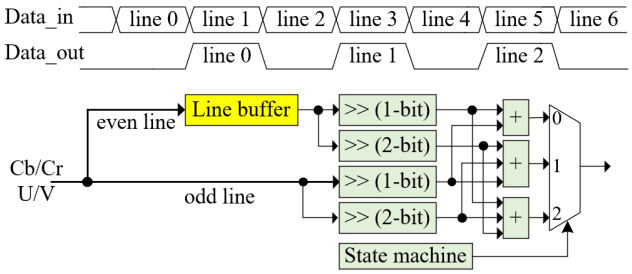

Figure 13. Circuit-level RTL diagram of the 4:2:2 to 4:2:0 converter.

The 4:2:0 to 4:2:2 converter combines the data of each chroma line with that of the previous line to interpolate two new chroma lines (Figure 14). For progressive video, the chroma data of the previous line are right-shifted by 1 bit and 2 bits, and the chroma data of the current line are right-shifted by 2 bits. These bit-shifted signals are then added by the adder to form the first interpolated data. This process is equivalent to the convolution of an FIR filter with weights [0.75 0.25]. To create the second interpolated value, the chroma data of the previous line are right-shifted by 2 bits, and the chroma data of the current line are right-shifted by 1 bit and 2 bits before being added; these operations are equivalent to the convolution of an FIR filter with weights [0.25 0.75].

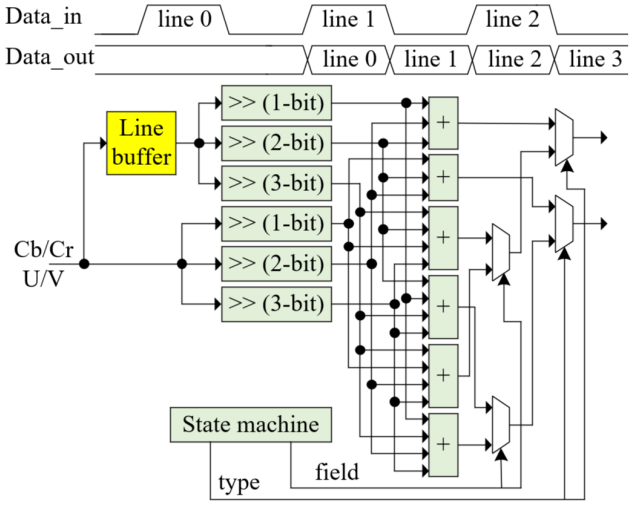

Figure 14. Circuit-level RTL diagram of the 4:2:0 to 4:2:2 converter.

For interlaced video, in the odd field, the previous chroma line data that are right-shifted by 2 bits and 3 bits, and the current line data that are right-shifted by 1 bit and 3 bits, are added to generate the first interpolated data. This process is equivalent to the convolution of an FIR filter with weights [3/8 5/8]. The second interpolated data are obtained by adding the previous chroma line data right-shifted by 1 bit, 2 bits, and 3 bits, and the current line data right-shifted by 3 bits; this process is equivalent to the convolution

Table III
THE CPSNR PERFORMANCE (DB)

Exp.	Methods	Baboon	Flowers	Flower	Tiffany	Pepper	Average
Exp.1	Method [8]	39.09	38.76	40.18	40.09	39.44	39.53
	Method [7]	42.26	42.03	43.20	42.94	42.89	42.66
	Proposed	44.32	44.17	45.43	45.46	45.50	44.98
Exp.2	Method [8]	38.07	37.89	39.97	40.02	39.33	39.06
	Method [7]	42.12	41.94	43.16	43.04	42.81	42.61
	Proposed	43.94	43.89	45.37	45.42	45.47	44.82
Exp.3	Method [8]	37.82	36.95	39.88	39.58	38.86	38.618
	Method [7]	41.96	41.17	42.81	42.88	42.07	42.178
	Proposed	43.86	43.65	45.08	45.37	45.20	44.632
Exp.4	Method [8]	37.68	36.96	39.83	39.61	38.52	38.52
	Method [7]	41.85	41.08	42.82	42.83	42.29	42.17
	Proposed	43.93	43.67	45.16	45.39	45.34	44.70

Table IV
RESOURCE UTILIZATION PERFORMANCE

Converter	Filter options	Slice LUTs	Slice Registers	RAMB36/ RAMB18	DSP48E1	Max. Clock Freq. (MHz)
4:4:4 to 4:2:2	Drop/replicate	203	150	0/0	0	298
	Fixed coefficient	375	285	0/0	0	282
4:4:4 to 4:2:0	Drop/replicate	270	180	0/0	0	298
	Fixed coefficient	507	468	0/2	0	316
4:2:2 to 4:4:4	Drop/replicate	197	163	0/0	0	314
	Fixed coefficient	362	268	0/0	0	298
4:2:2 to 4:2:0	Drop/replicate	180	119	0/0	0	288
	Fixed coefficient	336	271	0/2	0	298
4:2:0 to 4:4:4	Drop/replicate	304	242	0/1	0	306
	Fixed coefficient	612	544	0/3	0	322
4:2:0 to 4:2:2	Drop/replicate	209	155	0/1	0	324
	Fixed coefficient	407	323	0/3	0	262

of an FIR filter with weights [7/8 1/8]. For even-field interlaced video, the interpolated data are obtained similarly, with weights [1/8 7/8] and [5/8 3/8]. The state machine, based on the video type setting (progressive or interlaced), generates control signals for the multiplexers to connect the corresponding interpolated data to the output.

This study employs the conversion matrices from [1] and [2] to perform color space conversions between RGB and YCbCr/YUV. Experimental evaluations are conducted on five classic test images: Baboon, Flowers, Flower, Tiffany, and Pepper. For each test image, the following experimental scenarios are carried out to generate the corresponding output images:

- Experiment 1 1 (Exp.1): RGB → YCbCr/YUV 4:4:4 → format 4:2:2 → format 4:4:4.
- Experiment 1 2 (Exp.2): RGB → YCbCr/YUV 4:4:4 → format 4:2:0 → format 4:4:4.
- Experiment 1 3 (Exp.3): RGB → YCbCr/YUV 4:4:4 → format 4:2:2 → format 4:2:0 → format 4:4:4.
- Experiment 1 4 (Exp.4): RGB → YCbCr/YUV 4:4:4 → format 4:2:0 → format 4:2:2 → format 4:4:4.

Figure 15 shows the YCbCr component images after downsampling and upsampling in 4:4:4 format, before conversion back to the RGB color space. The results from four experiments exhibit no discernible visual differences to the naked eye. Therefore, Figure 15 presents only a representative experiment. Similarly, no significant visual variation was observed in the YUV color space across the experiments; hence, separate illustrations for these cases are omitted in this study.

To evaluate the proposed method's efficacy, this study employs the Color Peak Signal-to-Noise Ratio (CPSNR) [3]. CPSNR is computed from the color mean-squared error (CMSE) across the three color channels in 4:4:4 format, comparing the original and processed images from the four experiments. Specifically, CPSNR is defined as follows

$$CPSNR = 10\lg\frac{2^B-1}{CMSE},$$
$$CMSE = \frac{1}{3VH}\sum_{x,y}\sum_{c\in(\text{Chroma})}\left(P_c(x,y)-\tilde{P}_c(x,y)\right)^2, \tag{16}$$

where, B is the number of data bits per chroma channel; $H\times V$ denotes the image resolution (width × height); (x,y) are the pixel coordinates; c represents a chroma component of the image; P denotes the original image's chroma value; and $\tilde{P}$ represents the processed image's chroma in all four experiments.

Table III presents the average CPSNR results obtained from the five test images over the four experiments described above. As shown in Table III, our

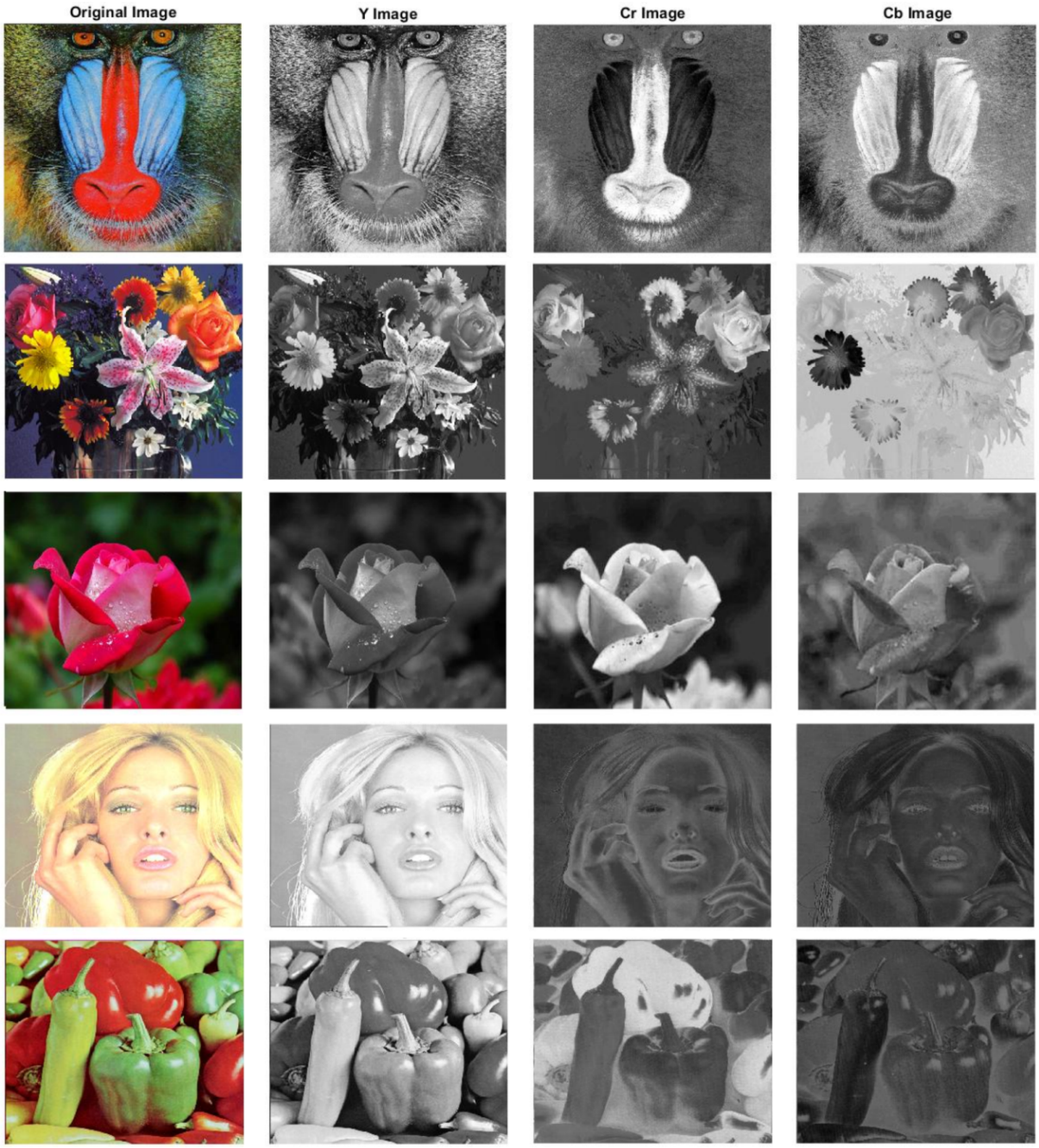

Figure 15. IP-core experiments using classical test images: Baboon, Flowers, Flower, Tiffany, and Pepper.

proposed FIR-filter-based resampling method delivers superior reconstruction performance compared to recent approaches. This improvement occurs because our method does not affect the luma component, thereby reducing the CMSE. These findings also demonstrate that our resampling method effectively preserves essential chromatic information and achieves high-quality image reconstruction.

The design is implemented on an FPGA chip using the ZCU104 Evaluation Kit. Table IV presents the synthesized resource utilization, generated using the Vivado Design Suite with all filter parameters set to their default values, a frame size of 1920×1080, and 10-bit data precision. From Table IV, it is evident that the proposed algorithm consumes relatively few FPGA resources compared to typical implementations.

5 Conclusion

The research proposes a low-complexity chroma upsampling/downsampling method based on FIR filters and an IP core design implemented on an FPGA. This approach enables efficient conversion between 4:4:4, 4:2:2, and 4:2:0 formats in the YCbCr/YUV color space while ensuring high reconstruction quality. Experimental results show that the proposed method does not affect the luma component, helps reduce the CMSE, and achieves higher CPSNR values than recent approaches. In addition, the IP core design is flexible in configuration, requires few FPGA resources, and meets real-time processing requirements. Consequently, this solution is highly appropriate for integration into contemporary video compression and processing systems.

References

[1] Z.-G. Liu, S.-Y. Du, Y. Yang, and X.-H. Ji, "A fast algorithm for color space conversion and rounding error analysis based on fixed-point digital signal processors," *Computers & Electrical Engineering*, vol. 40, no. 4, pp. 1405–1414, 2014.

[2] T. V. Nghia, "A design of the universal color-space converter IP-cores based on Field – Programmable Gate Array," *Journal of Vietnam Science and Technology*, vol. 1, no. 1, pp. 14–17, 2017.

[3] S. Zhu, C. Cui, R. Xiong, Y. Guo, and B. Zeng, "Efficient Chroma Sub-Sampling and Luma Modification for Color Image Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1559–1563, 2019.

[4] T.-L. Lin, K.-H. Jiang, and J.-S. Tu, "Low-Complexity Chroma Subsampling Using Optimal Lines of Subproblems of Pixel Distortion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 7, pp. 4778–4787, 2022.

[5] M. Azimi and M. T. Pourazad, "A Novel Chroma Processing Scheme for Improved Color Accuracy of HDR Video Content," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 718–728, 2020.

[6] T.-L. Lin, Y.-C. Yu, K.-H. Jiang, C.-F. Liang, and P.-S. Liaw, "Novel Chroma Sampling Methods for CFA Video Compression in AVC, HEVC and VVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 3167–3180, 2020.

[7] K.-L. Chung and S.-N. Chen, "An effective bilinear interpolation-based iterative chroma subsampling method for color images," *Multimedia Tools and Applications*, p. 32191–32213, 2022.

[8] A. Ahilan, B. P. Khanth, R. Ezhilarasi, and N. Muthukumaran, "Efficient luma modification-based chroma down-sampling and novel luma down-sampling with adaptive interpolation," *Signal, Image and Video Processing*, vol. 18, no. 2, p. 1415–1428, 2024.

[9] C. Guo, C. Li, J. Guo, R. Cong, H. Fu, and P. Han, "Hierarchical Features Driven Residual Learning for Depth Map Super-Resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2545–2557, 2019.

[10] X. Yang, H. Mei, J. Zhang, K. Xu, B. Yin, Q. Zhang, and X. Wei, "DRFN: Deep Recurrent Fusion Network for Single-Image Super-Resolution With Large Factors," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 328–337, 2019.

[11] "ITU-R BT.601–5—Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios," ITU-R, Tech. Rep., 1995.

[12] "ITU-R BT.709–5—Parameter values for the HDTV standards for production and international programme exchange," ITU-R, Tech. Rep., 2002.

Tran Van Nghia graduated from the University of Military Technology in 2009 and received a PhD from Moscow Technical University in 2018. Currently, the author is working at the Air Force-Air Defense Technical Institute.

Main research: signal processing, next generation radio telecommunications and television systems, algorithms for wideband signal dynamic range compression and signal predistortion/power amplifier linearization, machine learning, radar signal processing, electronic warfare, and digital microelectronic programming techniques FPGA.

Dang Trung Hieu graduated from the University of Communications and Transport in 2006. He obtained a Master's degree from the Military Technical Academy in 2009 and earned a Ph.D. from Electric Power University in 2024. He is currently a lecturer at Electric Power University.

Main research: Wireless Communications; Signal Processing for Communication.

Pham Viet Dung graduated from Le Quy Don Technical University in 2005 and received his M.Eng. from the same institution in 2012. He is currently working at the Air Defense–Air Force Technical Institute.

Main research: Signal processing, next generation radio telecommunications and television systems, algorithms for wideband signal dynamic range compression, radar signal processing, electronic warfare, and digital microelectronic programming techniques FPGA.