Research Article

Division-Free Multiquantization Scheme for Modern Video Codecs

Mousumi Das, Atahar Mostafa, and Khan Wahid

Department of Electrical and Computer Engineering, University of Saskatchewan Saskatoon, SK, Canada S7N5A9

Correspondence should be addressed to Khan Wahid, khan.wahid@usask.ca

Received 16 March 2012; Accepted 14 June 2012

Academic Editor: Dimitrios Tzovaras

Copyright © 2012 Mousumi Das et al. 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.

The current trend of digital convergence leads to the need of the video encoder/decoder (codec) that should support multiple video standards on a single platform as it is expensive to use dedicated video codec chip for each standard. The paper presents a high performance circuit shared architecture that can perform the quantization of five popular video codecs such as H.264/AVC, AVS, VC-1, MPEG-2/4, and JPEG. The proposed quantizer architecture is completely division-free as the division operation is replaced by shift and addition operations for all the standards. The design is implemented on FPGA and later synthesized in CMOS 0.18 μm technology. The results show that the proposed design satisfies the requirement of all five codecs with a maximum decoding capability of 60 fps at 187 MHz on Xilinx FPGA platform for 1080 p HD video.

1. Introduction

An evident trend in modern world is the digital convergence in the current electronic consumer products. People want the portable devices to have various functions like Video on Demand (VOD), Digital Multimedia Broadcasting (DMB), Global Positioning System (GPS) or the navigation system, Portable Multimedia Player (PMP), and so on. Due to such demand, it is necessary to support the widely used video compression standards in a single system-on-chip (SoC) platform. So the goal is to find a way so that the multicodec system achieves high performance, as well as low cost.

Most modern multimedia codecs (both encoder and decoder) employ transform-quantization pair as shown in Figure 1. A significant research has been conducted to combine and efficiently implement the transform units for multiple codecs, but little research is focused on the implementation of multiquantizer unit. A unified Inverse Discrete Cosine Transform (IDCT) architecture to support five standards (such as, AVS, H.264, VC-1, MPEG-2/4, and JPEG) is presented in [1]. A design to support the 4×4 transform and quantization of H.264 has been presented in [2]. The 8×8 transform and quantization for H.264 is presented in [3] and [4]. Several other designs based on H.264 codec have been reported in [5–10]. The authors

in [11] present a design for the quantization for AVS. The design in [12] describes an MPEG-2 encoder. In [13], another JPEG encoder is implemented for images where the quantization block is designed using multiplication and shift operation instead of division. The design in [14] describes a multistandard video decoder to support four codecs—AVS, H.264, VC-1, and MPEG-2. Silicon Image Inc. currently supplies a Multi-standard High-Definition Video Decoder (MSVD-HD) core that supports H.264, VC-1, and MPEG-1/2 codecs [15]. Their multiplexed decoder chip costs 970 K gates using TSMC 90 nm technology (including complete memory interfacing, stream reader functionality, and extra logic for context switch support).

However, none of the existing designs can compute the quantization of any video codecs. In this paper, we present a new division-free quantization algorithm (DFQA) and its efficient implementation to compute the quantization units for five multimedia codecs: JPEG [16], MPEG-2/4 [17], VC-1 [18], H.264/AVC [19], and AVS [20].

While developing the architecture, we have carefully considered all the quantization (Q) coefficients of the Q-tables of different standards and established a relationship between them. The quantization in MPEG-2/4 and JPEG is defined as the division of the DCT coefficient by the corresponding Q-values specified by the Q-matrices. On the other

TABLE 1: Description of different parameters of the proposed DFQA.

Parameters	AVS	H.264	VC-1	MPEG-2/4	JPEG
r	0	0	4	4	0
$offset$	1	0–0.5	0	0	0
$qbits$	14	$16 + (QP \bmod 6)$	0	0	0
n	15	$16 + (QP \bmod 6)$	8	8	8
qs_bit	0	0	5	5	0

hand, the two most popular video standards, H.264/AVC and AVS exploit multiplication and shift operation for the purpose of quantization to avoid the division operation for reduced computational complexity. The quantization in VC-1 is user-defined and similar to the process in MPEG-2 [21]. Based on the observation, we propose a new multiquantizer architecture to support these five codecs. The architecture is later synthesized into both FPGA and ASIC level and the cost is compared with existing designs. The design serves as a key unit in a multicodec system in transcoding applications [22] and [23].

2. Proposed Division-Free Quantization Algorithm (DFQA)

Quantization (Q) is defined as the division of the DCT coefficient by the corresponding Q -value. But in H.264 and AVS, it is done by multiplication and right shift operation. Hence these standards define their own Multiplication Factors (MF). These MFs are multiplied with the transform coefficients and finally right shifted. However, the quantization in VC-1, MPEG-2/4, and JPEG is defined as division operation only (using 8×8 matrices). As a result, it is challenge to establish a relationship that is general enough to merge all these schemes. After careful observation, a novel generalized algorithm is developed that is divided into three steps given as follows.

Step 1.

$$p = (w[i, j] \ll r) \cdot MF[i, j], \quad \text{for } i, j = 0, \dots, 7, \quad (1)$$

where $\ll r$ = left shift by r bits.

Step 2.

$$q = p + (offset \ll qbits). \quad (2)$$

Step 3.

$$y = q \gg (n + qs_bit),$$

$$\text{where } \gg (n + qs_bit) = \text{right shift by } (n + qs_bit) \text{ bits,} \quad (3)$$

where w denotes the transform coefficient, MF is the Multiplying factor, and y denotes the corresponding quantized value (level). The description of the rest of the parameters is listed in Table 1.

Moreover, QP is the quantization parameter which specifies MFs. In the next sections, we apply the general DFQA to individual codecs.

2.1. DFQA Applied to H.264. In this section, we apply the generalized DFQA to perform the quantization operation in H.264. Firstly, the transform coefficients coming from the transform unit is directly multiplied by MF as the value of r is equal to 0 for this standard and hence no left shift operation is applied to the transform coefficients. In the second stage $offset$ is left shifted by $qbits$ and then added to the result coming from the first stage. This value is finally right shifted by $n = (16 + QP \bmod 6)$ bits in the third stage, which is the final stage of the proposed DFQA. In case of H.264 as specified in [19], Multiplication Factor MF depends on $m = QP \bmod 6$ and the position (i, j) of the element as follows:

$$MF[m; i, j] = \begin{cases} M_{m0} \text{ for } (i, j) \text{ with } i = [0, 4], j = [0, 4], \\ M_{m1} \text{ for } (i, j) \text{ with } i = [1, 3, 5, 7], j = [1, 3, 5, 7], \\ M_{m2} \text{ for } (i, j) \text{ with } i = [2, 6], j = [2, 6], \\ M_{m3} \text{ for } (i, j) \text{ with } (i = [0, 4], j = [1, 3, 5, 7]) \cap (i = [1, 3, 5, 7], j = [0, 4]), \\ M_{m4} \text{ for } (i, j) \text{ with } (i = [0, 4], j = [2, 6]) \cap (i = [2, 6], j = [0, 4]), \\ M_{m5} \text{ for } (i, j) \text{ with } (i = [2, 6], j = [1, 3, 5, 7]) \cap (i = [1, 3, 5, 7], j = [2, 6]). \end{cases} \quad (4)$$

The matrix MF for H.264 is specified as

$$MF = \begin{bmatrix} 13107 & 11428 & 20972 & 12222 & 16777 & 15481 \\ 11916 & 10826 & 19174 & 11058 & 14980 & 14290 \\ 10082 & 8943 & 15978 & 9675 & 12710 & 11985 \\ 9362 & 8228 & 14913 & 8931 & 11984 & 11259 \\ 8192 & 7346 & 13159 & 7740 & 10486 & 9777 \\ 7282 & 6428 & 11570 & 6830 & 9118 & 8640 \end{bmatrix}. \quad (5)$$

2.2. DFQA Applied to AVS. Next, we apply the DFQA to perform the quantization operation of AVS. In this case only the DFQA parameters are changed according to Table 1. For AVS, based on [25], MF depends on QP. Each QP specifies one particular MF. The value of MF for the particular QP is given by Table 2. Once QP is specified the corresponding MF is multiplied with the transform coefficient in the first step of the DFQA. Again in case of AVS, the transform coefficient is

TABLE 2: Quantization table for AVS encoder.

QP	0	1	2	3	4	5	6	7
MF	32768	29775	27554	25268	23170	21247	19369	17770
QP	8	9	10	11	12	13	14	15
MF	16302	15024	13777	12634	11626	10624	9742	8958
QP	16	17	18	19	20	21	22	23
MF	8192	7512	6889	6305	5793	5303	4878	4467
QP	24	25	26	27	28	29	30	31
MF	4091	3756	3444	3161	2894	2654	2435	2235
QP	32	33	34	35	36	37	38	39
MF	2048	1878	1722	1579	1449	1329	1218	1117
QP	40	41	42	43	44	45	46	47
MF	1024	939	861	790	724	664	609	558
QP	48	49	50	51	52	53	54	55
MF	512	470	430	395	362	332	304	279
QP	56	57	58	59	60	61	62	63
MF	256	235	215	197	181	166	152	140

TABLE 3: Comparison with the existing designs (quantization only).

Arch.	Size	Tech.	No. of LUT	No. of slice	Freq. (MHz)	Supporting standards				
						H.264	MPEG-2/4	JPEG	VC-1	AVS
[2]	4 × 4	Virtex2(xc2vp7)	286	143	135	Y	o	o	o	o
[7]	4 × 4	XilinxV2P30FF896	956	—	108	Y	o	o	o	o
[8]	4 × 4	Virtex2(xc2vp7)	1954	977	94	Y	o	o	o	o
Proposed	8 × 8	Virtex4(LX60)	553	298	187	Y	Y	Y	Y	Y

Y: Yes; o: No; —: No information.

not left shifted by r bits as the value of r is 0 which is similar to the case of H.264. After that in second step this result is added to the 14 bits left shifted *offset* value, which is eventually right shifted by 15 bits in the final step.

2.3. DFQA Applied to VC-1 and MPEG-2/4. VC-1 uses multiple transform sizes but the same quantization rule is applied to all the coefficients. This standard allows both dead-zone and regular uniform quantization. In uniform quantization, the quantization intervals are identical. A dead-zone is an interval on the number line around zero, such that unquantized coefficients lying in the interval are quantized to zero. All the quantization intervals except the dead-zone are of same size—the dead-zone being typically larger. The use of dead-zone leads to substantially bit savings at low bitrates. However, at a high level the quantization process (scalar quantization where each transform coefficient is independently quantized and coded) in VC-1 is similar to the corresponding process in MPEG-2 standard. As the quantization in VC-1 is user defined and according to [21] this process is similar to the corresponding process in MPEG-2 standard in this proposed architecture, the quantization parameters for VC-1 and MPEG-2/4 standards are the same. As specified by [26], the MPEG-2/4 standard uses two quantization matrices, Intra matrix and non-intra-matrix. Here we focus only on the Intramatrix. The intraquantization matrix is shown in Figure 2:

However, we generate MF for each of the coefficient of the quantization matrix specified in Figure 2. This MF is then right shifted by 8 bits. For example, dividing a DCT coefficient coming from transform unit by a quantization value 19 can be expressed as

$$\frac{\text{DCT}}{19} = \frac{\text{DCT}}{19} * \frac{14}{14} \approx \frac{\text{DCT} * 14}{2^8}. \quad (6)$$

Hence for 19, the corresponding MF is 14. Moreover, the quantization in VC-1 and MPEG/2-4 needs the denominator of (6) to be multiplied by quantization step (QS), and in this proposed architecture for simplicity we choose $QS = 32 = 2^5$, where 5 is denoted as *qs_bit* in Table 1. So the right-hand side of (6) can be characterized as

$$\frac{\text{DCT} * 14}{2^8 * 32} = \frac{\text{DCT} * 14}{2^8 * 2^5} = \frac{\text{DCT} * 14}{2^{13}}. \quad (7)$$

The following matrix shows the MFs for the corresponding quantization matrix in Figure 2:

$$\text{MF} = \begin{bmatrix} 31 & 16 & 14 & 12 & 10 & 9 & 9 & 8 \\ 16 & 16 & 12 & 10 & 9 & 9 & 8 & 7 \\ 14 & 12 & 10 & 9 & 9 & 8 & 8 & 7 \\ 12 & 12 & 10 & 9 & 9 & 8 & 7 & 7 \\ 12 & 10 & 9 & 9 & 8 & 7 & 6 & 5 \\ 10 & 9 & 9 & 8 & 7 & 6 & 5 & 4 \\ 10 & 9 & 9 & 8 & 7 & 5 & 4 & 4 \\ 9 & 9 & 7 & 7 & 5 & 4 & 4 & 3 \end{bmatrix}. \quad (8)$$

TABLE 4: Comparison with existing designs (transform and quantization).

Arch.	Matrix size	Tech.	No. of LUT	No. of slice	Freq. (MHz)	Supporting standards				
						H.264	MPEG-2/4	JPEG	VC-1	AVS
[3]	8×8	Virtex2(XC2VP30)	2887	1624	112	Y	o	o	o	o
[4]	8×8	Virtex2(XC2V4000)	29018	14509	69	Y	o	o	o	o
[5]	8×8	Startix II	7323	—	100	Y	o	o	o	o
[6]	4×4	Virtex2(XC2V1500)	246	123	115	Y	o	o	o	o
[9]	4×4	Virtex2(2V8000FF1157)	2498	1249	81	Y	o	o	o	o
[10]	4×4	Virtex2(XC2V1500)	1386	693	99	Y	o	o	o	o
Proposed	8×8	Virtex4(LX60)	1722	972	187	Y	Y	Y	Y	Y

Y: Yes; o: No; —: No information.

FIGURE 1: Functional block diagram of a modern video encoder.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

FIGURE 2: Intra quantization matrix for VC-1 and MPEG-2/4.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

FIGURE 3: Luminance quantization matrix for JPEG.

TABLE 5: Comparison of multiquantization schemes (VLSI implementation).

Codecs	No. of gates
H.264 [2]	1.75 K
AVS [11]	7.2 K
MPEG-2/4 [12]	9 K
JPEG [24]	4.1 K
VC-1 [21]	9 K
Estimated total cost (5 codecs—standalone)	31.1 K
Proposed (all 5 codecs—shared)	19.6 K

As compared to the elements in the original intramatrix in Figure 2, the elements in (8) are smaller which helps to decrease the size of the RAM. Once MF is obtained, in the first step this MF is multiplied with the 4 bits left shifted transform coefficient as the value of r is 4 in this case. However, second step is not applicable to this standard and as a result the output of the first step is directly right shifted by $(n + qs_bit) = (8 + 5) = 13$ bits in the third step.

TABLE 6: Comparison of decoding capability of the multiquantizer schemes.

Arch.	1920 × 1080		1280 × 720		Comments
	Time to transmit 1 frame (msec)	Frame per second (fps)	Time to transmit 1 frame (msec)	Frame per second (fps)	
[2]	23.3	43	10.2	97	Supports only H.264
[7]	28.8	34	12.8	78	Supports only H.264
[8]	33.1	30	14.7	67	Supports only H.264
Proposed	16.6	60	7.4	135	Supports five codecs

FIGURE 4: (a) Overall block diagram of the proposed architecture, (b) core unit with a multiplier, and (c) lookup tables and data flow diagram.

FIGURE 5: Detailed block diagram of the entire operation with all the pipelining boundaries.

TABLE 7: Performance measure using standard images for different standards.

Sample images	PSNR (dB)			
	H.264	VC-1/MPEG-2/4	JPEG	AVS
Peppers	38.62	32.09	34.81	50.64
House	38.97	33.36	35.51	50.64
Girl	38.67	33.03	35.56	50.64
Mandrill	36.20	27.46	28.91	50.69
Tree	36.78	29.09	31.11	50.60
Lena	38.81	31.73	34.14	50.65
Lake	37.28	29.47	31.74	50.64
Mean	37.91	30.89	33.11	50.64

provided in Annex K of the JPEG standard for reference. Here we focus only on the Luminance quantization matrix shown in Figure 3.

The following matrix shows the MFs for the corresponding quantization matrix of Figure 3:

$$MF = \begin{bmatrix} 16 & 22 & 25 & 16 & 10 & 6 & 5 & 4 \\ 20 & 20 & 18 & 14 & 10 & 4 & 4 & 5 \\ 18 & 20 & 16 & 10 & 6 & 4 & 4 & 4 \\ 18 & 15 & 11 & 8 & 5 & 3 & 3 & 4 \\ 14 & 11 & 7 & 4 & 4 & 2 & 2 & 3 \\ 10 & 7 & 5 & 4 & 3 & 2 & 2 & 3 \\ 5 & 4 & 3 & 3 & 2 & 2 & 2 & 2 \\ 4 & 3 & 3 & 3 & 2 & 2 & 2 & 2 \end{bmatrix}. \quad (9)$$

Again the elements in (9) are smaller than those of the original Luminance matrix in Figure 3, which reduces the size of the RAM. After we calculate the MF, each of them is directly multiplied by the corresponding transform coefficient in the first step. Similar to VC-1 and MPEG-2/4 codecs, the second step is not applied to the JPEG standard. Finally, in third step of the proposed DFQA, the multiplied output of the initial step is right shifted by 8 bits which gives the desired quantized level. To reduce the quantization error in VC-1, MPEG-2/4, and JPEG, we also calculate MF with different amount of right shift operation. But that approach increases the hardware cost and cannot reduce the quantization error significantly. Hence we implement the quantization by using the MF with fixed 8 bit right shift.

3. Hardware Implementation of DFQA

The overall architecture of the proposed cost-sharing algorithm is shown in Figure 4(a). It can perform the 8×8 quantization of any of the five different standards as selected by the user (or another master system) using the select_standard pin. The proposed architecture contains three main blocks with four-stage pipelining: (1) lookup tables to hold the multiplying factors (MF), (2) one multiquantization unit (composed of only one shared multiplier), and (3) one finite

FIGURE 6: Comparison in terms gate count.

2.4. DFQA Applied to JPEG. Similarly for JPEG codec, we calculate MFs for each of the coefficients of the quantization matrix. The JPEG standard does not define any fixed quantization matrix. It is the prerogative of the user to select a quantization matrix. There are two quantization matrices

FIGURE 7: Cost reduction map of a decoder using the proposed architecture.

FIGURE 8: (a) Original Lena image; (b) reconstructed image using VC-1/MPEG-2/4; (c) using JPEG; (d) using H.264; (e) using AVS; (f) original Mandrill image; (g) reconstructed using VC-1/MPEG-2/4; (h) using JPEG; (i) using H.264; (j) using AVS.

state machine to control all the standards. The description of these blocks and their operations are described below. However, QP processing is not necessary for hardware implementation as it does not process data but calculates parameters used for data processing in the quantization block. Hence we assume that QP processing is previously done by software.

The core unit of the multiquantizer unit, as shown in Figure 4(b), contains one general-purpose multiplier, adder and shared right shifter. Right shift operation depends on the select_standard pin. The look-up tables in Figure 4(c) contain all the MF matrices for five standards. However, one look-up table is used at a time based on the specified standard. The multiplexer is used to select the valid data from the look-up tables. To reduce the power consumption, only one look-up table is activated at a time by the enable pin. Once the standard is chosen by the user, the desired look-up table is activated by the controller and all the other look-up tables go into the sleep mode. The control logic

assigns the enable signal as well as the MUX selection signals accordingly.

For H.264 standard according to Figure 4(c), there are six look-up tables—LUT_H_0, LUT_H_1, LUT_H_2, LUT_H_3, LUT_H_4, and LUT_H_5 for $QP = 0, 1, \dots, 5$. If QP is greater than 5 even then the same look-up tables will be used, only the q bits will be changed. For example, with $QP = 6$ and $QP = 7$, LUT_H_0 and LUT_H_1 will be used, respectively. Similarly the rest of the look-up tables will be reused as the value of QP is being increased. MUX1 selects the desired MF for H.264 standard from these six look-up tables based on QP . LUT_AVs, LUT_VC-1/MPEG, and LUT_JPEG contain MFs for AVS, VC-1/MPEG-2/4, and JPEG, respectively. The detailed block diagram of the entire operation with all the pipelining boundaries is shown in Figure 5.

According to Figure 5, the proposed architecture operates in four stages of pipeline, which consists of a row-column-generator to point the row and column of the look-up tables, several multiplexers to select the valid path, a shared

multiplier for all five standards, a shared adder, and shared shifter. Here quantization is performed by multiplication and right shift operation rather than the division operation. In the first stage of pipelining, the value of row and col is generated to point to the row and column of the look-up table by the Row-Column-Generator with the help of the controller. This row and col is used to get the Multiplication Factor, MF from look-up table. After that in second stage of the pipeline operation the multiplexer selects the valid MF. Next in stage 3 the multiplier multiplies the transform coefficient $w[i, j]$ coming from the transform unit with the desired MF $[i, j]$. Moreover, at the same time in this stage the *offset* value is left shifted by q bits by the left shifter only for H.264 and AVS standards. For VC-1, MPEG-2/4, and JPEG, this left shifter involved in stage 3 is not used. The output of the multiplier and the left shifter are added and finally right shifted in the fourth and final stage of the pipeline operation and the output register gives the quantized output. However, for VC-1, MPEG-2/4, and JPEG, the output of the multiplier is directly right shifted in this final stage and the output register returns the final quantized level. The left shift and right shift operation for AVS, H.264, VC-1, MPEG-2/4, or JPEG is chosen by the `select_standard` pin. In addition to the logic shown in Figure 5, there are pipelining registers between each stage of pipelining.

The design of the shared multiplier along with the shifter instead of the divider is the key part of the entire process. Although quantization is defined as the division operation, the AVS and H.264 standards define quantization as the multiplication and the right shift operation. To integrate the old standards like MPEG-2/4 and JPEG with AVS and H.264, we propose the whole architecture as a shared multiplication and right shift operation. Due to this strategy, the proposed architecture needs only a shared multiplier rather than both the multiplier (for AVS and H.264) and the divider (for VC-1, MPEG-2/4, and JPEG), which reduces the hardware complexity as well as the cost. Moreover, the whole design shares only one control circuit rather than using specific control circuit for each standard, which makes the design more cost effective.

4. Hardware Comparison

4.1. Performance Comparison in FPGA. The proposed architecture is implemented in Verilog HDL, and the operation is verified using Xilinx Vertex4 LX60 FPGA. The design is later synthesized using 0.18 μm CMOS technology. The proposed architecture costs 553 LUTs and 298 slices with a maximum operating frequency of 187.1 MHz. In Table 3, we summarize the performance (FPGA only) of our proposed multiquantizer supporting five standards in terms of hardware count, maximum working frequency, and supporting standards with other designs. The design in [2] has lesser hardware count than ours but supports only one standard, H.264.

For better evaluation, we have integrated the multitransform design (for five codecs) in [1] with the proposed multiquantization scheme. The combined design is implemented in Xilinx FPGA (Vertex4 LX60) and the results are shown in Table 4. This combined architecture costs

1722 LUTs (4-input), 972 slices, and 1036 registers with a maximum operating frequency of 187 MHz. Note that, the operating frequency of the multitransform design in [1] is 194 MHz. In Table 4, we compare the performance (FPGA only) of this combined multi-DCT and multiquantizer with the existing designs; those include both DCT and quantization block. The designs in [6] and [10] can support only one standard, H.264, and hence has lesser hardware than ours. Compared to the existing designs, our design has higher frequency of operation with comparable hardware count. Thus, it can be seen from the Tables 3 and 4 that the proposed design can support the highest number of popular and widely used video standards (i.e., AVS, H.264/AVC, VC-1, JPEG, and MPEG-2/4) and still consumes relatively lesser hardware cost and runs at higher operational frequency.

4.2. Performance Comparison in VLSI. The proposed design is synthesized in CMOS 0.18 μm technology using Artisan library cells. It consumes 176,911 μm^2 silicon area, 19.6 K gates, and 6.8 K standard cells. The frequency of operation is 88.5 MHz. As seen here, the operational frequency of our design is much higher in FPGA because of the use of optimized LUTs that are inherent to the type of FPGA chosen.

In Table 5, we compare the VLSI implementation of the proposed DFQA scheme with that of the estimated cost of the existing designs. Since, as of today, we have not come across to any design that can support all five codecs, we show an estimate of the projected cost. The costs of the standalone quantization units are added together to find the total estimated cost of five standalone codecs which is 31.1 K logic gates. The cost of the proposed circuit shared quantizer architecture is 19.6 K logic gates, which saves up to 36.7% than the estimated cost of five codecs. Figure 6 illustrates the comparison showing the percentage of reduction based on Table 5.

4.3. Estimated Area Saving in a Multicodec Design. In order to better assess the saving in hardware for the entire decoder, we have done a cost analysis as presented in Figure 7 where the costs of standalone and shared design are shown. The cost of a decoder for four codecs (H.264, VC-1, AVS, MPEG2/4) is taken from [14]. The cost of JPEG codec is taken from [24] and added with the previous to calculate the total cost for all five codecs. Here, MC is motion compensation, IP is intra prediction, VLD is variable length decoder, IQ is inverse quantization, and IT is inverse transform. To calculate the hypothetical cost of shared implementation, we have used the implementation cost for MC, IP, and VLD units from the shared design presented in [14]. Then the cost for IT (for shared design taken from our previous work [1]) and the cost for IQ (taken from the current work) are added. Thus, we can see that the shared design (that includes the proposed multicodec DFQA scheme) is estimated to save overall 41.1% area of a decoder compared to standalone design for five codecs.

In Table 6, we compare the decoding capability of the proposed multiquantization approach with other quantization only designs. While working at maximum capacity on Virtex4 LX60 FPGA, the proposed multiquantizer can achieve a frame rate of 60 fps (with 4:2:0 luma-chroma sampling, $187 \times 10^6 / (1,920 \times 1,080 + 2 \times 960 \times 540) = 60.1 \approx 60$). The decoding capability of other designs is also calculated using 4:2:0 sampling. It is seen in Table 6 that the proposed scheme achieves the highest decoding capacity.

4.4. Performance Evaluation Using Standard Images. In order to verify the functional correctness, in this section, we present the performance evaluation of the proposed algorithm using several standard gray-scale images. The images are first coded with transform (used in [1]) and quantization (presented here) operations, followed by a decoding (inverse) process. The quantization parameter (or quality factor) is set to 10 in all cases. The results are shown in terms of peak-signal-to-noise-ratio (PSNR) and presented below in Table 7. Figure 8 presents the original and reconstruction images of “Lena” and “Mandrill” for all five codecs.

5. Conclusion

In this paper, we present a high performance circuit shared architecture to perform the 8×8 quantization operation for five different multimedia codecs. The architecture replaces the hardware costly division operation with addition and shift operations. In addition only one control circuit is designed to control the entire architecture for all five standards. These strategies of using shared multiplier and shared control circuit result in a much lower hardware cost. The performance analysis shows that the proposed design satisfies the requirement of all codecs and achieves the competitive decoding capability. The scheme is later verified for functional correctness using standard images. Overall, the architecture is suitable for real-time application in modern multicodec systems.

References

- [1] K. A. Wahid, M. Martuza, M. Das, and C. McCrosky, “Efficient hardware implementation of 8×8 integer cosine transforms for multiple video codecs,” *Journal of Real-Time Image Processing*. In press.
- [2] R. C. Kordasiewicz and S. Shirani, “ASIC and FPGA implementations of H.264 DCT and quantization blocks,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP '05)*, pp. 1020–1023, September 2005.
- [3] S. P. Jeong and T. Ogunfunmi, “A new hardware implementation of the H.264 8×8 transform and quantization,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '09)*, pp. 585–588, April 2009.
- [4] I. Amer, W. Badawy, and G. Jullien, “A high-performance hardware implementation of the H.264 simplified 8×8 transformation and quantization,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, pp. II1137–II1140, March 2005.
- [5] G. Pastuszak, “Transforms and quantization in the high-throughput H.264/AVC encoder based on advanced mode selection,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: Trends in VLSI Technology and Design (ISVLSI '08)*, pp. 203–208, April 2008.
- [6] X. T. Tran and V. H. Tran, “Cost-efficient 130nm TSMC forward transform and quantization for H.264/AVC encoders,” in *Proceedings of the 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS '11)*, pp. 47–52, April 2011.
- [7] R. Husemann, M. Majolo, V. Guimaraes, A. Susin, V. Roesler, and J. V. Lima, “Hardware integrated quantization solution for improvement of computational H.264 encoder module,” in *Proceedings of the 18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC '10)*, pp. 316–321, September 2010.
- [8] R. Kordasiewicz and S. Shirani, “Hardware implementation of the optimized transform and quantization blocks of H.264,” in *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE '04)*, vol. 2, pp. 0943–0946, May 2004.
- [9] O. Tasdizen and I. Hamzaoglu, “A high performance and low cost hardware architecture for H. 264 transform and quantization algorithms,” in *Proceedings of the 13th European Signal Processing Conference*, pp. 4–8, September 2005.
- [10] C. P. Fan and Y. L. Cheng, “FPGA implementations of low latency and high throughput 4×4 block texture coding processor for H.264/AVC,” *Journal of the Chinese Institute of Engineers*, vol. 32, no. 1, pp. 33–44, 2009.
- [11] K. Zhang, Y. Zhu, and L. Yu, “Area-efficient quantization architecture with zero-prediction method for AVS encoders,” in *Proceedings of the Picture Coding Symposium (PCS '07)*, p. 4, November 2007.
- [12] K. Suh, K. Y. Min, K. Kim, J. S. Koh, and J. W. Chong, “Design of DPCM hybrid coding loop using single 1-D DCT in MPEG-2 video encoder,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, pp. V-279–V-282, June 1999.
- [13] H. Osman, W. Mahjoub, A. Nabih, and G. M. Aly, “JPEG encoder for low-cost FPGAs,” in *Proceedings of the International Conference on Computer Engineering and Systems (ICCES '07)*, pp. 406–411, November 2007.
- [14] C. C. Ju, Y. C. Chang, C. Y. Cheng et al., “A full-HD 60fps AVS/H.264/VC-1/MPEG-2 video decoder for digital home applications,” in *Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT '11)*, pp. 1–4, April 2011.
- [15] Silicon Image Inc., 2011, <http://www.siliconimage.com/products/index.aspx>.
- [16] CCITT recommendation T. 81, digital compression and coding continuous-tone still images, 1992.
- [17] ISO/IEC, Information technology—generic coding of moving pictures and associated audio information: video, 13818-2:1995.
- [18] Standard for television: VC-1 compressed video bitstream format and decoding process, SMPTE 421 M, 2006.
- [19] ITU-T Rec. H. 264/ISO/IEC, 14496-10 AVC, 2003.
- [20] GB/T, 20090. 1 Information technology—advanced coding of audio and video Part 1: system, Chinese AVS standard.
- [21] S. Srinivasan, P. Hsu, T. Holcomb et al., “Windows media video 9: overview and applications,” *Signal Processing*, vol. 19, no. 9, pp. 851–875, 2004.

- [22] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, 2003.
- [23] I. Ahmad, X. Wei, Y. Sun, and Y. Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, 2005.
- [24] C. J. Lian, L. G. Chen, H. C. Chang, and Y. C. Chang, "Design and implementation of JPEG encoder IP core," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 29–30, 2001.
- [25] L. Yu, F. Yi, J. Bong, and C. Zhang, "Overview of AVS-video: tools, performance and complexity," in *Visual Communications and Image Processing*, vol. 5960 of *Proceedings of SPIE*, pp. 679–690, July 2005.
- [26] J. B. Lee and H. Kalva, *The VC-1 and H.264 Video Compression Standards for Broadband Video Services*, vol. 32, Springer, Florida, Fla, USA, 1st edition, 2008.