IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 14, 2022, accepted May 9, 2022, date of publication May 12, 2022, date of current version May 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174553

Edge Computing-Based SAT-Video Coding for Remote Sensing

TRONG-AN BUI[1], (Member, IEEE), PEI-JUN LEE[2], (Senior Member, IEEE), KUAN-YU CHEN[3], CHIA-RAY CHEN[4], CYNTHIA S. J. LIU[4], AND HSIN-CHIA LIN[4]

[1]Electrical Engineering Department, National Chi Nan University, Puli, Nantou 54561, Taiwan
[2]Department of Electronics and Computer Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan
[3]Research and Development Department, Liscotech System Company Ltd., Taipei 115, Taiwan
[4]National Space Organization, Hsinchu Science Park, Hsinchu City 300, Taiwan

Corresponding authors: Pei-Jun Lee (pjlee@mail.ntust.edu.tw) and Trong-An Bui (trongan93@gmail.com)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Contract No. MOST 110-2221-E-011 -157 -MY3 and Contract NSPO-S-111089.

ABSTRACT This paper proposes an edge computing-based video coding implementation on an Earth observation satellite (SAT-video coding), which can encode video using limited resources and the power of mini/microsatellites. SAT-video coding proposes a hardware-related quantization (Q) function, hardware reduction of the motion estimation (ME) method, and simplified entropy coding (EC), which reduces the computation complexity. The hardware-related Q reduces hardware resource and power consumption by 72% and 55%, respectively, compared with traditional Q implementation. The hardware reduction of ME reduces resource use compared with regular ME implementation (59% of lookup tables [LUTs] and 79% of Registers). The total number of LUTs used for the simplified EC function is also much lower than other EC hardware implementations. The SAT-video encoder IP uses fewer hardware resources, and the power consumption is estimated at 0.0894 W at a high working frequency (125 MHz). The SAT-video encoding speed is 18.95 frames per second for 2560×2560 video. Therefore, the proposed SAT-video coding is an edge computation suitable for micro/minisatellites. The coding efficiency records the highest compression ratio at 33.8, with a peak signal-to-noise ratio of 34.46 dB. With the important task of designing edge computing based on satellite video encoding, these are adequate values for remote sensing video.

INDEX TERMS Video compression, remote sensing video, hardware acceleration, satellite.

I. INTRODUCTION

With the rapid development of satellites, the demand for remotely acquired video is overtaking that for remotely acquired images. However, onboard remote sensing video has a challenge: the contradiction between limited power, satellite hardware resource, and small capacity of transmission bandwidth between the Earth and orbit with video quality. Video compression standards such as MPEG-2, H.264/AVC, H.265/HEVC are limited in deployment to the resource-constrained onboard satellite because of the enormous computational complexity of these methods [1]–[3]. Therefore, interest in the successful development of methods with which to produce real-time remotely acquired video is growing [4], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Wenming Cao.

The downlink bandwidth is limited in video coding on a satellite architecture, so reducing the transmission capacity is essential. The proposed edge computing design for video coding on a satellite can be used to encode the raw video from the raw/uncompressed video recorded by a high-resolution satellite camera. The compressed video data can then be transmitted to the ground station via the download link.

A standardized video compression algorithm has four main features: discrete cosine transforms (DCT), coding and quantization (Q), motion compensation (MC), and entropy coding (EC). In hardware reduction methods, the motion estimation (ME) technique based on pipelined design and rapid computing of the minimum sum absolute difference (SAD) on FPGA are introduced to speed up computation [6]–[10]. Moreover, the hardware efficiencies of Q reduce the computation complexity [11], [12]. This paper proposes a highly efficient video encoding, named as SAT-video coding,

optimized for satellites' limited hardware resources and power and also achieves the target compression ratio (CR) and acceptable [13] peak signal-to-noise ratio (PSNR), based on hardware resource reduction with high-quality decompressed remotely sensing video. In this study, PSNR is determined as 25 [13], with CR is in the range between 2.03 and 8.13. The SAT-video encoder is proposed for monochrome video, and the mission with the encoding speed is at least 12 frames per second (fps).

In order to design an edge–computing video coding on satellite, increasing compression speed is the most critical research area. Parallel processing is used to reduce executing time, a graphic processing unit (GPU) and field-programmable gate array (FPGA) are commonly used. Besides, FPGAs are more efficient with less power consumption and better performance [14]. More reason, GPUs are not currently qualified against radiation. Thus, FPGAs are selected as target devices to perform the video compression onboard satellites.

Moreover, the Xilinx Kintex series is selected for edge computing on satellite because of its lower energy consumption than the Virtex series. Moreover, Kintex-7Q FPGA (defense grade) withstands temperatures of $[-55 : +125°C]$, which is radiation tolerant. Therefore, Xilinx Kintex-7 (K7) is chosen to evaluate the proposed SAT-video coding for remote sensing.

The proposed method not only focuses on presenting a new design of ME, but also on optimizing Q and EC functions to improve the performance of satellite hardware:

– Most research on improving Q/IQ in hardware implementations solves floating-point in the division, which takes up numerous resources. The proposed design focuses on solving the problem of floating points in a more straightforward direction that significantly reduces the resource usage of the hardware.

– The primary aim of this study is to provide a video coding method suitable for edge computing. Accordingly, this research proposes a simple ME method related to satellite hardware.

– This paper also presents an adaptive length coding by combining run-length and Huffman coding, which requires less computation complexity and low resources.

II. SAT-VIDEO CODING FOR REMOTE SENSING AND PROPOSED HARDWARE DESIGN

This section presents an SAT-video coding hardware implementation suitable for edge computing. Figure 1 displays the SAT-video encoding flowchart. Each block is represented for each sub-function of the video encoder process, including the Direct Memory Access (DMA) controller [15], pixel value truncation, frame switch, frame to blocks/blocks to frame, DCT/IDCT [16], Quantization/Inverse Quantization, ME, MC, residual generator, and package generator.

The SAT-video coding follows the hardware pipeline design [17] to increase the encoding speed. The new_frame_in and ref_frame_in are the the I_t and I_{t-1},

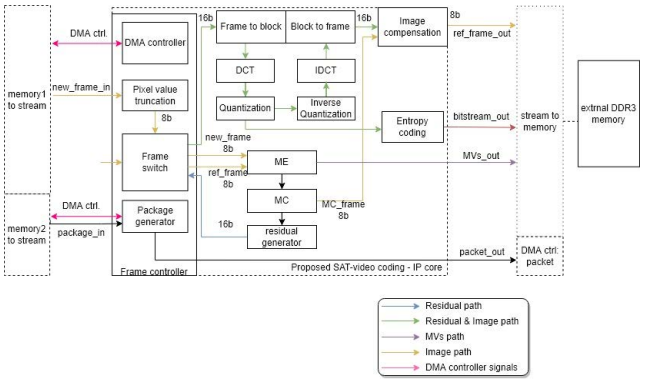


FIGURE 1. Video encoding in hardware design.

respectively. Moreover, the ref_frame_out is the I_{t-1} frame data in the case of I frame or the Image compensation of I_t and I_{t-1} in the case of P frame. The bitstream_out and MVs_outs are the entropy encoded and motion vector data. Additionally, the package generator function and packet_out handle the satellite sensors information, including satellite latitude/longitude and camera position/angle.

Moreover, the SAT-video encoding uses Advanced eXtensible Interface 4 (AXI4) [18], a high-speed data streaming, to stream data between each function. The 8 bits bandwidth is used between frame and blocks communication, ME and MC, and Entropy coding functions. Moreover, the 16 bits bandwidth is used for residual generation, transformation, and quantization functions. Moreover, DMA is used to control the cache data on DDR3 to cache reference frames, motion vectors and entropy encoded data.

A. HARDWARE DESIGN OF THE PROPOSED QUANTIZATION FUNCTION

In video coding, the quantization function is performed on the transformed coefficients $-\omega_k(x, y)$. The quantization parameter (QP) determines the step size for associating the transformed coefficients with a finite set of steps. This section presents a hardware-friendly quantization operation that combines algorithms and hardware implementations.

The quantization functions consist of equations on DC and AC coefficients. In which, AC coefficients on I and P frames are two different calculations. The quantization equation on the DC coefficient has a floor division of $\omega_k(x, y)$ by 8 [19] that requires a divider circuit – Equation (1).

$$\varphi_k(x, y) = \omega_k(x, y)//8 \tag{1}$$

With the AC coefficient, quantized values $\varphi_k(x, y)$ are quotients of division where dividends are $\omega_k(x, y)$ and $\omega_k(x, y) - \frac{\delta}{2}$ for I and P frames, respectively. The divisor is $2 \times \delta$, where δ is a variable in the range 1 to 31 [19]. Equations (2) and (3) denote the quantization method [19] of the AC coefficient for I and P frames.

$$\varphi_k(x, y) = \frac{|\omega_k(x, y)|}{(2 \times \delta)} \tag{2}$$

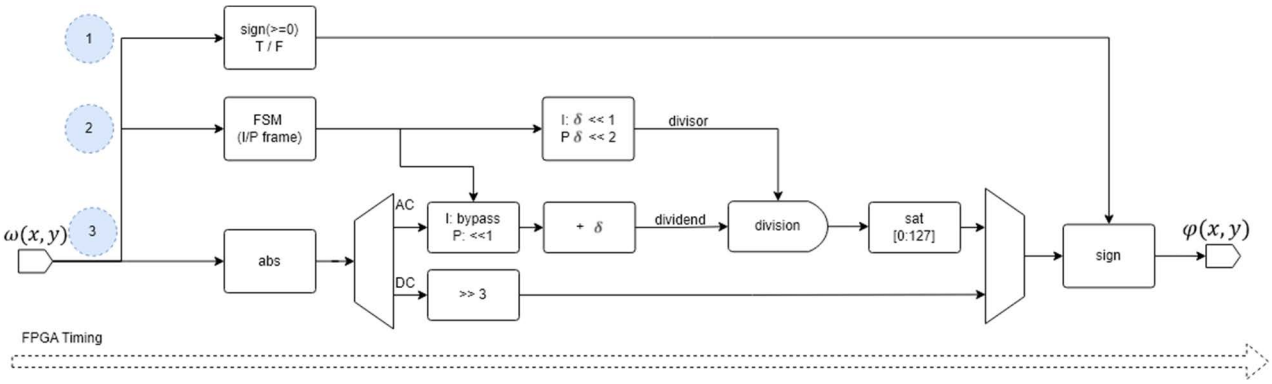


FIGURE 2. Hardware architecture of quantization function.

$$\varphi_k(x,y) = \frac{|\omega_k(x,y)| - \delta/2}{(2 \times \delta)} \tag{3}$$

Equation (4) proposes using a shifting operator instead of the floor division in Equation (1) to improve divider circuit computation.

$$\varphi_k(x,y) = \omega_k(x,y) \gg 3 \tag{4}$$

The division used in Equations (2) and (3) is a floating-point computation. Therefore, the quantization of AC coefficients requires multiple clock cycles and uses a large number of lookup tables (LUTs). Equations (5) and (6) are the rounding function of quantization functions for I and P frames, respectively, which reduce the computation complexity of floating-point division. The rounding of quantization functions is then shortened and replaced by shifting operators.

$$\begin{aligned}\varphi_k(x,y) &= \frac{2 \times \frac{|\omega_k(x,y)|}{2\times\delta} + 1}{2} \\ &= \frac{|\omega_k(x,y)| + \delta}{2 \times \delta} = \frac{|\omega_k(x,y)| + \delta}{\delta \ll 1}\end{aligned} \tag{5}$$

$$\begin{aligned}\varphi_k(x,y) &= \frac{2 \times \frac{|\omega_k(x,y)| - \frac{\delta}{2}}{2\times\delta} + 1}{2} \\ &= \frac{2 \times |\omega_k(x,y)| + \delta}{4 \times \delta} = \frac{|\omega_k(x,y)| \ll 1 + \delta}{\delta \ll 2}\end{aligned} \tag{6}$$

Proposed Equations (5) and (6) use rounding instead of floating-point divisions, which makes the proposed results different from traditional quantized values. The difference between proposed and traditional quantization functions results is in the range of [0:0.5]. However, the hardware implementation of the proposed functions significantly reduces the number of resources used for computation, as displayed in Table 1. Moreover, because the proposed hardware designs the 16 bits calculation for quantization function, the $\varphi_k(x,y)$ values domain is $[-1639{:}1639]$. Therefore, the difference between the proposed and traditional quantized values in the range of [0:0.5] is acceptable, and the proposed equations are preferable when implementing SAT-video coding edge computing.

Figure 2 is the hardware implementation of the proposed quantization function, an FPGA pipelining design, in which one column represents one stage and one row represents one pipeline in the hardware. The signal-in/out arrow, rectangle, and left/right trapezoid represent input/output of pipeline implementation, calculation in one cycle, and the split/combine signal data, respectively. The rectangular circle represents the division module.

The transformed coefficient $\omega_k(x,y)$ is input to the quantization hardware architecture. The quantization design consists of 3 parallel executions, which are denoted by ①, ② , ③. Process ① deals with the domain of positive or negative values. The most critical operation in process ① is the comparison operator to check if the value is positive or negative. This result decides the sign operator's calculation in process ③. Process ② contains two registers. The first is a finite-state machine (FSM) [20], which controls I or P frame stages. The second is the divisor determining, which is the divisor calculation in Equations (5) and (6). The main process in this design (process ③) contains the registers that calculate the absolute and dividend function of Equations (4), (5), and (6). The output of this progress calculation is the quantized value ($\omega_k(x,y)$). However, the division module is the most complicated stage in quantization function architecture. Therefore, Figure 3 presents the proposed division module implemented in Equations (5) and (6), represented by the rectangular circle in Figure 2.

The proposed division module, illustrated in Figure 3, is hardware friendly due to its use of integer to integer instead of integer to single-precision division. The hardware implementation is as follows:

- Step 1: Store dividends into the remainder buffer.
- Step 2: Subtract divisor buffer from the remainder buffer. Then, import the remainder value into the Controller.
- Step 3: Comparison: If the remainder ≥ 0, the Controller bypasses the remainder value into the remainder buffer. Then, shift the quotient buffer to the left and set the LSB bit to 1. If the remainder <0, the controller imports the previous remainder value into the remainder buffer. Then, shift the quotient buffer to the left and set the LSB bit to 0.

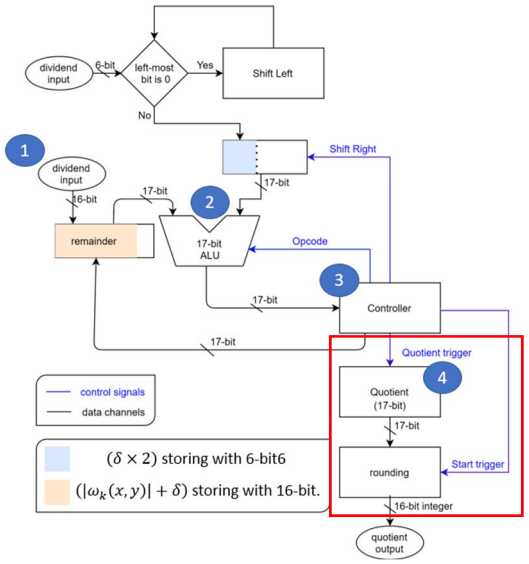


FIGURE 3. **Hardware implementation of divider operator in equations (5) and (6).**

TABLE 1. **Hardware comparison of the standard and proposed division module.**

		Q/IQ division [19] module with [21]	Q/IQ division module in the proposed design
Shifting-bit operation		At least 24 times	At most 17 times
Subtract operation		At least 24 times	At most 17 times
Memory occupation	divisor	32 bits	17 bits
	remainder	32 bits	17 bits
	quotient	24 bits	17 bits
	controller	32 bits	17 bits
Precision		High precision	Low precision
LUTs		1409	380
Registers		1787	515
Resource reduction		-	71.99%
On-Chip Power		6.3912	2.8273
Power reduction		-	55.76%

*Controller-buffer is used to replace the value into remainder-buffer if the value is less than 0.

- Step 4: The proposed hardware implementation (the red square) processes 17 times in operations of the divider to get a 17-bit quotient, which has one fraction bit in the LSB bit.

To implement the quantization function in Step 4 [19], the divider must process 24 times to obtain a 24-bits quotient to then obtain a single-precision floating-point, following the single-precision-floating point criterion from IEEE 754 standard single-precision floating-point multipliers designed [21]. The output of the division module is stored in 32 bits. However, proposed Equations (5) and (6) use rounding instead of floating-point divisions. Therefore, the proposed hardware implementation of the quantization function only processes 17 times in operations of the divider to get a 17-bit quotient. Furthermore, the proposed division output is cached in 16 bits.

The quantization method [19] implementation with the single-precision-floating point criterion (IEEE 754) and the proposed quantization division (integer-to-integer) module are compared in Table 1. Table 1 also compares simulation hardware and power resources between Q/IQ division [19] module with [21] and the proposed Q/IQ division module, implemented on Xilinx Kintex-7 XC7K410T. The use of hardware resource and power consumption are reduced by 71.99% and 55.76%, respectively.

In summary, with a small deviation ($\leq$0.5 per quantized value) but a considerable hardware improvement, the proposed quantization design is selected to implement video encoding on a satellite for edge computing applications.

B. MOTION ESTIMATION AND MOTION COMPENSATION

The video coding standards, such as H.264 and High Efficiency Video Coding (HEVC), introduce variable blocks concept with sizes from 4 × 4 to 64 × 64 to improve the ME accuracy by increasing the video compression efficiency [22]. However, this concept complicates the encoding process and uses higher hardware resources [23]. Using block 8 × 8 reduces the compressed video size at least four times compared with 4 × 4. Using a small block size increases the number of resources, the compressed data size and decompressed video quality. Therefore, the ME function uses a block size 8 × 8 to define the motion vectors (MVs) in the proposed SAT-video coding architecture.

In addition, the use of 8 × 8 blocks for DCT and Q is more optimized for hardware than other sizes [24]. To avoid excessive use of memory and resources to split blocks and optimize all functions during video encoding, all equations, including DCT, Q, ME, use a block-based structure 8 × 8. Moreover, section 2.3 proposes an EC method by combining RLE and Huffman coding to achieve suitable compression for 8 × 8 blocks. Therefore, the macroblock 8 × 8 is selected to implement edge computing for SAT-video coding for remote sensing.

Figure 4 presents the flowchart of the proposed ME function related to FPGA implementation. First, the block matching is processed from the current and reference frames using SAD calculation, presented in Equation (7), which considers a template block at position (x, y) in the current frame I_t and the candidate block at position $(x + u, y + \widehat{v})$ in the previous frame I_{t-1}.

$$SAD(u, \widehat{v}) = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} |g_t(x + i, y + i) - g_{t-1}(x + u + i, y + \widehat{v} + j)| \quad (7)$$

where $g_t(\cdot)$ and $g_{t-1}(\cdot)$ are pixel values in frames I_t, and frame I_{t-1}, respectively.

Assuming the computational complexity per block is $O(SAD(u, \widehat{v})) = O(f(n))$, then, with the video size 2560 × 2560 and a search window range (S) of 15 × 15, the complexity of block matching computation on the whole

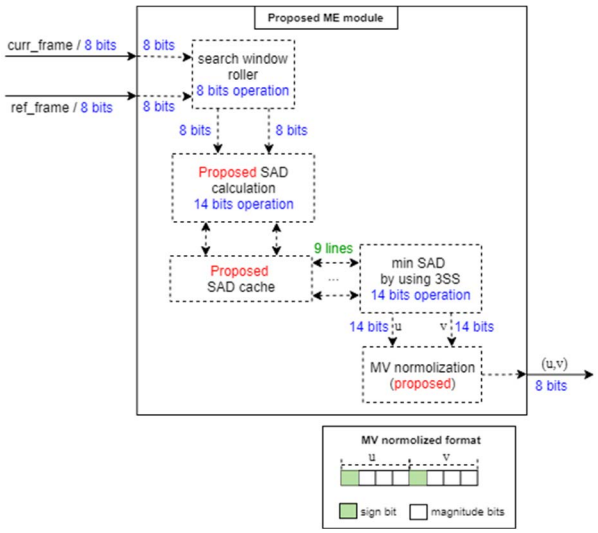


FIGURE 4. Motion vector normalization in the proposed ME module.

frame is

$$O(F(n))) = \frac{2560 \times 2560}{8 \times 8} \times (15 \times 15) \times O(f(n)) = 23040000(f(n)).$$

This paper proposes an SAD calculation and caching architecture to reduce the $O(f(n))$ and $O(F(n)))$.

In the comparative study of three-step search (3SS) and diamond search (DS) [25], 3SS has lower computation complexity and level of hardware utilization than DS [26]. Additionally, in comparing the accuracy of different search algorithms in the ME function performed on 18 various video tests, the average PSNR values of full search (FS), 3SS, and DS are 32.55, 32.25, 32.10, respectively [27]. This means that 3SS and DS have approximately 99.08%, 98.62% accuracy, respectively. Therefore, 3SS is a better solution than DS in edge computing on satellites with the accepted video quality. The proposed hardware implementation of the ME function uses a 3SS and a search window range (S) of 15×15. Therefore, the ***initial step*** determines the search step $\lambda = [4, 2, 1]$.

Figure 5 shows the detailed ME pipeline hardware design, including all the stages and the instructions in each stage.

Step 1: the central point is the position (x, y) of a pixel of I_t. Because the proposed hardware implementation uses the search step of λ, $(x + \widehat{\lambda}_k, y + \widehat{\lambda}_{k'})$ are the candidate block positions in the previous frame I_{t-1}. The proposed hardware-related SAD is represented by Equation (8).

$$SAD(\widehat{\lambda}_k, \widehat{\lambda}_{k'}) = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} |g_t(x+i, y+i) - g_{t-1}(x + \widehat{\lambda}_k + i, y + \widehat{\lambda}_{k'} + j)| \quad (8)$$

where k, k' is the λ index.

Furthermore, because of $\widehat{\lambda}_k, \widehat{\lambda}_{k'} \in [0, -\lambda_k, \lambda_k]$, the proposed ME function only needs to use $count([0, -\lambda_k, \lambda_k])^2 = 9$ search points instead of $15 \times 15 = 225$ search points

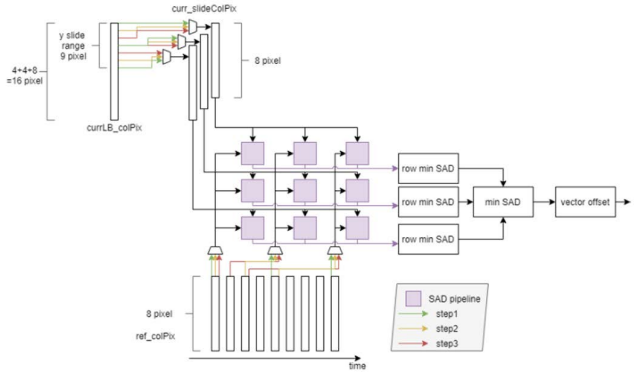


FIGURE 5. Hardware design of motion estimation function.

on traditional ME implementation. Therefore, the complexity of the proposed hardware implementation of block matching computation on the whole frame is

$$O(F'(n))) = \frac{2560 \times 2560}{8 \times 8} \times 9 \times O(f(n)) = 921600(f(n)),$$

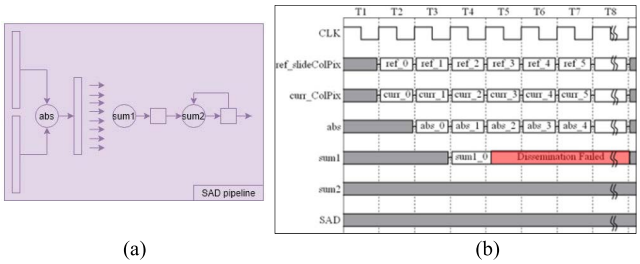
which is reduced 25 times compared with traditional block matching $O(F(n)))$.

Step 2: A hardware parallel accelerator is applied to optimize the computation complexity $f(n)$. Equation (8) is implemented to find the $SAD(\widehat{\lambda}_k, \widehat{\lambda}_{k'})$ at each search point.

Instead of calculating SAD using full-blocks, the proposed ME design process is performed on a column that takes full advantage of the parallel computation of the FPGA to calculate multiple values at the same time, thereby increasing the parallel executing speed without reducing accuracy. Thus, the input in each stage consists of two columns, one from the current frame and the other from the reference frame.

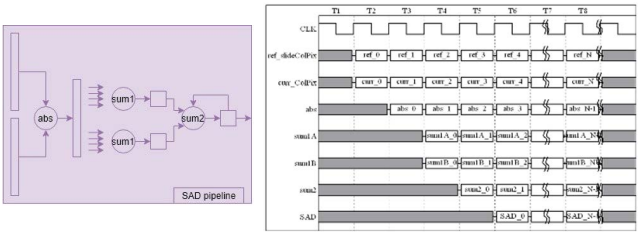
Figure 6 displays a traditional FPGA implementation of Equation (8), which uses one *abs* and two *sum* operations. Two pixels column values, one of which is taken from the reference block and the other from the current block, become the input of the SAD hardware design. The output is the SAD value between reference and current blocks. In this implementation, the *sum*1 operator sums the absolute pixel values in the reference and current columns. The sum value of the following two columns is then added to the cached *sum*1sum1using the *sum*2sum2 operator. However, with this design, the calculation *sum*1 cannot be performed in a single cycle with the 8 inputs values on each reference and current column because of dissemination failure, represented by the red regions in Figure 6 (b).

The first solution is using the smaller SAD block size [28] to solve the dissemination failure. However, this solution increases the memory required to cache the SAD values and the number of resources used for the whole video encoder. The second solution is to delay all the pipeline processes. The *sum*1 operator is delayed to the next cycle. However, extending the processing time to solve one operation calculation time latency is not desirable. Therefore, this paper proposes



(a) (b)

FIGURE 6. Time flow of the SAD design and the problems when implementing equation (7), represented by the red regions. (a) Traditional FPGA implementation of Equation (8). (b) Time flow of FPGA implementation in Figure 6 (a).



(a) (b)

FIGURE 7. Time flow of the proposed SAD pipeline design. (a) Proposed SAD's FPGA implementation. (b) Time flow of FPGA implementation in Figure 7 (a).

a SAD design related to the hardware flow, displayed in Figure 7 (a).

The data received from the *abs* operation is split into two groups. Each group contains a 1×4 column value. The *sum*1 operator is executed on each group in parallel. Figure 7 (b) illustrates the time flow of the proposed SAD pipeline method. The SAD calculation can be done in one cycle.

The proposed SAD design effectively avoids dissemination failure without delaying the process. Moreover, the proposed pipeline method indeed maintains the calculation time with less performance loss.

Step 3: find the $\min_{(\lambda_k,\lambda_{k'})\in S} SAD(\widehat{\lambda}_k, \widehat{\lambda}_{k'})$ from the 9 SAD cached values in step 2. The central point is updated to $(x + \widehat{\lambda}_k, y + \widehat{\lambda}_{k'})$ and we return to step 1 with $k + 1$.

Step 4: the motion vector is defined as follows:

$$(\lambda_k, \lambda_{k'}) = \arg \min_{(\lambda_k,\lambda_{k'})\in S} SAD(\widehat{\lambda}_k, \widehat{\lambda}_{k'}) \tag{9}$$

where k, k' is the λ index.

The ME implementation operates the calculation in 16 bits. However, because $\lambda = [4, 2, 1], \lambda_k, \lambda_{k'} \in [-7 : 7]$. Therefore, the $(\lambda_k, \lambda_{k'})$ is normalized from a (16-bits, 16-bits) structure to (4-bits, 4-bits), which makes the MV size decrease 3.5 times with the same quality, and the compressed video size is reduced.

The regular ME function caches full SAD values [29] to implement 3SS. Each search step uses 9 stages (search locations) to find the *min SAD* value. Therefore, the hardware uses 255 cache points and $9 \times 3 = 27$ stages to implement the 3SS in FPGA.

TABLE 2. The utilization comparison between 2 ME versions.

	Regular ME (without pipeline design and full SAD cache [29])	**Proposed** ME (pipeline design and optimize SAD cache)
search algorithm	3SS	3SS
stages	27	9
cache points	255	27
LUTs	9830	4030
Regs	15315	3284
BRAMs	28	32

This paper proposes an optimized ME module, which caches 27 search points (***initial and step 1***) and processes finding the min SAD by using 9 pipelines stages (**steps 2 and 3**). The speed of the ME function is significantly improved and the hardware used is optimized. Table 2 presents the results of a comparison of the performance of the ME function when using a three-step search without optimization of SAD pipeline generation and SAC cache with the proposed implementation.

Table 2 compares the regular and the proposed implementation on a test sequence of size 360×240 to test only the ME performance on the FPGA XILINX Kintex 7. Hardware resources, including LUTs and Registers (Regs), do not impact by caching resources with 2560×2560 videos on DDR. The LUTs and Regs are reduced by about 2.44 and 4.66 times, respectively. However, the block RAM (BRAM) is increased from 28 to 32 because the proposed solution uses BRAMs to cache the location of three-step search points. When designing edge computing for satellites, improvements to LUT and Reg that reduce calculation and power consumption are crucial. The proposed ME design improves the encoding performance of regular ME using the three-step search with a sequences size of 2560×2560.

Table 3 presents the utilization comparison between the proposed ME implementation and other architecture. A hardware accelerator of integer-pixel motion [29] is selected for comparison with the proposed ME, which introduces an FPGA-based SAD calculation on all the search range. A pipeline ME design [30] with eight parallel SAD calculation hardware is also compared with the proposed method. Real-time DS ME [22] accelerates the search algorithm using cache values of search points, similar to the proposed SAD caching.

The proposed ME function uses fewer LUTs and Registers than others. Although [29] has a larger video size, the LUTs, Registers, and BRAM values only represent a search range ± 16 of SAD block 16×16. Therefore, resource usage only represents a 256×256 video. Furthermore, using block size 16×16 reduces the amount of BRAM but also reduces decompressed video quality. Therefore, the performance of the proposed hardware is better than that of [29].

Moreover, the video size used for testing in [22] and [30] is smaller than that used in the simulation performed for the proposed function. The total pixel values that need to be cached and processed are only 1/3 of the proposed architecture. Therefore, the BRAM of the proposed architecture is

TABLE 3. The utilization comparison between the proposed and other ME architecture.

	Proposed ME	Integer-pixel ME [29]	ALC(M500) _APAR [30]	Real-time DS ME [22]
Frame Size	2560×2560	4096×2160	1920×1080	1920×1080
Block size	Fixed: 8×8	Fixed: 16×16	PU: 8×8 CU: 64×64	PU: 4×4 CU: 64×64
LUTs	6774	30010	36761	48258
Registers	5572	9589	12767	13351
BRAM	128	15	44	N/A
DSPs	0	0	146	N/A
Freq. (MHz)	125	100	125	199
FPGA chip	Xilinx Kintex-7	Xilinx Zynq ZC702	Xilinx Virtex 6	Xilinx Virtex-7

three times that of [30], which is acceptable. Although the video size is smaller and optimized for hardware, the LUTs and Registers used in [22] and [30] are still much higher than the proposed design. This proves that the optimization of SAD in the proposed implementation is superior, and ME is minimalist.

Xilinx Kintex consumes less energy than the Virtex series. Therefore, Xilinx Kintex-7 (K7) is selected to implement the edge computing-based SAT-video coding for remote sensing.

C. THE OPTIMIZED ENTROPY CODING

Entropy coding plays a vital role in the last stage of encoding and the first stage of the decoding process. Run-length encoding (RLE) and Huffman coding are widely used in image compression. The Octonary repetition tree (ORT) and physical-next-generation secure computing (PHY-NGSC) [31] are optimization algorithms proposed based on RLE to increase compress performance. The means of PSNR value of RLE is more or less same that of the CAVLC (<=6%) and higher than CBAC (31%) [31], [32]. However, CAVLC, CBAC, CABAC are not good in terms of compression speed. This paper proposes a compression algorithm based on RLE that reduces the computation complexity and improves the encoding speed.

Huffman coding is widely implemented on both software and VLSI design [33], [34]. Considering the hardware limitations is most important when designing code for satellite implementation. Therefore, this study proposes an adaptive length coding to combine the run-length coding and Huffman coding suitable for FPGAs implementation and complement each other.

Figure 8 presents a hardware process of EC, a FIFO progress whose input is a quantized block. This output is streamed to satellite memory. The proposed hardware implementation is a pipeline design, and all the pipeline blocks are running in parallel. Each stage follows the process outlined in Figure 8.

First, a zigzag scan [35] is applied to increase the continued zero values in the quantization coefficient. Run-length is then used to reduce the length of the quantization coefficient.

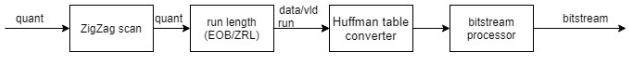


FIGURE 8. Hardware process of entropy coding.

TABLE 4. The utilization comparison of the proposed entropy coding and others.

	Proposed	Table-Based CABAC [37]	Adaptations CABAC [38]	5-stage Entropy CABAC[2]
Frame Size	2560×2560	3840×2160	1920×1080	1920×1080
Block size	Fixed: 8×8	PU: 4×4 CU: 16×16	PU: 4×4 CU: 64×64	PU: 8×8 CU: 32×32
LUTs	1749	3764	7802	3177
Freq. (MHz)	125	208	120	140
FPGA chip	Xilinx Kintex 7	Altera Stratix V GX	Altera Stratix V GX	Xilinx Zynq ZC706

Second, based on ISO/IEC 10918-1 [36], a standard Huffman lookup table (ISO/IEC 10918-1) is created and loaded on the BRAM as an initialized step. Huffman coding then encodes each value of the run-length coding results using Huffman lookup table which uses fewer resources than the original Huffman coding. Finally, the Huffman encoded data on each coefficient is concatenated to encoded frame data.

Table 4 presents the comparison of hardware performance when comparing the proposed EC with another FPGA EC. The methods in [37], [38], and [2] outline recent hardware, aware of ECs, which are calibrated with parallel processes on FPGA chips. The purpose of Entropy coding is to encode real numbers into binary values. Therefore, Table 4 compares the LUTs used in different EC designs.

Despite the video size being smaller than proposed video simulation, the number of LUTs used in [38] and [2] is much larger than the proposed EC design. Moreover, the larger video input size indicates that the total pixel values entered into EC in [37] are 1.26 times higher than the proposed design. The number of LUTs needed in [37] is 2.16 times higher than the proposed EC. Additional, the larger the block size, the larger the amount of LUTs needed. The higher the clock frequency, the more power is consumed [39]. Therefore, the proposed EC design uses fewer resources than [37], [38], and [2]. This proves that the proposed EC has a faster encoder speed but also consumes less power. Therefore, the proposed EC is suitable in edge computing-based SAT-video coding for remote sensing.

In summary, the SAT-video coding comprises a hardware-related quantization function that reduces resource usage and power consumption by up to 71.99% and 55.76%, respectively. Moreover, the amount of resource needed for the proposed SAD pipeline design in the ME function is only 1/3 of the non-pipelined design, and the SAT-video encoding uses a simple EC method, which only uses 1749 LUTs.

III. EXPERIMENT RESULTS

This section describes the test environment, the number of resources used, and the energy consumed by the

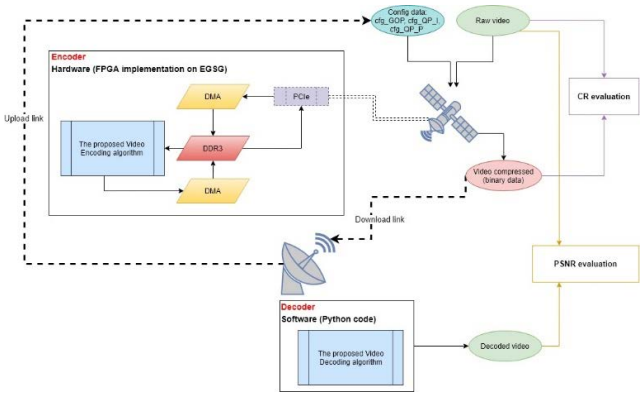


FIGURE 9. Testing/evaluation flowchart of satellite hardware.

proposed SAT-video algorithm. Values for CR and video quality (PSNR) are also presented in this section. The testing environment flowchart is presented in Figure 9.

Experiment testing uses two different systems (one acts as a satellite and the other acts as a ground station), interconnected through a network link (a signal transmission between the satellite and the ground station). First, satellite system caches raw video and video configs signals (GOP and QP) on memory, and sends it to Electrical ground support equipment (EGSE) via PCIe. The first DMA controls the signal received from the PCIe and caches video data and configuration into DDR3 (on-board EGSE). Then, SAT-video encoding is processed on EGSE, which adopts a Xilinx Kintex-7 XC7K410T and passes the radiation-tolerant test for a space mission. Verilog, a hardware description language, is used to implement SAT-video encoding. The second DMA controls the encoded data to the PCIe port. Then, the encoded data is returned to the satellite systems as a binary file. Binary data is transmitted to the ground station system via the network. After that, the SAT-video decoding, written in Python – a software description language, decompresses binary data to decompressed video.

The CR is calculated from the raw video size and compressed data size, which is the output from the PCIe port. After the decoding process, the decompressed video is used to evaluate the PSNR. The evaluation test uses QP and a group of pictures (GOP) to control the compressed video size and decompressed video quality.

Remotely sensed video coding on a satellite is a new research topic. Therefore, using uncompressed satellite videos of the right size to perform the test is impossible. All test videos have a preprocessing step to be suitable for testing. The first frame of each test video are presented in Figure 10. Table 5 provides additional information on the different testing videos. Four video sequences are used to evaluate the hardware resources used, power consumption, and CR/PSNR values. The videos include three remote sensing videos captured by satellite and one aerial video captured by UAV, each of which is 2560 × 2560 in size and has 12 frames.

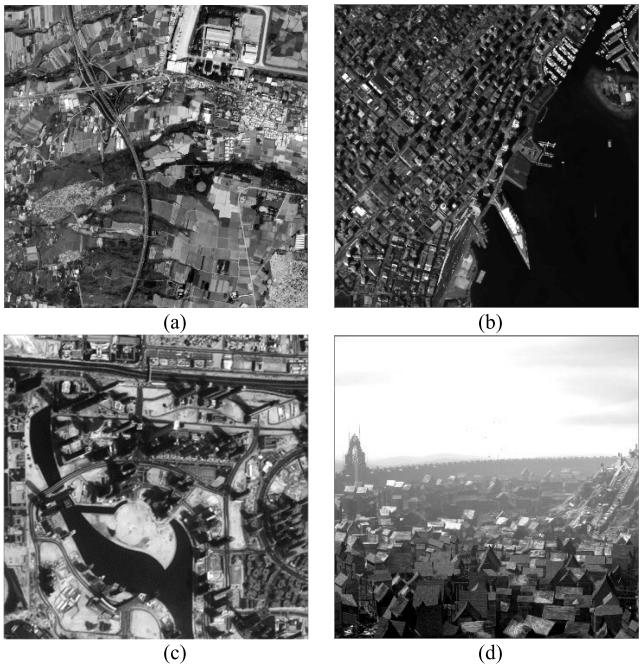


FIGURE 10. The first frames of the video test sequences. **(a) taipei_260_271 - S1, (b) vancouver_harbor - S2, (c) dubai - S3, (d) sintel - S4.**

TABLE 5. Test video sources.

Sequences	Video size	Video details
(S1) taipei_260_271	Non – crop images	From an uncompressed satellite image with satellite/image vibration
(S2) vancouver_harbor	FHD → up sample (bi-cubic) → Crop	Sample satellite video and preprocess on the Y channel only
(S3) dubai	FHD → up sample (bi-cubic)	Sample satellite video and preprocess on the Y channel only
(S4) sintel	4K → crop → up sample (bi-cubic)	Raw video with tiny moving objects and moving camera scenes captured by UAV

Table 6 and Figure 11 present the hardware resources used by each sub-function of the video encoding process. The experiment results are recorded on the EGSE board and follow the testing process outlined in Figure 9. TFB, IR, and FC in Table 6 and Figure 11 are the accessing coding block, image reconstruction, and input frame controller, respectively.

The ME uses 44% of the LUTs resource. The DCT and IDCT functions use approximately 21%. The proposed Q function optimized for floating-point only uses 2%. The EC with run-length and Huffman look-up table uses few resources at 12%.

Table 7 is the hardware implementation's resources comparison between Xilinx Kintex-7 (xc7k410) and Zynq-7000 Kintex-7 (xc7z035). Based on the data presented in Table 10, the proposed hardware implementation of the video encoder is available in both targets (Xilinx and Zynq).

Table 8 shows the percentage of resource usage compared with the total resources available in the EGSE board with XC7K410T. The entire proposed video coding algorithm uses

TABLE 6. The proposed SAT-video encoding hardware resources (unit values).

Function	Slice LUTs	Slice Registers	F7 [40] Muxes	F8 [40] Muxes	Block RAM	DSPs
EC	1749	5179	288	144	0	0
TFB	1004	1124	32	0	56	0
Q	357	398	0	0	0	0
ME	6774	5572	166	0	128	0
IR	1174	1248	24	0	48	0
IQ	202	207	0	0	0	1
IDCT	1554	7072	256	0	0	16
FC	778	1163	3	0	0	0
DCT	1724	6742	0	0	0	8
Total video encoding	15321	28708	769	144	232	25

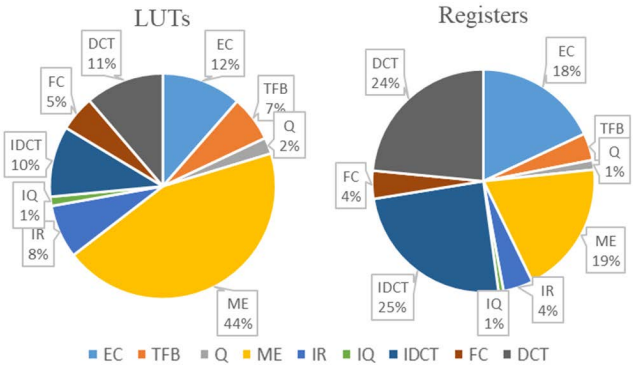


FIGURE 11. Resource usage of the proposed video encoding's sub-functions.

TABLE 7. The FPGA resources comparison on Xilinx and Zynq.

Resources/ FPGAs board	LUTs	Registers	BRAM	DSPs
proposed video encoder (used resources)	15321	28708	232	25
xc7k410 limitation (Xilinx Kintex-7)	254200	508400	795	15400
xc7z035 limitation (Zynq-7000 Kintex-7)	171900	343800	500	900

only 6.03% (LUTs), 5.65% (Registers), 29.18% (BRAM), and 1.62% (DSPs) of Xilinx Kintex-7 resources.

Therefore, the proposed SAT-video coding is a low-computation algorithm. This hardware design is, thus, a good model for remote sensing video coding on satellites.

The four test sequences in Table 5 are used to evaluate the coding efficiency. Intra coding uses resources but consumes minimal energy, whereas inter coding uses the maximum energy needed for the encoding process. Therefore, this paper considers intra-coding (GOP = 1) and inter-coding (GOP = 12, thus the coding process continued on one I frame and 11 P frames) to evaluate the hardware performance and coding efficiency. Figure 12 is the CR and PSNR evaluation of the proposed SAT-video coding on the four test sequences. QP configuration is used in the ranges between 1 and 29. Figure 12 shows the CR, PSNR, and RD curves of sequence S1, S2, S3, and S4, respectively; each row is represented for each test sequence.

TABLE 8. The proposed SAT-video encoding hardware resources (percentage).

Function	Slice LUTs	Slice Registers	F7 Muxes	F8 Muxes	Block RAM	DSPs
EC	0.69%	1.02%	0.23%	0.23%	0.00%	0.00%
TFB	0.39%	0.22%	0.03%	0.00%	7.04%	0.00%
Q	0.14%	0.08%	0.00%	0.00%	0.00%	0.00%
ME	2.66%	1.10%	0.13%	0.00%	16.10%	0.00%
IR	0.46%	0.25%	0.02%	0.00%	6.04%	0.00%
IQ	0.08%	0.04%	0.00%	0.00%	0.00%	0.06%
IDCT	0.61%	1.39%	0.20%	0.00%	0.00%	1.04%
FC	0.31%	0.23%	<0.01%	0.00%	0.00%	0.00%
DCT	0.68%	1.33%	0.00%	0.00%	0.00%	0.52%
Total video encoding	6.03%	5.65%	0.61%	0.23%	29.18%	1.62%

The results suggest that the CR and PSNR value ranges are 2.88 to 33.84 and 28.63 to 41, respectively. The proposed SAT-video coding method uses the least resource to match the satellite hardware and power consumption. Furthermore, the highest CR value can be used to reduce the amount of data transmitted from the satellite to the ground station. In addition, the PSNR value is in an acceptable range (42dB $\sim$ 28dB) with CR (2$\sim$10). The CR and PSNR recorded with four testing sequences are suitable for remote sensing video coding.

Figure 13 evaluates the CR and PSNR between proposed SAT-video coding with H.264 and HEVC on test sequence S1, which is a raw remote sensing video. In the intra-coding (GOP 1), the proposed algorithm's CR value is better than that of H.264 and HEVC. In inter-coding (GOP 6 and GOP 12), the CR only has better values in the test cases where $\delta < 16$. Because the primary research target is to provide a video coding algorithm related to a satellite, the ME is optimized by hardware equation and processing. For inter-coding, the CR value depends on both the entropy-coded and MV values. Especially for GOP configurations with more P than I frames, when increasing the δ value, the residual is decreased, then the CR value depends mainly on MV data. Thus, the CR values in the inter-coding are lower than HEVC and H.264 in cases of greater δ value, and the number of frame P is a more significant rate than I in the same GOP.

Figure 14 compares two decompressed videos with different levels of degradation; figures 14 (a) and (b) are decompressed from data of video compressed with the QP=2/GOP=1 and QP=16/GOP=12, respectively.

To design edge computing on satellites, the latency in signal processing must be optimized. Therefore, the edge computing-based SAT-video coding has the fastest encoding speed possible, providing a real-time video on the Earth station. Figure 15 displays the running time of the encoding process of one GOP, which has one I frame and one P frame. The running time is the sum of the active I frame and P frame.

The SAT-video coding uses a fixed block (8×8) for both intra- and inter-coding. Therefore, Equations (10), (11), and (12) are represented by clock cycles latencies, including latency#1 (ℓ_1), latency#2 (ℓ_2), and latency#3 (ℓ_3),

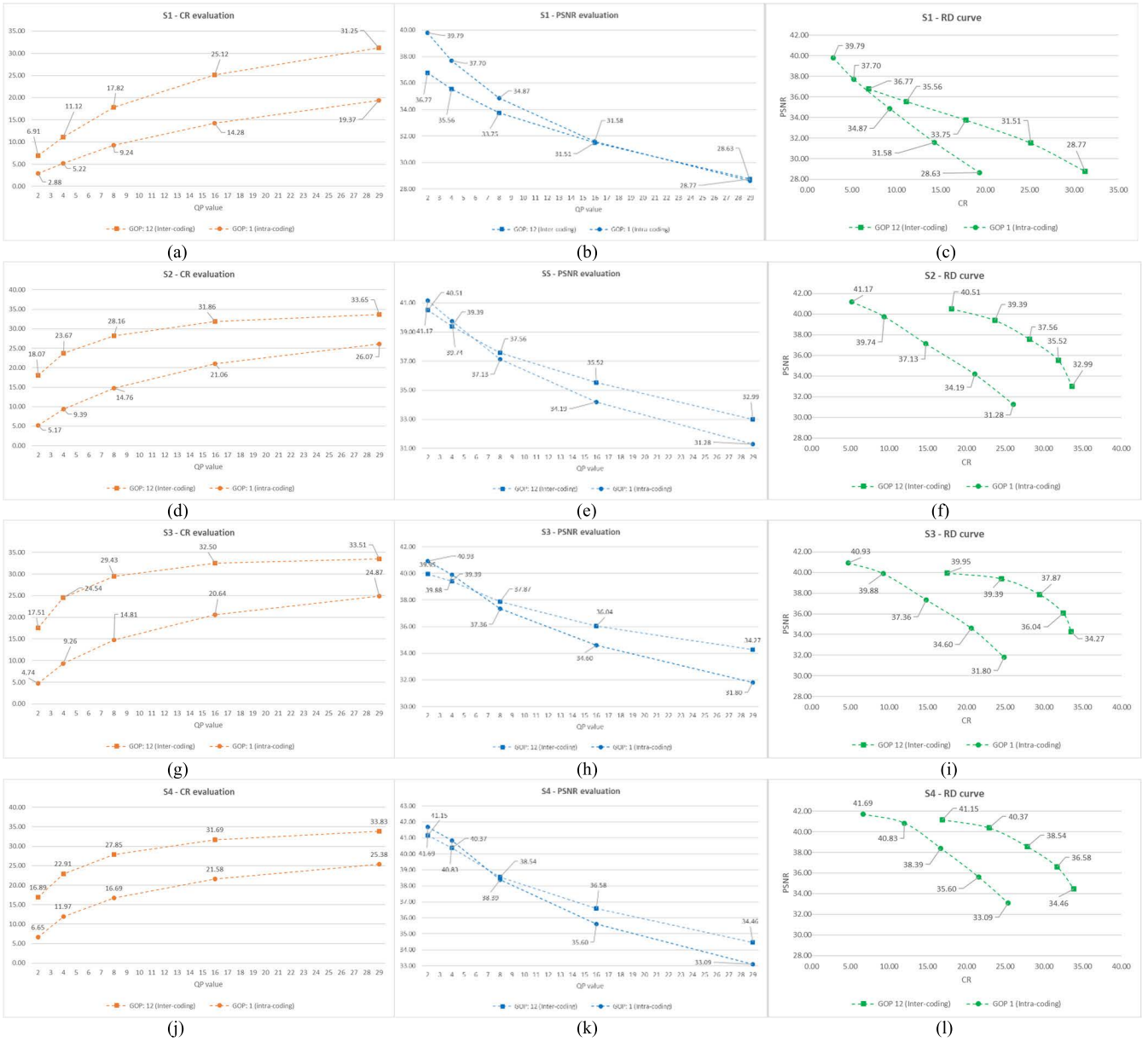


FIGURE 12. CR, PSNR, and RD curves of sequences (S1), (S2), (S3), and (S4). (a) CR evaluation of (S1). (b) PSNR evaluation of (S1). (c) RD curve of (S1). (d) CR evaluation of (S2). (e) PSNR evaluation of (S2). (f) RD curve of (S2). (g) CR evaluation of (S3). (h) PSNR evaluation of (S3). (i) RD curve of (S3). (j) CR evaluation of (S4). (k) PSNR evaluation of (S4). (l) RD curve of (S4).

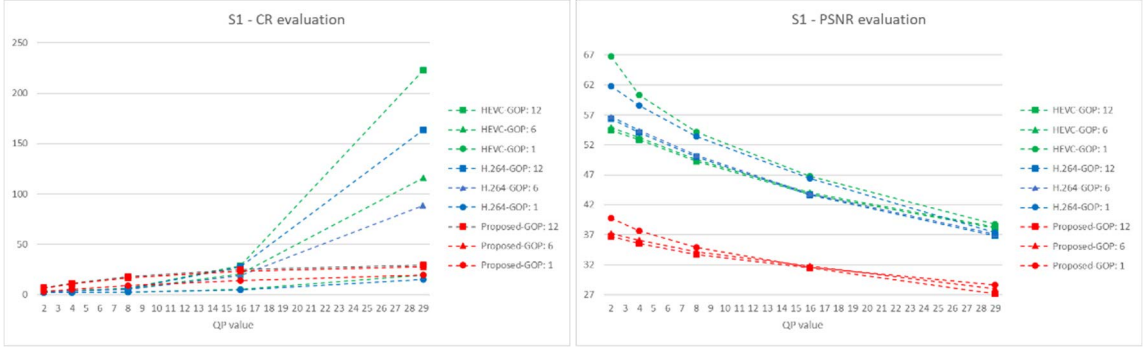


FIGURE 13. Comparing CR and PSNR values between H.264, HEVC, and the proposed algorithm.

(a) (b)

FIGURE 14. The decompressed frames of S1 with different QP and GOP configuration. (a) QP = 2/GOP = 1. (b) QP = 16/GOP = 12.

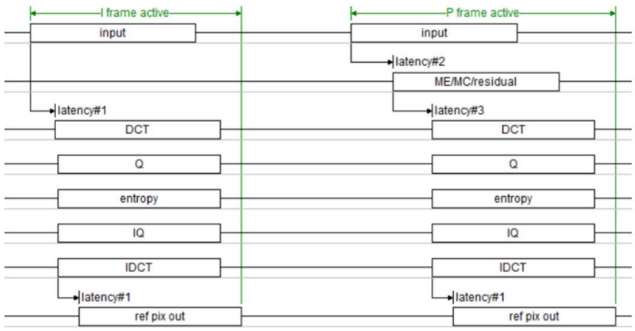


FIGURE 15. The encoding running time of one GOP has one I and one P frame.

respectively, as displayed in Figure 15.

$$\ell_1 = w \times 8 \tag{10}$$

$$\ell_2 = w \times 8 \times 2 \tag{11}$$

$$\ell_3 = w \times 32 \tag{12}$$

where w is the frame width.

According to the flow process shown in Figure 15, the frame activities of the I and P frames are defined by Equations (13) and (14), respectively.

$$\sigma_I = 2 \times \ell_1 + \rho = w \times 16 + \rho \tag{13}$$

$$\sigma_P = \ell_1 + \ell_2 + \ell_3 + \rho = w \times 56 + \rho \tag{14}$$

Finally, a GOP cycle (g) is defined by Equation (15).

$$g = (w \times 16 + \rho) + n \times (w \times 56 + \rho) \tag{15}$$

where n is the number of P frames in a GOP and ρ is the number of pixels in a frame ($\rho = 2560 \times 2560 = 6553600$).

In VLSI design in general and video coding implementation with FPGA in particular, Frequency is inversely proportional to Gates count but proportional to Power when deployed with the same method and algorithm [41]. Therefore, the proposed SAT-video coding uses a work clock frequency of 125 MHz.

Table 9 shows the GOP cycle and the running time for each GOP with a work-clock frequency of 125 MHz. Two GOP cases are used to record CR/PSNR evaluations. The running time is defined by Equation (16). The maximum encoding frame rate per second (fps) is τ– Equation (17).

$$\mathrm{T}_g = g/\varpi = g/125000000 \tag{16}$$

TABLE 9. Running time and encoding speed.

	GOP: 12	GOP: 1
g (cycle)	80261120	6594560
$\mathrm{T}_g(125MHz)$	0.64208896	0.05275648
$\tau(125MHz)$	18.69	18.95

TABLE 10. Design comparison with the state-of-the-art designs.

	Proposed	HEVC code [1]	RT intra code [2]	Accelerated inter/intra [3]
Size	2560×2560	4096×2160	1920×1080	4096×2160
LUTs	15,316	93,184	83,548	308,000
Registers	28,705	n/a	28303	n/a
DSPs	25	481	34	862
fps	18.69 ~18.95	30	30	30
FPGA chip	Xilinx Kintex-7	Arria II GX	Zynq ZC706	Arria 10 (CPU+FPGA)
Freq.	125 MHz	200 MHz	140 MHz	125 MHz

$$\tau = n/\mathrm{T}_g \tag{17}$$

where n is the total frames in a GOP.

The hardware designs of the proposed architecture and other video coding architectures are presented in Table 10. The implementations [1] and [2] have lower coding efficiency relative to HEVC standard, (PSNR) −0.16 dB and −3%, respectively. [3] has the same coding efficiency as HEVC. The efficient coding of [1], [2] and [3] are better than proposed SAT-video coding. However, the hardware required is much larger than the proposed SAT-video coding implementation. Studies [1] and [3] use test video sizes 1.35 times larger, but the number of LUTs used are 6 and 20 times higher. Moreover, the number of DSPs used in the proposed design has the lowest value among the methods investigated. The architecture proposed in [2] is designed for a smaller video size than the proposed architecture but the hardware consumption parameters are larger than that of the proposed video coding implementation.

The power consumption increases almost linearly with the clock frequency [39]. The proposed SAT-video coding is designed to work with the same frequency with [3], and a lower frequency than design [1], [2]. Therefore, the proposed hardware implementation and [3] have lower power consumption than other designs. The proposed SAT-video coding hardware only supports a maximum coding speed of 18.95 fps on the Xilinx Kintex 7 410T. This value is lower than [1], [2] and [3]. However, this design serves to deploy edge computing on satellites, and thus the energy consumption and resource usage are more important. Moreover, the speed of 18.95 fps is sufficient for real-time satellite remote sensing tasks.

Moreover, this research uses Xilinx Power Estimator to estimate the power consumption of the proposed video encoder. The total on-chip power consumption is equal to the

sum of device static power, and the design power accounts for 2.892 W, of which GTX, dynamic, and device static account for 0.182 W (6%), 2.462 W (85%), and 0.248 W (9%), respectively. The proposed video encoding in hardware design, presented in Figure 1, only uses 4.91% LUTs, 4.99% Registers, 27.17% BRAMs, and 1.62% DSPs. Therefore, the power consumption of video encoding IP core is estimated at 0.0894 W, which is a low consumption for satellite missions.

The proposed SAT-video coding is an optimized video coding and hardware design for real-time encoding on a satellite. The priority target is to use fewer resources and consume less power. Based on the comparison results displayed in Table 10, the proposed design has the lowest LUTs, Registers, DSPs, and frequency clock among most miniature chip design techniques, and thus, the proposed hardware consumes the least energy of all the designs. The fps recorded in the proposed design is not the highest. Nonetheless, the satellite video retains a wide variety of applications, especially in identifying moving objects on the Earth's surface.

IV. CONCLUSION AND DISCUSSION

This study designs an edge computing-based SAT-video coding for remote sensing. The proposed SAT-video coding algorithm related to hardware has a fast encoding speed (up to 18.95 fps) and low energy consumption (0.0894 W) for 2560×2560 video at 125-MHz, which adapts to a satellite's hardware. The proposed SAT-video coding has a CR of up to 33.8, which reduces the data transmission bandwidth to the Earth. Moreover, the decompressed video's quality is evaluated for remote sensing mission with PSNR in the range of [28.63:41.69]. The proposed algorithm is SAT-video coded and implemented on low-value hardware, suitable for the limit of micro/mini-satellites. CR value is within the allowable range of the download link from the satellite to the ground station. In addition, PSNR values and decompressed video perform well for remote sensing tasks.

REFERENCES

[1] G. Pastuszak and A. Abramowski, "Algorithm and architecture design of the H.265/HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 210–222, Jan. 2016, doi: 10.1109/TCSVT.2015.2428571.

[2] S. Atapattu, N. Liyanage, N. Menuka, I. Perera, and A. Pasqual, "Real time all intra HEVC HD encoder on FPGA," in *Proc. IEEE 27th Int. Conf. Appl.-Specific Syst., Architectures Processors (ASAP)*, Jul. 2016, pp. 191–195, doi: 10.1109/ASAP.2016.7760792.

[3] P. Sjovall, V. Viitamaki, A. Oinonen, J. Vanne, T. D. Hamalainen, and A. Kulmala, "Kvazaar 4K HEVC intra encoder on FPGA accelerated airframe server," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2017, doi: 10.1109/SiPS.2017.8109999.

[4] C. Duan, L. Tang, C. Wu, C. Li, C. Li, and B. Zhao, "Real-time remote sensing video compression for on-orbit heterogeneous platform," in *Proc. IEEE Int. Conf. Signal, Inf. Data Process. (ICSIDP)*, Dec. 2019, pp. 1–4, doi: 10.1109/ICSIDP47821.2019.9173512.

[5] X. Wang, R. Hu, Z. Wang, and J. Xiao, "Virtual background reference frame based satellite video coding," *IEEE Signal Process. Lett.*, vol. 25, no. 10, pp. 1445–1449, Oct. 2018, doi: 10.1109/LSP.2018.2862145.

[6] J. Olivares, J. Hormigo, J. Villalba, I. Benavides, and E. L. Zapata, "SAD computation based on online arithmetic for motion estimation," *Microprocessors Microsyst.*, vol. 30, no. 5, pp. 250–258, Aug. 2006, doi: 10.1016/j.micpro.2005.12.006.

[7] K. Babionitakis, G. A. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, and N. Vlassopoulos, "A real-time motion estimation FPGA architecture," *J. Real-Time Image Process.*, vol. 3, nos. 1–2, pp. 3–20, Mar. 2008, doi: 10.1007/s11554-007-0070-9.

[8] M. Kthiri, H. Loukil, A. B. Atitallah, P. Kadionik, D. Dallet, and N. Masmoudi, "FPGA architecture of the LDPS motion estimation for H.264/AVC video coding," *J. Signal Process. Syst.*, vol. 68, no. 2, pp. 273–285, Aug. 2012, doi: 10.1007/s11265-011-0614-x.

[9] S. Rehman, R. Young, C. Chatwin, and P. Birch, "An FPGA based generic framework for high speed sum of absolute difference implementation," *Eur. J. Sci. Res.*, vol. 33, no. 1, pp. 6–29, 2009.

[10] M. Kthiri, P. Kadionik, H. Levi, H. Loukil, A. B. Atitallah, and N. Masmoudi, "An FPGA implementation of motion estimation algorithm for H.264/AVC," in *Proc. 5th Int. Symp. I/V Commun. Mobile Netw.*, Sep. 2010, pp. 1–4, doi: 10.1109/ISVC.2010.5654826.

[11] L. Braatz, L. Agostini, B. Zatt, and M. Porto, "A multiplierless parallel HEVC quantization hardware for real-time UHD 8K video coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050704.

[12] T. Dias, N. Roma, and L. Sousa, "High performance IP core for HEVC quantization," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 2828–2831, doi: 10.1109/ISCAS.2015.7169275.

[13] N. Thomos, N. V. Boulgouris, and M. G. Strintzis, "Optimized transmission of JPEG2000 streams over wireless channels," *IEEE Trans. Image Process.*, vol. 15, no. 1, pp. 54–67, Jan. 2006, doi: 10.1109/TIP.2005.860338.

[14] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, 2012, pp. 47–56, doi: 10.1145/2145694.2145704.

[15] (2021). Xilinx. *AXI DMA Controller*. Vivado Design Suite User Guide. Accessed: Nov. 18, 2021. [Online]. Available: https://www.xilinx.com/products/intellectual-property/axi_dma.html

[16] E. P. Wu, T. A. Bui, K. Chen, and P. J. Lee, "Hardware implementation of DCT/IDCT sharing for HEVC/MPEG video coding," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2020, pp. 1–3, doi: 10.1109/ICCE46568.2020.9043100.

[17] S. Shilparani, K. S. Reddy, and V. S. Reddy, "FPGA implementation of FIR filter architecture using MCM technology with pipelining," in *Proc. 2nd Int. Conf. Emerg. Technol. (INCET)*, May 2021, pp. 1–5, doi: 10.1109/INCET51464.2021.9456384.

[18] *AXI Reference Guide UG761 (v13.1)*, Xilinx, San Jose, CA, USA, 2011.

[19] D. R. Bull, "Video coding standards," in *Communicating Pictures*. Dordrecht, The Netherlands: Elsevier, 2014, pp. 411–449, doi: 10.1016/b978-0-12-405906-1.00012-x.

[20] J. Wang and W. Tepfenhart, *Formal Methods in Computer Science*. Boca Raton, FL, USA: CRC Press, 2019.

[21] G. S. S. V. Prasada, G. Seshikala, and S. Niranjana, "Research on IEEE 754 standard single precision floating point multipliers designed using Urdhva Triyagbhyam sutra of vedic mathematics," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2, pp. 2990–2993, Sep. 2019, doi: 10.35940/IJRTE.B1382.0982S1119.

[22] R. Khemiri, H. Kibeya, H. Loukil, F. E. Sayadi, M. Atri, and N. Masmoudi, "Real-time motion estimation diamond search algorithm for the new high efficiency video coding on FPGA," *Anal. Integr. Circuits Signal Process.*, vol. 94, no. 2, pp. 259–276, Feb. 2018, doi: 10.1007/s10470-017-1072-6.

[23] K. Singh and S. R. Ahamed, "Computationally efficient motion estimation algorithm for HEVC," *J. Signal Process. Syst.*, vol. 90, no. 12, pp. 1713–1727, Dec. 2018, doi: 10.1007/s11265-017-1321-z.

[24] D. Puchala, "Approximate calculation of 8-point DCT for various scenarios of practical applications," *EURASIP J. Image Video Process.*, vol. 2021, no. 1, pp. 1–34, May 2021, doi: 10.1186/s13640-021-00557-3.

[25] C. Yadav and R. Kunwar, "Comparative study of three step search and diamond search algorithm for motion estimation," *J. Graphic Era Univ.*, vol. 4, pp. 66–71, Jan. 2016.

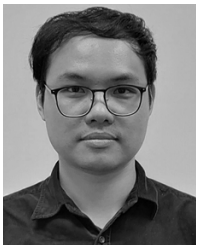
[26] K. Priyadarshini and A. K. T. Sulthana, "Improved fast block matching motion estimation using multiple frames," in *Proc. Int. Conf. Image Process. Capsule Netw.*, Bangkok, Thailand, May 2020, doi: 10.1007/978-3-030-51859-2_64.

[27] Z. Pan, R. Zhang, W. Ku, and Y. Wang, "Adaptive pattern selection strategy for diamond search algorithm in fast motion estimation," *Multimedia Tools Appl.*, vol. 78, no. 2, pp. 2447–2464, Jul. 2018, doi: 10.1007/S11042-018-6353-2.

[28] M. Nagaraju, S. K. Gupta, V. Bhadauria, and D. Shukla, "Design and implementation of an efficient mixed parallel-pipeline SAD architecture for HEVC motion estimation," in *Advances in VLSI, Communication, and Signal Processing* (Lecture Notes in Electrical Engineering), vol. 683, D. Harvey, H. Kar, S. Verma, and V. Bhadauria, Eds. Singapore: Springer, doi: 10.1007/978-981-15-6840-4_50.
[29] B. Mohamed, A. Shalaby, and M. S. Sayed, "High-level synthesis hardware accelerators of integer-pixel motion estimation of HEVC on SoC-FPGA platform," in *Proc. 2nd Eur. Middle East North Afr. Regional Conf. Int. Telecommun. Soc. (ITS), 'Leveraging Technol. Growth'*, Aswan, Egypt, Feb. 2019, pp. 1–8.
[30] F. A. Ghani, "FPGA implementations of motion estimation algorithms using vivado high-level synthesis," M.S. thesis, Graduate School Eng. Natural Sci., Sabancí Univ., Tuzla, Turkey, Aug. 2017.
[31] P. Girishwaingankar and S. M. Joshi, "The PHY-NGSC-based ORT run length encoding scheme for video compression," *Int. J. Image Graph.*, vol. 20, no. 2, Apr. 2020, Art. no. 2050007, doi: 10.1142/S0219467820500072.
[32] H. Jung, S. Choi, and S.-I. Chae, "Coding efficiency of the context-based arithmetic coding engine of AVS 2.0 in the HEVC encoder," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2015, pp. 377–378, doi: 10.1109/ICCE.2015.7066452.
[33] N. Sharma and U. Batra, "An enhanced Huffman-PSO based image optimization algorithm for image steganography," *Genet. Program. Evolvable Mach.*, vol. 22, pp. 1–17, Jan. 2021, doi: 10.1007/s10710-020-09396-z.
[34] A. Choudhary, P. Jain, and S. Pateriya, "Adaptive VLSI framework for image reducing algorithms," *Int. J. Res. Eng. Sci.*, vol. 8, no. 12, pp. 68–74, 2020.
[35] V. Sze and D. Marpe, *Entropy Coding in HEVC*. Cham, Switzerland: Springer, 2014, pp. 209–274, doi: 10.1007/978-3-319-06895-4_8.
[36] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, Standard ISO/IEC 10918-1, I.-T. Rec. T.81, 1994, p. 186.
[37] Y. Zhang and C. Lu, "A highly parallel hardware architecture of table-based CABAC bit rate estimator in an HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1544–1558, May 2019, doi: 10.1109/TCSVT.2018.2830126.
[38] Y. Zhang and C. Lu, "High-performance algorithm adaptations and hardware architecture for HEVC intra encoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 2138–2145, Jul. 2019, doi: 10.1109/TCSVT.2019.2913504.
[39] M.-C. Chien, J.-Y. Huang, and P.-C. Chang, "Complexity control for H.264 video encoding over power-scalable embedded systems," in *Proc. IEEE 13th Int. Symp. Consum. Electron.*, May 2009, pp. 221–224, doi: 10.1109/ISCE.2009.5157007.
[40] K. Chapman, "Multiplexer design techniques for datapath performance with minimized routing resources application note (XAPP522)," Xilinx, San Jose, CA, USA, Tech. Rep. XAPP522, Oct. 2014, p. 4.
[41] F. L. L. Ramos, A. V. P. Saggiorato, B. Zatt, M. Porto, and S. Bampi, "Residual syntax elements analysis and design targeting high-throughput HEVC CABAC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 475–488, Feb. 2020, doi: 10.1109/TCSI.2019.2932891.

TRONG-AN BUI (Member, IEEE) received the B.Sc. degree from the Faculty of Information Technology, Ho Chi Minh University of Education, Vietnam, in 2015, and the M.Sc. and Ph.D. degrees from the Electrical Engineering Department, National Chi Nan University, Taiwan, in 2018 and 2022, respectively. His current research interests include neural networks and artificial intelligence, computer vision and image processing, video coding and data compression, remote sensing, and satellite imagery.

PEI-JUN LEE (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, National Taiwan University, in 2004. In August 2015, she was promoted to a Distinguished Professor with the Department of Electrical Engineering, National Chi Nan University, Taiwan. Since August 2021, she has been a Professor with the Department of Electronics and Computer Engineering, National Taiwan University of Science and Technology, Taiwan. Her research interests include 2D/3D image conversion, 3D/multiview video coding, video communication, video compression, image processing, robot fish, and FPGA system application. She was a recipient of the First Runner-Up and the Outstanding Advisor of the Macronix Golden Silicon Awards Semiconductor Design and Application Competition from MXIC Company Ltd., in 2008.



KUAN-YU CHEN has been a Senior FPGA Engineer of Liscotech System Company Ltd., Research and Development Department, since 2016. His work focuses the FPGA design, IP design, video compression algorithm design, and embedded systems.

CHIA-RAY CHEN received the Ph.D. degree from the Institute of Aeronautics and Astronautics, National Cheng Kung University, Taiwan. Since October 1995, he has been working with the National Space Program Office (now National Space Organization), Taiwan. He was the Director of the Mechanical Engineering Division, NSPO, from 2008 to 2019. He has been promoted to the position of the Formosat-8 Program Director, since 2019. His research interests include the areas of satellite system engineering, optical remote sensing instrument, satellite image applications, and satellite thermal control design.

CYNTHIA S. J. LIU has been working on systems engineering, satellite data processing, and analysis for more than ten years at NSPO. She is currently the Director of the Satellite Image Division, National Space Organization (NSPO), Taiwan. Her expertise is mainly in remote sensing instrument (RSI) systems engineering and satellite image processing, especially on data injection, geometric/radiometric correction, and image restoration techniques. Her recent interests include the FORMOSAT-5 image application and promotion and also preparation for next-generation FORMOSAT-8 images system development.

HSIN-CHIA LIN received the Ph.D. degree in space science from the National Central University, Chungli, in 2012. Since 1992, he has been working in satellite development with the National Space Organization (NSPO), Taiwan. His research interests include the design of the command and data handling and the remote sensing instruments on FORMOSAT satellites. Currently, he is the CubeSat Project Manager with the National Space Organization.