Power-Efficient Error-Resiliency for H.264/AVC Context-Adaptive Variable Length Coding

Muhammad Shafique, Bruno Zatt, Semeen Rehman, Florian Kriebel, Jörg Henkel
Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems, Karlsruhe, Germany
muhammad.shafique@kit.edu

***Abstract*—Technology scaling has led to unreliable computing hardware due to high susceptibility against soft errors. In this paper, we propose an error-resilient architecture for Context-Adaptive Variable Length Coding (CAVLC) in H.264/AVC. Due to its context-adaptive nature and intricate control flow CAVLC is very sensitive to soft errors. An error during the CAVLC process (especially during the context adaptation or in VLC tables) may result in severe mismatch between encoder and decoder. The primary goal in our error-resilient CAVLC architecture is to protect codeword/codelength tables and context adaptation in a reliable yet power efficient manner. For reducing the power overhead, the tables are partitioned in various sub-tables each protected with variable-sized parity. Moreover, for further power reduction, our approach incorporates state-retentive power-gating of different sub-tables at run time depending upon the statistical distribution of syntax elements. Compared to the unprotected case, our scheme provides a video quality improvement of 18dB (averaged over various fault injection cases and video sequences) at the cost of a 35% area overhead and 45% performance overhead due to the error-detection logic. However, partitioned sub-tables increase the potential for power-gating, thus bring a leakage energy saving of 58%. Compared to state-of-the-art table protection, our scheme provides 2x reduced area and performance overhead. For functional verification and area comparison, the architecture is prototyped on a Xilinx Virtex-5 FPGA, though not limited to it. For the soft errors experiments, evaluation of error-resiliency and power efficiency, we have developed a fault injection and simulation setup.**

I. Introduction and Related Work

The H.264/AVC coding standard [1] enables a wide-range of video recording applications (from high-end cinema to low-end mobile devices) as it provides double compression at the same visual quality compared to earlier coding standards [2]. However, the H.264 encoder has an approximately 10× higher computational complexity compared to the MPEG-4 advanced simple profile [2]. This increased complexity has been well countered by the advanced multimedia computing platforms (or coprocessors [29][30]) fabricated using modern process technologies. However, a serious side-effect of these smaller transistors is their increased vulnerability to soft errors due to (i) lower-threshold and operating voltages, (ii) particle strikes[1] on the transistors, and (iii) tight noise margins [3][4]. These soft errors manifest themselves as bit flips in the hardware and jeopardize a correct program execution. This gives rise to the need of considering the computational reliability as a critical design parameter [4][5], which is not only crucial for the current H.264/AVC video coding standard but also for the upcoming video coding standards like H.265/HEVC [20].

State-of-the-art error-resilient techniques in image and video processing have primarily exploited the inherent resilience of different parts of the image and video codecs. The works in [13] and [14] target fault tolerant JPEG2000, where in [14] aggressive voltage scaling is applied at the encoder side to save power at the cost of errors and error-concealment methods are applied at the decoder side. Authors in [15] explore the decoded picture buffer of an H.264 decoder for memory failures under very low operational voltages. The approach in [16] performs error classification based on application annotation and uses conventional correction methods like copying the neighboring pixel/block data in case an error is being detected. Authors in [5] propose a technique for H.264 encoder that computes the block checksum of the original and residual data of Macroblocks in order to protect the prediction path. The soft error tolerance of motion estimation is investigated in [3][10]. These state-of-the-art techniques tolerate errors in *either* (a) main frame/picture buffer [13]-[16], which has inherent resilience as a bit flip in a pixel value of a frame is less visible; *or* (ii) motion estimation which is inherently resilient, as an error during the matching process may lead to a sub-optimal motion vector[2] (i.e. more residual data), but not the wrong prediction.

In contrast, *due to its context adaptive nature, complex bitstream structure, and multiple VLC tables, the Context-Adaptive Variable Length Coding (CAVLC) is highly susceptible to soft errors on advanced multimedia computing platforms*: A soft error during the CAVLC may lead to significant visual artifacts due to encoder-decoder mismatches. For instance bit flips in a codeword or part of the bitstream or a wrong table selection during CAVLC may result in wrong residual data (thus wrong reconstructed pixels) that may affect a complete Macroblock (MB, 16x16 pixel block) and propagate to the subsequent frames. Moreover, faults in certain syntax elements during the CAVLC may lead to decoder and/or encoder crashes (e.g., an abort due to an unrecognized codeword). Note: traditional error-concealment methods from communication reliability domain may not efficiently cope with these artifacts, as they model data corruption as packet losses over a noisy channel [11]. Contrarily, in case of computational reliability for soft errors, the data is available but it has an incorrect value which might still be decodable but leads to visual artifacts [5][6] that propagate to other parts of the video frame or even the subsequent frames, i.e. an encoder-decoder mismatch (see Fig. 1). Furthermore, these soft errors may occur in all profiles (like High or Main profiles) which do not exhibit the error concealment algorithms.

Conventional computational reliability techniques like dual or triple modular redundancy [4][5][7], pipeline protection [8], software level fault-tolerance [9] incur significant (≥2x) area, performance, and/or power overhead as they *ignore* the application-specific knowledge, e.g., the effect of quantization parameter on the vulnerability of CAVLC. The work of [17] provides error-tolerance for the header coding process in CAVLC. However, it does not provide comprehensive methods for reliable coefficient coding. Moreover, the work of [17] provides limited power-efficiency. In multimedia systems with advanced video codecs (like H.264), area and power are critical design parameters. **Therefore, power-efficient error-resilience for H.264 CAVLC is desirable, especially for coding the transformed quantized coefficients which occupy the major portion of the coded bitstream data**.

A. Motivational Case Study: Soft Error Analysis of CAVLC

We have performed a fault injection case study to analyze the effects of soft errors on the subjective video quality of various sequences (see details of the experimental setup in Section V). Faults are injected in the memory (storing VLC *codeword* and *codelength* tables) during the encoding of different syntax elements, i.e. level or quantized coeffi-

[1] High-energy neutron/proton from the cosmic rays or low-energy alpha particles from the packing materials [3].

[2] The probability of soft errors in the best motion vector is very low due to small amount of data. The work of [17] provides protection mechanisms for this.

978-3-9810801-8-6/DATE12/©2012 EDAA

cients, run, number of total non-zero coefficients, trailing ones, and total zeros (see an overview of the H.264 CAVLC in Section II). Our experiments unveiled that some syntax elements (like trailing ones, total zeros) and variables for context switching are highly sensitive to soft errors and may result in decoder crash, because a wrong context switch leads to an incorrect codeword and disturbs the decoding of the subsequent syntax elements. Therefore, protection of these syntax elements and context switch information is of paramount importance.

Fig. 1 Excerpts from different video sequences showing the effect of soft errors on the subjective video quality (white lines encircle the corrupted regions)

Soft errors in the VLC *codeword* and *codelength* tables during the coding of level and run values result in visual artifacts (see Fig. 1, encircled by the white lines) that may even propagate to the subsequent frames as the decoded data is used for prediction of subsequent frames. These artifacts are visually unpleasant to the end-user/viewer.

Our analysis illustrates that computational reliability needs to be considered for advanced video codecs targeting future unreliable multimedia platforms. Especially, for an error-resilient CAVLC hardware, the protection of VLC *codeword* and *codelength* tables is of key importance, as they are highly sensitive to soft errors and may even become more vulnerable as they reside for a long time on the on-chip memory[3]. *The goal of our work is to provide power-efficient error-resilience in H.264 CAVLC with protected VLC tables, while exploiting the application-specific knowledge, like inherent properties of the CAVLC algorithm and the input video data*.

B. Challenges and Our Novel Contributions

Since the VLC *codeword* and *codelength* tables are accessed frequently during the CAVLC computation, performing row-based or column-based or combined row-column-based block-wise parity (like in [12]) on a certain complete VLC table incurs significant delay, area, and/or power overhead due to parity computations and memory accesses as a result of large amount of data (see power savings in Section VI). Furthermore, this leads to a higher probability of parity mismatch as the soft error probability is higher in a large table due to an increased area and a soft error in a single value will always result in a mismatch. If only rows or columns are protected with parity, this might even lead to frequent reloading of data from the main memory which is power inefficient.

Our analysis of the statistical distribution of different syntax elements (see details in Section III) shows that not all entries in the *codeword* and *codelength* tables are accessed with the same frequency during the CAVLC of a given video frame. The number of times a certain value from the table is accessed highly depends upon the texture and motion properties of the video data and the quantization parameter. Therefore, *the challenge is to provide power-efficient reliability/error-resilience by partitioning the tables into multiple sub-tables that can be protected using low-overhead row- or column-based block-wise parity*. Moreover, the unused tables can be predicted and power-gated in the state-retentive sleep mode[4] (i.e. the contents of the tables are preserved while incurring low leakage) independently to obtain further power reduction. To guide such a partitioning scheme and power-gating algorithm, knowledge of statistical distribution of syntax elements is the key.

Our Novel Contributions: We propose a novel error-resilient CAVLC hardware for H.264/AVC that employs:

1). *parity-protected partitioned VLC codeword & codelength sub-tables* (Section IV.A) for reduced delay/power overhead, obtained by,
2). *a design-time partitioning algorithm* (Section IV.A) that exploits the design-time analysis of error probabilities and statistical distribution of different syntax elements coded using CAVLC (for various test video sequences and quantization parameters; Section III) to partition large VLC tables into multiple sub-tables.
3). *a run-time manager for error-resilience and power management* (Section IV.B) that accesses the data from the partitioned sub-tables, performs error detection using a reduced-sized block-wise parity, and reloads the data in case of a parity mismatch. Furthermore, it determines the power-gating decision for the unused VLC *codeword* & *codelength* sub-tables in the state-retentive sleep mode using the design-time analysis of statistical distribution and a run-time classification of Macroblocks.

Our experimental results demonstrate that compared to the unprotected case, our scheme provides on average 18 dB better PSNR at the cost of a 35% area and 45% performance overhead. However, partitioned sub-tables and state-retentive power gating provide 58% reduced leakage energy. Compared to state-of-the-art table protection [12], our scheme incurs 2x reduced area and performance overhead.

This is the first approach towards power-efficient reliability/error-resilience in the H.264 CAVLC against soft errors that exploits inherent properties of the CAVLC algorithm and input video sequence.

II. Overview of the H.264 CAVLC

The main CAVLC process works on a 4x4 sub-block level. The input is the transformed quantized coefficients of a 4x4 block in a zigzag scanning order, see Fig. 2. Different syntax elements (SE) for a 4x4 block are shown in Fig. 2. In CAVLC, values of syntax elements are replaced by *codewords* of a certain *codelength*. Shorter codes are used for frequently occurring values and longer codes are used for infrequently occurring values. CAVLC exploits the following properties: (i) the non-zero coefficients at the end of a zigzag scan are often patterns of +/-1, i.e. trailing ones, (ii) 4x4 blocks contain several zeros, (iii) there is a correlation in the number of total non-zero coefficients for the neighboring blocks.

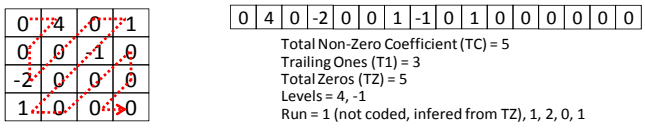

Fig. 2: An example showing syntax elements for a 4x4 block

The following syntax elements are coded using CAVLC (details in [1]):

- *Total non-zero coefficients (TC) and trailing ones (T1)* are jointly coded. The values of TC and T1 range from 0 to 16 and 0 to 3 (additional '*1*'s are represented as levels), respectively. The parameter *N* denotes the context switch used to select a row in the code/length tables and its value depends upon the TC in the upper (TC_{Top}) and left (TC_{Left}) coded 4x4 blocks. $N=(TC_{Left}+TC_{Top}+1)/2$ if both blocks are available. $N=TC_{Top}$ or $N=TC_{Left}$ if only the upper or left block is available, respectively. Otherwise: N=0.
- *Sign of Trailing ones*: The sign of each trailing one is coded as one bit: '0' = positive and '1' = negative.
- *Levels*: The sign and magnitude of the remaining non-zero coefficients (i.e. TC–T1) are encoded in reverse order (i.e. high frequency coefficients first). A large table with seven contexts (VLC_0, …, VLC_6) is used for encoding. The context switch depends upon the value of the previously coded levels.
- *Total zeros*: The number of zeros before the last highest-frequency non-zero coefficient is encoded.

[3] the probability of soft errors in memory elements is much higher compared to the combinatorial circuits due to the *logical masking effects* in the later [3].

[4] Sleep transistors with multiple sleep modes provide the necessary physical infrastructure means for state-retentive power-gating [21][22].

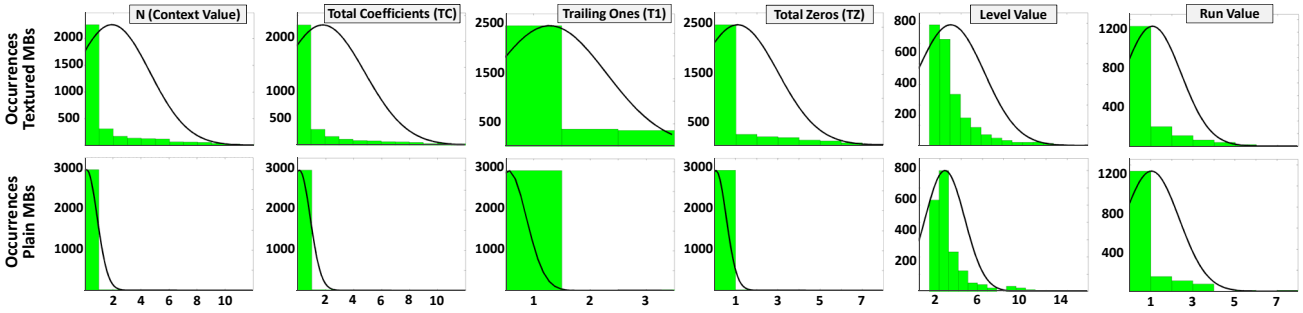

Fig. 3 Statistical distribution of different syntax elements in CAVLC for various test video sequences (QCIF–HD1080p resolutions)

- *Runs*: Finally, starting with the highest-frequency coefficient, the number of zeros before each (except the lowest-frequency) non-zero coefficient is encoded as a Run value.

III. Analysis of Error Probabilities and Statistical Distribution of Different Syntax Elements

Now, we discuss the error probabilities for the CAVLC process to analyze its relationship with the number of total coefficients (TC). Since different syntax elements are *statistically dependent*, for a 4x4 sub-block, we can represent the error probability (P_E) of the coded data (cData) as the *summation* of the error probabilities of different syntax elements.

$$P_E(cData) = P_E(TC) + P_E(T1) + P_E(Levels) + P_E(TZ) + P_E(Runs)$$

TC and T1 are coded in one codeword (T_{C1}) which depends upon TC_{Left} and TC_{Top}. Therefore, the error probability of the T_{C1} codeword is given as the joint conditional probability: $P_E(T_{C1}|TC_{Left})+P_E(T_{C1}|TC_{Top})$. Since the number of levels and runs depends upon TC, the P_E(Levels) and P_E(Runs) error probabilities can be represented as:

$$P_E(Levels) = \sum_{i=1\ to\ (TC-T1)} P_E(Level_i);\ P_E(Runs) = \sum_{i=1\ to\ (TC-T1)} P_E(Run_i)$$

Moreover, the context adaptation during the level coding (i.e. switching of VLC tables) results in a conditional probability for levels, since an error in a certain level value affects the coding of the subsequent levels.

$$P_E(Levels) = P_E(Level_1|TC,T1) + \sum_{i=2\ to\ (TC-T1)} P_E(Level_i|Level_{i-1})$$

The total error probability of the coded data can be represented as:

$$\begin{aligned} P_E(cData) = {} & \{P_E(T_{C1}|TC_{Left})+P_E(T_{C1}|TC_{Top})\} \\ & + \{P_E(Level_1|TC,T1) + \sum_{i=2\ to\ (TC-T1)} P_E(Level_i|Level_{i-1})\} \\ & + P_E(TZ) + \sum_{i=1\ to\ (TC-1)} P_E(Run_i) \end{aligned} \quad (1)$$

From this equation, we can draw the conclusion that **a higher value of TC leads to a higher probability of soft errors, thus leading to a greater degradation**. The value of TC mainly depends upon the quantization parameter (QP) and the spatial/temporal properties of a given video. To demonstrate this fact, we have performed a statistical analysis of the distribution of different syntax elements for homogeneous (like in background objects) and textured MBs for several test video sequences (ranging from QCIF to HD1080p with slow to fast motion, e.g., Foreman, Tractor). A block is categorized using Eq. 2. T_{LV} is the threshold for low variance and it is obtained using regression analysis [18]. μ denotes the average brightness of a given MB.

$$MB = \begin{cases} Homogeneous, & if\ (Variance_{MB} \le T_{LV}) \\ Textured, & Else \end{cases} \quad (2)$$

$$Variance_{MB} = \sum_{i=0to15} \sum_{j=0to15} (P(i,j) - \mu_{MB})^2 >> 8$$

If MB properties are known, knowledge of the highly-probable and less-probable values of syntax elements can be obtained from the probability density function (PDF) plots; see Fig. 3. Assuming these PDFs follow a *Gaussian* distribution, the zone of highly-probable values (with a probability of 0.975) is given as $F(\mu+3\sigma; \mu, \sigma^2) - F(0; \mu, \sigma^2)$, such that μ denotes the mean of distribution and σ denotes the standard deviation. For instance, the value of TC does not exceed '3' for a homogeneous MB. Correspondingly, it can be predicted which parts of the VLC tables for a given syntax element are more likely to be used. This prediction is helpful for table partitioning and power-gating of the unused tables. For instance, the *less-probable values are stored in a separate sub-table that can be power-gated*. The unused tables can then be predicted by analyzing the MB properties, which are typically obtained in a video preprocessing stage. Note, such an MB characterization and preprocessing stage is also required for fast mode decision and adaptive motion estimation in an H.264 encoder [18][19]. Therefore, in this work, we assume MB properties are already available to the run-time manager and do not introduce any additional power overhead for CAVLC. The goal of partitioning is to provide a sub-table organization, such that sub-tables not accessed in a certain time window can be power-gated for a long time; see Section IV.

Since the values of syntax elements also depend upon the QP, a highly-probable value of a given syntax element for homogeneous (H) and textured (T) MB can be predicted using Eqs. 3–8. These equations are obtained by computing the zone of highly-probable values (considering a *Gaussian* distribution) from various distributions obtained using various QPs and then applying a polynomial curve fitting.

$$N_H = 0.18QP^2 - 6.16QP + 69.34; \quad N_T = 0.24QP^2 - 0.21QP + 94.47 \quad (3)$$
$$TC_H = 0.22QP^2 - 7.33QP + 81.98; \quad TC_T = 0.28QP^2 - 9.61QP + 109.6 \quad (4)$$
$$T1_H = 0.11QP^2 - 3.60QP + 38.57; \quad T1_T = 0.14QP^2 - 4.57QP + 49.23 \quad (5)$$
$$L_H = 0.18QP^2 - 5.93QP + 64.17; \quad L_T = 0.23QP^2 - 7.55QP + 82.36 \quad (6)$$
$$TZ_H = 0.28QP^2 - 9.24QP + 101.6; \quad TZ_T = 0.35QP^2 - 11.9QP + 132.8 \quad (7)$$
$$R_H = 0.17QP^2 - 5.70QP + 62.87; \quad R_T = 0.20QP^2 - 6.84QP + 76.92 \quad (8)$$

IV. Error-Resilient CAVLC Architecture

Our error-resilient CAVLC architecture receives the transformed quantized coefficients after the quantization process in H.264 encoder and outputs the *codeword* and *codelength* of different syntax elements to the bitstream writing module. It consists of three main parts: (i) core CAVLC modules, (ii) a run-time manager for error-resilience and power management, and (iii) partitioned VLC sub-tables; see Fig. 4.

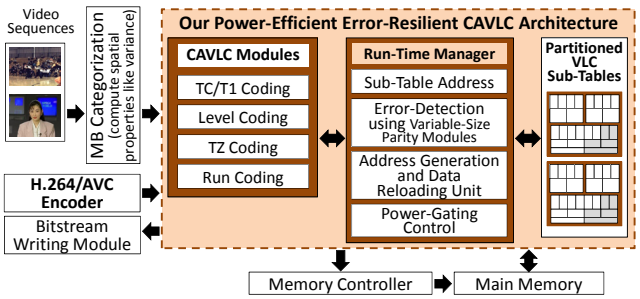

Fig. 4 Overview of our error-resilient CAVLC architecture

The core CAVLC modules compute the syntax values and their corresponding context. For instance, the TC/T1 coding module computes the number of total coefficients and trailing ones along with the context information *N* based on the values of TC_{Left} and TC_{Top} (see Section II). The syntax value and the context information are then forwarded to the run-time manager that computes the address for the appropriate *codeword* and *codelength* sub-tables. The values from the partitioned sub-tables along with the pre-computed parity values are then extracted and forwarded to the parity modules. The computed parity is compared with the stored parity. In case of a mismatch, an error is detected, and the corresponding entries of the sub-tables are

reloaded from the main memory[5]. Furthermore, based on the knowledge of MB categorization, the run-time manager predicts the unused sub-tables and puts them in the state-retentive sleep mode. In the following, we explain the table partitioning algorithm and the operational flow of the run-time manager for error-detection and power-management. The core CAVLC modules are simple state-machines implementing the standard compliant functionality [1].

A. Table Partitioning and Zero Value Elimination

The VLC *codeword* and *codelength* tables are partitioned into sub-tables using a partitioning algorithm (see Fig. 5) that employs the knowledge of the statistical distribution of different syntax elements (SE). The loop iterates over all syntax elements, i.e. T_{C1}, Levels, TZ, and Runs. First, highly-probable values of the context switching parameters and syntax values are estimated using equations 3–8 for homogeneous and textured MBs (see lines 3–4). The average values are computed (line 5), which are then used to determine the number of horizontal and vertical table partitions (lines 6, 8). Symmetric partitioning is required in order to enable a simple design of block-based parity hardware and to maximize the potential of power-gating. The *codeword* and *codelength* tables of a given syntax element '*se*' are *partitioned horizontally* (i.e. row-wise) using the average highly-probable value of the context switching parameter, like *N* in case of tables for coding TC/T1 and VLC_{num} (VLC_0-VLC_6) in case of level coding (lines 6–7). Afterwards, tables are *partitioned vertically* (i.e. column-wise) using the average highly-probable value of the syntax element (lines 8–9). Since the entries of *codeword* and *codelength* tables for a given syntax element are accessed from the same index, both tables of the given syntax element are merged (line 10).

```
1.  PartitionTable (Syntax Elements SE, Quantization Parameter QP)
2.  ∀se ∈ SE | SE={TC1,Levels,TZ,Runs}   {
      // Estimate highly-probable values of context switch and syntax values
3.    (cs_H, cs_T) := estimateContextSwitch(QP, se);      // Eq.3,4,7
4.    (sv_H, sv_T) := estimateSyntaxValue(QP, se);        // Eq.4–8
5.    cs=(cs_H+cs_T)/2;          sv=(sv_H+sv_T)/2;
      // Horizontal Partitioning
6.    numPart_Hz := ⌊numRows/cs⌋;
7.    {T_HzCod, T_HzLen} := partition(se.get(T_Cod, T_Len), numPart_Hz);
      // Vertical Partitioning
8.    numPart_Vt := ⌊numCols/sv⌋;
9.    {T_VtCod, T_VtLen} := partition({T_HzCod, T_HzLen}, numPart_Vt);
10.   T_se := merge(T_VtCod, T_VtLen);   // Merge codeword & codelength tables
      // Eliminate sub-tables with all Zero values
11.   T' := Ø;
12.   ∀t ∈ T_se
13.     numZE := 0;
14.     ∀c ∈ 1,...,t.getNumCols()   numZE+=isAllZeros(t.getCols(c));
15.     numZeroTablePartitions := ⌊numZE/4⌋;
16.     {t'} := partition(t, numZeroTablePartitions); // only, the sub-
          tables with non-zero entries are returned
17.     T' := T' U {t'};
18.   se.store(T');
19. }
```

Fig. 5 Pseudo-code of the table partitioning algorithm

Our analysis shows that there are several zero values in the sub-tables that can be eliminated by intelligent partitioning (see an example in Fig. 7). Therefore, for each partitioned sub-table, columns of all zero entries are determined (line 14). In case all columns of a sub-table are all-zero-entry columns, the complete sub-table is eliminated. Otherwise, the sub-tables are further partitioned in such a way that the number of sub-tables with all zero entries is maximized while preserving the symmetry and parity format used by other sub-tables (line 15–17), thus leading to a reduced leakage. To enable a fast access by the parity hardware, the sub-tables are rotated and a dedicated 50-bit wide port to the on-chip memory is provided to fetch 4 values of *codeword* and *codelength* each. Different sub-tables contain values of different bit widths. Therefore, variable-sized parity modules are provided to reduce the dynamic power overhead of parity computation. Moreover, a small block-size in the parity computation due to less number of table values (like, 4 values in Fig. 6) also results in low power for parity computation and a low probability of parity mismatch.

On overall, our proposed scheme reduces power by (i) reduced-sized parity computations, (ii) lesser number of table row/column values used for parity computations, (iii) reduced memory requirements and leakage energy due to zero value elimination, and (iv) power-gating the unused VLC sub-tables in a state-retentive sleep mode.

An Example: We discuss an example of partitioning the VLC table for coding the *Total Coefficients (TC) and Trailing Ones (T1)*. This table consists of 2x204 values (as specified in the standard [1]) with the largest value represented with 5 bits (Fig. 6). The access to this table is controlled by three parameters. The choice of a horizontal line is controlled by the parameters *N* and *T1* (see Section II). Referring to Fig. 3, in case of homogeneous and textured MBs, the values of *N* are 3 and 6, respectively, with a probability of 0.975. This leads to a horizontal partition of size '4' (see line 6 of Fig. 5). The access to a column is controlled by the value of TC. Fig. 3 shows that the TC values for homogeneous and textured MBs are 2 and 7, respectively, with a probability of 0.975. It gives a vertical partition of size '4' (see line 8 of Fig. 5). This leads to a partitioned sub-table organization of Fig. 6 (the last horizontal partition contains only two sub-tables as its access probability is low). Each partitioned sub-table is stored in a small memory. Each memory entry has (i) 4x5-bit codelength information plus parity and (ii) 4x5-bit codeword information plus parity. Similarly, the sub-tables for total zeros (Fig. 7), level values[6], and run values are partitioned. In case of total zeros six sub-tables with all-zero values (see grey boxes in Fig. 7) are eliminated to save the storage requirements.

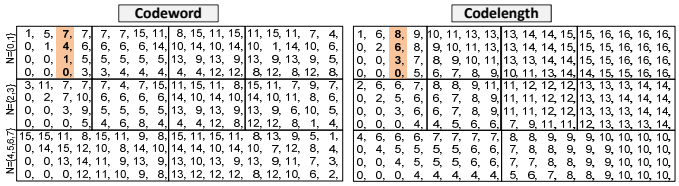

Fig. 6 Tables for coding Total Coefficients and Trailing Ones

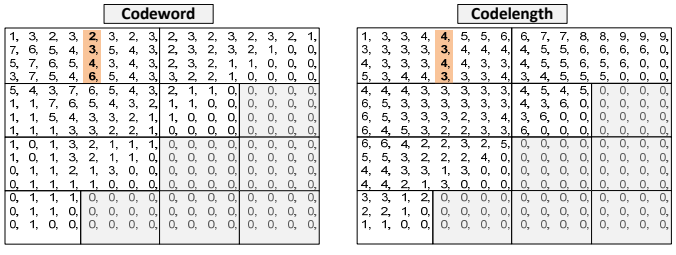

Fig. 7 Tables for coding Total Zeros

B. Run-time manager for error-resilience & power management

Our error-resilient CAVLC architecture is equipped with a run-time manager (Fig. 8) that performs the following tasks:

i) *Loading the codeword and codelength values*: The context information and the values of syntax elements from the CAVLC core modules

[5] Note, main memory is ECC protected, which is a well-established practice in various research and industrial projects (IBM [23], AMD [24]).

[6] Although the maximum value of level can be represented in 16-bits, for embedded video applications, our statistical analysis for various sequences shows that the value of level ranges from -14 to +14 due to the quantization effects for QP values ≥20. Note QP=20 is a typical for high-quality encoding.

are forwarded to the run-time manager for computing the address parameters: sub-table identifier $ID_{subTable}$, line identifier ID_{Line}, specific value ID_{Value}. These address parameters are used to fetch the appropriate *codeword* & *codelength* values from the respective sub-tables.

ii) *Error-detection and data reloading*: four data entries and one parity value are extracted for each of the *codeword* and *codelength* parts. Afterwards, respective parity values are computed using variable-sized parity modules and are compared to the stored parity values. In case of a parity match, the requested *codeword* and *codelength* values are extracted and output. In case of a mismatch, the address of the sub-table entry in the external memory is calculated (using the address generation unit) and the data values are reloaded.

iii) *Power-gating the temporarily unused VLC sub-tables*: in the last step, the run-time manager employs a power-gating algorithm (see Fig. 9). First the MB is characterized as homogeneous or textured (line 2). Afterwards, for all syntax elements, a prediction of the VLC sub-tables is performed by exploiting the statistical distribution of the syntax element and properties of the current MB (lines 4–6). Afterwards, the unused sub-tables are set into the state-retentive sleep mode, while evaluating the energy benefit to amortize the wakeup cost (line 7–9). Alternatively, the tables are kept in the power-on state.

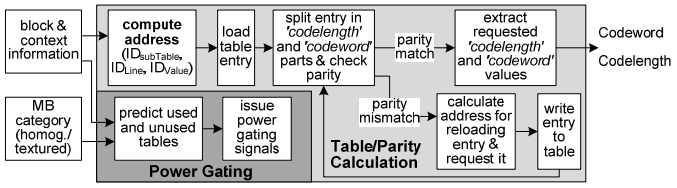

Fig. 8 Operational flow of the run-time manager

```
1.  PowerGateSubTables(SE, QP, MB)
2.  C_MB := evaluateMBCategory(MB);   // Eq.2
3.  ∀se ∈ SE | SE={TC1,Levels,TZ,Runs}   {
4.    cs_MB := predictContextSwitch(C_MB, QP, se);   // Eq.3,4,7
5.    sv_MB := predictSyntaxValue(C_MB, QP, se);   // Eq.4–8
6.    T_ID := getUsedTableID(se.get(T'), cs_MB, sv_MB);
7.    ∀t ∈ se.get(T')
8.      If ((t.ID!=T_ID)&((t.P_LeakxL_CAVLC4x4)>E_Wakeup)) powerGate(t);
9.      else powerON(t);
10. }
```

Fig. 9 Pseudo-code of the power-gating algorithm

V. Fault Injection and Simulation Setup

Though not limited to FPGAs, we have prototyped our error-resilient CAVLC architecture on a Xilinx Virtex-5 FPGA. Since, commercial FPGAs do not support power-gating features; the prototype is only used for area comparison and functional verification. For the final product, the whole system is envisaged to be implemented using an ASIC flow. For soft error analysis, fault injection, and power comparison, we have developed a fault injection and simulation method, see Fig. 10 (see further details in [28]). The inputs are: (i) transformed quantized coefficient data from the H.264 encoder, (ii) hardware footprint, power, and frequency information obtained after implementation and place & route for FPGA fabric (see Table I for hardware results) or from the ASIC flow, (iii) fault configuration in terms of fault rate (in #faults/MB). The fault rate is computed from the neutron flux (N in particles/mm^2/sec, [25]; it provides the flux information based on the coordinates of a city/location), fault probability (P_{Fault}), hardware area (A_{CAVLC} in mm^2), MB rate (MBR in MBs/sec):

$$N_{FaultsCAVLC} = (N \times P_{Fault} \times A_{CAVLC}) / MBR$$

For a given city's coordinates and three different altitude values (covering Terrestrial and Ariel use cases, also used by prominent related work [26]), three different fault rates are computed as R=1fault per *'N'* MBs with *N=5, 10, 20*. Single bit-flip faults are randomly injected in hardware during the processing of syntax elements. The outputs of the simulator are the fault-free bitstream and faulty bitstream, which are then decoded using the H.264 decoder [27] to obtain the decoded videos. The error logs maintain the point of mismatches and the decoded videos are used for the objective and subjective video quality experiments, see Section VI. CIF (Akiyo, Rafting) and HD1080p (Tractor, Pedestrian) sequences are used for the experiments as they exhibit slow-high motion content. Other test conditions are GOP=IPPP…, QP={20,25,30,35,40}. For the on-chip memory, we deploy the model of a 65nm SRAM with multiple sleep states [22] due to its reduced wakeup latency (3 cycles for a transition from the state-retentive sleep to the power-on mode).

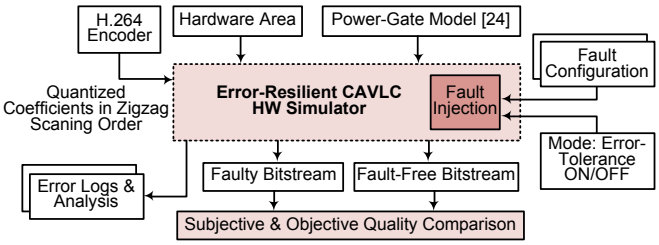

Fig. 10 Fault Injection and Simulation Environment

VI. Results and Discussion

Fig. 11 and Fig. 12 illustrate the objective and subjective quality comparisons between the fault-free case (Original), our error-resilient CAVLC at N=20, and unprotected cases with different faults rates (N=5, 10, 20). The rate-distortion (RD) curves in Fig. 11 show that our error-resilient CAVLC provides video quality results closer to the fault-free case. Since only tables are protected in the case (which occupy a major part of the hardware footprint), slight quality degradation are due to the faults during the computational hardware modules. Fig. 11 shows that our scheme provides a significant PSNR improvement (up to 10dB at N=20, 18 dB at N=5) over the faulty cases. The effect is also visible in the subjective quality results, Fig. 12.

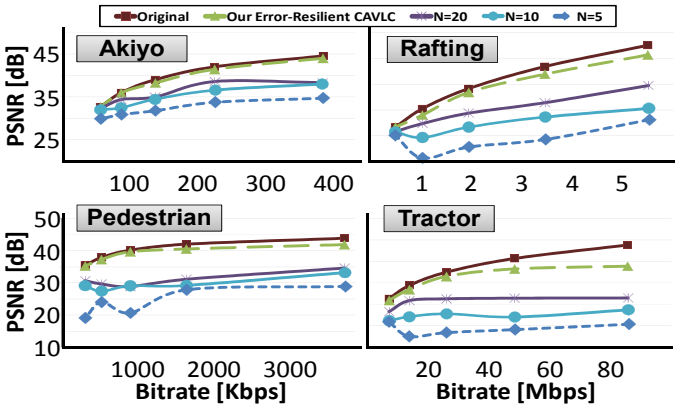

Fig. 11 Comparing RD-Curves with Original & Unprotected Cases

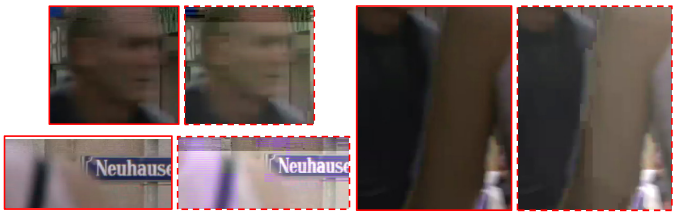

Fig. 12 Subjective quality comparison of various excerpts in the Pedestrian test sequence: PSNR of *Our* with red border= 40.22 vs. PSNR of *Unprotected* case with red dotted border=26.40)

Fig. 13 shows the leakage power savings of our partitioned VLC sub-tables with power-gating compared to the un-partitioned tables for various video sequences using different QP values. On average, our partitioned table scheme provides 58% leakage savings. High-textured sequences (Mobile, Tractor) provide lower leakage savings compared to the slow motion sequences, as more sub-tables are used frequently

due to relatively larger values of syntax elements. When comparing for different QPs, higher leakage savings are obtained on bigger QP values, because for bigger QPs, the values of syntax elements are smaller and there are fewer context switches.

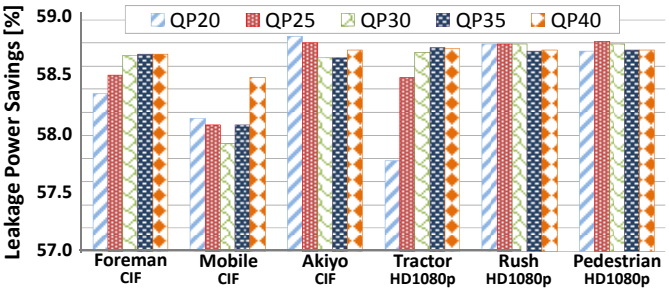

Fig. 13 Leakage power savings compared to un-partitioned tables

For a comparison of memory requirements, area, and performance, we prototyped our error-resilient CAVLC architecture, an unprotected CAVLC architecture, and CAVLC with state-of-the-art double-parity table protected [12] (i.e., combined row- and column-based parity of the complete table) on a Xilinx Virtex-5-vlx110 FPGA. The on-chip memories for the partitioned VLC sub-tables are implemented with LUTs. Table I shows that compared to an unprotected scheme, our approach incurs a 7% memory overhead, 34.8% area overhead, and 45% performance overhead due to parity calculation, parity storing, and an additional run-time manager module. However, this overhead is justified by the significant (up to 18 dB) PSNR improvements and 58% leakage energy reduction of our scheme. Still compared to state-of-the-art table protection scheme [12], our scheme provides a 12% reduced memory overhead, which is primarily due to elimination of all-zero value sub-tables and sub-tables with redundant values. However, it complicates the controller design. Moreover, Table I shows that compared to [12], our scheme provides 2x reduced area and performance overhead due to the reduced-complexity parity hardware and control logic.

TABLE I. COMPARING THE HARDWARE RESULTS WITH STATE-OF-THE-ART (L=NUMBER OF LEVELS, R= NUMBER RUNS TO BE ENCODED)

			Virtex-5-vlx110 FF1153	
	Memory	Latency	Area	
	[bit]	[Cycles]	Slices	LUT
Unprotected [5]	6400	8 + 4*L + 5*R	560	1,460
Double Parity [12]	7672	24 + 10*L + 11*R	1,739	4,924
Our scheme	6850	12 + 6*L + 7*R	755	2,217

VII. CONCLUSIONS

We presented an error-resilient architecture for H.264 CAVLC with improved power-efficiency. It employs partitioned VLC codeword and codelength tables with state-retentive power-gating support. The table partitioning and power-gating algorithms consider the statistical distribution of different syntax elements to achieve high power-efficiency. Our experimental results demonstrate an up to 18 dB improved PSNR compared to the unprotected case at the cost of 35% increased area, while table partitioning with power-gating brings 58% leakage energy savings. Compared to state-of-the-art [12], our architecture provides 2x reduced area and performance overhead.

This paper instantiates the need to investigate solutions for power-efficient error-resiliency for computationally reliable video coding in order to address the upcoming industrial challenges related to soft-error issues in the advanced/modern unreliable multimedia computing platforms. This is not only crucial for the current H.264 standard but also for the upcoming video coding standards like H.265/HEVC. This work illustrates that, for increased power-efficiency, reliability methods need to exploit the inherent algorithmic and data properties of video codecs.

ACKNOWLEDGEMENT

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu).

REFERENCES

[1] ITU-T Rec. H.264 and ISO/IEC 14496-10:2005 (E) (MPEG-4 AVC), "Advanced video coding for generic audiovisual services", 2005.

[2] J. Ostermann et. al., "Video coding with H.264/AVC: Tools, Performance, and Complexity", IEEE Circuits and System Magazine, vol. 4, no. 1, pp. 7-28, 2004.

[3] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", IEEE TDMR, vol. 5, no. 3, pp. 305-316, 2005.

[4] S.Borkar, T. Karnik, V. De, "Design and Reliability Challenges in Nanometer Technologies", IEEE DAC, pp. 75-75, 2004.

[5] J. W. Wells, J. Natarajan, A. Chatterjee, "Error resilient video encoding using Block-Frame Checksums", IEEE IOLTS, pp. 289-294, 2010.

[6] H.-Y. Cheong, I. S. Chong, A. Ortega, "Computation Error Tolerance in Motion Estimation Algorithms", IEEE ICIP, pp.3289-3292, 2006.

[7] R. Vadlamani et al.,"Multicore soft error rate stabilization using adaptive dual modular redundancy", IEEE DATE, pp. 27-32, 2010.

[8] D. Ernst et al., "Razor: circuit-level correction of timing errors for low-power operation", IEEE MICRO, vol. 24, no. 3, pp. 10-20, 2004.

[9] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, D. I. August, S. S. Mukherjee, "Software controlled fault tolerance", ACM TACO, vol. 2, pp. 366-396, 2005.

[10] G. V. Varatkar, N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance", IEEE ISLPED, pp. 113-118, 2006.

[11] Y. Wang, S. Wenger, J. Wen, A. Katsaggelos, "Review of error resilient coding techniques for real-time video communications", IEEE Signal Processing Magazine, vol. 17, no. 4, pp. 61–82, 2000.

[12] C. Nguyen, "Fault Tolerant Huffman Coding for JPEG Image Coding System", Technical Report, http://www.ece.ucdavis.edu/cerl/ReliableJPEG

[13] C. Nguyen, G. R. Redinbo, "Fault tolerance design in JPEG 2000 image compression system", IEEE Transactions on Dependable and Scure Computing, vol. 2, no. 1, pp. 57-75, 2005.

[14] M. A. Makhzan, A. Khajeh, A. Eltawil, F. J. Kurdahi, "A low power JPEG2000 encoder with iterative and fault tolerant error concealment", IEEE TVLSI, vol. 17, no. 6, pp. 827-837, 2009.

[15] A. K. Djahromi, A. Eltawil, F. J. Kurdahi, "Exploiting fault tolerance towards power efficient wireless multimedia applications", IEEE Consumer communications and networking conference, pp. 400-404, 2007.

[16] A. Heinig, M. Engel, F. Schmoll, P. Marwedel, "Improving transient memory fault resilience of an H.264 decoder", IEEE ESTIMedia, pp. 121-130, 2010.

[17] S. Rehman, M. Shafique, F. Kriebel, J. Henkel, "ReVC: Computationally Reliable Video Coding on Unreliable Hardware Platforms: A Case Study on Error-Tolerant H.264/AVC CAVLC Entropy Coding", IEEE ICIP, pp.405-408, 2011.

[18] M. Shafique, B. Molkenthin, J. Henkel, "An HVS-based Adaptive Computational Complexity Reduction Scheme for H.264/AVC Video Encoder using Prognostic Early Mode Exclusion", IEEE DATE, pp.1713-1718, 2010.

[19] M. Shafique, L. Bauer, J. Henkel, "enBudget: A Run-Time Adaptive Predictive Energy-Budgeting Scheme for Energy-Aware Motion Estimation in H.264/MPEG-4 AVC Video Encoder", IEEE DATE, pp.1725-1730, 2010.

[20] H.265/HEVC, High Efficiency Video Coding: http://www.h265.net/.

[21] S. Roy, N. Ranganathan, S. Katkoori, "State-Retentive Power Gating of Register Files in Multi-core Processors featuring Multithreaded In-Order Cores", IEEE Transaction on Computers, 2010.

[22] H. Singh, K. Agarwal, D. Sylvester, K. J. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating", IEEE TVLSI, vol. 15, no. 11, pp. 1215-1224, 2007.

[23] IBM® XIV®: http://publib.boulder.ibm.com/infocenter/ibmxiv/r2/index.jsp.

[24] AMD PhenomTM II Processor Product Data Sheet 2010.

[25] Flux calculator: www.seutest.com/cgi-bin/FluxCalculator.cgi.

[26] J. Hu, S. Wang, S. G. Ziavras, "In-Register Duplication: Exploiting Narrow-Width Value for Improving Register File Reliability", DSN, pp. 281-290, 2006.

[27] H.264 Codec JM 13.2: http://iphome.hhi.de/suehring/tml/index.htm.

[28] S. Rehman, M. Shafique, F. Kriebel, J. Henkel, "Reliable Software for Unreliable Hardware: Embedded Code Generation aiming at Reliability", IEEE CODES+ISSS, pp. 237-246, 2011.

[29] S. Saponara, M. Martina, M. Casula, L. Fanucci, G. Masera, "Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding", Microprocessors & Microsystems, vol. 34, no. 7-8, pp. 316-328, 2010.

[30] S. Saponara, L. Fanucci, S. Marsi, G. Ramponi, "Algorithmic and architectural design for real-time and power-efficient Retinex image/video processing", Journal of real-time image processing, vol. 1, no. 4, pp. 267-283, 2007.