dSVM: Energy-Efficient Distributed Scratchpad Video Memory Architecture for the Next-Generation High Efficiency Video Coding

Felipe Sampaio[1], Muhammad Shafique[2], Bruno Zatt[3], Sergio Bampi[1], Jörg Henkel[2]
[1]Informatics Institute, PPGC, Federal University of Rio Grande do Sul (UFRGS), Brazil
[2]Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
[3]GACI, PPGC, CDTec, Federal University of Pelotas (UFPel), Brazil
{felipe.sampaio, bampi}@inf.ufrgs.br, bzatt@inf.ufpel.edu.br, {muhammad.shafique, henkel}@kit.edu

***Abstract*— An energy-efficient distributed Scratchpad Video Memory Architecture (dSVM) for the next-generation parallel High Efficiency Video Coding is presented. Our dSVM combines private and overlapping (shared) Scratchpad Memories (SPMs) to support data reuse within and across different cores concurrently executing multiple parallel HEVC threads. We developed a statistical method to size and design the organization of the SPMs along with a supporting memory reading policy for energy efficiency. The key is to leverage the HEVC and video content knowledge. Furthermore, we integrate an adaptive power management policy for SPMs to manage the power states of different memory parts at run time depending upon the varying video content properties. Our experimental results illustrate that our dSVM architecture reduces the overall memory energy consumption by up to 51%-61% compared to parallelized state-of-the-art solutions [11]. The dSVM external memory energy savings increase with an increasing number of parallel HEVC threads and size of search window. Moreover, our SPM power management reacts to the current video properties and achieves up to 54% on-chip leakage energy savings.**

Keywords—Video Memory, Scratchpad Memory, HEVC, Application-Specific Optimizations, Energy Efficiency, Adaptivity.

I. Introduction

To bridge the increasing gaps between the processor and memory scaling/speed in many-cores era with memory-intensive applications, specialization of memory architectures has become one of most important design issues. Multiple cores simultaneously accessing the same memory infrastructure incur high energy consumption and contention. Meanwhile, embedded multi-/many-core processors are subjected to stringent energy constraints. These issues intricate when executing memory-intensive applications like video coding, image matching, etc.

The *High Efficiency Video Coding* (HEVC) is the next-generation video coding standard [1] that provides double compression compared to its predecessor H.264/AVC. However, this comes at the cost of >40% more computation effort compared to the H.264 encoder as shown by our experimental analysis in Fig. 1a. This increased complexity is due to the novel *Coding Tree Unit* (CTU) structure [2] and a plethora of new prediction modes that result in an increase mode decision space [3]. Moreover, these new coding features lead to >2x more memory accesses compared to H.264/AVC due to more intensive reference frames storage and transmission (as in Fig. 1b). *A large amount of off-/on-chip memory accesses and large-sized on-chip memories lead to high energy consumption in HEVC encoders*. To achieve high performance, HEVC encoders can be parallelized on multi-/many-core processing platforms. However, this may lead to further increase in the energy consumption and memory pressure due to *multiple encoding cores requiring the same data* from the memory infrastructure, posing new challenges for the embedded multimedia systems.

A large body of research explored efficient cache organizations and on-chip memory architectures for general purpose multi-/many-core processors [18]. To overcome/alleviate the hardware overhead of caches, Scratch-Pad Memories (SPMs) evolved for energy-constrained embedded systems [19]. Instead of providing hardware support for mapping data/code from off-chip to on-chip memory, SPM allows designer/compiler to perform content management saving up to 30% of energy compared to complete caches under certain operating scenarios [19][1]. *The challenge is to efficiently utilize the SPMs*.

Considering the above-discussed memory issues of HEVC, general-purpose techniques for SPM management [20]-[22] may not be energy efficient. Recent trends demonstrated benefits of application-specific SPMs management for low-power H.264 video encoding for single core or ASIC-based systems [4]-[7]. However, these works lack support for many-cores which are more memory restrictive and do not address memory contention in private vs. shared memories for cores synchronization. Moreover, these works do not account for the novel coding model of the advanced HEVC that can be leveraged to achieve even higher energy savings as we will motivate in Section I.B.

In summary, *there is a strong need for application-specific memory design targeting energy-efficient high efficiency video encoding on embedded multi-/many-core platforms*. Our goal is to leverage the application-specific characteristics of the emerging HEVC standard to increase the potential of energy savings.

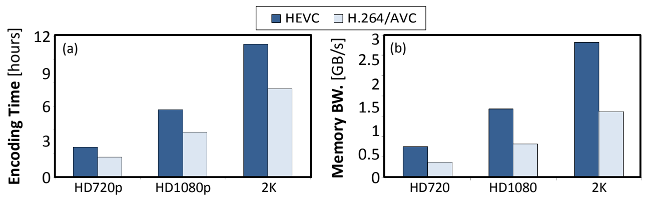


Fig. 1 HEVC vs. H.264/AVC encoder (a) encoding time; (b) memory accesses. (average results for commonly used test sequences [13], 128x128 search window, H.264/AVC and HEVC test models, 300 frames)

Before moving further, we will present basics of HEVC to the level of details necessary to understand our novel contribution.

A. Overview of HEVC Coding Tools and Related Memory Issues

To facilitate parallelization with minimal quality loss, the standardization committee (JCT-VC) introduced the novel concept of *Tiles*[2] in HEVC, which is different from slices that are used for video streaming [17]. Tiles divide one video frame into rectangular regions that can be coded independent of each other, thus increasing the thread level parallelism [15][16]. Fig. 2 presents an example of 4-Tile partitioning. Each Tile is assigned to a specific core *without* any data dependency with another Tile processing.

The inter-frame prediction with Motion Estimation (ME) is the most complex processing step in the HEVC encoder as it corresponds to >80% of the computation time and energy consumption of HEVC encoders. ME searches the best match of a block from the current frame in a set of so-called reference frames[3]. The search is performed in a restricted *search window*. The reference frames are typically stored in the external/off-chip memory while the search windows are stored in on-chip memories. Due to the memory management for fetching the search window samples from the off-chip and increased leakage energy for keeping them in the on-chip memory, the ME becomes the most

[1] *Examples:* IBM Cell Processor [23], ARM10E [24], TI TMS370CX7X [25], etc.
[2] These are video Tiles, i.e. different from hardware tiles in many-core processors.
[3] These are previously coded and reconstructed frames.

978-3-9815370-2-4/DATE14/©2014 EDAA

memory–intensive processing block [4]-[7]. As a result, 70%-90% of the ME energy is spent in the off-chip and on-chip memories (leakage and dynamic) [4]-[7]. Furthermore, multiple Tiles amplify the memory pressure since more data must be fetched/stored during the same time instant.

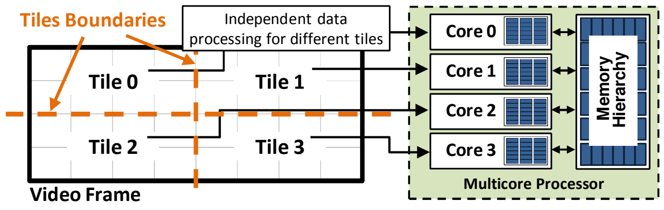


Fig. 2 Video Tiles and multicore organization for parallel HEVC.

Another novel coding tool of the HEVC that aggravates the memory problem is variable-sized *Coding Units* (CUs). The HEVC decision is based on a quad-tree structure (see Fig. 3). The root for the decision is the 64x64 CU, called *coding-tree block* (CTB). The encoder is responsible for deciding what is the best partitioning for the current CTB that provides the best coding efficiency, in terms of bitrate and coded video quality. Current HEVC draft also supports 32x32, 16x16 and 8x8 CU sizes [17].

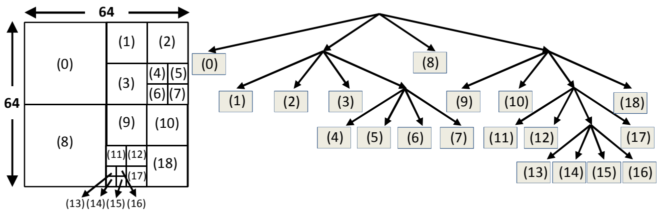


Fig. 3 An example of HEVC coding tree unit organization

B. Goals and Motivational Analysis

The main goal of our work is *to leverage the application-specific knowledge of the emerging HEVC standard (i.e. its new coding tools) and video content properties to develop an energy-efficient SPM-based on-chip video memory*. The key is (1) to analyze and exploit the memory access behavior in the video Tiles-based processing; and (2) the overlapping reference samples that define the shared access patterns for different cores processing different Tiles. The samples near to the tile boundaries in the reference frame must be fetched/stored by multiple cores, leading to external memory contention, redundant memory access and extra on-chip storage (causing energy wastage). An example in Fig. 4a depicts the overlapping accesses performed for more than one tile processing core (gray and black regions).

In the following, we highlight important memory related issues during the Tile-based HEVC processing with the help of our experimental case study and expose the potential of application-specific optimization with the help of several observations.

Analysis-1: The overlapping regions tend to grow for an increased number of Tiles (assuming 1 Tile per core). The overlap size trend is plotted for growing number of Tiles in the Fig. 4b. In the worst case, the overlap reaches 50% in a 16-core HEVC encoder. As larger is the overlapping area, more cores must concurrently access the same reference data from the external memory without any data reuse. Therefore, *it may be beneficial to design dedicated SPMs for the overlapping regions to avoid external memory retransmission of the Tiles shared reference data, saving off-chip memory energy.*

Analysis-2: Although the ME is performed within a search window, the search algorithm may not require all the samples. For instance, the TZ search algorithm in the HEVC software [14] does not necessarily explores the entire search window analysis [6]. Moreover, adaptive ME algorithms feature changing centering of the search window depending upon the already coded neighboring CUs. As a result, the *Tiles overlap shape may substantially vary according to the video content* as shown in Fig. 5. Furthermore, the *samples inside the overlapping regions have different access intensities*. It shows that, not all parts of the on-chip video memory (storing the overlapping samples) will be accessed for every CU depending on the video content. Even for the accessed sectors, the access distribution is different depending on the video content characteristics. Therefore, *the key is to leverage the overlapping memory access knowledge to predict the unused or less-frequently used memory sectors for adaptive power management of the SPMs.*

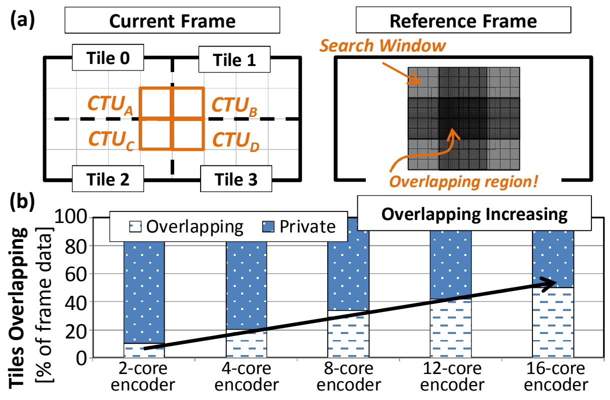


Fig. 4 (a) Example of Tile partitioning and of the overlapping problem; (b) Evaluation of overlapping accesses for different number of Cores (HD1080p; "BasketballDrive" sequence; 127x127 search window)

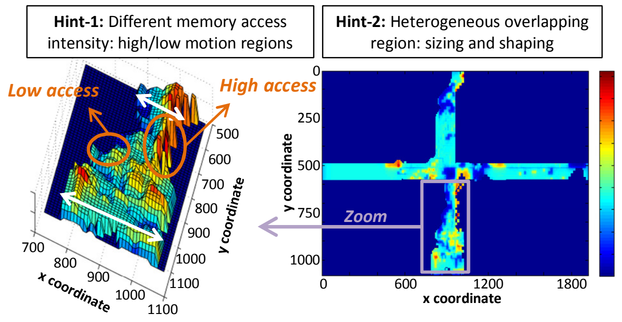


Fig. 5 Distribution of the overlapping samples (HD1080p; "BasketballDrive" sequence; 127x127 search window)

C. Our Novel Contributions

We propose an energy efficient distributed Scratchpad video memory architecture (dSVM) for the next-generation High-Efficiency Video Coding (HEVC) exploiting the video Tiles based parallel processing on multi/many-core processors. It employs:

- **A Distributed Scratchpad Video Memory Architecture (Section III)** that integrates several *private and overlapping (shared) SPMs* to support intra-Tile and inter-Tiles data reuse, respectively, among various cores. We develop a scheme that leverages the offline statistical analysis of HEVC and video content to size and design the organization of SPMs. A reading policy is designed for energy-efficient data fetching.
- **Adaptive Power Management of dSVM (Section IV)** that takes into account the size and the shape of the predicted overlapping area to select appropriate sleep states for different regions of private and overlapping SPMs.

We evaluate the energy efficiency of our dSVM architecture for various recommended test video sequences for different number of Tiles.

To the best of authors' knowledge, this is the first work towards energy-efficient on-chip memory hierarchy for the emerging Tile-based parallel HEVC encoders.

II. Memory Models and Notations

Every data transmission from/to memory is based on a fixed *basic access unit* (BU), which corresponds to a BU_{Size}*BU_{Size} picture block. When external memory communication is required, then *several BUs are accessed in one burst operation* to increase the energy efficiency.

On-Chip SRAM Organization Model: We adopt a bank-based partitioning Scratchpad memory (SPM) model to allow for parallel data accesses; see Fig. 6. Each SPM is composed of N_B number of banks. To facilitate parallel reading, different rows of a BU are stored in parallel banks. A bank B_i is composed of N_S sectors of size S_S. Each sector has N_L number of lines of size S_L.

Different sectors of the SPM can be individually power-gated using a multiple sleep-state transistor model supporting four power states [12]: *S0=OFF*, *[S1,S2]=Data Retentive* and *S3=ON*, where $E_{Static}(S0) < E_{Static}(S1) < E_{Static}(S2) < E_{Static}(S3)$. Still, each state have also increasing associated wake-up energies ($WE(S0)> WE(S1)> WE(S2)> WE(S3)= 0$).

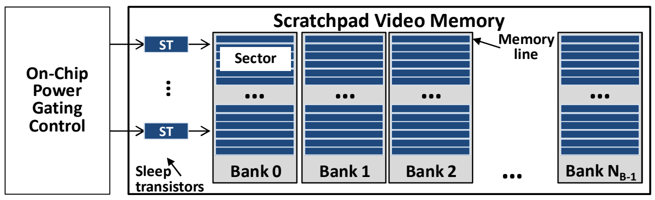


Fig. 6 Organization model of our scratchpad video memory

Off-Chip DDR DRAM Model: For energy estimation of the off-chip memory, we adopt the DDR (Dual Data Rate) DRAM model depicted in [9]-[10]. The total power is derived by the composition of six components: (1) page activation energy (E_{ACT}), (2) write energy (E_{WR}), (3) read energy (E_{RD}), (4) I/O pins energy (E_{DQ}), (5) refresh energy (E_{REF}), and (6) standby energy (E_{STBY}). In the experimental analysis, we assume that the memory will always operate in the active state and the standby energy will be equivalent to the E_{ACT_STBY} component.

III. Architecture of Our Distributed Scratchpad Video Memory

Fig. 7 depicts the block diagram of our distributed Scratchpad video memory architecture (dSVM) for multi-core HEVC encoding. Each core[4] is assigned the processing of one out of the *n* video Tiles. The SPMs are used to store different parts of the reference frame used for ME or other encoding blocks. We propose two levels of SPMs:

1) A core-private SPM (PrivSPM) to store the search window data corresponding to each CU for intra-Tile data reuse, and
2) A core-shared SPM (OvSPM) to store the Tiles overlapping reference data for inter-Tiles data reuse.

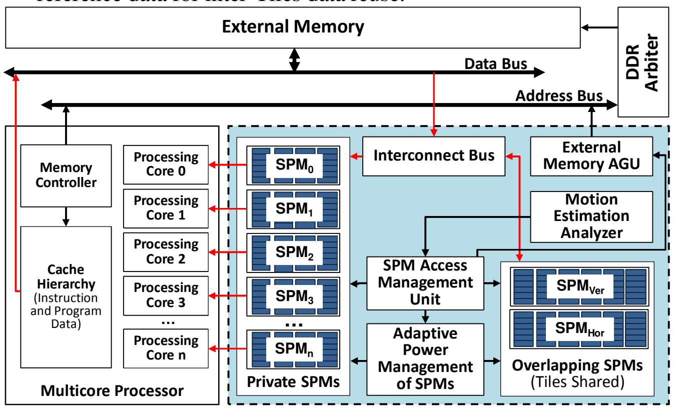


Fig. 7 Our dSVM architecture integrated in a HEVC encoder

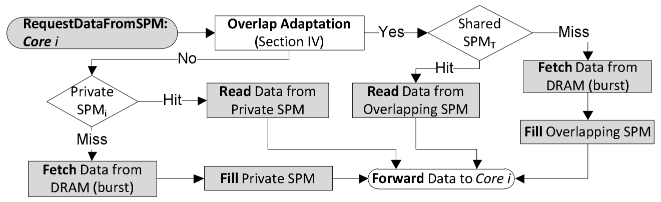
[4] A core has a small private instruction and data cache to store the program code and common data (like variables). The SPM is used for large data like reference frame.

Each core sends the search window data requests to the SPM access management module using the vertical/horizontal frame coordinates. The SPM access management unit will schedule the memory accesses according to our DRAM/SPM reading policy (see Section III.A). The overlap patters and size is extracted by the Motion Estimation Analyzer and forwarded to (i) the SPM access management module to map the overlapping region to the on-chip *OvSPMs*; and (ii) the adaptive power management unit for selecting an appropriate sleep state for the idle SPM regions. If external memory access is required, the frame positions are translated to physical DRAM memory position addresses by the Address Generation Unit (External Memory AGU in Fig. 7). The adaptive power management unit analyzes the Tiles overlap size to adaptively predict the less-likely accessed or idle memory sectors of the *PrivSPMs* and *OvSPMs* and to select an appropriate sleep state in order to save SPM leakage energy.

In the following sections, we detail the SPM access management module, SPM sizing and design, and adaptive power management policy.

A. *SPM Access Management Unit: Reading Policy and External Memory Arbitering*

Our SPM access management unit implements the memory reading policy (see flowchart of Fig. 8) that takes advantage from the tiles overlap to increase the data-reuse of the reference frames samples. If a core *i* requests data from the SPM memory organization, as the first step, the SPM access management unit checks along with the overlap prediction if the requested data potentially belongs to one tiles overlapping region. Assuming that the data is inside an overlap related to the tiles intersection *T*, the corresponding cores-shared $OvSPM_T$ is then accessed. In this case, the inter-Tiles data reuse is exploited, since all tiles that share the tile boundary *T* may request the same data. For non-overlapping regions, the $PrivSPM_i$ is accessed, leading to intra-Tile data reuse. Note that for each core data request, either the shared ($OvSPM_T$) or the private ($PrivSPM_i$) memory is accessed. In the case of a hit, the data is simply forwarded to core *i*. In case of a miss, the data must be fetched from the external memory and forwarded to the core *i*. For improved energy efficiency, the SPM access management unit requests a burst of samples from the DRAM memory, which reduces the DRAM page activation energy and amortizes the initial latency for memory random access [7]. Furthermore, the corresponding SPM is filled with the fetched data. To handle parallel accesses to the *OvSPM*, we employ a priority based scheduling.



RequestDataFromSPM: Core i
Overlap Adaptation (Section IV)
Yes
Shared SPM_T
Miss
No
Hit
Fetch Data from DRAM (burst)
Private SPM_i
Hit
Read Data from Private SPM
Read Data from Overlapping SPM
Fill Overlapping SPM
Miss
Fetch Data from DRAM (burst)
Fill Private SPM
Forward Data to Core i

Fig. 8 Flow of our SPM access management unit with the reading policy

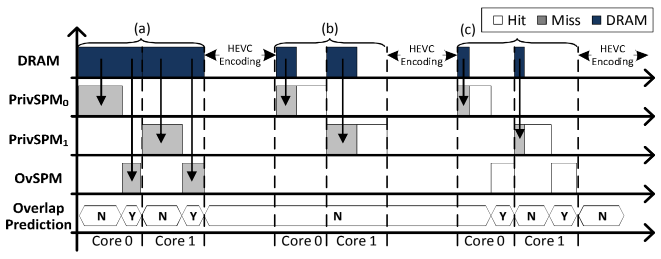


Fig. 9 An example of data interaction for a 2-core system

An Example: Fig. 9 illustrates an example for our memory reading policy in three different cases for a 2-core encoding system.

a) In the beginning, the on-chip SPMs are empty and each request will lead to external memory fetching (*OvSPM* and *PrivSPM* misses). Fig. 9 shows that the overlap prediction is analyzed to determine whether the reference data is stored in the $PrivSPM_i$ or in the $OvSPM_T$. During the frame processing, due to the intra-Tile and inter-Tiles reused data, more hits occur and even less external memory communication is needed.
b) The second case in Fig. 9 depicts tile-centering CUs processing where only the *PrivSPMs* is accessed (i.e. only intra-Tile data reuse).
c) The last case shows the best case of energy efficiency, where memory hits are observed for both *PrivSPMs* (i.e. intra-Tile data reuse) and *OvSPMs* (i.e. inter-Tiles data reuse) accessing.

B. *Design of Scratchpad Video Memories*

A key challenge is to determine an appropriate size and organization of different SPMs (*PrivSPMs* and *OvSPMs*) to optimize for leakage and dynamic energy. We propose an application-guided methodology that exploits the statistical analysis of memory access behavior Tile-parallelized HEVC in order to increase the energy efficiency of our dSVM architecture.

Our methodology leverages the Tiles overlap behavior that depends on the search window size and the video motion properties. Adaptive ME algorithms change the center of their search windows by using spatial predictors (i.e., motion vectors of previously-coded CUs). Moreover, low motion CUs will lead to less search window usage. Hence, the optimal overlapping memory size for each video sequence follows a statistical distribution of the near-boundaries ME motion predictors. Fig. 10a depicts statistics of the tiles overlap varying the search window size. On average, the overlap linearly increases with the increase n the search range. The more or less concentrated distribution around the average size hints towards the video motion properties. Different regions near the tile boundaries have different motion characteristic, which leads to more or less memory access overlaps.

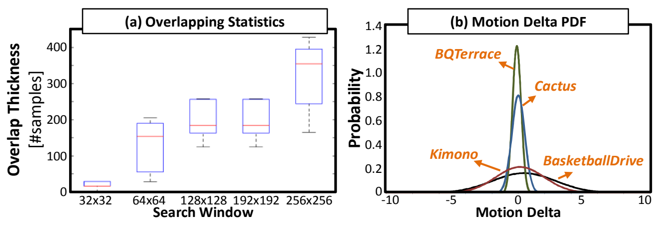


Fig. 10 (a) Overlapping statistics for increasing search window size for the "BasketballDrive" test video sequence;
(b) motion delta distribution for several test video sequences

To statistically define the motion property near a specific tile boundary of a given video, we define the Δ_{Motion} (motion delta) metric as being the *video correlated parameter used for determining the overlap size*, as presented in Fig. 11. For each frame of the video and for each defined Tile boundary, the algorithm obtains the used ME spatial predictors (*lines 7-8*). The difference of the predictors used by the near-boundary CUs from the two Tile boundary sides (*SideA* and *SideB*) is then calculated (*lines 12-17*). This difference will represent the access search range of *SideA* CUs in the *SideB* reference frame region, and vice-versa. The Probability Density Function (PDF) of the motion delta metric is then calculated (*line 20*), where μ_Δ and σ_Δ are the statistical average and standard deviation, respectively, of the motion delta parameters extracted from the video. The PDFs for HD1080p test sequences are plotted in Fig. 10b. We can note diverse behaviors depending on the input video: high motion videos like *BasketballDrive* and *Kimono* present more spread distributions, while low motion videos like *Cactus* and *BQTerrace* have more concentrated distributions. Using the motion parameter and the search window dimension, we define the Tiles overlap sizing formula for the overlap thickness ($Ov_{Thickness}$) and length (Ov_{Length}) in Eq. (1)-(2), respectively. The signal of the motion delta represents the video motion direction near the target tiles boundary. Negative values mean that we have opposite motion directions, which decreases the overlap size, while positive motion delta values increases the range of the overlap.

```
1.  determineMotionDelta(Video: V; TilePartitioning: TP):
2.  List_Δ = [ ];
3.  For all Frame ∈ V
4.    For all Tile_ID ∈ TP
5.      PredMap[Tile_ID] = [ ];
6.      For all CU ∈ Tile_ID
7.        CU.performMotionEstimation();
8.        PredMap[Tile_ID].insert(CU.getUsedPredictor());
9.      End For
10.   For all TileBoundary_ID ∈ TP
11.     //Let SideA and SideB the two tile boundary sides
12.     For all CU_SideA, CU_SideB ∈ TileBoundary_ID
13.       PredA := PredMap[Tile_SideA][CU_SideA][Coord_ID];
14.       PredB := PredMap[Tile_SideB][CU_SideB][Coord_ID];
15.       Δ_Value := |PredA – PredB|;
16.       List_Δ.append(Delta_Value);
17.     End For
18.   End For
19. End For
20. {μ_Δ, σ_Δ} = norm_dist(List_Δ);
21. return {μ_Δ, σ_Δ};
```

Fig. 11 Motion knowledge extraction for overlapping SPM sizing

$$\mathrm{Ov_{Thickness}}(\mathrm{TileBoundary_{ID}}) = 2 \times \mathrm{SW} + \Delta_{\mathrm{Motion}} \quad \text{where: } \Delta_{\mathrm{Motion}} = \mu_\Delta + 2 * \sigma_\Delta \tag{1}$$

$$\mathrm{Ov_{Length}}(\mathrm{TileBoundary_{ID}}) = \begin{cases} \mathrm{H_{Frame}} & \text{if vertical} \\ \mathrm{W_{Frame}} & \text{if horizontal} \end{cases} \tag{2}$$

Based on statistical evaluations and the memory organization model defined in the Section II, we determine the physical sizing for SPMs in our dSVM; see Eq. (3)-(7). For the overlapping data, each Tile boundary will leads to a specific *OvSPM* design. Our sizing formulation is based on the definition of the BU size (BU_{Size}), which is the smaller unit that can be accessed. For instance, a BU_{Size} equals to 16 means that the smaller data transmission unit is one 16x16 reference block. The BU size is a design decision for efficient power management depending on the adopted search window dimension. One BU in the overlap is mapped to a specific memory line (composed of $OvSPM_{SL}$ bytes) along the $OvSPM_{NB}$ memory banks. Each *OvSPM* sector groups specific rows of the BUs along the overlap thickness ($OvSPM_{SS}$). One entire line of BUs is completely stored into a group of same positioned sectors along the $OvSPM_{NB}$ memory blocks. In total, each *OvSPM* has $OvSPM_{NS}$, to store the complete overlapping data.

$$\mathrm{N_{OvSPM}} = \mathrm{N_{TilesBoundaries}} \tag{3}$$

$$\mathrm{OvSPM_{S_L}} = \mathrm{BU_{Size}} \tag{4}$$

$$\mathrm{OvSPM_{N_B}} = \mathrm{BU_{Size}} \tag{5}$$

$$\mathrm{OvSPM_{S_S}} = \lceil \mathrm{Ov_{Thikness}} / \mathrm{BU_{Size}} \rceil * \mathrm{S_L} \tag{6}$$

$$\mathrm{OvSPM_{N_S}} = \mathrm{OvSPM_{N_B}} * \lceil \mathrm{Ov_{Thikness}} / \mathrm{BU_{Size}} \rceil \tag{7}$$

The *PrivSPM* stores the search window samples, as expressed in Eq. (8)-(12). The data organization is similar to that presented for the *OvSPMs* except that the *PrivSPM* must store core-private search window instead of Tile overlaps.

$$\mathrm{N_{PrivSPM}} = \mathrm{N_{Tiles}} \tag{8}$$

$$\mathrm{PrivSPM_{S_L}} = \mathrm{BU_{Size}} \tag{9}$$

$$\mathrm{PrivSPM_{N_B}} = \mathrm{BU_{Size}} \tag{10}$$

$$\mathrm{PrivSPM_{S_S}} = \lceil \mathrm{SW_H} / \mathrm{BU_{Size}} \rceil * \mathrm{S_L} \tag{11}$$

$$\mathrm{PrivSPM_{N_S}} = \mathrm{PrivSPM_{N_B}} * \lceil SW_V / BU_{Size} \rceil \tag{12}$$

IV. Adaptive Power Management of SPMs

In case where the overlap size is reduced when low motion is captured around the tiles boundary, we propose an adaptive power management scheme for the *OvSPM* in our dSVM architecture to reduce its leakage energy. Furthermore, *PrivSPMs* are less accessed when CUs near the Tile boundaries are encoded since most memory requests are actually performed in the *OvSPMs*. Therefore, our scheme power-gates the *PrivSPMs* regions that are not accessed due to the overlap intersection.

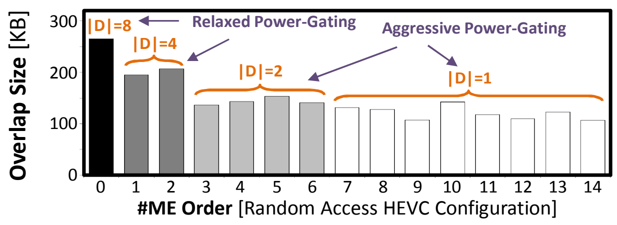


Fig. 12 Overlap sizing variation for several temporal distances (D factor).

To capture the impact of temporal distance for overlap size prediction, we define a term "D" as the distance between the current and reference frames. This distance directly affects our overlap prediction. More distant frames (i.e. high D values) lead to high overlap size due to more intense motion activity. Smaller overlaps can be noted when D is decreasing. Fig. 12 illustrates the overlap size for MEs with different D factors. Our power management selects an appropriate sleep state according to the motion behavior: relaxed power-gating (i.e. putting idle sectors in data retentive modes) is used when we have high motion overlaps. In case of low motion overlaps, aggressive power-gating (i.e. putting sectors in power-OFF mode) is applied to save more leakage energy.

```
1.  managePowerOverlapSPM (Frame: $F_{Current}$, $F_{Reference}$;
    TileBoundary: $TileBoundary_{ID}$)    //frame-level management
2.  currOverlapUsage := 0;
3.  {$\mu_{Usage}$, $\sigma_{Usage}$} := getOverlapUsages();    //run-time statistics
4.  $D_{ME}$ := getPoc($F_{Current}$) – getPoc($F_{Reference}$);    //overlap prediction (lines 3-4)
5.  PredOv($TileBoundary_{ID}$) := $\begin{cases} \mu_{Usage} - \sigma_{Usage} & \text{If } D_{ME} = 1 \\ \mu_{Usage} & \text{If } 2 \geq D_{ME} \geq 3 \\ \mu_{Usage} + \sigma_{Usage} & \text{If } 4 \geq D_{ME} \geq 7 \\ \mu_{Usage} + 2.\,\sigma_{Usage} & \text{If } D_{ME} \geq 8 \end{cases}$
6.  $PowerMap_{Ov}$(x,y) := $\begin{cases} S0 & \text{If } (x,y) \in \text{to predicted overlap} \\ S3 & \text{otherwise} \end{cases}$
7.  For all CTU ∈ {$Tile_0$, $Tile_1$, …, $Tile_{n-1}$}    //CTU-level management
8.      SearchLimits := getSearchLimits(CTU);
9.      $PowerMap_{Ov}$(x,y) := $\begin{cases} PowerMap_{Ov} & \text{If } (x,y) \notin (\text{PredOv} \cap \text{SearchLimits}) \\ S1 & \text{Else If } (x,y) \text{ shared by 2 tiles} \\ S2 & \text{Else If } (x,y) \text{ shared by} > 2 \textit{ tiles} \end{cases}$
10.     SPM[$TileBoundary_{ID}$].powerGate($PowerMap_{Ov}$);
11.     currOverlapUsage += performMotionEstimation();
12. End For
13. store(currOverlapUsage);
14. return;
```

Fig. 13 Adaptive power management policy for the Overlapping SPM.

```
1.  managePowerPrivateSPM(Frame Tile: $Tile_{ID}$)
2.  (∀ (x,y) ∈ $PowerMap_{SW}$, $PowerMap_{SW}$(x,y) := S3);
3.  For all CTU ∈ $TIle_{ID}$    //CU level power-gating
4.      For all $TileBoundary_{ID}$ ∈ TilePartitioning
5.          $PowerMap_{SW}$(x,y):= $\begin{cases} S0 & \text{if } (x,y) \in \text{PredOv}(TileBoundary_{ID}) \\ PowerMap_{SW} & \text{otherwise} \end{cases}$
6.      End For
7.      $SPM_{Priv}$[$Tile_{ID}$].powerGate($PowerMap_{SW}$);
8.      performMotionEstimation();
9.  End For
10. return;
```

Fig. 14 Adaptive power management policy for the Private SPM .

Fig. 13 depicts our adaptive power management policy for the *OvSPM*. At frame level, online statistics of overlap SPMs usages for previous ME are generated (*line 2*). As shown in Fig. 12, using the "D" factor of the current ME as parameter, we predict the current overlap size (*line 5*). For all SPM lines outside the predicted overlap, the *OFF* state (S0) is assigned to the *PowerMap*$_{Ov}$ corresponding position; otherwise, the ON state is assigned (S3). In CTU processing level (*line 7),* our power management checks for the non-accessed *OvSPM* positions that are inside the overlap prediction to put them in *data retentive* states (*lines 8-9*). S2 state is assigned for positions potentially accessed by more than two Tiles, while S1 state is used for overlap positions shared for only two Tiles. The overlap usage for the current ME is updated at every CTU processing (*line 11*) and saved to be used for future overlap predictions (*line 13*).

The adaptive power management policy for the *PrivSPMs* is depicted in Fig. 14. At the beginning of a CTU processing, it checks for intersected positions between the core-private search window and any predicted overlap. For each intersection, it power-gates the corresponding PrivSPM positions *(line 5* in Fig. 14). Note, both *OvSPM* and *PrivSPM* managements work in parallel in our dSVM system.

V. Experimental Results

A. *Experimental Setup*

The experimental analysis is based on the recommended HEVC common test conditions [13] using the HEVC test model (HM 11.0) [14]. We execute the experiments for 4-Tile and 8-Tile scenarios (each Tile executes on a dedicated core) with five search window dimensions: 64x64, 96x96, 128x128, 192x192, and 256x256. Six test video sequences with different properties were evaluated: *BasketballDrive (BDrive)*, *BQTerrace*, *Cactus* and *Kimono* (HD1080p: 1920x1080), *PeopleOnStreet (People)* and *NebutaFestival (Nebuta)* (2K: 2500x1600). Other encoder specifications are: GOP=8, CABAC, FRExt, Random Access configuration, and TZ Search algorithm.

For memory energy evaluation, we use the CACTI 6.5 leakage/dynamic energies estimation for a 32nm SRAM-based SPM. The leakage reduction and wake-up energies were derived from the analytical model presented in [12]. The 4-Gbit Low-Power DDR2 (LPDDR2) DRAM MT42L128M16D1GU-25WT electrical specifications [8] were used to determine all external memory energy components mentioned in Section II. As a design decision for combined coarse- and fine-grained SPM management, considering the most widely used video resolutions and search window sizes (as listed above), we adopt $BU_{Size}=16$.

To evaluate the savings of our dSVM architecture, we select two other comparison partners: (a) SPMs with Level C-based data reuse for each core, and (b) our dSVM with only the PrivSPMs and no shared OvSPMs. The energy evaluations consider the first 30 consecutive frames of each test video sequence.

B. *Energy Savings*

Tab. 1 presents the overall energy evaluation with a breakdown of off-chip and on-chip memory energy consumption.

Tab. 1 Overall Energy Consumption Evaluation

	SPMs Size [KB]	On-Chip Energy [mJ]	Off-Chip Energy [mJ]	Overall Energy [mJ]	Savings dSVM [%]
Scenario 1: *4-Tile HD1080, 129x129 search window*					
Level C [11]	144	16	587	603	36%
Our PrivSPM Only	144	16	469	485	21%
Our dSVM	614	33	351	384	-
Scenario 2: *8-Tile HD1080, 129x129 search window*					
Level C [11]	288	33	587	620	61%
Our PrivSPM Only	288	32	462	494	51%
Our dSVM	1098	63	179	242	-

Tab. 1 shows that our complete dSVM architecture provides the best energy efficiency for the two tested scenarios. Considering the accumulated size of SPM blocks (private plus overlapping), the dSVM architecture presents the highest memory usage. However, our adaptive power management is able to significantly reduce the leakage consumption and accordingly adapting the power states to the predicted overlap size and shape. Therefore, the dSVM architecture can reduce the on-chip energy consumption being competitive with the related non-shared memories approaches. Furthermore, this slight on-chip energy overhead is amortized by significant savings in the external memory transfers that leads to overall savings of 21%-36% compared to Level C and our PrivSPM Only solution (scenario 1), respectively. In the scenario 2, our energy savings even increase to 51%-61% compared to Level C and our PrivSPM Only solution, respectively. Note that our dSVM architecture provides increasing overall savings when more Tiles (i.e. parallel HEVC threads) are used (2x higher savings, on average). Extrapolating our results for more than 8 video Tiles (as more inter-Tiles data reuse potential can be exploited), our dSVM can achieve even higher memory energy savings.

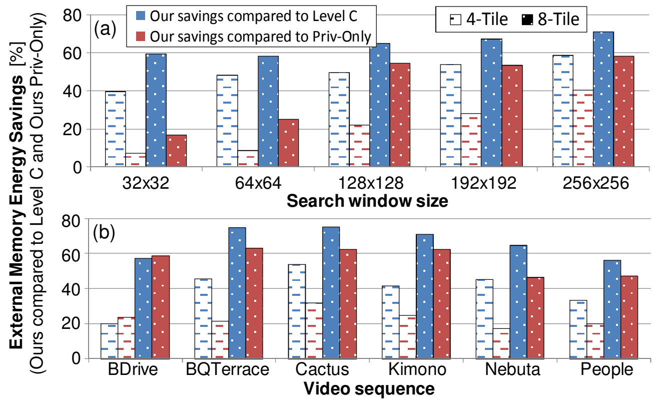


Fig. 15 External memory energy savings for 4-tile and 8-tile scenarios: (a) average savings for all sequences varying the search window size and (b) savings for each tested sequence (128x128 search window size)

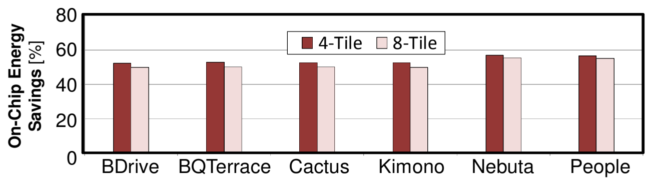


Fig. 16 Leakage energy savings due to our dynamic power management of the dSVM architecture (128x128 search window size)

Off-Chip Memory Energy Savings: Fig. 15 depicts the external memory energy savings of our dSVM for different search window sizes, input video test sequences, and the number of parallelized Tiles. Fig. 15 shows that as more Tiles are used, more external memory energy is saved due to the larger overlap. Our dSVM architecture supersedes other comparison partners by exploiting our novel concept of *both* intra-Tile and inter-Tiles data reuse. The dSVM savings increase with the growing search window from 7% to 58% for the 4-Tiles partitioning and from 17% to 71% for the 8-Tiles partitioning. This due to exploiting the shared memory accesses coming from different processing cores. Furthermore, there are savings also vary depending upon the video content: low motion videos leads to less overlap and less potential of reduction. In the best case, the *Cactus* sequence achieves an external memory energy reduction of 55% and 74% for 4-Tiles and 8-Tiles partitioning (using 128x128 search window size).

On-Chip Memory Energy Savings: Fig. 16 depicts the on-chip leakage energy savings of our dSVM architecture due to our adaptive power management policy. On average, our policy reduces the leakage energy by 54% and 52%, considering 4-Tile and 8-Tile scenarios. Part of the savings is related to the *PrivSPMs* energy management, which captures the intersections of the search window positions with the any predicted overlap. Regarding the *OvSPMs*, our scheme can significantly reduce the leakage energy for low motion ME, where the overlap tends to be small.

VI. Conclusion

This work presented a distributed Scratchpad Video Memory Architecture for the next-generation parallel High Efficiency Video Coding. It exploits intra- and inter- video Tile level data reuse jointly through private and shared SPMs of different cores executing parallel HEVC threads. The SPM design is based on application-specific knowledge of HEVC and statistical analysis of memory access behavior w.r.t. the video content properties. To further reduce the leakage energy, we integrated an adaptive power management policy for SPMs that exploit the prediction of the overlapping accesses from different cores and their relationship to the video content properties. Our dSVM architecture provides up to 61% reduction in the overall memory energy and 54% in the leakage energy compared to state-of-the-art. Our proposed contribution enables energy-efficient multimedia systems supporting multiple threads of the next-generation HEVC encoder.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89); http://invasic.de and partly supported by DAAD and CAPES as part of the PROBRAL project "VideoArch^{3D} - Power-Efficient Techniques for 3D-Multimedia".

References

[1] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, T. Wiegand, “High Efficiency Video Coding (HEVC) text specification draft 7”, May 2012.
[2] D. Marpe, et al, “Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding,” IEEE TCSVT, vol. 20, no. 12, pp. 1676–1687, 2010.
[3] B. M. T. Pourazad, C. Doutre, M. Azimi, P, Nasiopoulos, “HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?,” IEEE CEM, pp. 36-46, 2012.
[4] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, J. Henkel, "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding", IEEE/ACM/EDA DAC, pp. 1026-1031, 2011.
[5] M. Shafique, B. Zatt, F. L. Walter, S. Bampi, J. Henkel, “Adaptive Power Management of On-Chip Video Mamory for Multiview Video Coding”, IEEE/ACM/EDA DAC, pp. 866-875, 2012.
[6] B. Zatt, M. Shafique, S. Bampi, J. Henkel, “A Low-Power Memory Architecture with Application-Aware Power Management for Motion & Disparity Estimation in Multiview Video Coding”,IEEE/ACM ICCAD, pp. 40-47, 2011.
[7] F. Sampaio, B. Zatt, M. Shafique, J. Henkel, S. Bampi, “Energy-Efficient Memory Hierarchy for Motion and Disparity Estimation in Multiview Video Coding”, IEEE/ACM DATE, pp. 665-670, 2013.
[8] Micron. “4Gb: x16, x32 Mob. LPDDR2 SDRAM S4”. Rev. N 05/13 EN, 168p, 2013.
[9] Micron. “TN-46-03 – Calc. DDR Mem. System Power”. Rev. B 3/05 EN, 26p, 2005.
[10] Micron. “TN-46-12: Mob. DRAM Power-Sav. Features/Calc.”, 10p, 2009.
[11] C.-Y. Chen, C.-Y. Chen, C.-T. Huang, L.-G. Chen. “Level C+ Data Reuse Scheme for Motion Estimation with Corresponding Coding Orders”, IEEE TCSVT, vol. 16, no. 4, p. 553-558, 2006.
[12] H. Singh, L. 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.
[13] F. Bossen, “Common test conditions and software reference configurations”, ITU-T/ISO/IEC JCTVC-K1100, October 2012.
[14] JCT-VC. HEVC Software SVN, 2011. Available in: <https://hevc.hhi.fraunhofer.de/>
[15] Misra, K.; Segall, A.; Horowitz, M.; Xu, S.; Fuldseth, A.; Zhou, M., "An overview of tiles in HEVC," IEEE JSTSP, no.99, 2013
[16] C. Blumenberg, D. Palomino, B. Zatt, S. Bampi. “Adaptive Content-Based Tile Partitioning Algorithm for the HEVC Standard”, PCS, p. 185-188, 2014.
[17] JCT-VC, “High Efficiency Video Coding (HEVC) text spec. draft 10”, 2013.
[18] Iyengar, A., "Design and performance of a general-purpose software cache," IEEE IPCCC, vol., no., pp.329,336, 1999.
[19] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, P. Marwedel, “Scratchpad Memory: Design Alternative for Cache on-chip Memory in Embedded Systems,” CODES+ISSS, pp. 73–78, 2002.
[20] K. Bai and A. Shrivastava, “Automatic and efficient heap data management for limited local memory multicore architectures,” IEEE DATE, 2013, 2013, pp. 593-598.
[21] N. Deng, W. Ji, J. Li, F. Shi, and Y. Wang, “A Novel Adaptive Scratchpad Memory Management Strategy,” IEEE RTCSA pp. 236–241, 2009.
[22] B. Egger, S. Kim, C. Jang, J. Lee, S. L. Min, and H. Shin, “Scratchpad Memory Management Techniques for Code in Embedded Systems without an MMU” IEEE TC, vol. 59, no. 8, pp.1047-1062, 2010.
[23] IBM, “The Cell Project”, Last Accessed: Sep. 2013, <http://researcher.watson.ibm.com/researcher/view_project.php?id=2649>.
[24] ARM, “ARM10 Family: An Overview”, pp. 11, 2005
[25] Texas Instruments, “TMS370CX7X from Texas Instruments”, <www.ti.com/mcu/docs/mcuorphan.tsp?contentId=15364>.