Software Architecture of High Efficiency Video Coding for Many-Core Systems with Power-Efficient Workload Balancing

Muhammad Usman Karim Khan, Muhammad Shafique, Jörg Henkel
Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{muhammad.khan, muhammad.shafique, henkel} @ kit.edu

***Abstract*—The High Efficiency Video Coding (HEVC) standard aims at providing ~50% better compression compared to its predecessor (H.264) at the cost of high computational complexity. To enable HEVC video encoding in real-time scenarios, special coding support for parallelization is provided in HEVC that can be exploited by many-core systems. In this work, we present a HEVC software architecture where a video frame is adaptively divided into independent video frame regions (i.e. so-called video tiles) which are processed concurrently on multiple cores. By balancing the workload of each video tile mapped to a particular core, the total power consumption of a system is reduced (through dynamically scaling the operating frequency) under a given frame-rate constraint. We also exploit user tolerance to further curtail the HEVC workload with insignificant video quality degradation. Experimental results illustrate that the proposed approach results in ~43% power savings on a many-core system.**

I. Introduction and Our Novel Contributions

The growth of sub-micron fabrication technologies has led to significant increase of video devices in the consumer market. With the introduction of high-quality cameras and video recorders, and demands of high quality video transmission, the essential necessity of efficient video compression has commenced a competition among different video encoding standards. For example, VP9 [1] available since June 17th 2013, Daala [2] available since May 30th 2013 and High Efficiency Video Coding (HEVC) [3] available since June 7th 2013, are the newest video coding standards. These encoders aim to reduce the bit-rate by ~50% as compared to the current video encoding standards while maintaining the same video quality.

HEVC is a successor of the industry's current video coding standard, H.264 or Advanced Video Coding (AVC) [4]. The growth in digital entertainment, security and video communication industry has led the HEVC standard developers to provide efficient compression means to encode large video content (e.g., Ultra HD 7680×4320 pixels at 120 frames per second). Furthermore, due to an anticipated increase in HEVC employment, electronic vendors have already started shipping HEVC compatible decoders in the market (Samsung's F8500 plasma TVs [7] and S4 smart phones [8]), but there is a long way to go for power-efficient HEVC encoders (HEVC reference encoder is ~100x more complex compared to the reference decoder [9]).

Problem Statement: By aiming at 50% bit-rate reduction and preserving the same subjective video quality as H.264, HEVC has become a prime candidate to replace H.264 encoders [5][6]. Conversely, this gain in compression efficiency comes at a high cost of computational complexity due to the inclusion of numerous additional encoding tools (e.g. coded tree block, numerous new intra prediction modes, see details in Section III.A). However, in real-world video encoding systems, the video must be compressed under tight constraints of time budget [10] and output bit-rate. This requirement gathers more ground for HEVC as compared to H.264, due to its heavy workload, large data-structures and limitations in high processing power available to support real-time HEVC encoding [11]. The additional tools and timing-constraints give rise to several challenges for implementing a HEVC system on a hardware platform [12]. Our simulations with reference software have shown that HEVC encoding is ~1.7× more complex, providing a 33% increase in the compression efficiency compared to H.264. Moreover, one CIF frame (352×288) for HEVC intra encoding takes about 1.5 seconds and 6.94 Joules on an *x86* core running at 2.66 GHz. Therefore, workload balancing and power reduction is a fundamental requisite of HEVC.

In the current many-core era (where multiple cores are even available in the smart phones), it is beneficial to utilize the inherent parallelism offered by HEVC. By distributing the workload of HEVC encoder on multiple cores, the total encoding time can be reduced that may potentially improve the overall energy efficiency. HEVC standard allows exploiting parallel encoding tools, like slices and *video tiles*[1] to fulfill these requirements [13]. A tile (rectangular part of a video frame) is treated as an autonomous entity and can be encoded independently of other tiles within the same video frame. However, ignoring the encoding hardware characteristics during HEVC parallelization may lead to high power consumption, deadline misses, and video quality loss. System parameters like total number of cores and maximum operating frequency along with the video characteristics determine the maximum sustainable workload. Additionally, workload of each tile differs from that of the other tiles and even for one video tile it varies at run-time due to the varying video-properties (see our analysis in Section III). Thus, **the challenge is** to distribute the workload of HEVC selectively and adaptively on multiple cores. Moreover, HEVC video encoder on a many-core system must exploit the changing workload to reduce the total power consumption while meeting the quality of service demands. The power consumption can be dynamically reduced by using a workload-driven operating frequency adaptation scheme. Also, the user's tolerance to the output bit-rate can be exploited to adapt the workload, by reducing the coding complexity.

A. Our Novel Contributions

To enable power-efficient real-time HEVC video encoding on many-core systems with minimum video quality degradation, *we propose a novel software architecture to adaptively select the tile structure and consider user tolerance for run-time workload management of each core.* We account for the HEVC workload and the hardware platform properties (like maximum operating frequency and total number of cores) of the many-core system. Our software architecture employs:

- **A tile-generator** to select the tile structure and the maximum workload of each core under constraints of quality of service (frame-rate) and available system resources.
- **An online workload allocator** to curtail workloads (with minimal video quality distortion) of different cores at run-time, by considering user tolerance.
- **A tile-based workload manager** to manage the operating frequency of the underlying core (associated with an independent

[1] Throughout the text, a tile refers to a *video tile.*

978-3-9815370-2-4/DATE14/©2014 EDAA

tile), in order to fulfill the workload and encoding time requirements while minimizing the power consumption.

Our system overview outlining the above-mentioned novel contributions is shown in Fig. 1. Each tile is processing on a dedicated core. To the best of the authors' knowledge, this is the first software architecture which provides power-efficient workload balancing of HEVC encoding on many-core platform.

Paper Organization: State-of-the-art schemes in HEVC workload balancing and complexity/power reduction are discussed in Section II. Section III briefly discusses HEVC background and outlines our motivational analysis. In Section IV, the proposed scheme is discussed in detail. Experimental results are presented in Section V. We conclude the paper in Section VI.

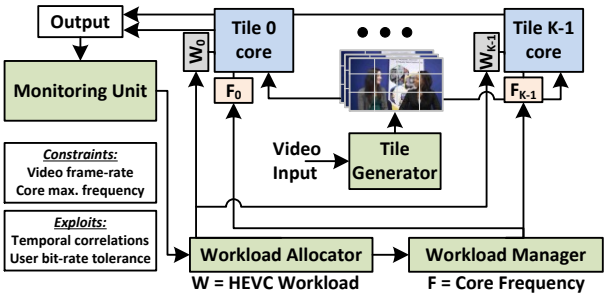

Fig. 1: Overview diagram of our novel contributions

II. Related Work

As pointed out in the previous section, real-time encoding via HEVC is difficult to achieve. Therefore, efficient algorithms and architectures are required for processing HEVC workload. Usually, coding efficiency is compromised, HEVC specific ASICs are proposed or large silicon area is consumed for gains in computational complexity. In addition, due to the high workload of HEVC, high power consumption and the elevated temperatures of the underlying hardware platform are expected for HEVC. A comprehensive analysis of these and additional issues with HEVC encoding is detailed in [12]. General categorization of workload mapping techniques on many-core systems is presented in [14]. Unlike general purpose techniques, our approach exploits the application-specific properties of HEVC for workload mapping and improved power efficiency. In addition, a number of schemes regarding the workload management of video encoders have been proposed in the literature [15]. However, they do not exploit the advantages offered by many-core systems for video encoding. For example, in [16], authors have proposed to limit search comparisons of the best coding configuration in HEVC, depending upon the encoding history. In [17], the workload of H.264 encoder on a single core processor is controlled by analyzing the Motion Estimation (ME) process. In [18], a joined distortion-complexity scheme is proposed for H.264, by configuring ME and mode decision. The above mentioned works focus only on single core implementation, and do not consider the underlying hardware properties. A SIMD implementation of H.264 on many-core processor is given in [19], but it does not consider workload mapping and balancing. Additionally, a multitude of works in the direction of reducing the complexity of HEVC encoding also exist. [20] implements early skip approaches to reduce HEVC configuration searches. Software and hardware collaboration is exploited in [11] to reduce the total time complexity and energy consumption of HEVC Intra encoder, by mapping the compute intensive part of the encoder to hardware. [21] and [22] configure the total Intra modes tested for the HEVC encoding process and reduce the time complexity. VLSI architecture for efficient Intra prediction is given in [23]. In short, state-of-the-art complexity reduction and workload balancing schemes of HEVC do not consider workload balancing on multiple cores to exploit the inherent parallelism offered by the HEVC standard.

III. Our Analysis of HEVC Encoder

A. Brief Background of HEVC

Compared to H.264, additional compression tools are added to HEVC encoding engine to boost its compression efficiency. In Fig. 2, the video partition hierarchy of HEVC encoder is shown. The frames to be encoded are grouped in Group of Pictures (GOP). Each GOP contains M video frames and there are K tiles (T) in each frame, where $K \geq 1$. A frame can also be divided into independent slices.

In HEVC, each tile is divided into a Coding Tree Units (CTU) of size 64×64 pixels or smaller. A CTU is recursively subdivided into smaller, square sized blocks called Coding Units (CU). Maximum CU size can be equal to the CTU size and minimum CU size is 8×8 pixels. CU is the basic entity of compression in HEVC. Intra mode predictions and motion estimation is performed for individual CUs.

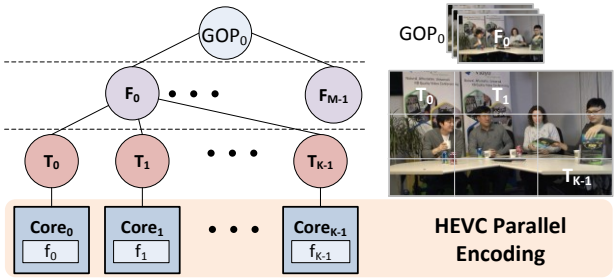

Fig. 2: HEVC video partition hierarchy

Similar to H.264, CU encoding has sequential dependencies with its neighboring CUs, and hence forbids parallel encoding of CUs. On the other hand, a tile encompasses a set of CTUs for HEVC encoding, and CUs in one tile are independent of CUs in the other tiles. Same is true for slices. However, note that each slice is treated as an independent entry in the bitstream, thus requiring addition bits to present the slice-headers.

B. HEVC Analysis

From the discussion in Section III.A, it is evident that tiles can be processed via independent encoding threads. Although GOP, frame and slice [13] level parallelism can also be exploited in place of tiles, but tiles are preferred for 2 reasons:

1- Tiles are at the lowest level of coding hierarchy. Therefore, they will consume least system memory, and hence, will be fastest among other parallelisms.

2- Unlike slices, tiles do not have their associated headers. Thus, tiles exhibit the potential to provide relatively better output video quality compared to slices.

Therefore, in order to achieve fast encoding at minimal video quality loss, we propose utilizing HEVC parallelism at tile-level.

Fig. 3 shows the histogram of percentage difference in time complexity and output compressed bytes generated between collocated tiles within consecutive frames of a video sequence (see Fig. 9). For Fig. 3(a), the horizontal axis presents the percentage time difference, β, between collocated tile number 0 of consecutive video frames. β is given by the following formula:

$$\beta = \frac{T_0(t,i) - T_0(t,i-1)}{T_0(t,i)} \times 100 \quad (1)$$

Here, $T_0(t,i)$ presents the time taken to encode the 0^{th} tile (first tile) of frame i. As seen, the time complexity for collocated tile is highly correlated. Identical formula is used for Fig. 3(b) by replacing

time with total bytes. Similar to the time complexity, the number of compressed bytes generated by collocated tiles is well correlated. Moreover, these curves can be estimated via a Normal (Gaussian) distribution. *Thus, the correlation between collocated tiles can be exploited to determine the estimated bit-rate and time complexity of the current tile, which can be translated to determine the workload.*

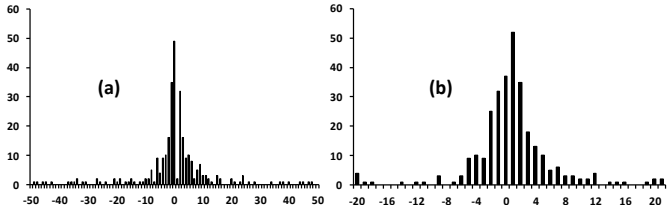

Fig. 3: Difference histogram of (a) Time and (b) Bytes for collocated tile number 0 of Foreman sequence (352×288) for 300 frames

In Fig. 4, time complexities of individual tiles per frame are plotted for Keiba sequence (832×480). A total of 4 tiles are used with 2 tile rows and 2 tile columns per video frame. We notice that the complexity of each tile not only differs from the other tiles, it also varies at run-time. Hence, the power consumption and idle times of each core may fluctuate over time owing to the changing workload. *Therefore, the complexity of each tile is different and must be separately adapted to balance workload among multiple cores. This adaption must consider the frame-rate requirement and compute constraints. Further, number of tiles must not be unnecessarily increased as it will increase the power consumption of the system.*

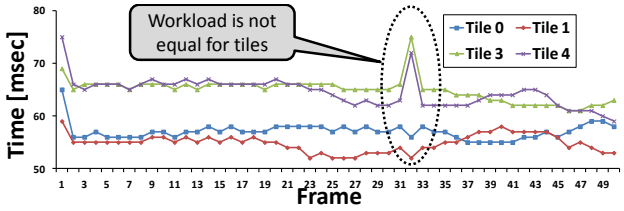

Fig. 4: Time consumption of each tile (4 tiles per frame, of 2×2 tile structure) of first 50 frames of Keiba sequence (832×480)

In Fig. 5, the average time complexity and video quality curves for different tile structures for Foreman sequence (352×288) is plotted. We notice that increasing the number of tiles results in time complexity improvement (Fig. 5(a)). However, at the same time it results in small quality degradation as shown in Fig. 5(b). In fact, a single tile per frame generates the best quality. For T^2 tiles per frame, $T\times T$ tile structure (i.e. T tile rows and T tile columns per slice) results in the best video quality [24]. *Therefore, if possible, the total number of tiles within a slice must be minimized and the total tile-rows and tile-columns within a slice must be equal, in order to increase the video quality.*

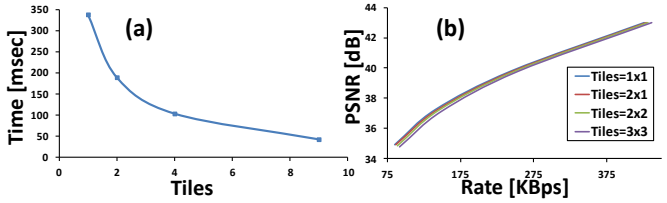

Fig. 5: (a) Average time per frame and (b) rate-distortion curves for Foreman sequence (352×288) with different tile structures

IV. HEVC Tile-Based Workload Balancing

The proposed software architecture for tile-based workload balancing of HEVC-Intra is shown in Fig. 6. The software is responsible for:

1- Setting up the tile structure by considering hardware resources (operating frequency, total number of cores) and quality of service demands (frames per second).

2- Allocating workload to each core by utilizing user's tolerance of the output bit-rate.

3- Managing the workload by adapting the operating frequency of each core in order to reduce power consumption.

Before start of encoding, the total number of tiles required and the maximum workload of the cores is determined. While encoding, the input video is converted into tiles and forwarded to the respective cores. Cores process each tile individually and generate the compressed output, as well as the statistics required for workload balancing. These statistics are provided to the workload allocator. The workload allocator adaptively curtails or increases the computational workload of each tile, within the user's tolerance of the bit-rate. The allocator's output is passed to the workload manager, which dynamically scales the operating frequency of the cores, depending upon the estimated workload. In short, the software adapts the operating frequency and thus, controls the power consumption of the many-core system.

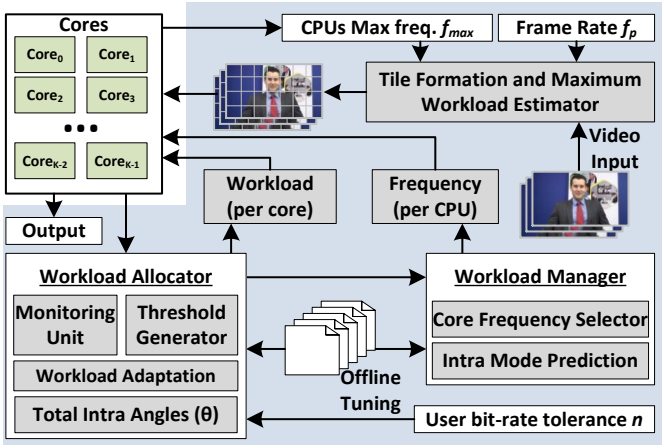

Fig. 6: Tile-based workload balancing for HEVC on many-core systems

A. Tile Formation and Maximum Workload Estimation

Before running HEVC-Intra encoding, the proposed scheme interacts with the hardware to determine workload mapping policy. Workload balancing of HEVC-Intra proposed in this work is a two step approach. First, the number of cores is determined to distribute the HEVC-Intra application's workload. If cores are insufficient, the complexity of the HEVC-Intra encoder is curtailed itself, effectively curtailing the total workload.

There are multiple complexity knobs in HEVC-Intra encoder which can be adjusted to reduce the total workload at the cost of increased output bit-rate. In this work, we propose to adjust the number of Intra predictions (or directions, given by θ) to curtail the computational complexity (these predictors are shown on the left side of Fig. 10). For HEVC-Intra, a set of angular predictors is searched to select only one prediction out of θ which results in the best rate-distortion metric. Note that θ can reach 35, and is a major bottleneck for HEVC-Intra encoders.

The workload, ρ_k, of tile number k, with a frame-rate demand of f_p frames per second is given by:

$$\rho_k(\theta, QP) = N_{CTU,k} \times C_{\theta,QP,K} \times f_p \qquad (2)$$

Here, $C_{\theta,QP,K}$ is the number of cycles consumed by a CTU of a frame with K tiles, with the given θ and QP values. $N_{CTU,k}$ is the total number of CTUs of tile k. Note that ρ_k actually denotes the total number of cycles consumed per second for the given tile.

To initialize the total number of tiles (cores) and maximum possible θ ($\theta_{init,k}$) of tile k of the encoder, we use an algorithm as outlined in Fig. 7. Initially, we test if it is possible to support 1 tile per frame with θ=35 using the maximum operating frequency, f_{max} (line 2), to support best video quality (see Fig. 5). If not, we adaptively increase the number of tiles (line 4) and hence involve more cores in encoding. This reduces the encoding pressure on cores. However, only a specific number of cores (K) can be used for HEVC encoding which is allotted by the user of the many-core system. The total number of tiles is adjusted for a better tile arrangement by reading a lookup table (lines 7-9). For example, higher video quality is obtained by having 4×3=12 tiles instead of 1×11=11 tiles. Afterwards, we individually test the cores if they can fulfill their allotted workload (lines 11-12). If the workload is still not supportable, we start reducing maximum θ (lines 12 to 16) till a minimum possible value (5 in our case).

For best compression efficiency, we try to utilize all the available cores with maximum θ (lines 6 and 20). We exit the program if minimum θ is not satisfied while using all cores (line 17), as it is impossible to sustain the current frame-rate. If the current cores support the workload, the function returns (line 19) and the HEVC-Intra encoding can start.

B. Workload Allocator

The workload allocator engine is responsible for power-efficient workload adjustment of each tile by tuning the complexity knobs of HEVC-Intra encoder.

For workload balancing, an adaptation interval is defined (see Fig. 8). The starting tile of this interval is always a fully searched ($\theta = \theta_{init,k}$) tile to achieve best compression. This tile is known as the Key-Tile (KT). Workload of the same collocated tile in the future frames is gradually adjusted down ($\theta \le \theta_{init,k}$) to reduce the tile-workload and power consumption. These tiles are termed as Non-Key-Tiles (NKT). However, if the total number of compressed bytes (B) for the current NKT increases beyond a certain threshold, τ, we increase θ, thereby increasing the workload (see Fig. 9(b)). The threshold is set statistically using the following equation:

$$\tau = \mu(B)_{\forall\ NKTs\ \in \mathrm{RI}} + \sqrt{\upsilon(B)_{\forall\ NKTs\ \in \mathrm{RI}}} \tag{3}$$

In this equation, μ is the average bit-rate and υ is the variance of B, for all the NKTs in RI. For on-the-fly calculation of mean and variance of B, Knuth's formula is used [25]. If a certain number of frames have been processed or B exceeds a threshold (τ_{init}), adaptation and KT insertion is required. Mathematically, adaptation is done if:

$$B > (1+n)\,\tau_{init} \tag{4}$$

Initial τ of a RI, τ_{init}, equals the bit-rate of KT. Here, n is a user-defined parameter for tolerance in the bit-rate parameter. Higher tolerance will result in reduced workload and vice versa. For example, if the bit-rate of NKT is 25% more (n=0.25) than nearest KT's bit-rate, KT is reinserted.

For every CTU, if threshold in equation 4 is satisfied, θ is adjusted as:

$$\theta = \left\lfloor \mu(\theta)_{\forall\ NKTs\ \in \mathrm{RI}} + \gamma(B-\tau) \right\rfloor_{\theta_{init}}$$
$$\gamma(x) = \begin{cases} +u/2 & \text{if } x > 0 \\ -u & \text{if } x \le 0 \end{cases} \tag{5}$$

Here, u is a user defined parameter. Note that while employing equation 5, $\theta_{init,k}$ also saturates θ as it is the maximum number of modes tested for the maximum operating frequency of the core for the tile k.

The impact of θ on bit-rate and time complexity is shown in Fig. 9. Extensive offline analysis is performed for different HEVC-Intra settings to predict the workload (given in equation 2), depending upon θ size. We can estimate the total cycles consumed per CTU, $C_{\theta,QP,K}$ using a regression fit of Fig. 9(a) and this is given by:

$$C_{\theta,QP,K} = \begin{pmatrix} 32.39 - 0.32QP + 2.20\theta \\ -0.085K + 0.12K^2 \end{pmatrix} \times 10^5 \tag{6}$$

TileInitialize ( ):
Input: Maximum number of cores K; Maximum frequency of a core f_{max}; Image dimensions W×H; Quantization Parameter QP; Frame rate in frames per second f_p;
Output: Total tiles T_{tot} and initial θ_{init} per tile

```
1.  ρ ← N_CTU,1 × C_35,QP,1 × f_p;                    //Equation 2
2.  if(ρ < f_max){T_tot ← 1; θ_init,1 ← 35;}          //Best RD configuration
3.  else{
4.      T_t ← 1; x ← ρ; Core_Avail ← true;
5.      ∀i∈{1 to K cores}{if(x > f_max){T_t ← T_t + 1; x ← x − f_max;}}
6.      while(Core_Avail){                             //Some cores still available
7.          T_tot ← TileMap[T_t];                      //Actual number of tiles
8.          while(T_tot > K){
9.              T_t ← T_t − 1; T_tot ← TileMap[T_t];}
10.         ∀k∈{1 to T_tot}{                           //Test every core
11.             ρ_k ← N_CTU,k × C_35,QP,T_tot × f_p; θ ← 35;
12.             if(ρ_k > f_max){                       //Curtail workload (reduce θ_init)
13.                 while(θ > 5){
14.                     θ ← θ − 5;
15.                     ρ_k ← N_CTU,k × C_θ,QP,T_tot × f_p;
16.                     if(ρ_k < f_max) break; }
17.                 if(θ ≤ 5 and Core_Avail = false){ErrorAndExit( );}}
18.             θ_init,k ← θ;}
19.         if(∀k∈{1 to T_tot}, θ_init,k = 35){break;}
20.         if(T_tot = K){Core_Avail ← false;} else{T_t ← T_tot + 1;}
21. }
```

Fig. 7: Total tiles and θ initialization

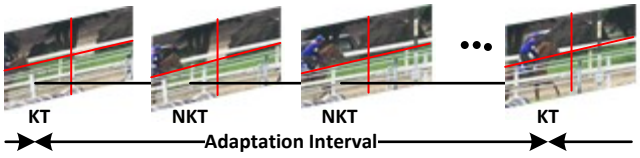

Fig. 8: Example adaptation interval for sequence frames with 4 tiles

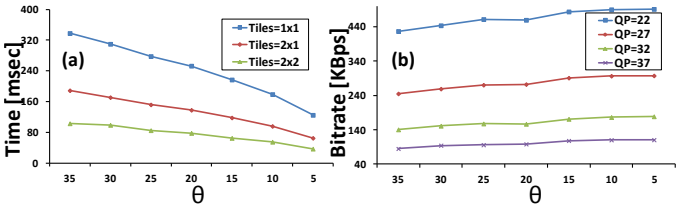

Fig. 9: Impact of θ on (a) time per frame and (b) bit-rate for different tile settings and QPs using the Foreman sequence (352×288)

C. Workload Manager

The workload manager adapts the operating frequency of each core, depending upon the workload of the core. From the discussion in Section IV.A, it is clear that the operating frequency of the core for tile k should be at least $\rho_k(\theta,QP,K)$. Given the user-defined threshold and θ adjustment, we can reduce the operating frequency even further, thus reducing the power consumption.

Intra Mode Prediction: Adjusting θ will increase or decrease the workload. However, notice that the most probable best prediction/direction must be intelligently included in θ to achieve the best RD metric. Usually, the intra prediction mode selected

corresponds to the direction of texture flow. Therefore, we also determine the most probable prediction and θ is centered on this prediction. This prediction/direction is obtained by sorting a histogram created by gradients of each individual pixel [22][26]. The gradient computation involves overhead as each individual CU pixel needs to be processed. However, we propose a much simpler solution, similar to the one presented in [27], as demonstrated in Fig. 10. Computing running differences over the boundary of current CU under test can give a good estimate of the location of a line passing through the CU, and thus, the texture flow or edge direction is obtained, as shown in the figure. This flow direction is translated into intra prediction mode. Only the predictors in the vicinity of this direction are tested, where the size of the neighborhood, θ, is adjustable.

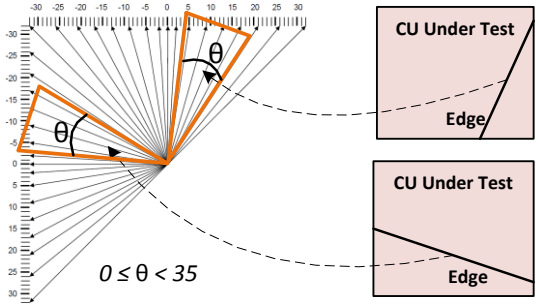

Fig. 10: Intra mode prediction via edge extraction for HEVC

V. Experimental Setup and Results

A. Software Architecture and Simulation Setup

HEVC reference encoding software (latest: HM-12.1) is provided by Fraunhofer institute on their website [28]. However, the reference software is not optimized for any specific processor. There are no threading capabilities available and it is very difficult to map the source code to include tile-based threading. Additionally, the times consumed by setup and irrelevant test conditions are too large and needlessly intrude into the coding complexity. There are other open-source encoders available (e.g. x265 [29]), however, these encoders do not include the workload balancing and adaptations required by this work.

Table I: Runtime dynamic power and power savings for different sequences with and without workload adaptation on Sniper x86 simulator

Sequence	Dimensions	Cores Used	Power [W]		Power Savings
			Adaptive	Non Adaptive	
Foreman	352×288	8	53.0985	121.306	53.23%
Flower	352×288	8	61.4772	112.62	45.41%
Coastguard	352×288	8	75.2263	126.032	40.31%
Vassar	640×480	24	271.1	440.523	38.46%
Ballroom	640×480	24	249.685	437.067	42.87%
Flamenco	640×480	24	252.813	418.359	39.57%

Table II: Average quality PSNR [dB], Rate [KBps] and time per frame [msec] on a real-world core i7 processor (n=0.2, QP=22 till 37, f_p=23, 4 threads)

Sequence	Adaptive θ			Non Adaptive θ		
	PSNR	Rate	Time	PSNR	Rate	Time
Ballroom	38.35	542.1	138	38.39	517.4	181
Exit	39.67	350.4	94	39.87	300.0	138
Flamenco	39.66	388.1	102	39.68	379.7	133
Vassar	38.35	479.4	100	38.40	447.1	157
Keiba	39.22	624.4	127	39.32	555.3	168

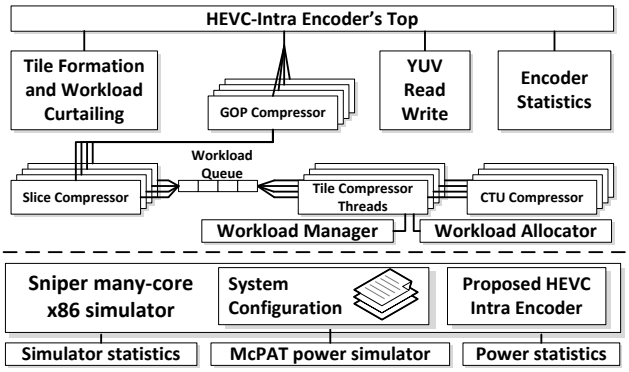

Fig. 11: (Top) Proposed HEVC-Intra encoder and (Bottom) simulation setup

Therefore, we have developed a C++ based multi-threaded HEVC Intra-encoder in our lab. **It is open source and available for download**[2]. With 1-tile (single thread) configuration, our software is ~13× faster than HM-9.2 reference software for full-HD (1920×1080) video sequences. Thus, it facilitates the embedded systems, architecture, and video coding communities to perform (1) research, development, and testing of various video related concepts at both hardware and software levels, (2) fast simulation and evaluation, (3) accurate complexity analysis. Source can be compiled on Windows and Linux-based operating systems. The parallelization schemes and tile formation can be configured by the user during start-up. Tile jobs are pushed to a work queue and the available cores pop the encoding jobs from this queue. Currently, we only utilized tile-level parallelism Inter-encoding and further extensions for inter-prediction part and advanced management and adaptations are currently under development.

Hardware platform simulation is performed via the Sniper many-core simulator [30]. Sniper simulates *x86* instruction set and supports DVFS. Only frequency scaling is considered, with a step size of ±133MHz. Power numbers are generated via McPAT [31].

B. Results and Discussion

Table I shows the total power consumption for the supported frame-rate f_p=23, with n=0.2 (equation 4) and f_{max}=5GHz, for multiple sequences. For comparison, we have also shown power consumption on a baseline implementation without workload adaptation (no workload allocator and manager). We notice that the workload allocator and manager contribute significantly to power reduction. The power savings are given by:

$$\text{Power Saving (\%)} = \left(1 - P_{Adaptive} / P_{NonAdaptive}\right) \times 100 \qquad (6)$$

Although these results are demonstrated for up to VGA (640×480) resolutions for a reasonable simulation time, similar results in power savings are expected for HD (1920×1080) resolutions and above. A detailed analysis of core frequencies and θ for the Foreman and Ballroom sequence is shown in Fig. 12 for different tiles. For the Foreman sequence, proposed HEVC-Intra software is given a total of 8 cores. The tile generator algorithm (see Fig. 7) chooses 6 out of the 8 cores with maximum coding efficiency (θ_{init}=35) to support given f_p. Thus, 2 cores are not utilized (only 6 tile compressor threads are created) and they can be shut-down or used by other applications. Ballroom sequence is provided with 24 cores. Algorithm in Fig. 7 selects all 24 cores and creates 24 instances of the tile compressor. Moreover, θ_{init} for half of the cores is set to 25 instead of 35, due to high workload. Notice how the workload allocator of each tile adapts θ with each frame by testing equation 3

[2] http://sourceforge.net/projects/ces265

and 4. For tile 0 of Foreman sequence and Tile 14 of Ballroom sequence (Fig. 12(a) and (d)), the bit-rate smoothly decreases, thus the reduction in θ by the workload allocator and the operating frequency by the manager. We notice that with every frame, θ and the operating frequency of the core increases or decreases, depending upon the bytes threshold.

The proposed software architecture is also run on a real-world core i7 computer, running Ubuntu, without operating frequency scaling, using only 4 static threads. Average video quality, bit-rate and time complexity for various sequences is tabulated in Table II. Notice that even on the above real-world platform (without SIMD or vector instructions), our software architecture produces high throughput (6-8 frames per second). Fig. 13 shows the video quality and time relationship with the bit-rate. We achieve performance improvement of ~1.3-1.57× with small video quality degradation.

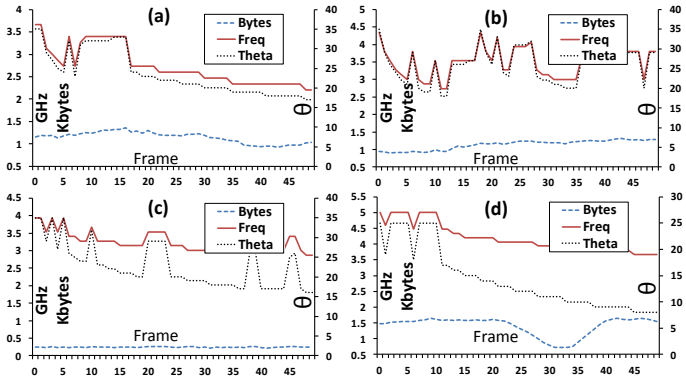

Fig. 12: Bit-rate, frequency and θ adaptation of (a) tile 0, (b) tile 4 for Foreman sequence and (c) tile 0, and (d) tile 14 of Ballroom sequence

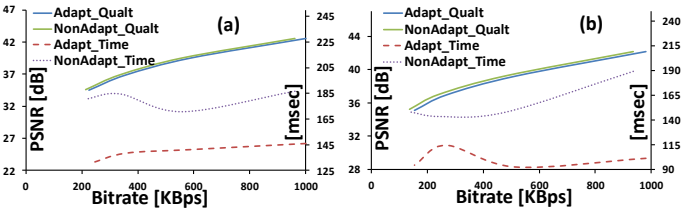

Fig. 13: Impact of θ adaptation on video quality and average time per frame for (a) Ballroom (VGA: 640×480) (b) Vassar (VGA: 640×480)

VI. Conclusion

In this work, a novel software architecture of HEVC-Intra encoding with run-time power-efficient workload balancing on many-core systems is presented. A video frame is adaptively divided into tiles. This division aims at distributing the workload on multiple cores, based upon the frame-rate requirement as well as the maximum operating frequency constraint of the underlying encoding hardware. Afterwards, power savings are obtained by exploiting user's tolerance to the bit-rate increase. Workload of the cores is curtailed or increased with every frame at run-time by adjusting the total intra modes tested. This adjusted workload is used to adapt operating frequency, thereby reducing the power consumption of the many-core system. Also, to reduce the video quality degradation, the set of predictors is intelligently selected by edge extraction. Other HEVC complexity reduction scheme can be used in conjunction with the proposed approach, as long as the proper workload adjustment knobs are identified, allowing for more workload and frequency settings.

Acknowledgement

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.

References

[1] http://www.webmproject.org/vp9/. WebM VP9 project, accessed on 28.08.2013

[2] http://people.xiph.org/~xiphmont/demo/daala/demo1.shtml. Xiph.Org Foundation's Daala Video Encoder, accessed on 28.08.2013

[3] G. J. Sullivan, J. Ohm, W. Han, T. Wiegand, "Overview of the High Efficiency Video Coding," in *IEEE TCSVT*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity," in *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[5] 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?" in *IEEE Consumer Electronics Magazine,* pp. 36–46, 2012.

[6] T. Nguyen, D. Marpe, "Performance analysis of HEVC-based intra coding for still image compression," in *PCS*, pp.233–236, 2012.

[7] Ben Drawbaugh, "Samsung 2013 LCDs and plasmas revealed: quad core CPU, new menus and more," *Engadget*. accessed on 28.08.2013.

[8] Campbell Simpson, "Samsung's Galaxy S4 has a next-gen video codec," *Pcworld*, accessed on 28.08.2013.

[9] F. Bossen, B. Bross, K. Suhring, D. Flynn, "HEVC Complexity and Implementation Analysis," in IEEE TCSVT, vol.22, no.12, pp.1685–1696, 2012.

[10] C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng, L. G. Chen, "Power-aware multimedia: concepts and design perspectives," in *IEEE Circuits and Systems Magazine*, vol. 7, pp. 26–34, 2007.

[11] M. U. K. Khan, M. Shafique, M. G. da Silva, J. Henkel, "Hardware-Software Collaborative Complexity Reduction Scheme for the Emerging HEVC Intra Encoder," in *DATE*, pp. 125–128, 2013.

[12] M. Shafique, J. Henkel, "Low Power Design of the Next-Generation High Efficiency Video Coding," in *ASP-DAC*, 2014.

[13] B. Bross et al., "High Efficiency Video Coding (HEVC) text specification draft 10," 2013.

[14] A. Singh, M. Shafique, A. Kumar, J. Henkel, "Mapping on Multi/Many-Core Systems: Survey of Current and Emerging Trends," in *DAC*, pp.1–10, 2013.

[15] 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 Encode," in *DATE*, pp. 1726–1730, 2010.

[16] G. Corrêa, P. Assuncao, L. Agostini, L. A. S. Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," in *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.

[17] W. Kim, J. You, J. Jeong, "Complexity control strategy for real-time H.264/AVC encoder," in *IEEE Transactions on Consumer Electronics*, vol.56, no.2, pp.1137–1143, 2010.

[18] L. Su, Y. Lu, F. Wu, S. Li, W. Gao, "Complexity-Constrained H.264 Video Encoding," in *IEEE TCSVT*, vol.19, no.4, pp.477–490, 2009.

[19] M. Bariani, P. Lambruschini, M. Raggio, "An Efficient Multi-Core SIMD Implementation for H.264/AVC Encoder," in *VLSI Design*, pp. 1–14, 2012.

[20] M. B. Cassa, M. Naccari, F. Pereira, "Fast Rate Distortion Optimization for the Emerging HEVC Standard," in *PCS*, pp. 493–496, 2012.

[21] H. Zhang, Z. Ma, "Fast Intra Prediction for High Efficiency Video Coding," in *PCM*, vol. 7674, pp. 568–577, 2012.

[22] W. Jiang, H. Ma, Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *CECNet*, pp. 1836–1840, 2012.

[23] F. Li, G. Shi, F. Wu, "An efficient VLSI architecture for 4×4 intra prediction in the High Efficiency Video Coding (HEVC) standard," in *ICIP*, pp. 373–376, 2011.

[24] C. Chi et al., "Improving the parallelization efficiency of HEVC decoding," in *ICIP*, pp. 213–216, 2012.

[25] D. E. Knuth, "The Art of Computer Programming," *Addison-Wesley*, pp. 232, 1998.

[26] 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," in *DATE,* pp.1713–1718, 2010.

[27] M. U. K. Khan, J. M. Borrmann, L. Bauer, M. Shafique, J. Henkel, "An H.264 Quad-FullHD low-latency intra video encoder," in *DATE*, pp.115–120, 2013.

[28] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/. HEVC reference software, Fraunhofer Institute, accessed on 29.08.2013.

[29] https://code.google.com/p/x265/. HEVC x265 encoder, Google code, accessed on 29.08.2013.

[30] T.E. Carlson, W. Heirman, L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *SC,* pp. 1–12, 2011.

[31] S. Li, J. H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, N.P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, pp.469–480, 2009.