Study of Architectures for High Quality Video Realization

By

Rishab Goel (UFID: 11436836), Anirudh Pathak (UFID: 64091067), Krishit Shah (UFID: 15531845) and Harshit Shah (UFID: 12116976)

Abstract: HD video content is being generated and consumed worldwide now. UHD video playback and streaming have also gained impetus in recent times. This advancement in Ultra HD videos is driven by improving video codec standards, hardware architecture computation capabilities and network advances. This paper discusses about the bottlenecks and challenges in generating, consuming and transmitting Ultra HD videos. Our main focus would be the different hardware architectures that can leverage the parallelization in the video codecs to achieve real time performance. The CPU and GPU architecture model, which is the most widely used model for video compression, is our center of focus. Multicore CPU-only architecture, ASIC/FPGA accelerator architectures, Cloud architectures and networking architectures are other models which would also be discussed. We focus on the complexity and performance of the proposed algorithms for different architectures or approaches and evaluate their significance.

1. Introduction

Videos have come into account for the large portions of the content over the Internet. You will find videos everywhere on social networks, content sharing applications, multimedia hosting sites like YouTube etc. With the demand of quality of video experience getting higher and swiftly changing, the multimedia content are no longer downloaded and stored on the devices. In recent years, there is a general dip in the sales of DVDs , Blu-Ray or Downloads of multimedia content for entertainment purposes like movies, television series, etc. This decline is mainly attributed to growth of multimedia proprietary content providers sites like Netflix, HULU, Vudu and Amazon Prime. We human beings do not tend to compromise on quality a lot even when adopting new trends and technology. These content providers made the concept of video on demand popular among masses at acceptable quality of experience and service charges. Video on Demand allowed the people to watch their favourite shows whenever they wanted to, instead of scheduling their day according to the broadcast time. **[3]** The Cisco Visual Networking Index predicted that more than two thirds of the consumer internet traffic would be from video by 2017. In some surveys, it has been noted **[24]** that at peak hours, Netflix accounted for about 30 percent of the total internet traffic in USA!

This high consumption of video content could be attributed to the availability of affordable, easy and smooth video-streaming solutions. The slew of such video streaming solutions is driven by high bandwidth provided by Internet Service Providers, increased computing power of devices like PCs and growth of smartphones in the recent years. Innovative video compression techniques and efficient video delivery mechanisms utilize the available bandwidth, computing power and outreach of devices, leading to the dominance of video content in the internet traffic. The innovations in video compression and delivery mechanisms have made the video streaming and video on demand possible in today's world possible. Before reaching the user for consumption, the video stream goes through multiple phases. These phases **[24]** could be enlisted as *Transcoding*, *File Transfer*, *Content Delivery* and *Video decoding & Playback*. The parallelization techniques and computing power available are used in each of these phases to provide affordable good quality of experience of video streaming.

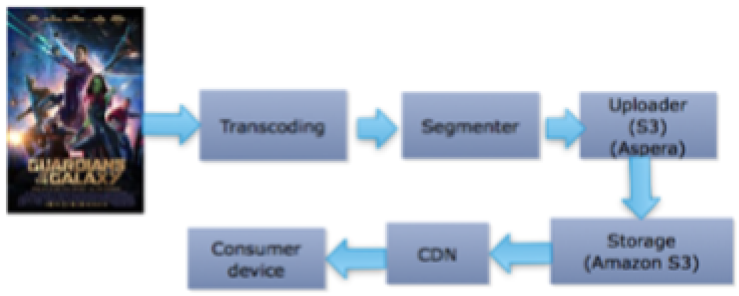


Fig 1. **Video Streaming Steps [3]**

Transcoding is a step in which raw video content is encoded into a video codec standard like H.264, HEVC and VP9 to a lower file size for transmission. Transcoding is the foremost step to provide high quality video streaming as the efficiency of codec standard and encoding algorithm will determine whether it is even possible to transfer the video stream over network. These encoded and compressed files need to be stored efficiently to cloud and data servers. This phase of a video streaming system is called file transfer. File storage services like Amazon S3 have the capability to break the files into chunks, upload it parallelly and reconstruct the files losslessly from these uploaded segments. The video content is delivered to client devices upon request efficiently using content delivery networks. They differ from the standard network servers in providing cached data and faster access by maintaining redundant copies of content at different global locations. An essential part of the content delivery network is the media streaming protocols like RTP (Real-time Transport Protocol), RTSP (Real Time Streaming Protocol) and HLS (HTTP Live Streaming) which would allow the client to change bit-rate automatically or manually based upon network traffic and contention. Video Decoding happens at the client device by the very same standards which were used initially to encode the content, such as H.264, HEVC , VP9. The device and applications available should have an efficient parallel implementation of decoding standard to deliver real-time playback of video content on the respective device. It is clearly visible that video compression is the most critical part at the both ends of the video streaming system.

The video compression has always been a field of continuous development in terms of subjective and objective quality, storage space requirement, system design for real-time performance and bandwidth requirement. High Definition Content became prevalent less than a decade back with the widespread adoption of H.264/AVC codec standard. HEVC and VP9 claim to support the encoding of even 8K content. They claim to have the same video quality experience or better as compared to H.264 for same resolutions with almost half the bandwidth. **[3]** suggests that VP9 could become very popular due to its licence free implementation and being backed by Google and being the major codec for YouTube videos. Video compression is basically encoding of the video content at the source for transmission and decoding of the video content for playback at two ends of the video streaming chain. Video compression standard have evolved from MPEG1, MPEG2, MPEG4, H.264, VP8 and now HEVC and VP9. With the evolution of each codec, the quality and compression ratio for video content has improved. The complexity, resolution and size of video input raw streams for such encoding systems have increased along with the demand for high quality real time video content consumption on varied devices. Video compression in turn, has become a more compute intensive operation over time. To handle these compute intensive tasks, many parallelization strategies are part of the defined video codec standard, which are exploited to speed up the compression. Multiple compute intensive devices like multi-core CPUs, GPUs, FPGAs, and ASICs are used for different components and different usages of the compression standard. The encoding is ten times more complex than decoding, and parallelization strategy used in both encoding and decoding are different. Thus,the encoding is normally targeted for more compute intensive devices than a decoder.

In this paper we focus on the architectures and video compression standards that have the capability to produce and consume high quality video content. In the next section, we give a basic background of high quality video resolution, compression standards and architectures we would be focussing on. In third section, we discuss about the parallelization tools described in the video standards that can be leveraged to gain enhanced performance on different architectures. In section 4, we focus upon how the encoder and decoder are implemented on multi-core CPU architectures. In section 5, we focus on host + accelerator model in which host is multi-core CPU and accelerator is a GPU or FPGA/ASIC. We focus upon which components needs to be target for acceleration in the HEVC codec to get the most speed up. We discuss some complete encoder/decoder implementations for the host + accelerator model. In section 6, important partition techniques and their performance comparison is discussed for cloud computing architectures. The networking aspects of the high quality video streaming are touched upon in section 7. We conclude the paper with some recent developments in this field and the the summary of our survey.

2. Background

4K resolution, also called **4K**, refers to a horizontal resolution of the order of 4,000 pixels and vertical resolution on the order of 2,000 pixels. Previous generations of video resolutions were described by the vertical resolution (e.g., 1080p refers to a signal with 1080 vertical lines). Had the naming convention for SD and HD been used, 4K video might instead be referred to as "2160p". Several 4K resolutions exist in the fields of digital television and digital cinematography. In the movie projection industry, Digital Cinema Initiatives (DCI) is the dominant 4K standard. In television and consumer media, 4K UHD (Ultra High Definition) or UHD-1 is the dominant 4K standard. The increase in resolution presented by 4K is tremendous, as characterized in this diagram **[5]**. 4K video delivers great resolution and beautiful experience. But like most new technologies, it comes with challenges to overcome as well. Enormous bandwidth (10 Gbit/s) is required to deliver 4K video. This places huge demands on infrastructure. To be successful in real world applications, 4K distribution systems must be designed and engineered to address challenges such as: mismatched resolutions, new frame rate considerations, signal integrity issues, new cable length restrictions and source/display compatibility. By 2015, 4K television market share had increased greatly as prices fell dramatically during 2014 and 2015. By 2025, more than half of US households are expected to have a 4K-capable TV, which would be a much faster adoption rate than that of FullHD (1080p).

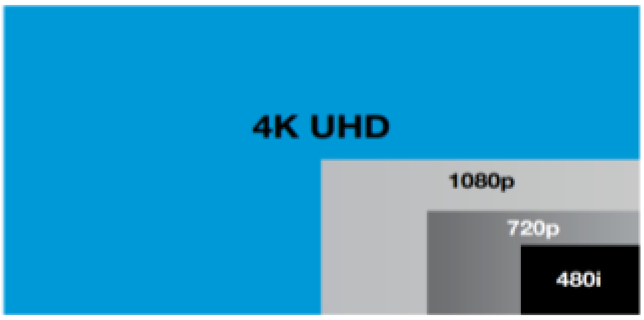


Fig 2. **4K resolution [5]**

HD resolution is one that is substantially higher than Standard definition. It transmits in 3 formats: 1080p, 1080i and 720p. When transmitted at two megapixels per frame, HDTV provides about five times as many pixels as SD (standard-definition television).
Given that unprecedented bandwidth requirements for 4K streaming, it is important to have efficient encoding techniques. There are 3 compression techniques that are used most widely for 4K compression: H.264/MPEG-AVC, H.265/MPEG-HEVC and VP9 Encoders.
H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) is a block-oriented motion-compensation-based video compression standard. As of 2014 it is one of the most commonly used formats for the recording, compression, and distribution of video content. The intent of the H.264/AVC project was to create a standard capable of providing good video quality at lower bit rates than previous standards (i.e., half or less the bitrate of MPEG-2, H.263), without increasing the complexity of design so much that it would be impractical or excessively expensive to implement.
H.265 or HEVC (High Efficiency Video Encoding), the successor of H.264, offers about double the data compression ratio at the same level of video quality, or substantially improved video quality at the same bit rate. HEVC has many design elements similar to H.264 such as Quadtree based Block Partitioning and intra picture prediction**[9]**. On the encoding side, HEVC has 35 modes vs H.264 having 9, meaning good mode heuristics need to be employed for better performance. HEVC employs fixed point matrix multiplication rather than sequences of shift and add operations as used in H.264. HEVC employs CABAC (Context-adaptive binary arithmetic coding) as the only coding method, in comparison with H.264 that additionally employs CAVLC (Context-adaptive variable-length coding). The overall complexity is reduced in HEVC by further optimizations done in CABAC. HEVC also employs three concepts that lead to some degree of high level parallelism which are slices, tiles and wavefronts. While an HEVC decoder would not be much expensive to implement as compared to an H.264 decoder, an HEVC encoder that exploits all its capabilities would be highly complex as compared to its H.264 counterpart.
VP9 is an open and royalty free video coding format developed by Google. It is a successor to VP8 and competes with MPEG's High Efficiency Video Coding (HEVC/H.265). At first, VP9 was mainly used on Google's popular video platform YouTube. The emergence of the Alliance for Open Media and it supporting the ongoing development of the successor AV1 led to growing interest in the format.
Dan Grois et al **[6]** compared these 3 most popular schemes for compression. According to their experimental results, which were obtained for a whole test set of video sequences, MPEG-HEVC provides significant average bit-rate savings of 43.3% and 39.3% relative to VP9 and MPEG-AVC, respectively. The VP9 encoder produces an average bit-rate overhead of 8.4% at the same objective quality, when compared to an open MPEG AVC encoder implementation – the x264 encoder. On the other hand, the typical encoding times of the VP9 encoder are more than 100 times higher than those measured for the x264 encoder. When compared to the full-fledged HEVC reference software encoder implementation, the VP9 encoding times are lower by a factor of 7.35, on average.
To implement these complex compression techniques for real-time encoding and playback, we require highly parallelized architectures like GPUs, multi-core CPUs, FPGAs, ASICs and Cloud Architectures. GPU, FPGA and ASIC are accelerators that, combined with a host CPU, can be used to implement these compression techniques. They come under the umbrella of 'host + accelerator programming' model. Another model used for the implementations is the 'multi-core CPU model' in which the tasks are distributed across various threads on single or multiple cores and things are

done parallely or concurrently depending upon the number of processors. A third model used is the 'cloud computing model', which is internet based computing that provides shared computer processing resources and data to computers and other devices on demand.
A graphics processing unit (GPU), occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. Companies like Nvidia, AMD, Intel and Apple design GPUs and their tools and Nvidia being the market leader in GPU market. A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". An application-specific integrated circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. FPGA is extensively used for emulation of the ASIC circuits for functionality test before their physical design is done.

3. Parallelization tools

The parallelization of any algorithm is a complex problem and scope of parallelism is determined by redundancy and independence of operations. **Task parallelism** is the simultaneous execution on multiple units of many different functions across same or different datasets. **Data parallelism** (SIMD- Single Instruction Multiple data) is the simultaneous execution on multiple cores of the same function across the elements of a dataset. The parallelism could be coarser when the each individual partition is complex, while it would be finer when an individual partition have less complexity. The parallel efficiency of any application is governed by the Amdahl's Law:

SpeedUp = 1 / ((1-p) + p/n) ;where p is the component that is parallelized

H.264 is a highly popular and effective video compression format, but as many other compression standards, it also suffers with cost of complexity. The cost of complexity could be termed as the computation cost required to achieve desired performance in speed for a given quality standard. These codecs do offer high compression efficiency with quality acceptable for human consumption. But to achieve the speed and quality standard desired, the video codec standard needs to provide parallelization strategies, tools and structures to be exploited. These strategies discussed in the standard would serve as the basis of highly parallelized hardware and software architectures.
H.264 discussed four main strategies:**[17]** frame level parallelism, GOP level parallelism, slice level parallelism and macroblock level parallelism, which could also be partially extended to HEVC. Frame level Parallelism basically exploits the strategy that different frames can be processed simultaneously on varied multicore platforms with minimal control code overload and with simplicity. The parallelization achieved using frame level parallelism is dependent majorly on length of the motion vectors, so fast motion videos have limited scope of parallelism. Workload imbalance among processing cores may occur due to varied complexity of each frame. It is already been employed in many H.264 decoders and encoders implementations, with close eye on satisfying the motion compensation dependencies. Frame parallelism may help in frame rate increase, but does not reduce frame latency. GOP(Group of Pictures) level parallelism is an extension to frame level parallelism in which instead of a single frame a complete GOP structure can be scheduled on a processing node. This avoids problem of workload imbalance for fixed GOP size and motion estimation dependency structure for closed GOP structures. Though GOP level parallelism is not effective for the decoder size. This is because it would cause high latency in playback . GOP level

Parallelism becomes effective for encoder implementations on server grade machines or cloud architectures for HEVC as discussed later in section 6. Slice level Parallelism is another form of parallelism that could be exploited. Slice could be defined a continuous raster scan partition in a frame which is normally independent any other slice and adds robustness to the bitstream. Slice level parallelism leads to high compression inefficiency. This issue may lead the encoder to avoid slice partitioning in a single frame or only two to three slices, thus limiting the scope of parallelization. Slices are slightly attractive tools for distributed systems for fine grained parallelism because of their lack of inter-dependence. All intra encoding and decoding implementations could combine slice and frame parallelism to get good amount of speed up. Though, HEVC also introduces the concept of dependent slices which introduces extra check for parallelization. Deblocking filter in H.264 can be applied across slices, which goes against slice based parallelism. Macro block Parallelism can be scaled up to many cores as independent macroblocks inside a frame and across frames can be reconstructed in parallel. For macroblock parallelization in frame, wavefronts are required and across frames there should be no motion compensation dependencies. It needs to be noted that macroblock parallelism can only be applied when entropy decoding is decoupled as it can be parallelized across frames only. This constraint reduces the frame latency at the reconstruction stage not only at the entropy decoding stage.
HEVC introduces parallelization tools **[17]** like tiles and wavefronts. Tiles are basically a rectangular group of CTBs separated by vertical and horizontal independent boundaries. In **Fig. 3**, we can see a picture divided into nine Tiles and the scanning order of the CTBs **[17].** The number of tiles and their boundary location can be specified for the entire sequence or can be changed from picture to picture. There is no parse and prediction dependencies across tile boundaries similar to a slice boundary, although loop filters (deblocking and SAO) can be applied across the tile boundaries. Tiles don't require communication across processors for decoding and reconstruction if the loop filters are 'non-crossing'. The 'non-crossing' sometimes result in visual artifacts, so 'crossing' loop filters also exist. The choice of 'crossing' and 'non-crossing' is a tradeoff between performance and quality, a choice that depends on the designer and the applications. Tiles don't have raster scan order, and it complicates the processing on a single core. It is to be noted that slices and tiles are related with a constraint that all CTBs in a tile should be a part of a single slice and all CTBs in a slice should be a part of the same tile. Tiles have better coding efficiency compared to slices because picture partition samples for tiles have a higher correlation and lesser header overhead than slices. Tiles, similar to slices suffer from a higher rate-distortion loss due to breaking of dependences along the tile boundaries and resetting of CABAC probabilities at each partition. Wavefronts Parallel Processing(WPP) is the method of processing a frame such that each CTB row is considered a separate partition and can be processed independently. For processing each CTB top, top-left and left CTB should have been processed, which leads to a scenario in **Fig 4[12]** in which, every new row is 2 CTB behind its previous row. This two CTB limitations is due to the two level dependency between neighbouring CUs for intra encoding:

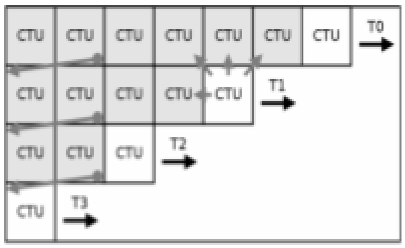


Fig 4. **Principle of wavefronts and CTU Dependency [12]**

1) CTU level dependency: Each CTU waits until its left and top-right neighbour CTU finish reconstruction. Thus the current CTU row is always two-CTU latent than its adjacent upper row and the latency persists until the upper-row finishes its rightmost CTU.
2) CU level dependency:- For each CU, the encoder decides the best mode by performing intra prediction with 35 modes and each CUs cost, then splits into four sub-CU's and repeats the

process until it reaches the maximum allowed depth. The calculation is proposed from top to bottom in quad-tree traverse manner. The encoder decides best partition by checking if the cost of its parent CU is smaller than the cost sum of its children's. This process is sequential and highly time consuming. This introduces parallelization inefficiencies in terms of time difference in spawning and seizing of the threads. Though in steady state, all threads workload is evenly distributed. CABAC probabilities of current CTB row are only available after processing the 2nd CTB of the row above. There is no change in raster scan, so the rate-distortion loss of a WPP unit is small compared to a non-parallel bitstream.A WPP bitstream can be a loosely transcoded with only an entropy conversion to/from a non parallel bitstream. In order to simplify the implementation, the HEVC standard does not allow to use Tiles and WPP simultaneously in the same compressed video sequence. WPP is generally well suited for the parallelization of the encoder and decoder because it allows to have a high number of picture partitions with low compression losses. Additionally it does not introduce artifacts at partition boundaries as is the case for slices and Tiles **[17]**. WPP can also be used for low-delay applications, specially those requiring subpicture delay (also called ultra low-delay). In conversational applications, for example, Tiles in combination with a tracking algorithm, can be used to dynamically adjust the size and error protection of the ROIs

The encoder has multiple components which utilizes redundancy, correlation and perceptual quality optimization in the raw image data to achieve compression. Except task parallelism and data parallelism at frame or inter-frame level, we have to also look component level parallelism and finer parallelism. SIMD and Hardware Acceleration by a GPU, FPGA or ASIC are few methods/ devices that are used to speed up these components. According to the **Fig 5[11]** , the motion compensation module occupy more than 50 %, while Intra and Transform & Quantization are close to 5% and Entropy encoding and loop filters less than 1%. The rest of 20 % include search range, RD cost, SAD/SATD (Sum of Absolute Differences/ Sum of Transform Differences), motion compensation modules and other modules. For motion estimation, a best MV (Motion Vector) is selected for each PU (Prediction Unit) with quarter pel accuracy. In order to get the motion vector, BMA (Block Matching Algorithm) is performed. At first, IME (Integer Motion Estimation) is conducted with 8 tap half-pel search around integer pel and 8 tap filter of quarter pel around half pel. The motion vector that makes the cost minimum is selected as the best MV of the PU. The cost for prediction parameter decision is specified as follow:

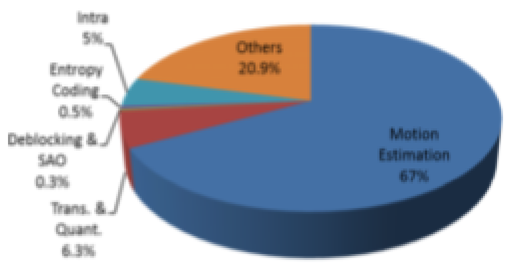


Fig 5. **Encoding Time Distribution of the HEVC HM 10.0 encoder [11]**

Jpred =SAD/SATD + λpred*Bpred; where SAD and SATD are used IME (Integral Motion Estimation) and SATD (Hadamard Transformed SAD) is used in FME (Fractional Motion Estimation) respectively. Bpred specifies the bit cost which is determined by the difference between current MV and MVP (Motion Vector Predictor). There is no data dependency between neighboring PUs in SAD and SATD. The calculation of SAD and SATD is independent from neighboring PUs, and along with search range optimization, it provides a large potential to make it highly parallel.

4. Multi-core architecture

With the evolution of CPU architecture, the multi-core CPU has become widespread. The applications and standards have been defined keeping this in mind. In the paper **[12]**, the authors have proposed a multi-threaded encoder implementation which is based on HEVC reference test model HM11, which makes full use of the Wavefront Parallel Processing (WPP) mechanism running on regular consumer hardware. The authors first implemented a component called BitstreamVerifier in order to make sure that their prototype doesn't contain any threading-related problems like data
races and deadlocks to compare the generated output data to a reference bitstream created in advance with the unmodified HM11 software during runtime.

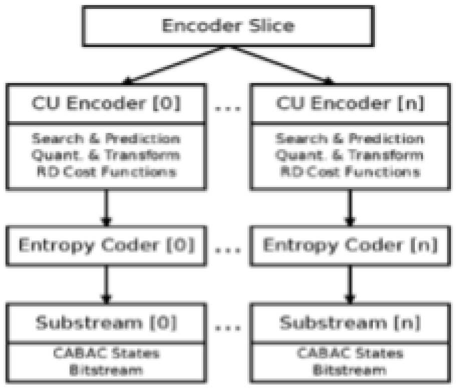


Fig 6. **Parallel Architecture [12]**

The authors **[12]** implemented a thread pool system that can manage an arbitrary amount of worker threads in order to make their prototype scalable and flexible. The degree of parallelism is set according to the number of available processing cores, however any other number of threads can also be used. Using mutual exclusion, the individual CTU lines are kept in an ordered task-queue and the worker threads autonomously grab the one or more CTU-lines out of the list and process all their CTUs accordingly. Instead of using one temporary state to realize the probability synchronization, the individual contexts must be used which means that the state after the second CTU of line n is directly loaded into the CABAC state of line n+1. After that, line n+1 can be processed by its associated worker thread. If current CTBs = width of CTB row, then semaphore counter is width of the CTB row. If current CTBs is more than 2 ,then semaphore counter is incremented by one, else set to 0. In order to avoid synchronization completely, the authors duplicated all needed encoder modules for every CTU line in order to make them work totally independent from one another. To generate the needed subpixels for the respective fractional MVs, an Interpolation Filter (IF) is utilized. The resultant subpixel-blocks are stored in many temporary buffers and serve as a basis for the cost estimation. These buffers must be duplicated in a multithreaded environment so that every thread is able to work on its own individual area of memory independently to avoid non deterministic results.
[12] The entropy coder and the CU encoder, which itself contains modules for search, prediction, quantization, transform and RD cost computation as in **Fig 6**, are the subsystems that need to be copied. For this purpose, the authors employ a *Deep-copy* mechanism means that not only the instance of a class is cloned, but all its respective member objects as well. Accordingly, new memory needs to be allocated for every class member and their data must be copied respectively. The results are totally independent new objects with no shared members, buffers, or pointers, thus cross-reference reconstruction is necessary. All needed aggregations and references between the newly cloned objects need to be set correctly to properly work for eg: the CU encoder needs to know its search module. Thus, the threads can work independently from one another and only need to be synchronized just to stay within the limitations of the WPP scheme. Also the WPP substreams do not need to be duplicated as the original HM11 software already has multiple

instances of them. Experimental results show in **Fig 7** that the fully compliant software gives speedups of up to 5.5 times on a 6-core CPU.

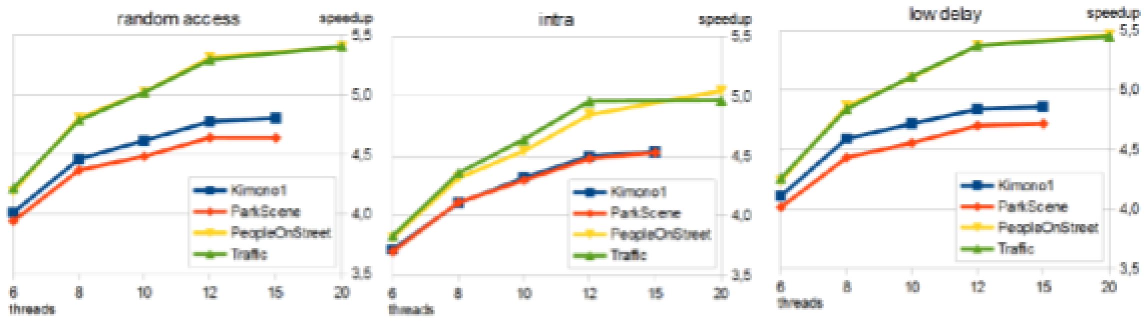


Fig 7. **Average Speedups of WPP for the Profiles (random access), (intra), and (low delay) for utilized Qps [12]**

The author **[17]** present a novel approach called Overlapped WaveFront (OWF) which achieves higher performance and efficiency than Tiles and WPP. The overlapped wavefront (OWF) suggests the wavefronts to start processing the next frame's same CTB row when current CTB ends with a constraint on the vertical length of the motion vector. The restriction ensures that the reference area of the motion vector is already encoded. Vertical motion constraint is part of the profile and level definition of the standard. Though, HEVC standard has not specified any such constraint on the motion vector. The author has presented the approach of overlapped wavefront from the perspective of the decoder.

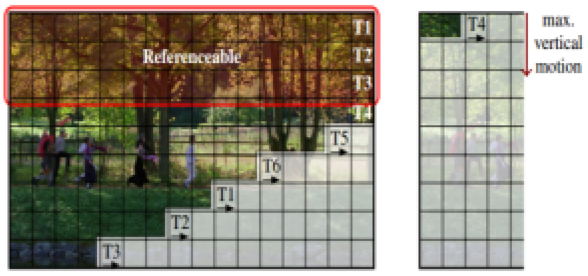


Fig 8. **overlapped frame decoding [17]**

Fig 8 illustrates overlapped frame decoding as well as the relation between the maximum motion vector length and the admissible number of CTB row decoding threads **[17]**. In the given example, there are 9 CTBs in a frame with 6 decoding threads and a vertical motion constraint to 1/4th of the picture height. Experiments in **[17]** conducted on a 12-core system running at 3.33 GHz show that the implementations for the HEVC decoder achieve average speedups, for 4k sequences, of 8.7, 9.3, and 10.7 for WPP, Tiles, and OWF, respectively. Comparing WPP to Tiles, the experiments show that the Tiles approach achieves slightly higher performance than WPP (7% higher on average over all resolutions at 12 cores). WPP has higher memory bandwidth efficiency than Tiles. Implementing a memory bandwidth optimized Tiles decoder can reduce the memory bandwidth requirements significantly, but is paired with high implementation complexity. The proposed OWF decoder achieves the highest performance (which is 28% higher on average than Tiles) and 1080p50 videos achieve real-time performance, but "only" 25.4 fps for 2160p.

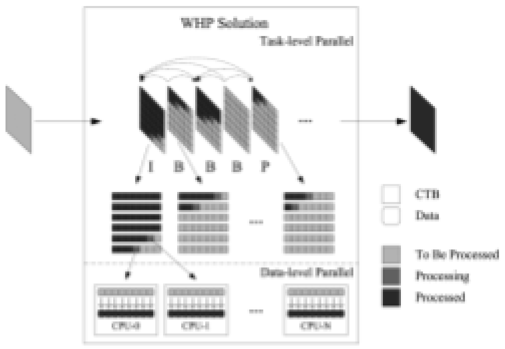


Fig 9. **Overall design of WHP solution for HEVC encoding [25]**

A novel inter-frame wavefront (IFW)**[25]** method is developed on task level by effectively decreasing the dependence of wavefront parallel processing (WPP). Additionally, in this particular paper, to prove the superiority of IFW method compared with other HEVC representative parallel methods, a coding tree block level parallelism analysis method has been presented. Moreover, to best exploit the parallelism of IFW method and achieve corresponding encoding speedup, a three-level thread management scheme is proposed in **Fig 9 [25]**. The author's proposed solution has also applied in many big video companies in China, providing HEVC video service to more than 1.3 million users everyday. To meet the performance requirements, the author employs SIMD optimizations, transpose-free integer transform, data-reuse in SSD/SAD calculation as well as replacing multiplication with add and shift in motion compensation.

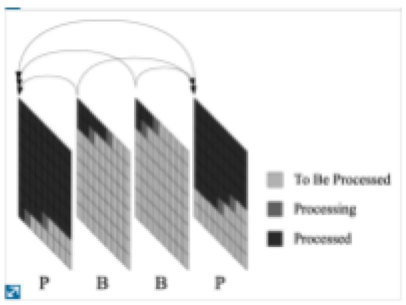


Fig 10. **wavefront achieved with IFW [25]**

A typical example**[25]** of wavefront coding order achieved with IFW method is illustrated in given **Fig 10** [25] , where four frames are encoded with eight threads in total at the same time, showing a general wavefront coding order over all the CTBs. Within each frame, CTB rows are encoded in intra-frame wavefront coding order based on wavefront parallel method. To maintain the dependence of the vertical and horizontal deblocking filters (DFs) in IFW method, the improved encoding process of each CTB row is applied in three steps: 1) Rate Distortion Optimization (RDO) process of the CTB row; 2) vertical edge DF; and 3) horizontal edge DF. The DF of the bottom border of current CTB row is delayed until the RDO process of next CTB row is finished. In this design, DF is applied in advance right after the RDO process of each CTB row, so that the reconstructed samples can be ready for reference as early as possible. To ensure that the Motion vector utilized in the RDO process of each inter-frame within the safe range, the following condition should be satisfied**[13]**:

*MVY 4 (sample rows of a CTB) L_IFW 28(Safe Range 7*4)*

where MVY is the vertical component of Motion Vectors in quarter pel units, L_IFW is a positive integer parameter indicating the L_IFWCTB rows being referenced, and the safe range is 7∗4=28 quarter-pixel rows. With the precise MV range given in above eq, the inter-dependence of IFW method can be expressed as **[25]** : $DEP_{IFW}, inter(C_{i,j,k})=\{C_{i\ ,j\ ,k} \ | i \quad REF(i), 0 \ j \quad < j+L\_IFW, 0 \ k \quad < W\}$;where REF(i) is the reference frame set of frame i , and W is the width of a frame measured by CTB. When CTB_HEIGHT is 64, even L_IFW is set to 1, the maximum MVY can be 228 in quarter pixel (57 samples). For videos with resolution of 1080p or smaller, this MV range restriction only causes negligible compression efficiency loss (the worst case of the standard sequences is only 0.1% BD-rate loss) compared with the standard MV range setting of HEVC, while the size of dependent CTB set is greatly refined. With the CTB-level dependence description of WPP, OWF, and IFW methods obtained, the parallelism of WPP, OWF, and IFW is:

$$Parallelism_{IFW} \approx 2*Parallelism_{OWF} \approx 4*Parallelism_{WPP}.$$

To give a more intuitive evaluation of the above equation, the parallelism of different methods using are listed in the below **Fig 11 [25].**

On data level, the authors have well analyzed the algorithms of the enhanced tools in HEVC and optimized them with novel SIMD algorithms over ×86 platform, reducing the encoding time of a CTB on each CPU. **On task level**, based on the theoretical deduction, detailed analysis and efficient design, the authors propose the IFW parallel method which is proved and experimentally validated to be of higher parallelism than other methods, and further improve the encoding speed using the computational resources of multicore platform. The WHP Solution indeed seems a promising technology for large-scale HEVC video applications as it brings up to 88.17 times speedup on 1080p sequences, 65.55 times speedup on 720p sequences, and 57.65 times speedup on WVGA sequences compared with the serial implementation.

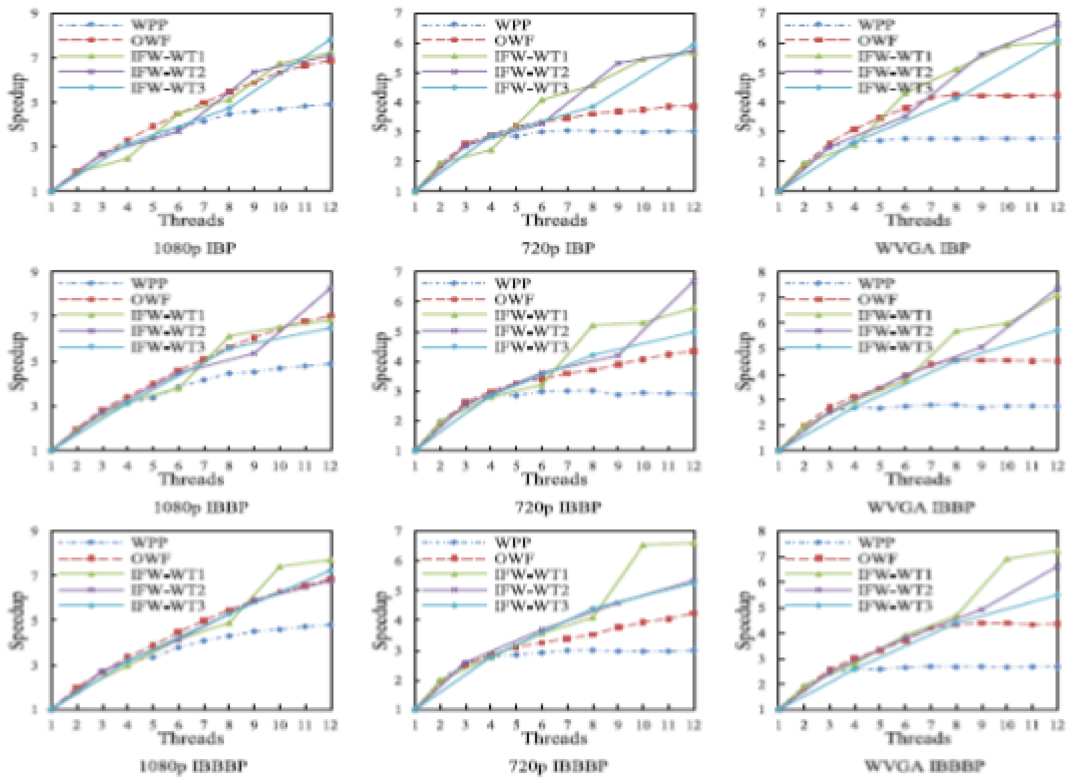


Fig 11. **Encoding speedup comparison of WPP, OWF, and IFW parallel methods based on WHPTB [25]**

Motion Estimation and Motion Compensation are followed by Intra prediction/encoding in terms of cycle occupancy of encoder/decoder. Though the intra encoding percentage occupancy is fairly less it seems some streams do have high spatial correlation like in minimum loss RAW video storage, where the intra encoding occupancy may be substantial. Moreover, when dominant module for inter-frame prediction and motion compensation are optimized, the percentage occupancy of intra encoding also shoots up. The paper **[10]** provides full analysis of parallelism for HEVC intra coding on 8 cores: Intel(R) Xeon(R) CPU E5-2670@2.6 GHz, with 24GB memory, 64 bit Windows 7, based on spatial parallelism, and then an efficient parallelism encoding strategy is proposed and implemented on x265. The intra-prediction extends to at most 36 prediction modes of which 3 are directional, 1 DC mode, 1 planar mode and 1 special mode used for chroma. As discussed before, intra encoding has CTU and CU level of dependency.

In the parallel processing stage **[10]**, multiple threads are launched to perform intra-prediction with each thread processing one CTU (coding tree unit) row and this intra information is stored when CTU is over. In the second stage (entropy coding stage) one additional thread is used to encode all CTU's within the whole picture in raster scan order. in the first stage, called parallel processing stage, multiple threads are launched to perform intra prediction with each thread processing one CTU row and proceeding under the same dependency constraint. Once one CTU is over, the intra information is stored first. In the second stage, called entropy coding stage, one additional thread is used to encode all CTUs within the whole picture in raster scan order. The WPP based two stage scheme for intra encoding provides three benefits: maximizes acceleration ratio, minimizing performance loss (independent thread encodes all CTUs in the second stage), and increasing speed up gain (entropy encoding of current CTU can be done if preceding CTUs encoded).

The algorithm**[10]** is implemented on x265, an open source HEVC encoder. The experiment also studies relationship between speedup ratios and thread numbers. The performance comparison between x265 and HM9.0 for intra-only coding is shown in **Fig 12 [10]** it can be concluded that the speedup gain is 80% when two threads are used and increases with increase in number of threads and reaches maximum value of 398% at 7 threads.The paper states that the scheme could achieve on an average 502% speedup gains for different video sequences at no performance loss. It is to be noted that such high gains are extracted because of dedicated intra only encoding.

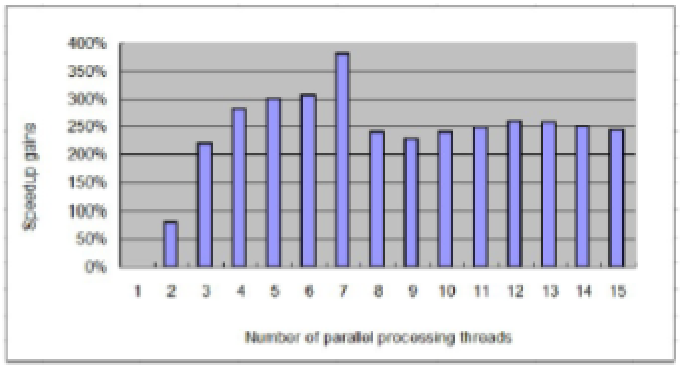


Fig 12. **speed up gain with thread numbers [10]**

5. Host + Accelerator architectures

In Host + Accelerator Model, the combination GPU as accelerator and CPU as Host is most widely used due to its widespread low cost availability and comparatively low implementation cycle. It is to be noted that motion estimation is the major module in encoder which gives you the most compression and occupies the most percentage of encoder cycles. In the decoder, motion estimation is not present so motion compensation modules dominate the cycles(> 35%) in the decoder. Motion Estimation and its search range optimization are basically the modules where the cycles are consumed in a HEVC encoder. Thus, most of the host + accelerator models focus on optimization of the Motion Estimation and Search Range keeping in mind the fact that the parallel efficiency of every application is mainly driven by Amdahl's Law.

5A. GPU

The GPU algorithms for an encoder are mainly targeted towards motion estimation only. HEVC encoding is a more severe challenge than H.264 because of complicated CU(coding unit), PU (Prediction Unit) and TU (Transform Unit) partitions. Wang *et. al* **[23]** paper first proposed the variable block size motion estimation on Multicore CPU and GPU architecture. The three search candidates are searched on the GPU with the lowest SADs (Sum of Absolute Differences) on multiple Streaming Processors (SP) and are returned to the CPU for the final decision. The compression efficiency loss is about 0.73 dB due to partial exploitation of GPU computational resources at CU and PU partition and also due to the inability to convey useful info from GPU for CPU optimization. Wei Xai *et.al* [20] proposes a new algorithm which targets motion estimation optimization on GPU and fast mode decision algorithm on the CPU. The decision to use the combination of CU and PU partition is a complicated process. For a 64×64 CTU, there is a total of $(2^4+1)^4+1$ = 83,522 possible CU partitions as CU size varies from 64x64 to 8x8. Every CU has three to seven possible PU partitions, thus it is impossible to evaluate all possible CU and PU combinations using RDO. They suggest that if each 8x8 CU is partitioned into 3 PU each, then 64x64 CU will have 24 PU partition and this would be the most effective partitioning. They model

this behaviour on the 3072 SPs in Nvidia GeForce GTX 690 with one thread in CPU controlling the data exchange for the GPU while rest do other HEVC encoding tasks. ME (Motion Estimation) search algorithm is implemented on GPU which returns the best MV (Motion Vector) and lowest MC (Motion Compensation) cost of every possible PU to CPU memory. These values are read by CPU for to derive the PU and CU cost data to make the final decision.
The GPU used has 8 Streaming Multiprocessors (SMP) and each one has 192 SPs on it with the instruction cache shared. The shared memory (B_M) and L1 memory of 48kB and 16 kB could be configured on each SMX respectively. According to the GPU hardware constraints, the following conditions have to be satisfied in the algorithm design[20]:
$K = Gcd(SP_n, S_n)$; where S_n is the no. of SPs to parallelly process selected PU area
$G \times K = SP_n$; where G is no. of searching points in parallel SP
$B_r + B \leq B_M$; where B_r is NxN buffer for reference in the shared memory, B is MxM buffer for current block in the shared memory .

Estimated motion vector MV_e and Initial MV is set to zero initially with search range R (-64,64) and number points in diamond search range. The search algorithm has three stages: presearch, small block size and large block size. In the presearch algorithm, we find the best integer motion vector over the diamond search range **Fig 13** (where D denotes the search point distance to the pattern center). Predicted Motion Vector (MV_p) of the 64x64. Load a CTU of original pixels into B and load reference pixels specified by MV_p and R into B_r

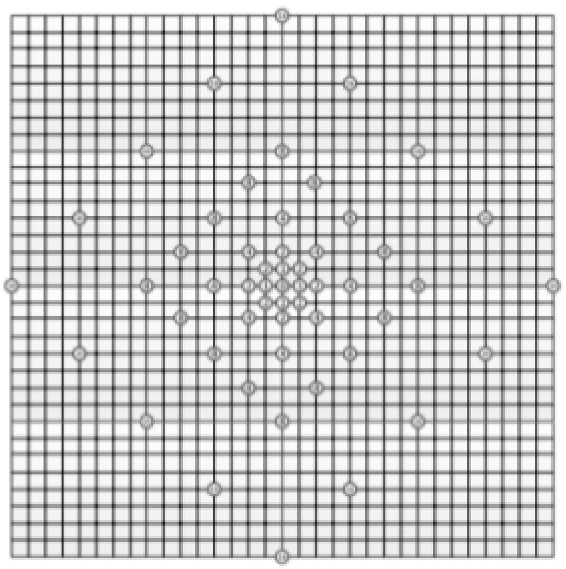


Fig 13. **Partial diamond search range for presearch (D<=16) [23]**

Follow the steps ahead recursively for all PUs until MV_0 initial MV) is not equal to MV_{pre}:
For every Gth points P=85 in the search pattern:
Calculate SADs of the PU at G=3 searching points by SP_n SPs in parallel and store SADs of 4 × 4 blocks in the shared memory. Sum SADs of 4×4 blocks in parallel to get the SADs of the PU at G searching points Update MV_{pre} by comparing SADs.
The large block search is done for 64x64 to 16x16 PUs with search range (-32,32) for the proposed points in **Fig 14** with different parameters using pre search algorithm. For the small block, search is done for PU partition less than 16x16. The 4x4 SADs are calculated first and summed up for higher PUs using required 4x4 SAD cost. The fractional motion search **[20]** is then applied at half pel search followed by quarter pel search. The search pattern is simplified from 7x7 to 5x5 pattern. The cost is measured as $J = SSD + {}_{MV} \times R(MV)$ where ${}_{MV}$ is the lagrangian factor. Find fractional motion vector MV_{min} of all PUs for a given reference and load original pixels to B and reference pixels to B_r. For all PUs in the PU set start MV_{min} as integer MV and search the 25 points in nine stages. Apply the following steps for each points calculate SSDs of 4 × 4 blocks at G search points in parallel summing SSDs of 4 × 4 blocks to the size of PU and add bits cost to get the MC cost of the PU.

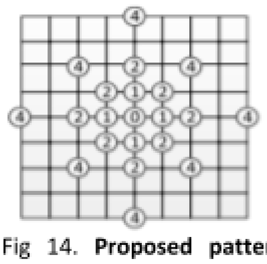


Fig 14. **Proposed pattern with 21 search points in the large block search [20]**

The best MV and lowest MV cost of every possible PUs are given to the CPUs.

Algorithm A) **[20]** For a CU determine if the CU is split or not and return its RD cost C_{CU} ;

1) If CU is the element of $G_{non-split}$,then the current CU is not split , perform RDO to encode the current CU and get its RD cost $C_{non-split}$ and $C_{CU} = C_{non-split}$.
2) If CU is the element of G_{split} , then the current CU is split into four child CUs and for each four child CUs call recursively Algorithm A for each child CU to determine its structure and get its RD cost. Sum four child CUs RD cost to get the cost C_{split}. Set $C_{CU} = C_{split}$.
3) If CU is the element of G_{tbd} ,then encode CU as $G_{non-split}$ and get CU cost $C_{non-split}$. Encode CU as G_{split} and get CU cost C_{split}. If Csplit < Cnon−split, then the current CU is split into four child CUs and $C_{CU} = C_{split}$, else the current CU is not split and $C_{CU} = C_{non-split}$.

When the CU cost is determined, each CU's PU partitions needs to be calculated but since PU MC cost is sent by GPU no ME needs to be performed but PU MC cost needs to be refined.

$J_{PU} = SSD_G + \lambda_{MV} \times R(MV_G - MV_P)$;where MV_g is motion vector sent from GPU a and MV_p predicted MV for neighbouring PUs. Some other fast mode decision are made for CU encoding for faster search range.

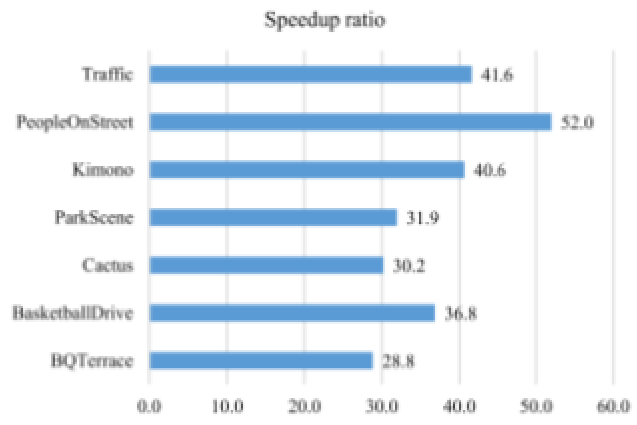


Fig 15. **SpeedUp ratios for streams in Class A and Class B [20]**

The proposed encoder implementation **[20]** is based on HM 10.0 using (IPPP) scheme and experiments were carried out on the Windows 8.1 64-bit OS platform with two Intel Xeon E5-2687W at 3.10-GHz processors, each with eight cores (hyper-threading disabled) with NVIDIA GeForce GTX 690. The curves of the proposed scheme and those of the HM anchor with WPP are nearly coincident with each other except Johnny 1920x1080p stream which has the loss due to WPP (7.5% of 7.8%). **Fig 15** describes the SpeedUp ratios for streams in Class A and Class B **[20]**. For Class A and Class B sequences, with 15 cores for WPP parallel encoding, the average encoding times are only 2.1% and 3.0% on average, which means an average speedup ratio of 48 times and 33 times is achieved for Class A sequences and for Class B sequences respectively.

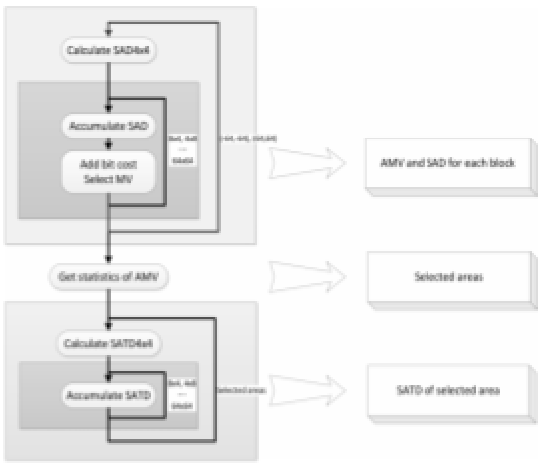
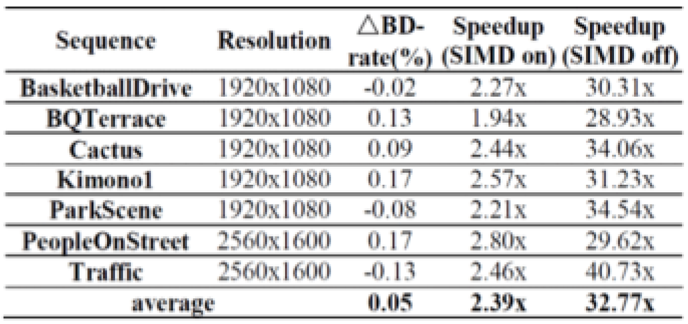


Fig 16. **Workflow Motion Vector Prediction on GPU [22]**

The implementation discussed in paper **[20]** doesn't seem to affect RD loss, but test sequences and speed up obtained seemed to be limited as there are no IPBB sequence in the test suite**.** It gives a very detailed description abouts its GPU architecture and mapping, but the testbed used for evaluation needs to more descriptive.

The motion estimation is the module that needs to be focussed on to get speed for a HEVC encoder or decoder implementation. The author's **[22]** proposed motion estimation approach divided motion estimation into two

parts: One on GPU using estimated MVP and the other on CPU using accurate MVP. On GPU, a full search is conducted to get AMVs for each block from 4x8, 8x4 to 64x64. After that, these AMVs are used to predict the motion vector areas where FME will be performed. In these predicted areas SATDs are calculated. The workflow as described in **Fig 16** shows clearly SAD happens first from top to bottom (4x4 , 8x4, 4x8 and so on till 64x64) and each higher SAD size blocks is calculated using the smaller SAD block calculation. The irregular SAD block calculation is also handled in a similar fashion. Similar process is applied for
calculating for SATD cost calculation, but it has a narrower search range. The sum of SAD and bit cost of a MV is calculated on the GPU by using the difference between the current MV and zero MV and MV having the minimum cost is the selected AMV after searching in the range of (-64,-64) to (64,64). It was observed that for block size less than 8x8 AMVS were scattered. AMVs were more accurate for block size higher than 8x8 as SAD had large proportion of total cost calculation compared to bit cost for large sizes. They divided the search range of (-64,-64) to (64,64) into squares of 8x8 and each AMV is tested for which square it is in. The higher weight is given to larger blocks, while lower weight for smaller block sizes which results in the square with higher value being selected FME prediction needs to be done in that area more times. They select only 5 areas of SATD calculation which leads search candidates reduction from 16641 to 320 and fractional points SATD is calculated in the selected area to support CPU. Work items that share data are tried to be placed in the same work group to ensure better performance. CPU sets the AMV as center and the search range as 8 and CPU does FME calculation for vectors for the points not covered by the GPU. CPU gets J_{pred} by adding up the SATD and bit cost and the motion vector with lowest J_{pred} after half-pel and quarter pel search is selected. The work of CPU in motion estimation is simplified to refine the motion vector with the support from GPU. CPU is concentrated on dealing with the logic to choose a best MV from a small range of candidates with the prepared AMV, SAD/SATD and real MVP. To make CPU and GPU cooperate properly, hide overlapped execution time and minimize communication time by overlapping tasks between CPU and GPU, a synchronization mechanism is constructed. The mechanism allocates a queue of FIFO buffers, where the CPU writes to the tail and GPU reads from the tail.



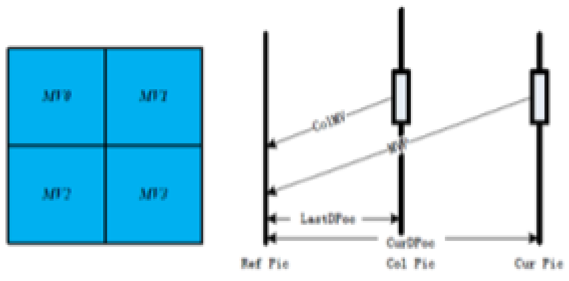
Sequence	Resolution	△BD-rate(%)	Speedup (SIMD on)	Speedup (SIMD off)
BasketballDrive	1920x1080	-0.02	2.27x	30.31x
BQTerrace	1920x1080	0.13	1.94x	28.93x
Cactus	1920x1080	0.09	2.44x	34.06x
Kimono1	1920x1080	0.17	2.57x	31.23x
ParkScene	1920x1080	-0.08	2.21x	34.54x
PeopleOnStreet	2560x1600	0.17	2.80x	29.62x
Traffic	2560x1600	-0.13	2.46x	40.73x
average		**0.05**	**2.39x**	**32.77x**

Fig 17 **[22]**

In their work **[22]** of mapping HEVC Motion Estimation on GPU with OpenCL, x265 is used as the encoding software. The Intel Core i7–3770, at clock rate 3.7 GHz, is selected as the CPU and the AMD R9 290x, at clock rate 1 GHz, is selected as the GPU. Firstly, we evaluate this work in comparison with the original x265. in this evaluation, all the 5 sequences of resolution 1920×1080 and 2 8-bit sequences of resolution 2560×1600 in HEVC standard test sequences set are tested. Experimental result in **Fig 17** shows that the speedup achieves 2.39 times and 32.77 times in the whole x265 encoder with CPU SIMD (Single Instruction Multiple Data) on and off, respectively. At the same time, quality degradation is negligible with only 0.05% increase of BD-rate. When compared to **[23],** another effective popular implementation which for resolution >1080p has a PSNR loss of 0.7 db, the **[22]** implementation has a PSNR loss of 0.003 db.

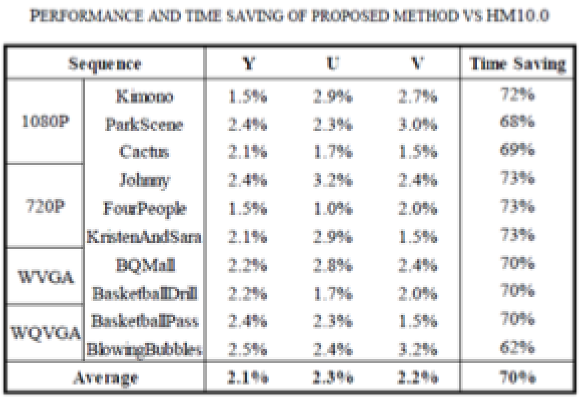
The author **[11]** proposes a flexible Coding Tree Unit (CTU) – level parallel ME method through CPU and GPU pipeline collaboration to reduce ME module's complexity in HEVC. The proposed design solves the issue of GPU resource wastage due to fixed size data structure by setting variable size parallel CTU group (CTU window) and adaptive search. In this scheme the variable size prediction units (PU) in a CTU are processed in only one CUDA (Computer Unified Device Architecture) thread block. The CUDA thread block is scheduled by GPU system and thus the number of CTU's can be configured to be of any size. The original picture is transmitted from CPU to GPU, then the parallel ME process of the first CTU window is launched. The CPU and GPU are synchronized for the MVs and SADs info of all PUs in current CTU window. GPUs process the next CTU window after synchronization After that, CPU executes mode decision and other encoder tasks, while GPU deal with CTU window i+1 simultaneously. Finally, when the last CTU window is finished by CPU, the whole reconstructed picture is transmitted to GPU for interpolation and saved as a new reference picture. Because of certain hardware constraints like fixed size data structure for specific architectures, it results in waste of GPU time to search for the best motion vector for slow moving sequences. Full search range assign initially for GPU due to regularity with 3 optimized search range of 8,16 and 32. The adaptive search range scheme is applied as suggested for CTU search: Default search range 8 is assigned for the current CTU window. If the partition depth of the co-located CTU is bigger than 0 and |MVC|> 16, then assign search range 16 for the current CTU window. If |MVC| > 32, then assign search range 32 for the current CTU window and exit the traversal.

In this method, ME of one CTU is implemented in exactly one block and the CTU window can be set to any length by organizing the blocks in the grid. Before performing motion search on GPU, the MVP is derived at CTU level from four 16 x 16 PU's in the temporal CTU. **Fig 18** shows CTU level MV derivation.



a) MVs of four 16x16 PUs in a CTU b) temporal MV

Fig 18. **CTU level MPV derivation [11]**



PERFORMANCE AND TIME SAVING OF PROPOSED METHOD VS HM10.0

Sequence		Y	U	V	Time Saving
1080P	Kimono	1.5%	2.9%	2.7%	72%
	ParkScene	2.4%	2.3%	3.0%	68%
	Cactus	2.1%	1.7%	1.5%	69%
720P	Johnny	2.4%	3.2%	2.4%	73%
	FourPeople	1.5%	1.0%	2.0%	73%
	KristenAndSara	2.1%	2.9%	1.5%	73%
WVGA	BQMall	2.2%	2.8%	2.4%	70%
	BasketballDrill	2.2%	1.7%	2.0%	70%
WQVGA	BasketballPass	2.4%	2.3%	1.5%	70%
	BlowingBubbles	2.5%	2.4%	3.2%	62%
Average		2.1%	2.3%	2.2%	70%

Fig 19. **Performance and Time Saving of Proposed Method vs HM10 [11]**

Based on the experiment results shown in **Fig 19**, it can seen that the proposed parallel architecture with GPU and CPU pipeline can achieve up to 73% reduction in complexity as compared to HM 10.0 which uses only CPU. To verify the acceleration and compression performance of the proposed ME paralleling method, it has been implemented into HEVC reference software HM 10.0. The author ran simulations with reference software HM 10 on a desktop with AMD Phenom (tm) II X4 830 CPU Processor @ 2.80 GHz plus NVIDIA GeForce GTX 560 Ti GPU.

When we analyze **[22]** and **[11]**, we see that both were published very close to each other and both focus on motion estimation. Approach in

[22] approximates motion estimation so would suffer PSNR loss compared to **[11]**. The quantization of speed up comparison of both techniques as the comparison base is different, but **[22]** tends to be faster than **[11]** because of synchronization loss per CTU and increased scope of parallelization in **[22]**. GPU works in threading model which do same task but the amount may vary. This variation happens a lot in Motion Estimation module and hence results in preventing GPU to achieve maximum performance. Due to this reason, FPGA/ASIC presents an interesting alternative for Motion Estimation acceleration as variation of load doesn't affect the performance. Though other components like IQ/IT or looping filter don't have sufficient encoder occupancy, in case of decoders they do have substantial contribution to squeeze out performance. The original IQ/IT (inverse Quantization / inverse transform) process in HEVC decoder also uses GPU to gain some level of parallelism as proposed in **[21]**. For the IQ/IT (inverse quantization/inverse transform) part, HEVC processes Transform Units (TU) blocks one by one. TU block size can be 4x4, 8x8, 16x16 and 32x32 and this causes an increase in decoding time using this approach. The authors present a parallel way for processing IQ/IT part by processing one 64x64 size Luma block and two 32x32 size Chroma block parallelly by using GPU.
CPU can still only process TU blocks one by one because of the data dependence in IQ process. But, it can process every pixel parallelly in one TU block by using GPU. So, it can't get a sufficiently high level of parallelism following this way. The approach proposed by authors is called the mark_QP_table data mapping approach. The CABAC (Context-adaptive binary arithmetic coding) decoder can only decode one 64x64 CTU transform coefficients in one cycle. The proposed algorithm realizes the max parallelism: 64x64 CTU (Coding Tree Units). It includes three parts.

- QP_TU[256]: This table is used to store the QP value by the CPU for each TU block with max size of 256 as 64x64 can split into 256 4x4 blocks. The other calculation of IQ/IT happens in GPU
- Block_flag[20]: The flag is marked for the 16x16 block and the 32x32 block and use 5 numbers to explain the split condition for each 32x32 block in the table. These tables are used to map the data into the GPU.
- Mapping algorithm: it shows how to use these two tables to map the data into GPU. A 64x64 luma block has been split into many TUs block in CPU side. In the proposed algorithm, one work-item is used to process one pixel in GPU so for 64x64 luma block 4096 work items. As the TU partitions into 4 blocks of 32x32 size, the work items also get partitioned recursively until minimum TU size is 4x4. The minimum TU size would vary from 4x4 to 32x32 depends on the split flag. That is, every TU block's QP value's position in the QP_TU[] table is known . Following this way, we can know each TU block's QP value and its transform matrix.

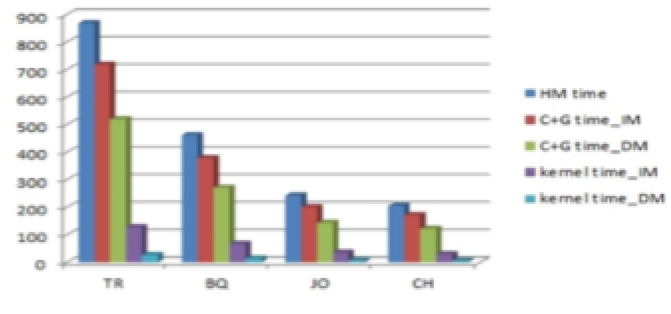


Fig 20 . **comparison among HM, IM and DM [21]**

The **Fig 20 shows the comparison among HM, IM and DM**.
HM: HEVC test model result.
IM: Original implementation of IQ/IT process result.
DM:Proposed approach which implements IQ/IT process into GPU.
HM time: Running time of computing the result by using GPU.
C+G time: Running time of control the kernel function by using CPU.
kernel time: Running time of computing the result by using GPU.

The buffer copies and OpenCL APis reduce the decoder performance. The author claims that DM's computing time and decoding time is 33.56 times and 40.7% faster than HM's respectively. DM can reduce 80% computing time compared with IM's. The author claims appears optimistic as HM has really bad IQ/IT implementation considering the scope of parallelism individually. Though ,the overall impact claimed on the HEVC decoder appears too large considering the overall contribution, the paper presents an approach to squeeze more performance using IQ/IT implementation.

Diego F. de Souza *et. al* paper **[12]** proposes an efficient parallelization of the in-loop filtering (Deblocking (non overlapped deblocking blocks processed) and SAO (custom data structure definition)) modules of the HEVC decoder by using low power GPU accelerators of embedded systems. The proposed parallel algorithms provide a significant reduction in the overall processing time being able to filter each Ultra HD 4k intra frame in less than 20 ms (about 50fps).

5B. <u>FPGA/ASIC</u>

The motion estimation clearly makes sense for FPGA/ASIC implementation because of its sluggishness on CPU and loss in RD cost. Compared to many GPU implementations FPGA/ASIC with additional advantage in terms of power gain, but the development cycle is very long. The video decoder are extensively designed in hardware as it avoids the CPU being heavily loaded in case of video playback. Similar approach is being applied for HEVC decoders by researchers and companies involved in ASIC implementation. The decoder does not have motion estimation and RD optimization but has other modules such as IQ/IT, Inter/Intra Prediction, Looping Filters and Entropy decoding. One such ASIC implementation is suggested by Wagner Penny *et. al* **[15]** using TSMC 65nm technology at 495.5 MHz with a throughput of 4320@60 fps for looping filters. For the HEVC encoders, motion estimation remains the main attraction for acceleration even for ASIC/FPGA implementation.

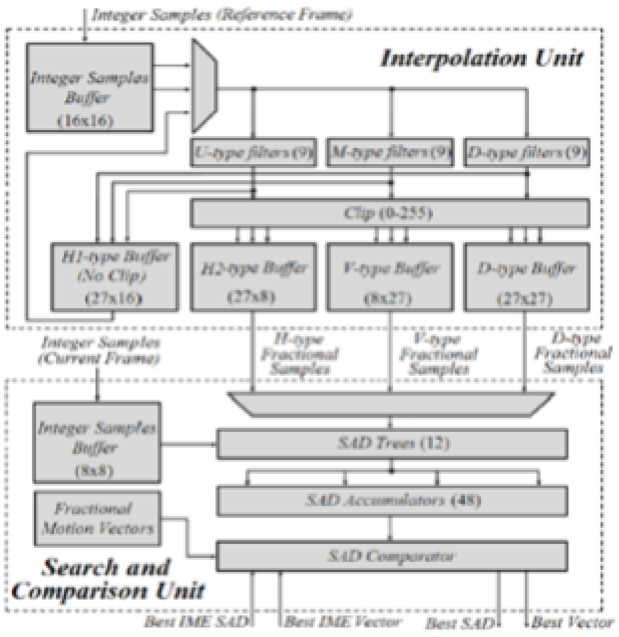

Integer Samples (Reference Frame)
Interpolation Unit
Integer Samples Buffer (16x16)
U-type filters (9)
M-type filters (9)
D-type filters (9)
Clip (0-255)
H1-type Buffer (No Clip) (27x16)
H2-type Buffer (27x8)
V-type Buffer (8x27)
D-type Buffer (27x27)
Integer Samples (Current Frame)
H-type Fractional Samples
V-type Fractional Samples
D-type Fractional Samples
Integer Samples Buffer (8x8)
SAD Trees (12)
SAD Accumulators (48)
Fractional Motion Vectors
SAD Comparator
Search and Comparison Unit
Best IME SAD
Best IME Vector
Best SAD
Best Vector

Fig 21. **HEVC FME Hardware Design [13]**

In the paper **[13]**, the author proposes the architecture using FME that implements a strategy to reduce the number of accesses to the reference frame in the memory by up to 49.22%. The FME technique is applied after the Integer Motion Estimation (IME) in order to refine the results in terms of compression. VHDL is used to describe the hardware design and it is synthesised for FPGA and ASIC technologies.

HEVC ME Mode Distribution: The ME of HEVC standard allows the use of 24 different PU sizes which is costly for real-time processing of UHD videos and this impact is reduced if we avoid the evaluation of most of the PU sizes. During the experiments with reduced PU sizes, it was observed that when ME process was limited to 32x32 and 64x64, it led to coding efficiency degradation.

HEVC FME Hardware design : The proposed architecture was designed to perform the FME only at the PU size for which the best IME model was presented. The hardware is divided into 2 main units: The Interpolation Unit and the Search and Comparison Unit. The fractional samples are calculated using the following 2 equations. (O represents output samples and I represents input samples)

$O_0 =(-1*I_0 + 4 *I_1 - 11*I_2 + 40*I_3 + 40*I_4 - 11*I_5 + 4*I_6 - 1*I_7 + 32)/64$

$O_1 =(-1*I_0 + 4 *I_1 - 10*I_2 + 58*I_3 + 17*I_4 - 5*I_5 + 1*I_6 + 32)/64$

Three different types of filters: Up-type, Middle type, and Down-type were implemented to reach the desired amount of parallelism.

Each filter is responsible for calculating 5 fractional positions. The filter's output is connected to four fractional sample buffers: Horizontal-type(2-buffers), Vertical-type, and Diagonal types. Once the pipeline is filled, the interpolation unit processes a new 8x8 block in 51 clock cycles, of which 16 are used to calculate the horizontal samples, 8 for the vertical samples and 27 for the diagonal samples.

The Search and Comparison Unit accumulates the Sum of Absolute Differences (SAD) of the 48 fractional blocks generated by the Interpolation Unit and compares it with best IME SAD. The proposed hardware design is developed for 8x8 PUs. For PUs of size 16x16, 32x32 and 64x64 samples, there is a reduction of 37.5%, 46.88% and 49.22% respectively in total number of accesses to the reference frames stored in external memory.

The synthesis results of FPGA and ASIC shows the ability of FME design to process UHD 2160p videos at 60fps with a power consumption of 48.67mW. The paper **[16]** presents a Fractional Motion Estimation (FME) design in high efficiency video coding for ultra high definition video (Ultra HD). The paper introduces an exhaustive size-Hadamard Transform(ES-HAD) to improve coding quality. The design is implemented in 65nm CMOS chip and verified by FPGA based evaluation system.

FME algorithm in HEVC reference software: In HEVC, an input picture is divided into coding tree blocks (CTBs) of size 64 x 64 which are further divided into coding blocks of size 8 x 8 using a recursive quad tree structure. For each prediction mode for CB, the procedure of motion estimation is classified into 3 steps: 1) *Integer Motion Estimation (IME) performs motion search to find integer motion vectors*, 2) *FME provides the sub-pixel refinements around IMVs* 3) *At last Residual quadtree (RQT) processes the transform coding*

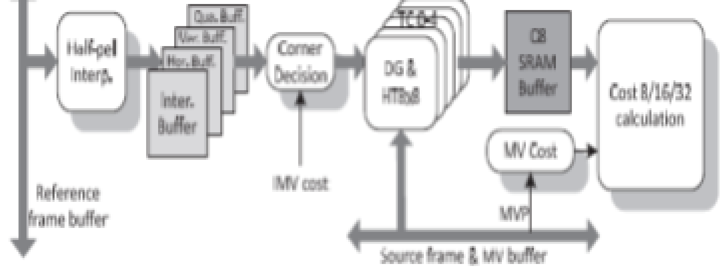


Fig 22. **Hardware architecture for FME [16]**

Proposed FME algorithm

A – Bilinear Quarter Pixel Approximation (BQA) scheme: BQA scheme employs a half-then-quarter model and uses bilinear filter for quarter pixels.

B - 5T12S Search Pattern: The search pattern in referenced software takes two iterations and it also introduces long latency which is costly. To reduce the hardware cost, the authors propose a 5T12S search pattern.

C – Exhaustive Size-HAD: In the proposed design exhaustive size HAD are used to process the residuals by HT8x8, HT16x16, HT32x32 which is then compared recursively to get rate distortion cost

Hardware Architecture for FME: The figure shows the block diagram of the proposed FME design. The architecture incorporates the following features:

A – Parallelism: The most efficient way for the interpolation is to design the processing unit based on the smallest block. The authors double the parallelism by adopting two horizontal units and embedding two 8-tap filters into a vertical unit, as shown

B – Data reuse in ES-HAD: Residuals are processed using ES-HAD with HT8x8, HT16x16, HT32x32. With individual implementation, six, eight, ten addition(ADD) layers are needed for pipeline design.

C – Memory Organization: HT8x8 adopts 16-pixel parallelism and processes a 16x16 block row by row. The design employs a two port SRAM to store the coefficients of HT8x8 (C8).

In comparison with the state-of- art design, this design reaches 995 Mpixels/s high throughput for 7680x4320 30fps video, at least 4.7 times faster than previous designs. The chip achieves better power efficiency (0.2nJ/pixel) than previous works in H.264 with 56% improvement.

6. Cloud based architectures[7][8]

Cloud-based distributed HEVC encoding has gained increasing attention because dedicated hardware solutions are expensive and lack flexibility. Distributed video encoding requires the input video to be partitioned into multiple chunks, which are then encoded independently via computational nodes. Video partitioning is an essential step in the encoding process. Hence, exploring the effects of video partitioning methods on encoding time and bitrate is important. The authors study performance considerations for two video partitioning methods.

1. Uniform-partitioning is a basic partitioning method used in many existing distributed encoding systems. In this method, the input video is partitioned into N video segments of the same size (M/N), where M and N represent the total number of frames to encode and the total number of nodes to participate in encoding, respectively. This has a few disadvantages:
   a. Individual segments can contain frames belonging to different GOPs. In that case, the frames from some 'GOP Y' cannot reference frames at the tail area of some 'GOP X', leading to inefficient inter-coding.
   b. The number of I-frames may increase in accordance with the segments formed. Hence, the bit rate would increase

2. GOP based partitioning: The unit of distribution is the GOP; therefore, the size of the segment is equal to that of the GOP and there are M/G segments in total, where G denotes the GOP size.
   a. Disadvantage: GOP-based partitioning also suffers from limited use of reference pictures between GOPs as a result of partitioning.
   b. Advantage over Uniform-partitioning: It does not allow the GOP itself to be partitioned, so there is no additional increase in the number of I-frames.

The Proposed MapReduce-based distributed HEVC encoding system has 3 stages:

a. In the partitioning stage, the uncompressed video file is partitioned into several non-overlapping segments.

b. In the encoding stage, individual Map tasks perform HEVC encoding (HEVC test model (HM 14.0)).
c. In the combining stage, the Reduce task receives HEVC encoded segments from Map tasks and merges them into a final HEVC encoded video file.

Experimental Results:
Type I represents scenes with many motion changes between frames
Type II represents static scenes with relatively few movements and similar adjacent frames.

The uniform partitioning method requires less encoding time than the GOP-based method. This is because, in the uniform video partitioning, all of the segments are processed in one cycle, whereas in GOP-based partitioning, as the number of segments exceeds the maximum allowable nodes, some segments can only be processed only after the 'Map' tasks finishes encoding the current segments.
The Fig 24 shows changes in the bitrate:

> The Uniform video partitioning:
As the number of computational nodes performing encoding increases, the bitrate also increases due to the increased number of I-frames and limited use of reference pictures.
> The GOP-based partitioning :
It shows the same bit rate regardless of the nodes (except in the single node case) because the number of segments and thus the coding structure remain the same.

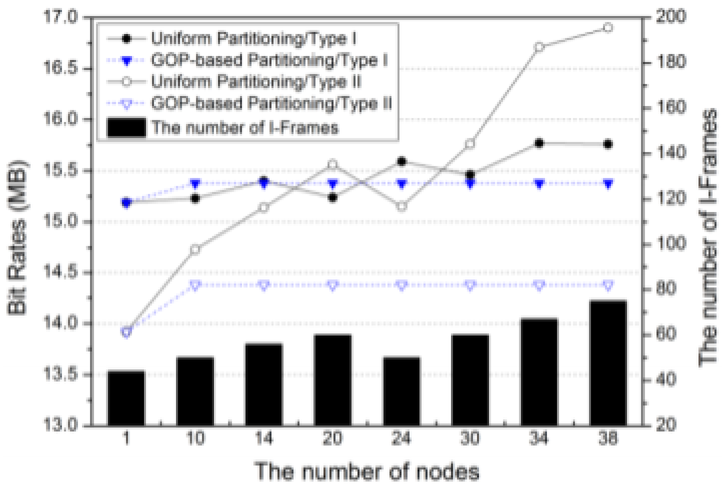


Fig 24. **Changes in bit rate [7]**

To conclude, distributed video encoding requires partitioning and allocating chunks of the input video to appropriate computational nodes for subsequent encoding. The results suggests that, depending on how the input video is partitioned, it might be possible to avoid performance loss in terms of both the encoding time and the bitrate. The experimental results show that distributed processing generally reduces total encoding time significantly at the cost of increased bit rate, due to increased I-frames and/or limited use of reference pictures. However, for uniform partitioning, a case with fewer segments will not always outperform one with more segments in terms of bitrate.

7. Networking aspects

The real-time video streaming not only has implementation-based architectural challenges, but also is constrained by network-based architectural and physical limitations. Video streaming packets over a network suffers through bandwidth, latency, packet loss and congestion issues which may require separate modification in the container formats and packetisation of the video stream. These networking issues are the nature of the medium and need to be handled effectively for better quality of experience. Due to varied bandwidth and internet traffic over time, place and device, it requires that the video streaming should have support for multi-bitrate and multi-resolution streaming . This kind of support leads to a smoother quality of experience at the end-user, as the video consumption is not halted completely by network issues. In this section we will be discussing the proposed implementations and architectures of multi-bitrate and multi-resolution high quality streaming. We will also discuss the proposed techniques for the efficient bandwidth usage, congestion control and packet loss

7A. Multi-bitrate and multi-resolution support

The SHVC or Scalable High Efficiency Video Encoding standard is an extension over HEVC standard which proposes method and design to support added functionality over HEVC standard. It is being rapidly used for delivery 4K resolution content especially for video streaming in overlay or over-the-top (OTT) networks using HTTP or adaptive bitrate streaming technologies. The SHVC standard definition allows the HEVC to support features like multi-bitrate and multi-resolution streaming. It uses inter-layer predictions to take advantage of the spatial correlations. SHVC video transmission can be done over satellite and terrestrial broadcast networks. Further, It has two major advantages which makes it an attractive option for UHD TV broadcasting and IPTV or HTTP streaming for UHD-TV:

- The SHVC coding into two layers introduces a rate-distortion loss of only 10%-30% in respect to the simulcast HEVC coding configuration.
- SHVC standard supports the base layer coded in the legacy AVC standard.

HTTP streaming technologies such as the Dynamic Adaptive Streaming over HTTP (MPEG-DASH) Standard are being deployed for high bandwidth high resolution video streaming. The savings and the overhead of encoding UHD content as an SHVC enhancement layer (EL) with AVC or HEVC HD base layer (BL) have been investigated in **[1]**, primarily focusing on the spatial scalability aspect of SHVC in the context of HTTP streaming and how UHD video content can be delivered in a deployment scenario with heterogeneous device and network characteristics. According to rigorous experiments done by the authors, It was concluded that SHVC performs better with HEVC as base layer compared to using AVC as the base layer. Also, as we increase the number of layers, we get better bitrate savings, however the scalability overhead increases. Hence, UHD videos can be efficiently deployed using SHVC.

4Kp30 end-to-end video streaming has also been demonstrated based on the SHVC standard in **[2]**. In this model, the authors transmit at high bitrates two SHVC layers over separate PIDs over satellite channel for full HD and 4K resolutions and at low bitrates, only one SHVC base layer over terrestrial channel enabling only full HD. This use-case uses SHVC capability to adapt the bitrate of the video content to the channel bandwidth constraint, and avoids using resource-consuming transcoding solutions.

All decoding operations including motion compensation, inverse transform and up-sampling filters are optimized in Single Instruction Multiple Data (SIMD) Intel SSE methods for x86 architecture. The decoder is friendly-parallel and leverages multi-core architecture to improve both the decoding frame rate and the decoding frame latency. This hybrid parallelism solution performs a high speedup performance with a good trade-off between decoding frame rate, decoding latency and memory usage.

7B. Efficient bandwidth usage, congestion control and packet loss

First we discuss the system architecture for UHD streaming and two main ideas proposed by authors from **[3]** (i) picture prioritization method, (ii) error concealment mode signaling (ECMS). Because of the importance of UHD TV in the current era, robust video streaming technologies including video packet error protection and error concealment (EC) are essential. The current work on video coding standards is mostly concentrating on compression while not giving enough importance to video transmission. The paper **[3]** describes some picture prioritization and error concealment methods, then expands over the proposed robust 4K UHD video streaming system and concludes with the gains in the video quality by using the proposed scheme.

Picture prioritization is of utmost importance for the role it plays in UEP (Unequal Error Protection), picture dropping for bandwidth adaptation, as well as quantization parameter (QP) control for enhanced video quality. It can be implemented in four ways: (a) Use picture type information which is related to temporal reference dependency; (b) Use temporal level information in hierarchical B structure, and higher layer will not be referenced by lower layer; (c) Use location information of slice groups (SGs) (SG-level prioritization); and (d) Use the layer information of the SVC (Scalable Video Coding). It is used for several QoS purposes such as dropping less important pictures in the transmitter or scheduler of the server for bandwidth adaptation and allocating more important pictures to more stable channels (or antennas) in multi-channel networks or MIMO. The authors talk about their proposed error resilient video streaming system in two parts:

1.Picture Prioritization and Unequal Error Protection (UEP): The uniform prioritization (applies same priority to pictures in the same temporal level of hierarchical B structure) presents a problem when the importance of two (or more) POCs (Picture Order Count) in each group of picture (GOP) varies according to both the reference picture set (RPS) and the size of the reference picture lists (RPL). To solve this problem, the paper proposes an implicit picture prioritization method that the encoder assigns priorities to pictures according to the RPS and the size of RPL of the encoding option without any additional delays.The proposed method shows the gains in video quality from 2.2 to 7.5 dB in Y-PSNR (Peak Signal to Noise ratio)

2. Error Concealment with Mode Signalling: The EC method is important for a scalable video coding transmission system over error-prone network. Applying the best EC method can improve not only the quality of the lost picture, but also the quality of the other pictures that are affected due to error propagation. The proposed EC mode signaling method enables the video encoder to (i) simulate various EC methods on a damaged picture, (ii) determine the best EC method that provides minimal disparity between an original image and a reconstructed image, and (iii) signal the best EC mode to the video decoder at the client. This does not increase the computational complexity at encoder side. On comparison of the proposed EC mode with other picture copy EC modes, the gains in video quality vary from 0.2dB to 2.5 dB in PSNR along with the corresponding subjective improvements in noticeable visual differences.

Next, we talk about congestion control for UHD streaming over the internet **[4]**. The advances in multimedia and network technology to enable Ultra HD video streaming have increased the importance of good congestion control over the network. Previous congestion control schemes have several limitations to guarantee the quality of UHD video streaming such as its sluggish behaviour disturbs the rapid increase in video quality and there are burst packet losses leading to degradation of the quality of video. We look at a proposed scheme that quickly increases the video quality and reduces packet losses by using concave, convex, and linear function.

Recent TCP based streaming technology such as DASH has been widely adopted as a multimedia streaming protocol but has some problems for UHD video streaming over the Internet because of the sluggish behavior. Several congestion control algorithms such as STCP(Scalable TCP), HSTCP(High Speed TCP), HTCP(High-speed TCP), BIC-TCP, and CUBIC have been proposed. The main idea behind these algorithms is the following: When no congestion occurs, the sender increases its congestion window(cwnd) quickly. In principle, a convex growth function is used to ramp up the cwnd to very large values. However, a convex function results in a very large window increment around the point of saturation and can cause a large burst of packet losses that degrade quality of video. Therefore, new congestion control scheme should be deployed in high-resolution smart devices to guarantee the quality of UHD video streaming service.

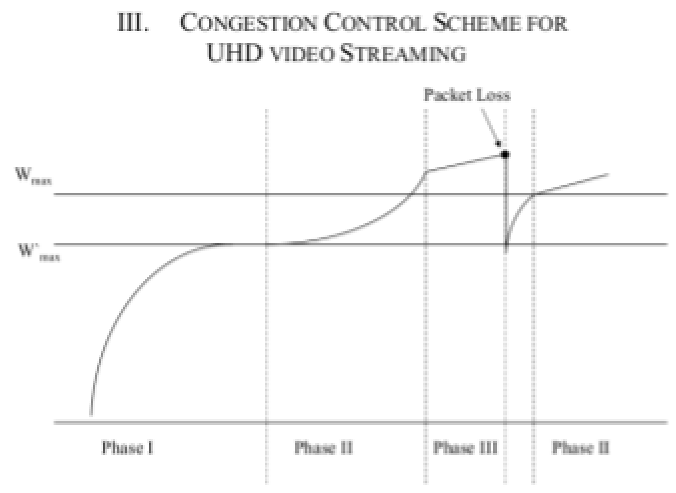


Fig 25. **Transmission rate and packet loss of slow-start [3]**

The authors describe the new scheme: to provide high bitrate quickly while reducing the packet losses caused by overshooting of transmission rate, the proposed scheme adjusts the cwnd on the basis of the packet loss and network buffer status.This finite state automata has been described with a graph. The proposed scheme quickly increases transmission rate after slow-start state and has the highest throughput while having lower packet losses than other schemes except TCP. However, it is hard for TCP to guarantee the quality of UHD video streaming service because the throughput is almost half of the proposed scheme.
Hence, we now appreciate the importance of increasing speed of transmission rate and reducing packet losses for QoS for UHD streaming.

8. Recent developments

Lots of research and industrial investments have been done to achieve ultra HD encoding, streaming and playback. Some of the recent developments highlight the tremendous progress in this area. YouTube now supports 4K live-streaming, allowing content creators to broadcast high-resolution video in both 360-degree and standard video formats at a rate of 60 frames per second. This update is a complimentary addition to the news that YouTube has also added 4K HDR support. This is a great update for both YouTube creators and viewers as standard 4K content is comparatively easier to film and is more accessible than 4K HDR content.
In E3 2016 Microsoft announced 4K Xbox code-named Project Scorpio. Alongside the Scorpio, Microsoft also has the Xbox One S which will upscale HD content to 4K as well as play Ultra HD Blu-ray discs. Microsoft isn't the only console manufacturer with a 3840 x 2160 resolution on its mind, however. Sony just announced a 4K console of its own called the PS4 Pro that not only plays 4K Ultra HD movies and TV shows from streaming services like Netflix, but can play games in 4K, too.
Also, Ittiam Systems (2015) which is a leading technology provider to the broadcast and media industry, recently announced the availability of its i265 H.265/HEVC Encoder for the new Intel Visual Compute Accelerator (Intel® VCA) platform.

Sony launched its Video Unlimited 4K service in 2013, which offers more than 70 films and TV shows for rental or purchase. It requires Sony's 4K Ultra HD Media Player.
The developments show that 4k60 fps playback, encoding and decoding have been achieved on varied complexity of devices for some streams. Though video streaming of only 4K30fps is currently supported by content providers.

9. Conclusion

In our study we illustrated the challenges and bottlenecks for high quality encoding as well as decoding. We described that parallelization strategies described by the video codec standard to implement these codecs efficiently for varied architectures and applications. We narrowed that for HEVC encoding and decoding wavefront parallel processing (WPP) (or its variants like OWP, IFW) is a performance and coding efficient approach for multicore architectures. The extension of WPP, IWP even exploits the interframe parallelism available. Tiles based encoding is superior to slice encoding in coding efficiency but has a higher implementation complexity. Tile based encoding can be particularly useful for many-core CPUs and distributed computers to exploit spatial level parallelism. The motion estimation and RD cost search range optimization have been found to be main performance bottlenecks in HEVC encoder implementation. Host + accelerator basically exploits this bottleneck and implements motion estimation and coordinates with CPU for RD cost search range optimization. It could be conferred that if motion estimation is done on accelerator with fast search range algorithms on CPU along with WPP and SIMD optimization for other kernels a substantial fast real-time encoding is possible. We detail how the motion estimation is implemented in GPUs and ASICs with concentration on the design approach that needs to be taken. We touch a cloud computing architecture implementation which details that how GOP partitioning is beneficial in maintaining coding efficiency and achieve speedup. For the high quality video streaming, the networking aspects are as crucial as encoding or playback aspects. In fact the streaming criteria like bandwidth, congestion, packet loss, are some of the factors that limits live ultra hd streaming. With the recent developments, the realization of live ultra HD streaming is not a dream too far.

References

1. Investigating Scalable High Efficiency Video Coding for HTTP streaming (Uma Sagar Madhugiri Dayananda, Dept. of Elec. Eng., University of Texas, Arlington, Viswanathan Swaminathan, Adobe Research, Adobe Systems Incorporated)

2. 4K real time video streaming with SHVC decoder and GPAC player (W. Hamidouche, G. Cocherel, J. Le Feuvre, M. Raulet and O. D ́eforges, IETR/INSA, Rennes, France and T ́el ́ecom Paris Tech, Paris, France)

3. Towards robust UHD video streaming systems using scalable high efficiency video coding(Eun-Seok Ryu (a) , Yeongil Ryu(a) , Hyun-Joon Roh(a) , Joongheon Kim(b) , Bok-Gi Lee(a))
(a)Department of Computer Engineering, Gachon University, Rep. of Korea
(b)Platform Engineering Group, Intel Corporation, Santa Clara, California, USA

4. Congestion control scheme for UHD video streaming over the Internet(Sunghee Lee, Hyunwoo Lee, Won Ryu Intelligent Convergence Media Research Department Electronics and Telecommunications Research Institute Daejeon, Korea)

5. Challenges of Distributing 4K Video - Crestron

6. Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders (Dan Grois, Senior Member, IEEE, Detlev Marpe, Senior Member, IEEE, Amit Mulayoff, Benaya Itzhaky, and Ofer Hadar, Senior Member, IEEE).

7. Cloud-based Distributed HEVC Video Encoding (Byoung-Dai Lee, Department of Computer Science, Kyonggi University)

8. Empirical Analysis of Video Partitioning Methods for Distributed HEVC Encoding (Byoung-Dai Lee, Department of Computer Science, Kyonggi University, Suwon, Korea)

9. HEVC Complexity and Implementation Analysis (Frank Bossen, Member, IEEE, Benjamin Bross, Student Member, IEEE, Karsten Suhring, and David Flynn)

10. EFFICIENT REALIZATION OF PARALLEL HEVC INTRA ENCODING (Yanan Zhao (a), Li Song (a) , Xiangwen Wang (b), Min Chen (b), Jia Wang (a))
(a)Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University,
(b) Shanghai University of Electric Power

11. Flexible CTU-level Parallel Motion Estimation by CPU and GPU Pipeline for HEVC(Juncheng Ma, Falei Luo, Shanshe Wang, Siwei Ma)
Institute of Digital Media, Peking University, Beijing 100871, China
University of Chinese Academy of Sciences, Beijing, China

12. HEVC In-Loop Filters GPU Parallelization in Embedded Systems (Diego F. de Souza, Aleksandar Ilic, Nuno Roma and Leonel Sousa) Universidade de Lisboa, Rua Alves Redol 9, 1000-029, Lisbon, Portugal

13. Memory-Aware and High-Throughput Hardware Design for the HEVC Fractional Motion Estimation(Vladimir Afonso, Henrique Maich, Luan Audibert, Bruno Zatt, Marcelo Porto, Luciano Agostini)
Federal University of Pelotas, Group of Architectures and Integrated Circuits Pelotas, RS, Brazil
14. A Parallel H.264 Encoder with CUDA: Mapping and Evaluation (Nan Wu , Mei Wen , Huayou Su , Ju Ren , Chunyuan Zhang) ,Computer School, National University of Defense Technology, P. R. of China
15. Real-Time Architecture for HEVC Motion Compensation Sample Interpolator for UHD Videos (Diego F. de SouzaWagner Penny, Guilherme Paim, Marcelo Porto, Luciano Agostini, Bruno Zatt)
Federal University of Pelotas Group of Architectures and Integrated Circuits Pelotas, Brazil
16. A 995Mpixels/s 0.2nJ/pixel fractional motion estimation architecture in HEVC for Ultra-HD (Gang He, Dajiang Zhou, Zhixiang Chen, Tianruo Zhang and Satoshi Goto)
Graduate School of Information Production and Systems, Waseda University, Kitakyushu, Japan
17. Parallel Scalability and Efficiency of HEVC Parallelization Approaches (Chi Ching Chi, Mauricio Alvarez-Mesa, Member, IEEE, Ben Juurlink, Senior Member, IEEE, Gordon Clare, Felix ´ Henry, Stephane Pateux and Thomas Schierl, Member, IEEE)
18. Hybrid Parallelization for HEVC Decoder (Hyunho Jo, Donggyu Sim Computer Engineering Kwangwoon University Seoul, Korea Byeungwoo Jeon School of Electronic and Electrical Engineering Sungkyunkwan University Suwon, Korea)
19. A High Parallel Way for Processing IQ/IT part of HEVC Decoder Based on GPU (Lang-ping He,and Satoshi Goto from Waseda University, Japan;2014 IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS))
20. HEVC Encoding Optimization Using Multicore CPUs and GPUs (Wei Xiao, Bin Li, Member, IEEE, Jizheng Xu, Senior Member, IEEE, Guangming Shi, Senior Member, IEEE, and Feng Wu,IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 25, NO. 11, NOVEMBER 2015)
21. A High Parallel Way for Processing IQ/IT part of HEVC Decoder Based on GPU (Lang-ping He Kitakyushu, Fukuoka, Waseda University, Japan, IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) December 1-4, 2014)
22. OpenCL based high-quality HEVC motion estimation on GPU(Fan Wang, Dajiang Zhou, Satoshi Goto)
Graduate School of Information, Production and Systems, Waseda University. 2-7 Hibikino, Kitakyushu 808-0135, Japan.
23. Xiangwen Wang, Li Song, Min Chen, Junjie Yang, "Paralleling variable block size motion estimation of HEVC on CPU plus GPU platform", ***Multimedia and Expo Workshops (ICMEW) 2013 IEEE International Conference on*****, pp. 1-5, 15–19 July 2013**
24. Parallelism In Video Streaming - Cameron Baharloo
25. A Novel Wavefront-Based High Parallel Solution for HEVC Encoding(Keji Chen, Jun Sun, Member, IEEE, Yizhou Duan, and Zongming Guo, Member, IEEE)