Design and Implementation of Next Generation Video Coding Systems (H.265/HEVC Tutorial)

Vivienne Sze (sze@mit.edu)
Madhukar Budagavi (m.budagavi@samsung.com)

ISCAS Tutorial 2014
Instructors

• Vivienne Sze (Assistant Professor at MIT)
  – Involved with video implementation research and standards for 7+ years
    • Contributed over 70 technical documents to HEVC.
    • Within JCT-VC Committee, Primary Coordinator of the core experiments on coefficient scanning and coding; chairman of ad hoc groups on topics related to entropy coding and parallel processing.
    • Published over 25 journal and conference papers.

• Madhukar Budagavi (Research Director at Samsung Research America)
  – Involved with video standards and product development for 15+ years
    • Contributed over 100 technical documents to HEVC.
    • Within JCT-VC Committee, Chaired and co-chaired sub-group activities on spatial transforms, quantization, entropy coding, in-loop filtering, intra prediction, screen content coding and scalable HEVC (SHVC).
    • Published over 40 journal and conference papers, book chapters.
Outline of Tutorial

• Part I: Overview of current video coding technology and systems
• Part II: High Efficiency Video Coding (HEVC)
• Part III: Video Codec Implementations
• Part IV: Emerging Applications and HEVC Extensions
Part I: Overview of current video coding technology and systems
Growing Demand for Video

- Video exceeds half of internet traffic and will grow to 86 percent by 2016. Increase in applications, content, fidelity, etc. → Need higher coding efficiency!
- 25x increase in mobile data traffic over next five years. Video is a “must have” on portable devices. → Need lower power!

Sources: Cisco Visual Networking Index
Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update
Digital Video

4:2:0

W × H

Y

Cb

Cr
Video Compression

• Uncompressed 1080p high definition (HD) video at 24 frames/second
  – Pixels per frame: 1920x1080
  – Bits per pixel: 8-bits x 3 (RGB)
  – 1.5 hours: 806 GB
  – Bit-rate: 1.2 Gbits/s

• Blu-Ray DVD
  – Capacity: 25 GB (single layer)
  – Read rate: 36 Mbits/s

• Video Streaming or TV Broadcast
  – 1 Mbits/s to 20 Mbits/s

• Require 30x to 1200x compression
Video Compression Basics

- Compression is achieved by removing redundant information from the video sequence.
- Types of redundancies in video sequences:
  - Spatial redundancy
  - Perceptual redundancy
  - Statistical redundancy
  - Temporal redundancy
Spatial Redundancy Removal (1)

- Intra prediction
Spatial Redundancy Removal (2)

- Block Transforms
  - Typically matrix operations
  - Used for correlation reduction and energy compaction in the block

8x8 2D Discrete Cosine Transform (DCT)
• Not all video data are equally significant from a perceptual point of view

• Make use of the properties of the Human Visual System (HVS)
  – HVS is more sensitive to low frequency information
Perceptual Redundancy Removal (2)

- Quantization is a good tool for perceptual redundancy removal
  - Most significant bits (MSBs) are perceptually more important than least significant bits (LSBs)
  - Coefficient dropping (quantization with zero bits) example:

Original frame

Image obtained by retaining 36 DCT coefficients for each 8x8 block
• Not all pixel values in an image (or in the transformed image) occur with equal probability

• Use entropy coding (e.g. variable length coding)
  – Shorter codewords used to represent more frequent values
  – Longer codewords used to represent less frequent value
Statistical Redundancy Removal (2)

- Original image: 8 bits/pixel, Entropy coding: 7.14 bits/pixel

- Results more dramatic when entropy coding is applied on transformed and quantized image: 1.82 bits/pixel
Temporal Redundancy Removal (1)

- Inter prediction
- Frame difference coding
  - Difference can be encoded using DCT + Quantization + Entropy Coding
Temporal Redundancy Removal (2)

• Inter prediction using Motion compensated prediction

– Divide the frame into blocks and apply block motion estimation/compensation
– For each block find out the relative motion between the current block and a matching block of the same size in the previous frame
– Transmit the motion vector(s) for each block
Temporal Prediction and Picture Coding Types

- **Intra Picture (I)**
  - Picture is coded without reference to other pictures

- **Inter picture (P, B, b)**
  - Uni-directionally predicted (P) Picture
    - Picture is predicted from one prior coded picture
  - Bi-directionally predicted (B, b) Picture
    - Picture is coded from two prior coded pictures
Summary of Key Steps in Video Coding

- Intra Prediction and Inter Prediction

- Transform and Quantization of residual (prediction error)

- Entropy coding on syntax elements
e.g. prediction modes, motion vectors, coefficients

- In-loop filtering to reduce coding artifacts

* Residual figure from J. Apostolopoulos, “Video Compression,” MIT 6.344 Lecture, Spring 2004
Video Compression Standards

- Ensures inter-operability between encoder and decoder
- Support multiple use cases and applications
- Levels and Profiles
- Video coding standard specifies decoder: mapping of bits to pixels
- ~2x improvement in compression every decade

Source → Pre-Processing → Encoding → Decoding → Post-Processing → Destination

Scope of Standard

bit-rate

- MPEG-2
- H.264/AVC
- HEVC

1994 2003 2013
History of Video Coding Standards

- MPEG: Moving Picture Experts Group (ISO/IEC)
- VCEG: Video Coding Experts Group (ITU-T)
- Other standards: VC1, VP8/VP9, China AVS, RealVideo
Video Coding Progress

- **Variable block size** (16x16 – 4x4) + quarter-pel + multi-frame motion compensation (H.264/AVC, 2003)
- **Variable block size** (16x16 – 8x8) (H.263, 1996) + quarter-pel motion compensation (MPEG-4, 1998)
- **Half-pel motion compensation** (MPEG-1 1993, MPEG-2 1994)
- **Integer-pel motion compensation** (H.261, 1991)
- **Intraframe DCT coding** (JPEG, 1990)

Source: T. Wiegand, JVT-W132, 2007
H.264/MPEG-4 AVC

• Completed (version 1) in May 2003
• H.264/AVC is the most popular video standard in market
  – 80% of video on the internet is encoded with H.264/AVC
• Applications include
  – HDTV broadcast satellite, cable, and terrestrial
  – video content acquisition and editing
  – camcorders, security applications, Internet and mobile network video, Blu-ray Discs
  – real-time video chat, video conferencing, and telepresence
• ~50% higher coding efficiency than MPEG-2 (used in DVD, US terrestrial broadcast)
Improvements of H.264/MPEG-4 AVC over previous standards

• Prediction
  – Intra prediction using neighboring samples
  – Temporal prediction using multiple frames
  – Motion compensation on variable block size, quarter-pel

• Transform
  – 4x4/8x8 Integer transform, 2x2/4x4 Secondary Hadamard

• Quantization
  – Finer quantization supported

• Entropy coding
  – Context adaptive variable length coding (CAVLC) and arithmetic coding (CABAC)

• In-loop deblocking filter
Part II: High Efficiency Video Coding (HEVC)
High Efficiency Video Coding (HEVC)

- Achieves 2x higher compression compared to H.264/AVC
- High throughput (Ultra-HD 8K @ 120fps) & low power
  - Implementation friendly features (e.g. built-in parallelism)
- Benefits include
  - reduce the burden on global networks
  - easier streaming of HD video to mobile devices
  - account for advancing screen resolutions (e.g. Ultra-HD)

“HEVC will provide a flexible, reliable and robust solution, future-proofed to support the next decade of video”

Activity in JCT-VC Committee

- Chairs
  - G. J. Sullivan (Microsoft)
  - J. R. Ohm (Aachen University)
- Meet Quarterly
  - 1st meeting (A) [January 2010]
  - 12th meeting (L) [January 2013]
- ~250 attendees per meeting representing ~70 companies
- Several hundred contributions per meeting
- Each meeting is around 9 - 10 days (14+ hours/day)
- Multiple parallel tracks
HEVC Reference Documents

• Meeting Contributions
  – http://phenix.int-evry.fr/jct/

• Specification

• Reference Software (HM)
  – https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

• References
### Coding Efficiency of HEVC (Objective)

**TABLE VI**

**AVERAGE BIT-RATE SAVINGS FOR EQUAL PSNR FOR ENTERTAINMENT APPLICATIONS**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Bit-Rate Savings Relative to</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H.264/MPEG-4 AVC HP</td>
</tr>
<tr>
<td>HEVC MP</td>
<td>35.4%</td>
</tr>
<tr>
<td>H.264/MPEG-4 AVC HP</td>
<td>–</td>
</tr>
<tr>
<td>MPEG-4 ASP</td>
<td>–</td>
</tr>
<tr>
<td>H.263 HLP</td>
<td>–</td>
</tr>
</tbody>
</table>

**PSNR** = 10 log\(_{10}\) \(\frac{(2^{\text{bitdepth}} - 1)^2 \cdot W \cdot H}{\sum_i \{O_i - D_i\}^2}\)

## Coding Efficiency of HEVC (Subjective)

Subjective Tests for Entertainment Applications (Random Access)

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Bit-rate Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>BQ Terrace</td>
<td>63.1%</td>
</tr>
<tr>
<td>Basketball Drive</td>
<td>66.6%</td>
</tr>
<tr>
<td>Kimono1</td>
<td>55.2%</td>
</tr>
<tr>
<td>Park Scene</td>
<td>49.7%</td>
</tr>
<tr>
<td>Cactus</td>
<td>50.2%</td>
</tr>
<tr>
<td>BQ Mall</td>
<td>41.6%</td>
</tr>
<tr>
<td>Basketball Drill</td>
<td>44.9%</td>
</tr>
<tr>
<td>Party Scene</td>
<td>29.8%</td>
</tr>
<tr>
<td>Race Horse</td>
<td>42.7%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>49.3%</strong></td>
</tr>
</tbody>
</table>

J. Ohm et al., "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, 2012
H.265/HEVC vs. H.264/AVC Decoder

- Entropy Decoder
- Picture Buffer
- Motion Comp.
- Intra Prediction
- Q^{-1} + T^{-1}
- In-loop Filter
- Deblocking Filter
- Sample Adaptive Offset
- Fewer Edges
- Larger and Flexible Coding Block Size
- High Throughput CABAC & Advanced Motion Vector Prediction
- Larger Transforms and More Sizes
- More Prediction Modes

Encoded bitstream

Decoded pixels

64x64
# Key Features In HEVC

<table>
<thead>
<tr>
<th>Feature</th>
<th>High Coding Efficiency</th>
<th>High Throughput / Low Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larger and Flexible Coding Block Size</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>More Sophisticated Intra Prediction</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Larger Interpolation Filter for Motion Compensation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Larger Transform Size</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Parallel Deblocking Filter</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sample Adaptive Offset</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>High Throughput CABAC</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>High Level Parallel Tools</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Parallel Merge/Skip</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Larger Coding Blocks

- Each frame is broken up into blocks
- Large block sizes reduce signaling overhead
- In H.264/AVC, **macroblock** is always 16x16 pixels
  - Each macroblock is either inter or intra coded
- In HEVC, **Coding Tree Unit (CTU)** can have up to 64x64 pixels
  - CTU can have a combination of inter and intra coded blocks

\[
N = 16, 32, \text{ or } 64
\]
Flexible Coding Block Structure

- Better adaptation to different video content
- CTU divided into Coding Units (CU) with Quad tree
- Coding units divided into prediction units (PU)
- PU have different motion data or prediction modes

**Coding Tree** composed of Coding Units (CU)

**Prediction Unit** (PU)

**Coding Tree Unit** (CTU)

Asymmetric Motion Partition
Prediction Units

• Intra-Coded CU can only be divided into square partition units
  – For a CU, make decision to split into four PU (8x8 CUs only) or single PU

• Inter-Coded CU can be divide into square and non-square PU as long as one side is at least 4 pixels wide (note: no 4x4 PU)
Large Transforms

- HEVC supports 4x4, 8x8, 16x16, 32x32 integer transforms
  - Two types of 4x4 transforms (IDST-based for Intra, IDCT-based for Inter); IDCT-based transform for 8x8, 16x16, 32x32 block sizes
  - Integer transform avoids encoder-decoder mismatch and drift caused by slightly different floating point representations.
  - Parallel friendly matrix multiplication/partial butterfly implementation
  - Transform size signaled using Residual Quad Tree

- Achieves 5 to 10% increase in coding efficiency

- Increased complexity compared to H.264/AVC
  - 8x more computations per coefficient
  - 16x larger transpose memory

Intra Prediction

• H.264/AVC has 10 modes
  – angular (8 modes), DC, planar
• HEVC has 35 modes
  – angular (33 modes), DC, planar
• Angular prediction
  – Interpolate from reference pixels at locations based on angle
• DC
  – Constant value which is an average of neighboring pixels (reference samples)
• Planar
  – Average of horizontal and vertical prediction
Intra Prediction Modes

Removing Intra Artifacts (Pre-Processing)

• Reference Sample Smoothing
  – Smooth out neighboring pixels (i.e., reference samples) before using them for prediction
  – Reduce contouring artifacts caused by edges in the reference sample arrays
  – Two modes
    • Three-tap smoothing filter
    • Strong intra smoothing with corner reference pixels
  – Application of smoothing depends on PU size and prediction mode


Image source: M. Wien, TCSVT, July 2003
• Boundary Smoothing
  – Intra prediction may introduce discontinuities along block boundaries
  – Filter first prediction row and column with three-tap filter for DC prediction, and two-tap for horizontal and vertical prediction
Inter Prediction

- Motion vectors can have up to $\frac{1}{4}$ pixel accuracy (interpolation required)

- In H.264/AVC, luma uses 6-tap filter, and chroma uses bilinear filter
- In HEVC, luma uses 8/7-tap and chroma uses 4-tap
  - Different coefficients for $\frac{1}{4}$ and $\frac{1}{2}$ positions
- Restricted prediction on small PU sizes
Interpolation Filter

Require integer pixels *(highlighted in red)* to interpolate fractional pixels *(highlighted in blue)*

To interpolate \( N \times N \) pixels requires up to \((N+7)\times(N+7)\) reference pixels

Use 1-D filters *(order matters for greater than 8-bit video)*
Mode Coding

- Predict modes from neighbors to reduce syntax element bits
  - Intra Prediction Mode

- Advance Motion Vector Prediction (AMVP), Merge/Skip Mode
Merge Mode

(a) Moving Object

(b) Without Merge
   (many extra motion parameters)

(c) With Merge

# AMVP, Merge, Skip Mode

<table>
<thead>
<tr>
<th></th>
<th>AMVP</th>
<th>Merge</th>
<th>Skip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax elements</td>
<td>mvp_l0_flag, mvp_l1_flag</td>
<td>merge_flag, merge_idx</td>
<td>cu_skip_flag, merge_idx</td>
</tr>
<tr>
<td>Use of neighbors candidates</td>
<td>Predict motion vector</td>
<td>Copy motion data (motion vector, reference index, direction)</td>
<td>Copy motion data (motion vector, reference index, direction); no residual</td>
</tr>
<tr>
<td>Number of Candidates</td>
<td>Up to 2</td>
<td>Up to 5 (signaled in slice header)</td>
<td></td>
</tr>
<tr>
<td>Spatial</td>
<td>Up to 2 of 5 (scaling if reference index different)</td>
<td>Up to 4 of 5 (no scaling, only redundancy check)</td>
<td></td>
</tr>
<tr>
<td>Temporal</td>
<td>Up to 1 of 2 (if &lt; 2 spatial candidates)</td>
<td>Up to 1 of 2 (always added to list if available)</td>
<td></td>
</tr>
<tr>
<td>Additional</td>
<td>Zero motion vector (if &lt; 2 spatial or temp candidates)</td>
<td>Bi-predictive candidates and zero motion vector</td>
<td></td>
</tr>
</tbody>
</table>
In-loop Filtering: Deblocking Filter

- Removes blocking artifacts due to block based processing
  - Computationally intensive in H.264/AVC

In H.264/AVC, performed on every 4x4 block edge
- Each macroblock has 128 pixel edges, 32 edge calculations
- Each 4x4 depends on neighboring 4x4

In HEVC, performed on every 8x8 block edge
- Each 16x16 CTU has 64 pixel edges, 8 edge calculations
- All 8x8 are independent (can be processed in parallel)
In-loop Filtering: Sample Adaptive Offset (SAO)

- Filter to address local discontinuities
  - Edge Offset and Band Offset

- Check neighbors in one of 4 directions (0, 90, 135, 45 degrees)

- Based on the values of the neighbors, apply one of 4 offsets
In-loop Filtering: Sample Adaptive Offset (SAO)

With SAO

Without SAO

Entropy Coding

- Lossless compression of syntax elements
- HEVC uses Context Adaptive Binary Arithmetic Coding (CABAC)
  - 10 to 15% higher coding efficiency compared to CAVLC

CABAC Throughput Improvements

- Reduce total number of bins
- Reduce context coded bins
- Reduce context dependencies
- **Grouping bypass bins**
- Reduce parsing dependencies
- Reduce memory requirements

Reduction in **worst case** bins for 16x16 pixels

<table>
<thead>
<tr>
<th></th>
<th>Total bins</th>
<th>Context bins</th>
<th>Bypass bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264/AVC</td>
<td>20861</td>
<td>7805</td>
<td>13056</td>
</tr>
<tr>
<td>HEVC</td>
<td>14301</td>
<td>884</td>
<td>13417</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.5x</td>
<td>9x</td>
<td>1x</td>
</tr>
</tbody>
</table>

- 3x reduction in context memory
- 20x reduction in line buffer for context selection

High Level Parallel Tools (Multi-Core)

Slices (also in H.264/AVC)

Tiles

Wavefront Parallel Processing (Interleaved Entropy Slices*)

Additional Modes

- For wireless display and cloud computing, screen content coding should be considered.
- Screen content typically has more edges.
- Lossless:
  - Bypass transform, quantization and in-loop filters.
- Transform Skip:
  - Bypass transform, but continue to perform quantization and in-loop filters.
- I_PCM:
  - Signal raw pixels.
Profiles, Levels, Tiers

- Profile defines set of tools for different applications
  - Main, Main 10, Main Still Picture
  - 8-bits/sample → 16.78 million colors
  - 10-bits/sample → 1.07 billion colors

- Level defines the maximum supported resolution and frame rate
  - e.g. Level 4.0, 1920x1080 @ 32 fps
  - Level 5.0, 4096x2160 @ 30 fps

- Bit-rates defined by level and tier
  - Main and High (professional)
Main Still Picture (Intra Coding Only)

- HEVC also provides improved compression for still images

<table>
<thead>
<tr>
<th></th>
<th>BD-Rate Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264/AVC (intra only)</td>
<td>15.8%</td>
</tr>
<tr>
<td>JPEG 2000</td>
<td>22.6%</td>
</tr>
<tr>
<td>JPEG XR</td>
<td>30.0%</td>
</tr>
<tr>
<td>Web P</td>
<td>31.0%</td>
</tr>
<tr>
<td>JPEG</td>
<td>43.0%</td>
</tr>
</tbody>
</table>

Part III: Video Codec Implementations
Decoder Design Considerations

• Function
  – Mapping of bitstream to pixels fixed by the standard

• Implementation Requirements
  – *Conformance*: Support all tools for a given profile in the standard
  – *Throughput*: Real-time processing for video playback; **level** specifies pixel-rate and bit-rate
Encoder Design Considerations (1)

• Function
  – Mapping of pixels to standard compliant bitstream
  – Flexibility of selecting which set of encoding tools to use and how to use them (e.g. how to search for best compression mode)

Encoder

10101011
bitstream
at specified
bit-rate or
compression
ratio

pixels at
specified
pixel-rate for
real-time
applications
Encoder Design Considerations (2)

• Implementation Requirements
  – Conformance: Must generate a bitstream that is decodable by a standard compliant decoder (for a given profile)
  – Throughput: For real-time applications, need to meet pixel-rate requirements; can be done off-line for storage applications
  – Bit-rate/Compression Ratio: For given application, must meet minimum compression requirements
  – Compression ratio vs. Complexity: Find compression mode that meets compression requirements under complexity constraint

Decoder design requires architecture innovations, while encoder design requires both algorithm and architecture innovations
## Multimedia Platforms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flexibility</strong></td>
<td>High</td>
<td>High</td>
<td>Med/High</td>
<td>Med</td>
<td>Med</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Development Cost</strong></td>
<td>Low</td>
<td>Low</td>
<td>Low/Med</td>
<td>Med</td>
<td>Med</td>
<td>High</td>
</tr>
<tr>
<td><strong>Speed/ Throughput</strong></td>
<td>Low/Med</td>
<td>Low</td>
<td>Med</td>
<td>Med</td>
<td>Med</td>
<td>High</td>
</tr>
<tr>
<td><strong>Power Consumption</strong></td>
<td>High</td>
<td>Med</td>
<td>High</td>
<td>Med</td>
<td>Med</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Examples of HEVC implementations

Implementation Requirements

• Throughput
  – Achieve target pixel-rate and bit-rate for real-time applications
  – Reduce latency of bits to pixels and pixels to bits for interactive applications
  – **Techniques:** parallelism, pipelining, eliminate stalls

• Energy and Power Consumption
  – Minimize energy consumption to extend battery life for portable devices
  – Minimize power consumption to reduce heat dissipation
  – **Techniques:** voltage scaling, frequency scaling, power gating, number of ops

• Platform Cost
  – Reduce amount of data to be stored in memory and amount of logic (e.g. gates in ASIC, number of cores for processors) to reduce size of chip
  – Reduce bandwidth requirements such as reads/writes from memory to reduce demands on off-chip components
  – **Techniques:** shared computations, on-the-fly processing, caching
ARMv7 1.3GHz (mobile processor) [Bossen, JCTVC-K0327, 2012]
  - Dual core, but decoding on single thread (other thread for display)
  - 1080p @ 24 fps at 2Mbps (16 picture buffer to average workload)

Intel i7 Core 2.6 GHz (desktop processor) [Bossen et al., TCSVT, 2012]
  - Single core, single thread
  - 1080p @ 60 fps at 7Mbps

Multi-thread Intel Core i7 2.7 GHz [Suzuki et al., JCTVC-L0098, 2013]
  - 4 cores / 4 threads (parallel GOPs)
  - 3840x2160 @ 76 fps at 12Mbps [cropped 8K content]

Multi-thread Intel X5680 3.3 GHz [Chi et al., TCSVT, 2012]
  - 2x6 cores/12 threads (parallel Tiles, WPP)
  - 3840x2160 @ 24 fps at ~12Mbps (QP=37)
  - 3840x2160 @ 14 fps at ~170Mbps (QP=22)
Software HEVC Decoder

Workload for different modules

Hardware HEVC Decoder Architecture

Pipelining HEVC Decoder

- Variable-size pipelining to support a diverse set of CTU, CU, and PU sizes (select size to balance memory cost vs. data reuse)

**System level pipeline**
(between Inv. Transform, Prediction and In-Loop Filters)

**Prediction level pipeline**
(within Prediction module)

Decoupling Entropy Coding

• Workload of entropy decoding based on bit-rate (bin-rate), while rest of decoder depends on pixel-rate

• Use FIFO to absorb variations in workload
  – Higher FIFO depth results in less stalls due to averaging, but longer latency and higher memory cost

Intra Prediction

• Reference sample processing
  – Reference pixel buffer to store neighboring pixels (padding when not available)
  – Apply smoothing filter on pixels depending on mode

• Feedback loop at TU granularity
  – Update reference pixel buffer accordingly

• Read samples from reference picture (typically stored in off-chip picture buffer)
  – Use cache to reduce off-chip memory bandwidth
• Interpolation pixels used a 2-D separable filter for fractional motion vectors
  – Multiple pixels can be interpolated in parallel (share input pixels)
• Smaller blocks have larger read overhead (for fractional mv)
  – $N \times N$ requires $(N+7) \times (N+7)$ pixel reads $\Rightarrow$ 4x4 inter-PU not supported in HEVC
MC Cache and Picture Buffer

- Minimize redundant reads from off-chip memory (DRAM)
- MC Cache design considerations
  - Sufficient throughput to support worst case PU
  - Detect redundant reads and handle latency of DRAM
- Store pixels in DRAM to minimize row changes (cycle overhead)
  - Avoid reading two rows from same bank for a given reference region

Inverse Transform

- Larger transform $\rightarrow$ More computation
  - Share coefficients across transform sizes and within transform to reduce area cost

Inverse Transform

- Larger transform → Larger transpose memory
  - Use SRAM rather than registers to reduce area cost
  - SRAM has limited read/write ports (requires careful mapping)

4 pixels/cycle throughput per 1-D transform

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Coding Standard</td>
<td>HEVC (HM4)</td>
</tr>
<tr>
<td>Technology</td>
<td>TSMC 40-nm</td>
</tr>
<tr>
<td>Core Area</td>
<td>1.33 x 1.33 mm</td>
</tr>
<tr>
<td>Gate Count</td>
<td>715k</td>
</tr>
<tr>
<td>On-Chip Memory (SRAM)</td>
<td>124 kB</td>
</tr>
<tr>
<td>Resolution / Frame Rate</td>
<td>4kx2k @ 30fps (3840x2160)</td>
</tr>
<tr>
<td>Frequency</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Core Voltage</td>
<td>0.9 V</td>
</tr>
<tr>
<td>Power</td>
<td>76 mW</td>
</tr>
</tbody>
</table>

Area Breakdown

Logic [kgates]
- MC cache: 126
- Deblock: 49.9
- Entropy Decoder: 94.5
- Inverse Transform: 121.1
- Memory Interface Arbiter: 13.7
- Others: 42
- RegFiles: 75.5
- Others: 42

Memory (SRAM) [kbits]
- Pipeline Buffers: 447.3
- Line Buffers: 337
- MC-related SRAM: 200.4
- Others: 32.8

The following is a power breakdown for HEVC (High Efficiency Video Coding) decoder hardware architecture:

- **Prediction**: 23%
- **MC Cache**: 26%
- **Inverse Transform**: 17%
- **Pipeline Buffers**: 10%
- **Line Buffers**: 2%
- **Entropy Decoder**: 3%
- **Memory Interface Arbiter**: 2%
- **Others**: 13%

Hardware vs. Software

Hardware (power)

- Prediction: 23%
- Deblocking: 3%
- MC Cache: 26%
- Inverse Transform: 17%
- Others: 13%
- Pipeline Buffers: 10%
- Line Buffers: 2%
- Entropy Decoder: 3%
- Memory Interface Arbiter: 2%

Software (cycles)

Random Access (ARM)

- Motion compensation: 43%
- Entropy decoding: 24%
- Intra prediction: 6%
- Inv. quant. & transform: 4%
- SAO filter: 4%
- Rest: 2%
# ASIC Decoder Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Max Specification</strong></td>
<td>3840x2160 @30fps</td>
<td>7680x4320 @60fps</td>
<td>4096x2160 @24fps</td>
<td>1920x1080 @30fps</td>
</tr>
<tr>
<td><strong>Gate Count</strong></td>
<td>715K</td>
<td>1338K</td>
<td>414K</td>
<td>160K</td>
</tr>
<tr>
<td><strong>On-Chip SRAM</strong></td>
<td>124KB</td>
<td>80KB</td>
<td>9KB</td>
<td>5KB</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td>40nm/0.9V</td>
<td>65nm/1.2V</td>
<td>90nm/1.0V</td>
<td>0.18µm/1.8V</td>
</tr>
<tr>
<td><strong>Normalized Core Power</strong></td>
<td>0.31nJ/pixel</td>
<td>0.21nJ/pixel</td>
<td>0.28nJ/pixel</td>
<td>5.11nJ/pixel</td>
</tr>
<tr>
<td><strong>Normalized DRAM Power</strong></td>
<td>0.88nJ/pixel**</td>
<td>1.27nJ/pixel</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Normalized System Power</strong></td>
<td>1.19nJ/pixel***</td>
<td>1.48nJ/pixel</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>DRAM Configuration</strong></td>
<td>32b DDR3</td>
<td>64b DDR2</td>
<td>N/A</td>
<td>32b DDR + 32b SDR</td>
</tr>
</tbody>
</table>

* Power for max specification
** Modeled by [5]
*** System Power = Core Power + DRAM Power

---

Decoder Power Comparison

TSMC 40nm, 0.9V Ultra-HD 4K @ 30 fps

- H.264/AVC Decoder (51mW)
P.K. Tsung et al. (NTU), ISSCC 2011

- H.265/HEVC Decoder (76mW)
C.T. Huang et al. (MIT), ISSCC 2013

- H.264/AVC Decoder (2mW)
Sze et al. (MIT), JSSC 2009

Energy per pixel (nJ)

Year

Low Power Approaches

• Operate at voltage near minimum energy point
• Utilize parallelism and pipelining to achieve performance
• Adaptive/Dynamic voltage frequency scaling
• Optimize access patterns to reduce memory power

Encoder Decisions

- Encoder must search for mode that gives the “best” compression. Some of the key decisions include:
  - CU and PU size
  - Inter or Intra CU
  - Motion Vector
  - Intra Prediction Mode

- “Best” compression is defined using a rate-distortion cost:

\[ D + \lambda \cdot R \]

- Perform rate-distortion optimization (RDO)

- where:
  - \( D \) is the distortion between the original and the compressed image (a measure of the visual quality of the compression)
  - \( R \) is a measure of the number of bits required to signal the compressed image
  - \( \lambda \) is the Lagrangian multiplier that weighs the distortion and rate costs
Full vs. Fast RDO

• Full RDO
  – Distortion based on sum of squared differences (SSD), includes quantization
  – Rate based on entropy coded bits of prediction info and quantized coefficients

• Fast RDO
  – Distortion approximation based on sum of absolute differences (SAD) or sum of absolute transformed differences (SATD)
  – Rate approximation based on prediction info bits (intra mode or motion vector); Can include number of non-zero coefficients to predict coefficient bits

RDO Flow in HM

CU and PU decisions

- The encoder must decide to how best divide a CTU into CU, and how to divide the CUs into PUs (based on full RDO in HM)

- For CTU of 64x64
  - CU options: 64x64, 32x32, 16x16, 8x8

- For Inter-coded CU
  - PU options

- For Intra-coded CU
  - PU options
Motion Estimation

• Search for block in reference frame(s) to predict current block with least rate-distortion cost
  – Signal block in previous frame using a motion vector
• Typically most computationally intensive function in encoder

Search algorithm considerations
1. Number of candidates
   – Number of computations
   – Number of memory accesses
2. Off-chip bandwidth
3. On-chip bandwidth
• Integer pixel motion estimation
  – Rate is the bits required to transmit the motion data
    (including impact of motion predictor)
  – Distortion is calculated from the SAD of original and motion-
    compensated prediction (subsampled when block size > 8)

\[
\arg\min_{MV, REF} \sum_{i,j} |\text{Diff}(i, j)| + \lambda \cdot R(MV, REF)
\]

where

– MV = motion vector (include
  impact of advanced mv predictor)
– REF = reference index

K. McCann et al “High Efficiency Video Coding (HEVC) Test Model 14 (HM 14) Encoder
Description,” JCTVC-P1002, 2014
Motion Estimation in HM

- Integer pixel motion estimation
  - Search Strategy
  1. Search center is motion vector predictor
  2. Diamond search around center (search range = 64 \rightarrow 7 \text{ steps} [1, 2, 4.. 64]); early termination if best candidate doesn’t change in 3 steps.
  3. If best candidate > 5 pixels away from search center, do raster scan search (5 pixel steps).
  4. Perform diamond search around best candidate from step 2 or 3. If new best candidate found repeat 4.

Reference
Motion Estimation in HM

• Half pixel motion estimation
  – Rate is the bits required to transmit the motion data (including impact of motion predictor)
  – Distortion is calculated from SATD
    • Block-wise 4x4 or 8x8 Hadamard transform on difference between original and motion-compensated prediction, and sum absolute coefficients
  – Search 8 points surrounding best integer motion vector

• Quarter pixel motion estimation
  – Same rate and distortion calculation as half pixel
  – Search 8 points surrounding best half pixel motion vector

• Also do search for merge/skip candidates

Multiple Searches in Parallel

Parallel Motion Estimation

- Perform motion estimation for each PU in inter-coded CU
- Process CUs in parallel to increase throughput
  - Share search pixels across engines to reduce memory bandwidth by 8x

Reduce Number of PUs Processed

<table>
<thead>
<tr>
<th>Configuration #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>64x64</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>64x32</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>32x64</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>32x32</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>32x16</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>16x32</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>16x16</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>16x8</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>8x16</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>8x8</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>8x4</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4x8</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4x4</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Ref. Buffer Size (KB): 680, 565, 248, 439, 208, 234, 163, 356, 170, 201, 115
On-Chip BW (GB/s): 1581, 429, 209, 121, 59, 32.5, 17.3, 409, 205, 351, 192
Off-Chip BW (GB/s): 159, 69, 30.2, 27.4, 12.7, 8.5, 5.1, 64, 28.7, 49.1, 25.1
Bit-Rate Increase (%): 0, 2, 3, 12, 12, 34, 34, 3, 4, 7, 11

Trade-off between coding efficiency (BD-rate) and complexity (area cost) for different number of inter predicted partitions units

Smallest slope provides best trade-off: #3

Only Square PUs

Motion Estimation with CU

- In HM, motion estimation done serially for PU within CU to get AMVP for accurate rate estimate

Can’t process PU1 and PU2 in parallel
Parallel Motion Estimation

• HEVC has “Parallel Motion Estimation” feature to turn off dependency within an Motion Estimation Region (MER)
  – PU within region cannot use data from other PU in region
  – All PUs in region can be processed in parallel at encoder

Can process PU1 and PU2 in parallel

CTU Processing Order

- In HM, CTU processed in raster scan order
- Change CTU Processing Order to reduce reads from picture buffer (off-chip memory bandwidth) due to increased data locality
- Requires frame decoupling with entropy encoder (as entropy encoder must generate bitstream in raster scan order to be standard compliant)

Additional Complexity Reductions

• Bottoms up approach
  – Derive distortion cost for PU from sub-PUs (e.g. compute distortion of 16x16 PU from four 8x8 PU)
  – Requires storage of SAD sub-PUs

• Reduce bit-width for distortion calculation

• Use bilinear interpolation for fractional motion estimation

\[
\text{SAD16}(X) = \text{SAD8}(A) + \text{SAD8}(B) + \text{SAD8}(C) + \text{SAD8}(D)
\]
Intra Prediction Search in HM

• Rough mode decision: select N best mode out of 35
  – N equals 8 for 4x4, 8x8
  – N equals 4 for 16x16, 32x32, 64x64
  – Hadamard Cost Ranking (SATD distortion and mode bits for rate)

• Determine three Most Probable Modes (MPM)
  – Spatial neighbors to the left (A) and above (B)
  – If neighbors not available or redundant (A=B), use DC, Planar, vertical or adjacent angles (+/- 1)

• Decide between rough mode + MPM candidates
  – Full RDO (SSD for distortion and mode + coefficient bits for rate)

Additional Complexity Reduction

• To reduce search space, use coarse search with angular prediction, then refinement around coarse angles

• Skip 64x64 PU size
  – Since max TU is 32x32, prediction done at 32x32; thus only benefit of 64x64 intra-PU is signaling

• To increase throughput, use original pixels for intra prediction (rather than reconstructed pixels) to avoid dependence on reconstruction feedback loop

Above techniques have cumulative coding loss of 1%

Hardware-Friendly RDO Pipeline

Only do full RDO on best Inter and Intra mode for each CU-depth (6% coding loss)

**Hardware HEVC Encoder**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Video Coding Standard</strong></td>
<td>HEVC (WD4)</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td>TSMC 28-nm HPM</td>
</tr>
<tr>
<td><strong>Core Area</strong></td>
<td>5x5mm²</td>
</tr>
<tr>
<td><strong>Gate Count</strong></td>
<td>8350k</td>
</tr>
<tr>
<td><strong>On-Chip Memory (SRAM)</strong></td>
<td>7.14 MB</td>
</tr>
<tr>
<td><strong>Resolution / Frame Rate</strong></td>
<td>8192x4320@30fps</td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td>312 MHz</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>708 mW</td>
</tr>
</tbody>
</table>

### ASIC Encoder Comparison

<table>
<thead>
<tr>
<th></th>
<th>ISSCC’09[22]</th>
<th>VLSIC’12[6]</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>4096x2160@24fps</td>
<td>7680x4320@60fps</td>
<td>8192x4320@30fps</td>
</tr>
<tr>
<td>Throughput</td>
<td>212Mpixels/s</td>
<td>1991Mpixels/s</td>
<td>1062Mpixels/s</td>
</tr>
<tr>
<td>Standard</td>
<td>H.264 High @ Level 5.1</td>
<td>H.264 Intra</td>
<td>HEVC</td>
</tr>
<tr>
<td>Search Range</td>
<td>[-255,+255]/[-255,+255]</td>
<td>N/A</td>
<td>[-512,+511]/[-128,+127] (Predictor CENTERED)</td>
</tr>
<tr>
<td>Technology</td>
<td>TSMC 90nm</td>
<td>e-Shuttle 65nm</td>
<td>TSMC 28nm HPM</td>
</tr>
<tr>
<td>Core Size</td>
<td>3.95x2.90mm2</td>
<td>3.95x2.90mm2</td>
<td>5x5mm2</td>
</tr>
<tr>
<td>Gate Count</td>
<td>1732K</td>
<td>678.8K</td>
<td>8350K</td>
</tr>
<tr>
<td>Power</td>
<td>522mW@280MHz</td>
<td>139.9mW@280MHz</td>
<td>708mW@312MHz</td>
</tr>
</tbody>
</table>

S.-F. Tsai et al., "A 1062Mpixels/s 8192x4320p High Efficiency Video Coding (H.265) encoder chip," *2013 Symposium on VLSIC*, 2013
Part IV: Emerging applications and HEVC extensions
What’s Next

• More compression efficiency
  – Yes, in 5-10 years. Especially since video delivery is moving from traditional broadcast model to IP delivery and one-to-one streaming
  – Analogy: Public transport versus individual cars

• Other considerations have become important too:
  – Power consumption, complexity, throughput
  – Ability to support new functionalities, modalities etc.
• Need for supporting diverse clients with varying capabilities (resolution, computational power etc.)
• Immersive experience
  – Multiple cameras and at higher video resolutions (1080p ➔ 4K ➔ 8K)
  – Multiple displays, Bigger displays (1080p ➔ 4K ➔ 8K)

  – Free-viewpoint video, 360degree video, augmented reality, 3D movies

  – Demos
    • http://replay-technologies.com/
    • http://www.kolor.com/video

Image source: Cisco, Kolor
• Growing requirement to support mixed format content consisting of natural video + graphics/text
Scalable Video Coding
Supporting Diverse Clients - Simulcasting

Can we do better?
Scalable Video Coding

Temporal scalability

Spatial scalability

Single Bitstream

Quality (SNR) scalability

... 0110111 ...

Single Bitstream

MIT

Samsung
Spatial Scalability

- Layered coding
- Higher layers have higher spatial resolution when compared to lower layers
- Upper layers re-uses data from lower layers

Layer N+1 – 1280x960 (Enhancement layer)
Layer N – E.g. 640x480 (Base layer)

Figure source: T. Wiegand, JVT-W132 [1].
Temporal Scalability

IPPP coding

IBBP coding

Hierarchical P-frames

Hierarchical B-frames

• p, b – Non-reference frames
HEVC Scalable Extension (SHVC)

- SHVC: Scalable extension: Expected July 2014
- EL – Enhancement layer, BL – Base layer

```
  Base layer decoder
  \__________________________
     \                     \    
     \                   BL
  \__________________________
     \__________________________
     \                     \    
     \                   EL
  \__________________________
     \                     \    
  EL Frame buffer
  \__________________________
     \                     \    
  EL Bitstream

  Upsampler
  \__________________________
     \                     \    
     \                   BL
  \__________________________
     \__________________________
     \                     \    
     \                   EL
  \__________________________
     \__________________________
     \                     \    
     \                   BL
  \__________________________
     \__________________________
     \                     \    
  BL Frame buffer
  \__________________________
     \                     \    
  BL Bitstream

  BL decoded pictures
   \__________________________
       \                     \    
       \                   EL
  \__________________________
       \__________________________
       \                     \    
       \                   BL
  \__________________________
       \__________________________
       \                     \    
       \                   BL
  \__________________________
       \__________________________
       \                     \    
       \                   EL
  \__________________________
       \__________________________
       \                     \    
       \                   BL
  \__________________________
       \__________________________
       \                     \    
  EL decoded pictures
```
SHVC Performance

• 2x scalability (i.e. base layer is half the size of enhancement layer) compared to simulcast

<table>
<thead>
<tr>
<th>Coding configuration</th>
<th>BD-Rate savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Intra coding</td>
<td>23%</td>
</tr>
<tr>
<td>Random access (Hierarchical-B)</td>
<td>16%</td>
</tr>
</tbody>
</table>

• Quality (SNR) scalability compared to simulcast

<table>
<thead>
<tr>
<th>Coding configuration</th>
<th>BD-Rate savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Intra coding</td>
<td>28%</td>
</tr>
<tr>
<td>Random access (Hierarchical-B)</td>
<td>20%</td>
</tr>
</tbody>
</table>

Multiview Video Coding
Multiview Video Capture

Stereo, 3D video

360degree video

Free viewpoint video

Image source: Fuji, Kolor
Stereoscopic Video Coding

Camera modules

Left View

Stereo Video encoding

Stereo video bitstream

Right View

Stereo Video decoding

Left View

Right View

3D display

Image source: Samsung
Redundancy in Stereo Video

Left view

Right view
Multiview Video Coding – Picture Prediction Structures (1)

- Linear camera array
  
  S0 \[\rightarrow \text{I}_0 \rightarrow \text{b}_3 \rightarrow \text{B}_2 \rightarrow \text{b}_3 \rightarrow \text{B}_1 \rightarrow \text{b}_3 \rightarrow \text{B}_2 \rightarrow \text{b}_3 \rightarrow \text{I}_0 \rightarrow \text{b}_3 \rightarrow \text{B}_2 \rightarrow \text{b}_3\]

S1

S2

S3

S4

S5

S6

S7

Simulcast
Multiview Video Coding – Picture Prediction Structures (1)

- Linear camera array

![Diagram showing picture prediction structures for a linear camera array with anchor frames S0 to S7. The structure includes I0, P0, B0, B1, and B2 frames with prediction arrows between them.](image)

Interview prediction of anchor frames
Both anchor and non-anchor views predicted from other views
**HEVC Multiview Extension (MV-HEVC)**

- **MV-HEVC**: Multiview extension: Expected July 2014
- **View 0**: Left view, **View 1**: Right view

![Diagram of HEVC Multiview Extension](image_url)

- **View 0** Bitstream → **View 0 decoder** → **View 0 decoded pictures**
- **View 1** Bitstream → **View 1 decoder** → **View 1 decoded pictures**
- **View 0** Framebuffer
- **View 1** Framebuffer

3D display
Combined Scalable and Mutiview Extension of HEVC

- Applications of the combined scalable and multiview HEVC coding include:
  - Scalable stereoscopic video (e.g. 1080p stereo to the emerging 4K stereo),
  - Mixed resolution multiview coding
- H.264/AVC does not support combined scalable and multiview coding
- HEVC allows for combined scalable and multiview coding

Combined Scalable and Mutiview Extension of HEVC


### Table IV. ‘BL-D + EL-D’ BD-Rate (%) of RefIdx SHVC + MV-HEVC W.R.T MV-HEVC.

<table>
<thead>
<tr>
<th></th>
<th>2x</th>
<th></th>
<th></th>
<th>SNR</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Y</td>
<td>Cb</td>
<td>Cr</td>
<td>Y</td>
<td>Cb</td>
<td>Cr</td>
</tr>
<tr>
<td>AI</td>
<td>-19.5</td>
<td>-17.1</td>
<td>-17.5</td>
<td>-24.4</td>
<td>-22.3</td>
<td>-22.6</td>
</tr>
<tr>
<td>RA</td>
<td>-12.7</td>
<td>-5.0</td>
<td>-5.6</td>
<td>-16.4</td>
<td>-7.8</td>
<td>-8.9</td>
</tr>
<tr>
<td>LDP</td>
<td>-7.9</td>
<td>-0.1</td>
<td>-1.5</td>
<td>-9.0</td>
<td>-1.8</td>
<td>-3.0</td>
</tr>
</tbody>
</table>
MV-HEVC + Depth (3D-HTM)

- Standardization in on-going
MV-HEVC + Depth Encoding

• Views that are transmitted will be coded using MV-HEVC

• Expect additional 20% gain
MV-HEVC + Depth Decoding

- View decoding
- Depth decoding

View synthesis

Multiple views
Screen Content Video Coding
• Applications such as automotive infotainment, wireless displays, remote desktop, remote gaming, cloud computing etc. are becoming popular.

• Video in these applications often has mixed content consisting of natural video, text, graphics etc.
  
  – In text and graphics regions, patterns (e.g. text characters, icons, lines etc.) can repeat within a picture.
  
  – Also blocks with limited set of colors are possible.
Intra Block Copy

Bit-rate savings

<table>
<thead>
<tr>
<th></th>
<th>Intra</th>
<th>Random access</th>
<th>Low delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC RGB 444</td>
<td>27.0%</td>
<td>21.5%</td>
<td>17.0%</td>
</tr>
<tr>
<td>SC YUV 444</td>
<td>23.5%</td>
<td>20.2%</td>
<td>15.9%</td>
</tr>
</tbody>
</table>

Palette Coding

• Input video:
  – 8 bits per pixel, per color component
  – 4x4 block: $8 \times 3 \times 16 = 384$ bits

• Palette coding:
  – Color palette: 2 Colors in our example:
    $2 \times 24 = 48$ bits
  – Color index: 1 bit per pixel in our example: 16 bits
  – Total bits: 64 bits

• Note: This slide shows a very simple example for explaining purposes. Techniques being evaluated currently can use more colors in palette and more bits for color index.
**HEVC Screen Content coding**

- HEVC Screen content coding activity
  - Started in April 2014
  - Expected completion early-mid 2015
- Key tools being studied
  - Intra Block Copy with extended search area
  - Palette based coding
Summary

• Video content continues to impose a severe burden on today’s global networks
  – Rapid growth in the usage and diversity of video applications and services
  – Increasing popularity of HD video and emergence of beyond-HD formats accompanied by stereo and multi-view content

• HEVC is the latest video coding standard, which gives 50% improvement in coding efficiency, and is expected to support video applications for the next decade.

• In addition to improving coding efficiency, implementation challenges were also considered to maximize processing speed and minimize hardware cost.
References


• J. Ohm et al., "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," IEEE Transactions on Circuits and Systems for Video Technology, 2012
HEVC Book

- Introduction
- High-Level Syntax in HEVC
- Block Structures and Parallelism Features in HEVC
- Intra-Picture Prediction in HEVC
- Inter-Picture Prediction in HEVC
- Transform and Quantization in HEVC
- In-Loop Filters in HEVC
- Entropy Coding in HEVC
- Compression Performance Analysis in HEVC
- Decoder Hardware Architecture in HEVC
- Encoder Hardware Architecture in HEVC

http://www.springer.com/engineering/signals/book/978-3-319-06894-7
HEVC Book

The book serves the video engineering community by:

• Providing video application developers an invaluable reference to the latest video standard, High Efficiency Video Coding (HEVC);

• Serving as a companion reference that is complementary to the HEVC standards document produced by the JCT-VC – a joint team of ITU-T VCEG and ISO/IEC MPEG;

• Including in-depth discussion of algorithms and architectures for HEVC by some of the key video experts who have been directly involved in developing and deploying the standard;

• Giving insight into the reasoning behind the development of the HEVC feature set, which will aid in understanding the standard and how to use it.