

# JVET-H0023

## Implementation and design aspects of xvc



- ♦ Introduction
  - ♦ Implementation
  - ♦ Restriction flags
  - ♦ Complexity
  - ♦ Binary splits in xvc
  - ♦ Caching of mode information
  - ♦ Performance
- 
- ♦ Real-time decoder demo on mobile device



- ♦ Informational contribution
- ♦ Independent implementation of technology from HEVC and JVET, in particular QTBT
- ♦ Source code available for research and standardization
- ♦ The contribution highlights some implementation aspects which may be relevant for JVET activities:
  - ♦ Source code language, style guide and API
  - ♦ Restriction flags evaluated during run-time
  - ♦ Decoder complexity
- ♦ The contribution also describes some of the differences of the QTBT implementation in xvc and the QTBT implementation in JVET



- Built from scratch in C++11 following the Google C++ Style Guide
- C++11 offers advantages such as improved type management, improved pointer handling, lambda expressions, threading, and useful keywords such as *auto*
- Implemented with a strict separation between libraries and apps with an API in C
- Supports 8, 10 and 12 bits
- Supports Monochrome, 4:2:0, 4:2:2 and 4:4:4
- ~20,000 lines of code (HM has ~60,000 and JEM ~120,000)
- Tested in FFmpeg, VLC and ExoPlayer



- Each tool and feature in xvc is contained in conditional clauses making it easy to disable individual tools – even during run-time
- In the first version of xvc there are 62 "restriction flags" to turn off different tools
- The flags are signalled in sequence header information
- Run-time conditionals ensure that the whole codebase is exercised during compilation
- Only one macro-defined constant: `XVC_HIGH_BITDEPTH`



- ♦ Decoder carefully designed to run fast
- ♦ Frame level parallelism
- ♦ SIMD operations for interpolation filtering for x86 and ARM
- ♦ Increased battery consumption on mobile devices
- ♦ Example 720p
  - ♦ H.264 hardware decoder (800 kbps): 8 h 45 min
  - ♦ xvc software decoder (400 kbps): 7 h



# Binary splits in xvc

- ♦ Very similar to QTBT in JEM
- ♦ A few differences:
  - ♦ The default binary split depth in xvc is 2 instead of 3. Binary split depth of 3 only used in “placebo”.
  - ♦ 5 of the QTBT shortcuts/speedups are not performed (corresponding to LCUFast, JVET\_C0024\_AMAX\_BT, JVET\_C0024\_PBINTRA\_FAST and 2 speed-ups without any defines related to early termination based on horizontal split and re-using motion vectors from previous blocks).
  - ♦ The “do not evaluate quad-split if the best evaluated mode so far had no further splits”-speedup is only performed if the best binary split had no further splits.



# Caching of mode information

- Mode information is stored in QTBT to speed up the process when the same coding unit is evaluated again (due to a different series of splits).
- In xvc, only 320 bytes is used to store modes
- In JEM, 110592 bytes is used to store modes
- In xvc, mode information is stored only for the split combinations that are at risk of occurring again instead of storing mode information for all blocks.





- Around 4% bitrate reduction compared to HM using JCT-VC CTC

	Random Access Configuration		
	Y	U	V
Class A	-1.6%	-1.0%	-2.4%
Class B	-2.3%	2.0%	0.0%
Class C	-6.2%	2.0%	2.6%
Class D	-6.3%	0.0%	-0.4%
Overall	-4.0%	0.8%	0.0%
Enc Time	+4%		
Dec Time	-49%		

- Encoding speed similar to HM
- Decoding speed without threading twice as fast as HM



# Thank you!

