الجامعة المصرية اليابانية للعلوم والتكنولوجيا
E-JUST
Egypt-Japan University of Science and Technology
エジプト日本科学技術大学

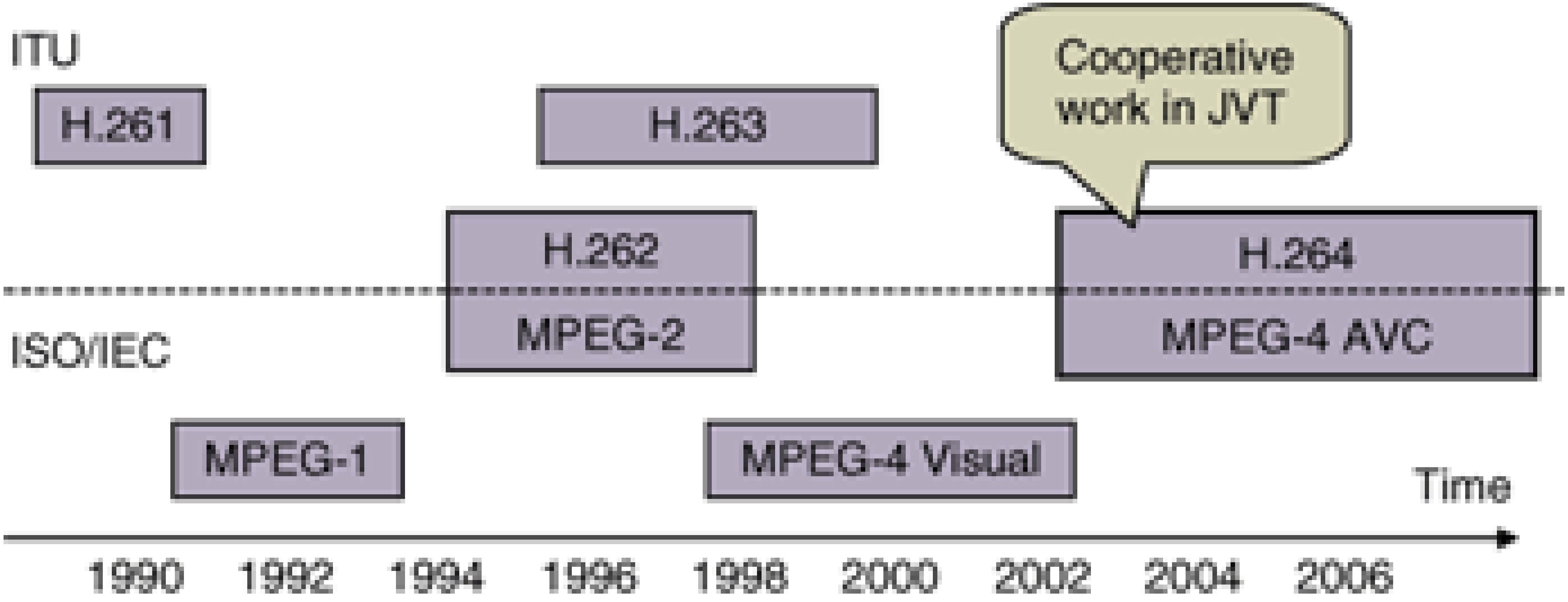
High Efficiency Video Coding (HEVC)

Mohammed Sharaf Sayed
mohammed.sayed@ejust.edu.eg

Outline

- Introduction
- Improvements in coding efficiency
  - Coding Tree Structure
  - Inter Prediction
  - Intra Prediction
  - Motion Vector coding
  - In-loop filters
- Parallel Processing Tools
  - Slices
  - Tiles
  - Wavefront parallel processing (WPP)
- HEVC Coding Complexity

Video Coding Standards



Motivations

- More than 50% of the current network traffic is video
- Popularity of HD videos
- Beyond HD format (4k x 2k , 8k x 4k)
- High resolution 3D or multiview

HEVC

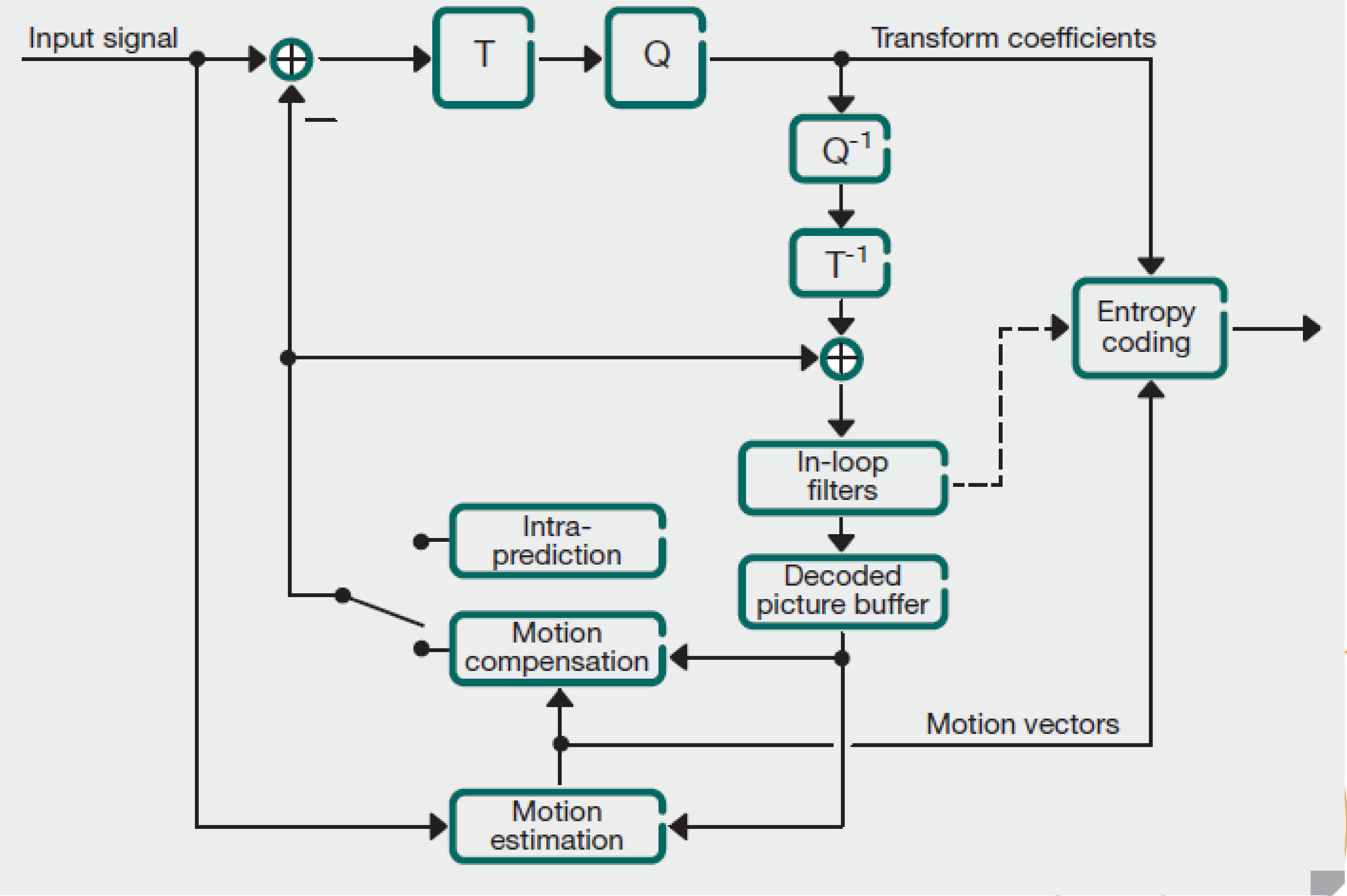
- 50% bit-rate reduction → **Same bandwidth, double the data !**
- HEVC is suitable for high resolution videos

HEVC

- **H.265** or **MPEG-H Part2**: The new joint video coding standard
- First edition finalized on Jan 2013
- Additional work planned to extend the standard …
  - 3D and multiview → expected in 2014/2015
  - Scalable extensions(SVC) → expected in July 2014
  - Range extensions (several color formats, increased bit depth)

HEVC

FIGURE 1 Simplified HEVC encoder diagram



HEVC

- *Mainly focus on:*
    - **Doubling the coding efficiency**
    - **Parallel processing architectures**

Improvements in coding efficiency

- Coding Tree Structure
- Inter Prediction
- Intra Prediction
- Motion Vector coding
- In-loop filters

Coding Tree Structure

*Coding Tree Units (**CTU**) instead of Macro Blocks (**MB**)*

→ Size of CTU can be larger than traditional MB

Coding Tree Structure

- Coding tree blocks (CTBs):
  - Picture is partitioned into CTBs, each luma CTB covers a rectangular picture area of NxN samples (N=16, 32, 64)
- Coding Tree Units (CTU):
  - The luma CTB and the two chroma CTBs, together with the associated syntax, form a CTU

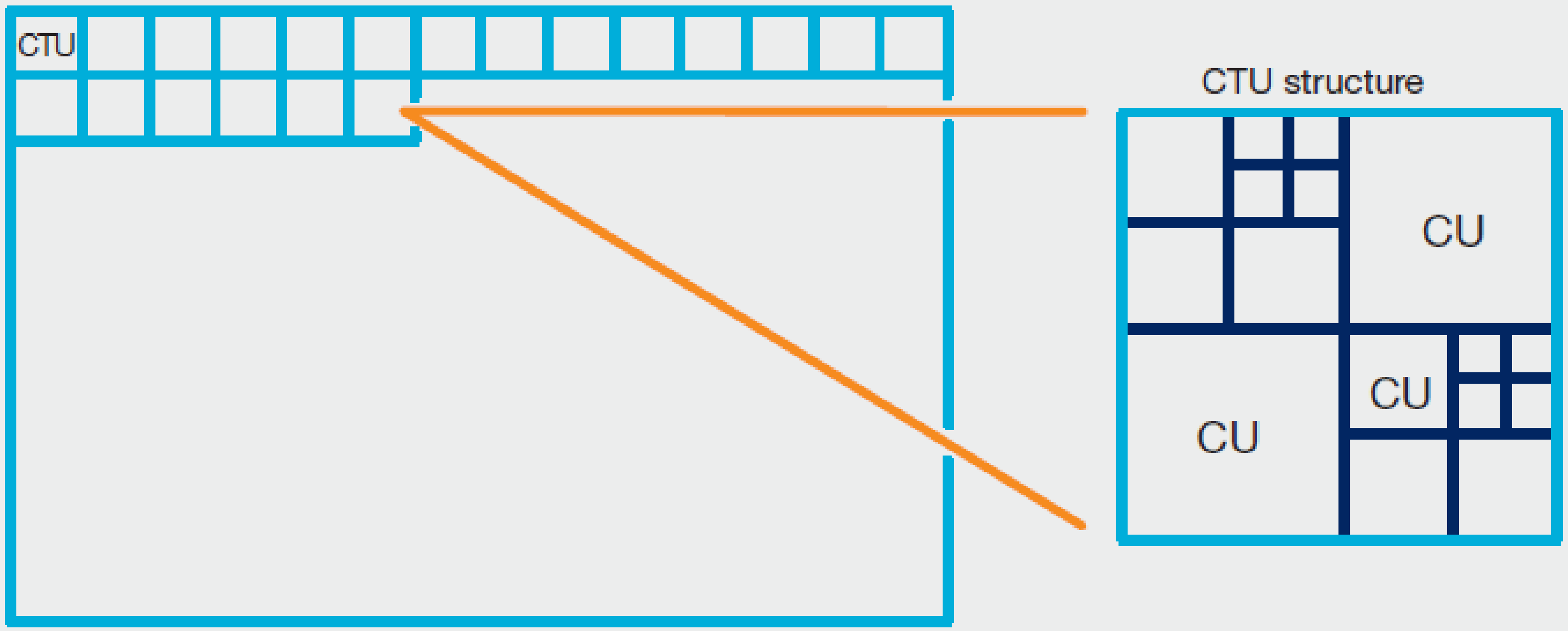
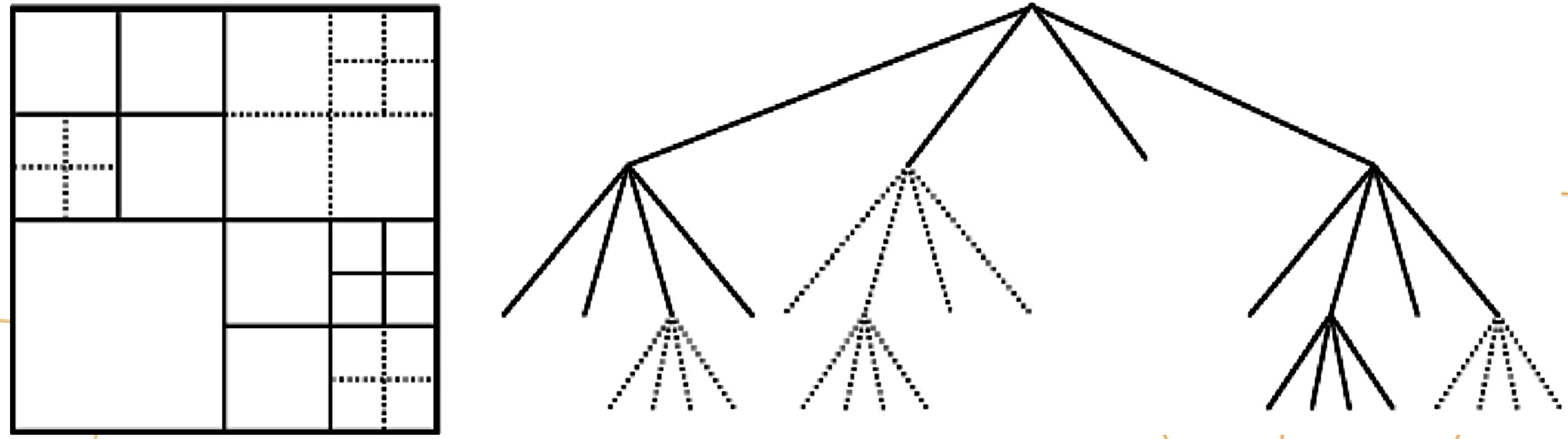


FIGURE 2 Example of the coding-tree unit structure in HEVC

Coding Tree Structure

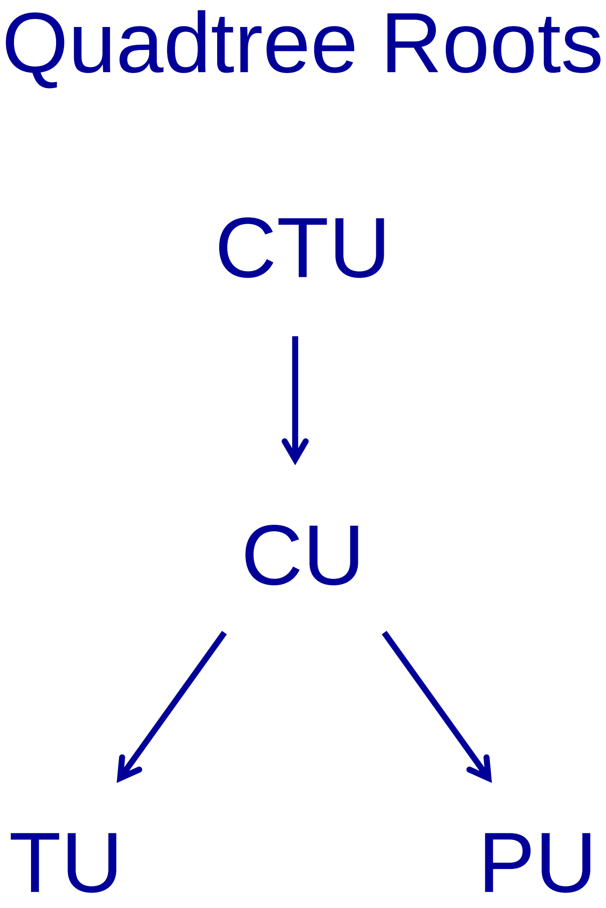
- Coding Blocks (CB):
  - CTB can be partitioned into multiple CBs
  - The syntax in CTU specifies the size and positions
- Coding Units (CU):
  - The luma CB and the two chroma CBs, with the associated syntax, form a CU

$8x8 \le$ CB size $\le$ CTB size



Coding Tree Structure

- The decision whether to code a picture area using **inter** or **intra** prediction is made at the **CU level**



Quadtree Roots

CTU

CU

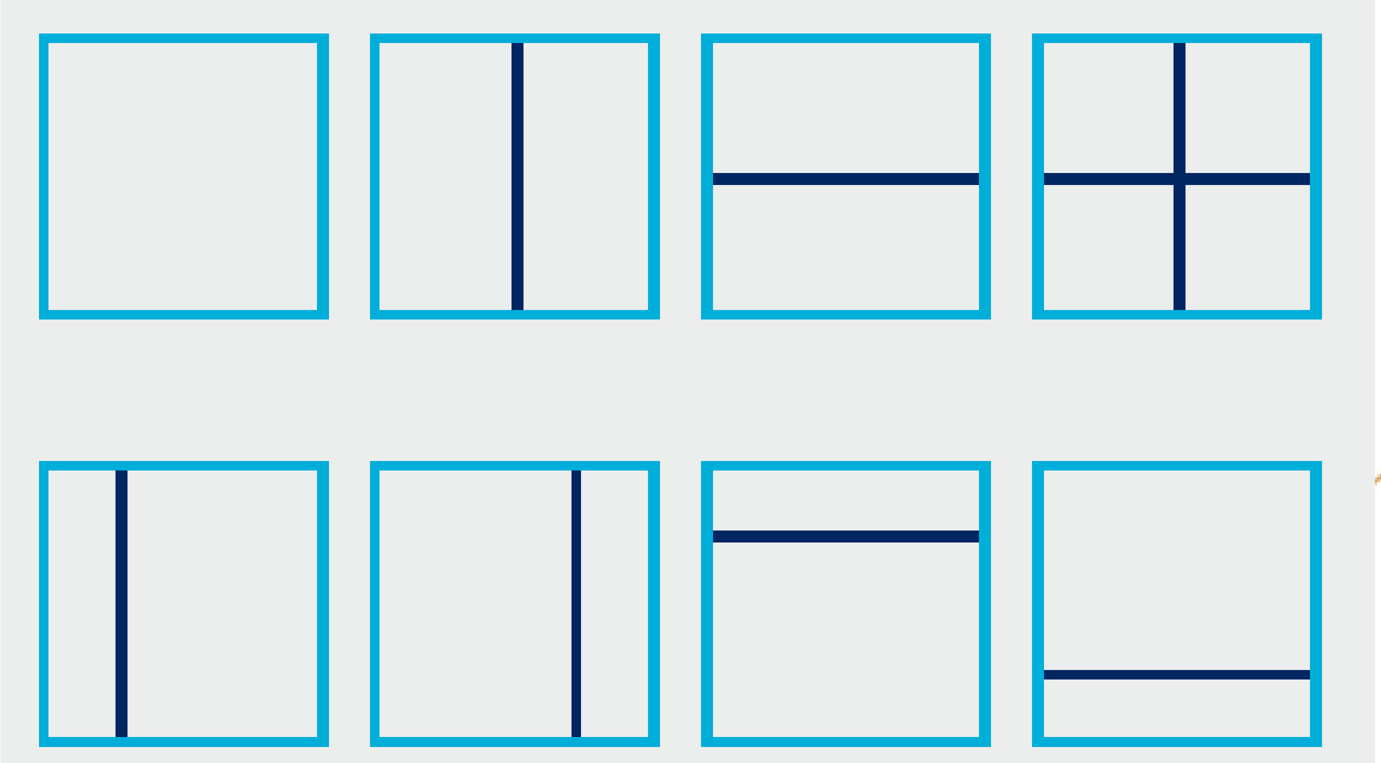
TU

PU

Coding Tree Structure

- Prediction Blocks (PB):
  - Depending on the prediction type CBs can be splitted to PBs.
  - Each PB contains one motion vector (if in a P slice).
- Prediction Unit (PU):
  - Again, the luma and chroma PBs, with the associated syntax, form a PU

4x4 ≤ PB size ≤ CB size



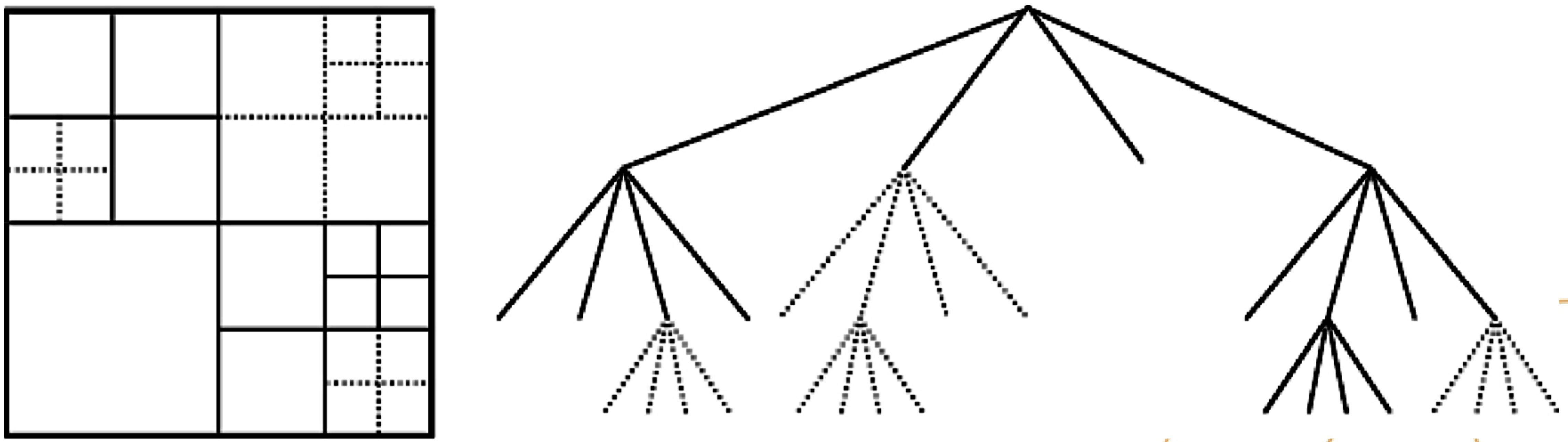
Asymmetric motion partitions are shown in the bottom row.

Coding Tree Structure

- Transform Blocks (TB):

TB size ≤ CB size

  - Blocks for applying DCT transform: 4x4 ≤ size ≤ 32x32
  - Integer transform for 4x4 intra blocks.
- Transform Unit (TU):
  - Again, the luma and chroma TBs, with the associated syntax, form a TU



TB can span across multiple PBs

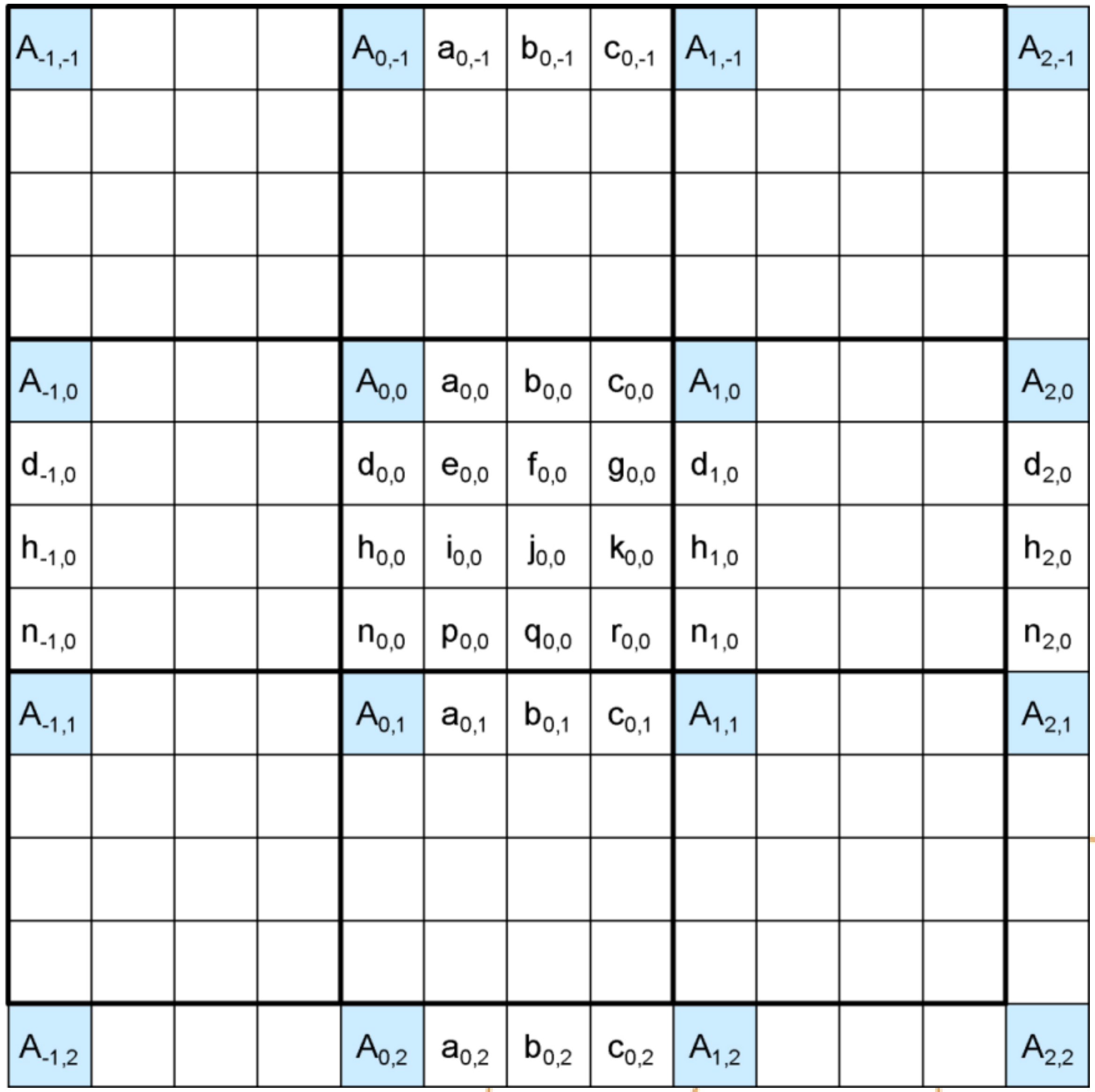
Coding Tree Structure

- **Large CTB** sizes are even more important for coding efficiency when **higher resolution** video are used
- Large CTB sizes increase **coding efficiency** while also reducing **decoding time**.
- HEVC supports **variable PB sizes** from **64x64 to 4x4** samples**.**

Inter Prediction

Fractional sample:

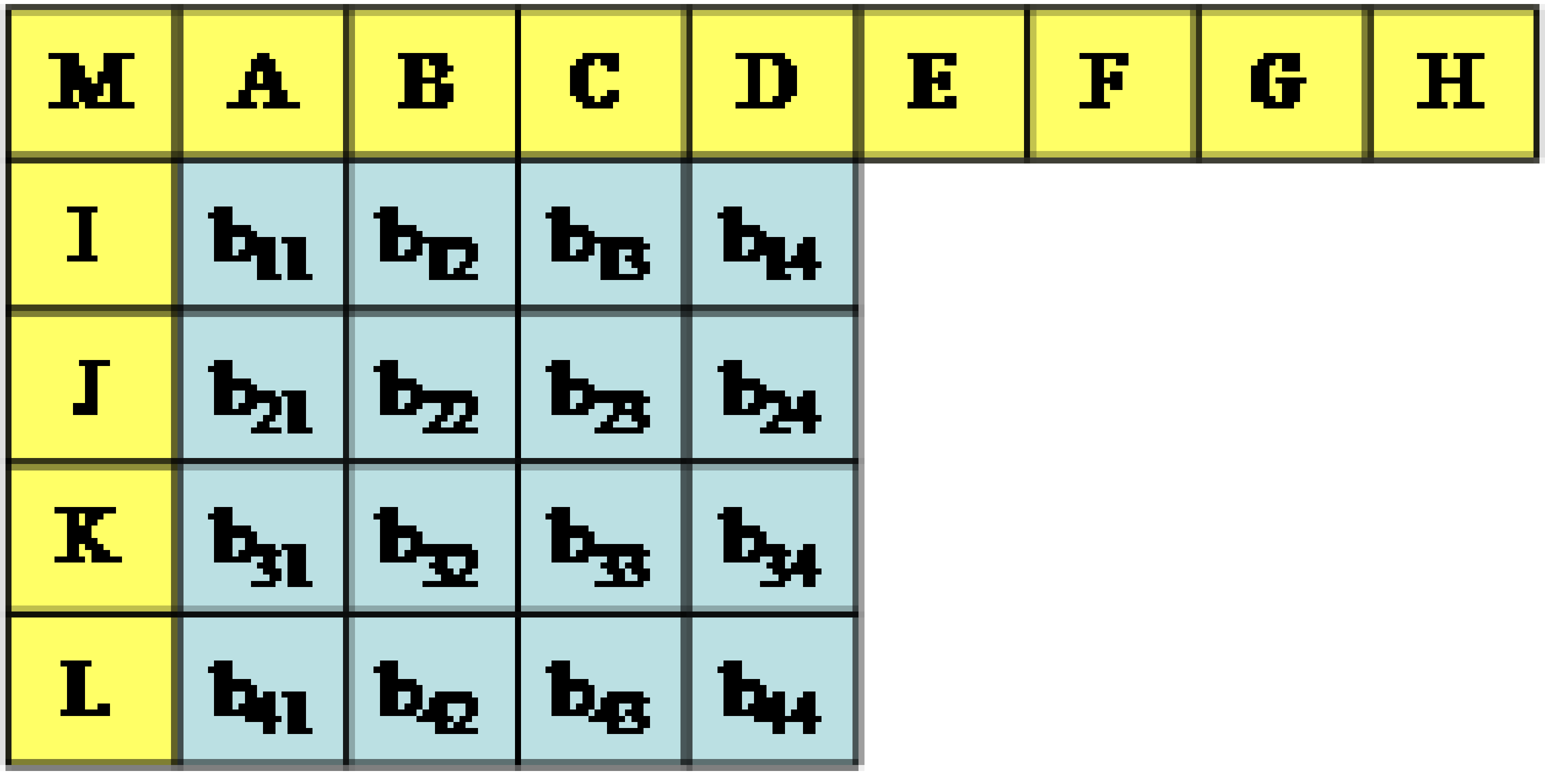
- ➢8 tap filter for half-sample
- ➢7 tap filter for quarter-sample
- ➢4 tap for chroma one-eight-sample



$A_{-1,-1}$				$A_{0,-1}$	$a_{0,-1}$	$b_{0,-1}$	$c_{0,-1}$	$A_{1,-1}$				$A_{2,-1}$
$A_{-1,0}$				$A_{0,0}$	$a_{0,0}$	$b_{0,0}$	$c_{0,0}$	$A_{1,0}$				$A_{2,0}$
$d_{-1,0}$				$d_{0,0}$	$e_{0,0}$	$f_{0,0}$	$g_{0,0}$	$d_{1,0}$				$d_{2,0}$
$h_{-1,0}$				$h_{0,0}$	$i_{0,0}$	$j_{0,0}$	$k_{0,0}$	$h_{1,0}$				$h_{2,0}$
$n_{-1,0}$				$n_{0,0}$	$p_{0,0}$	$q_{0,0}$	$r_{0,0}$	$n_{1,0}$				$n_{2,0}$
$A_{-1,1}$				$A_{0,1}$	$a_{0,1}$	$b_{0,1}$	$c_{0,1}$	$A_{1,1}$				$A_{2,1}$
$A_{-1,2}$				$A_{0,2}$	$a_{0,2}$	$b_{0,2}$	$c_{0,2}$	$A_{1,2}$				$A_{2,2}$

Intra Prediction

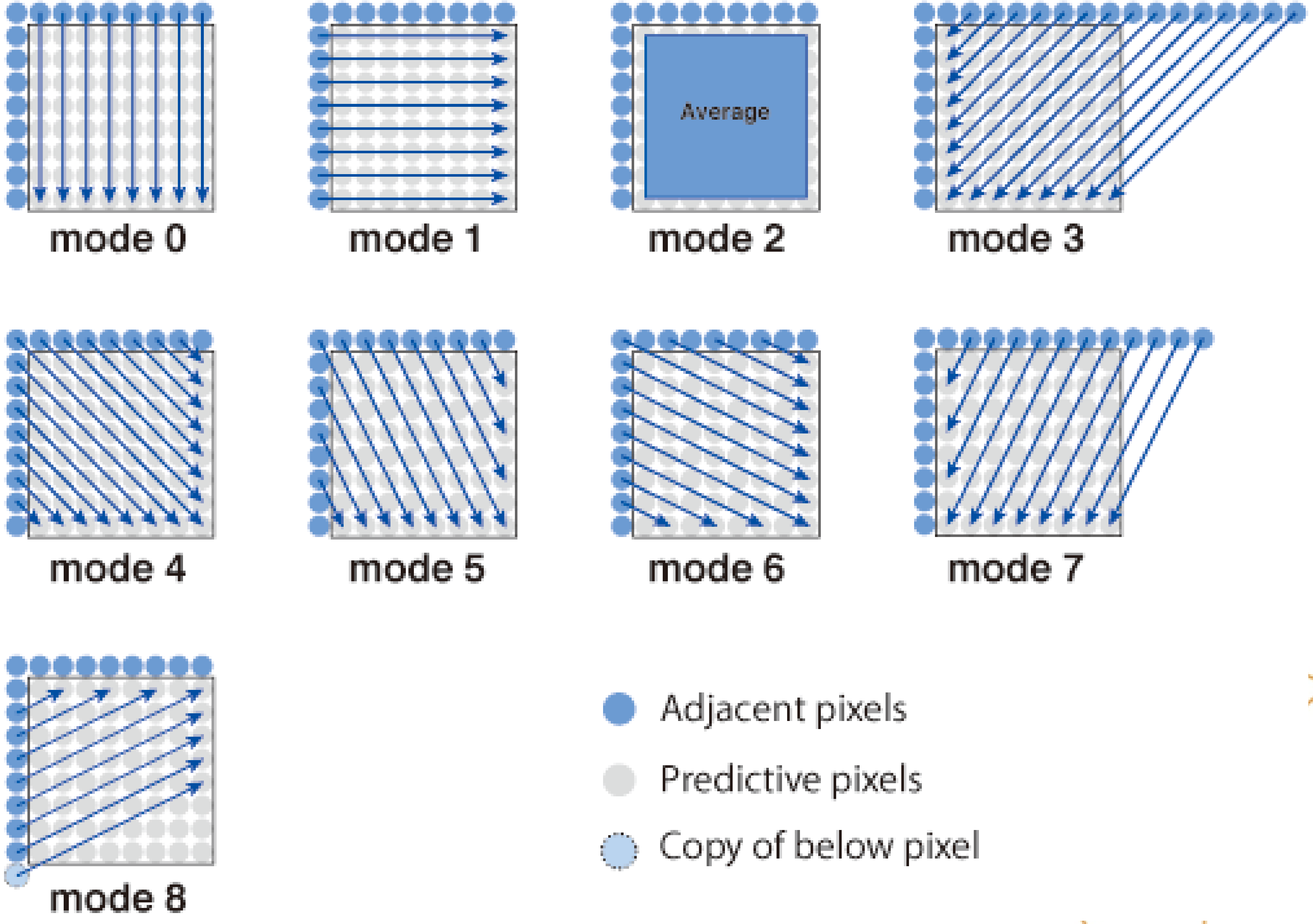
- What is Intra Prediction?



M	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
I	b_{11}	b_{12}	b_{13}	b_{14}				
J	b_{21}	b_{22}	b_{23}	b_{24}				
K	b_{31}	b_{32}	b_{33}	b_{34}				
L	b_{41}	b_{42}	b_{43}	b_{44}				

Intra Prediction

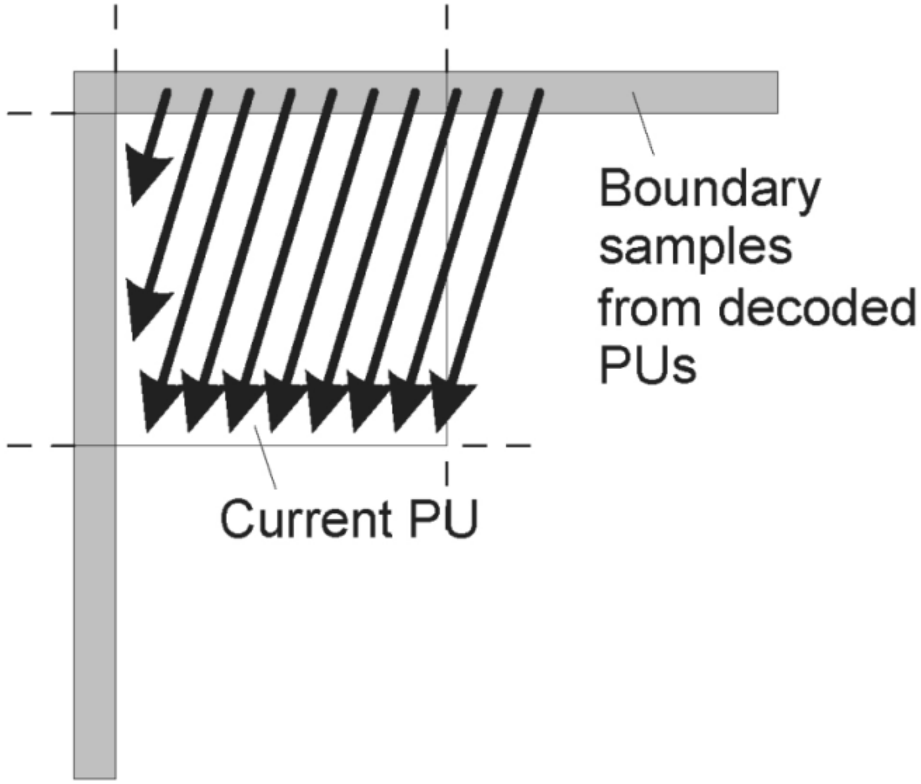
- Prior to HEVC

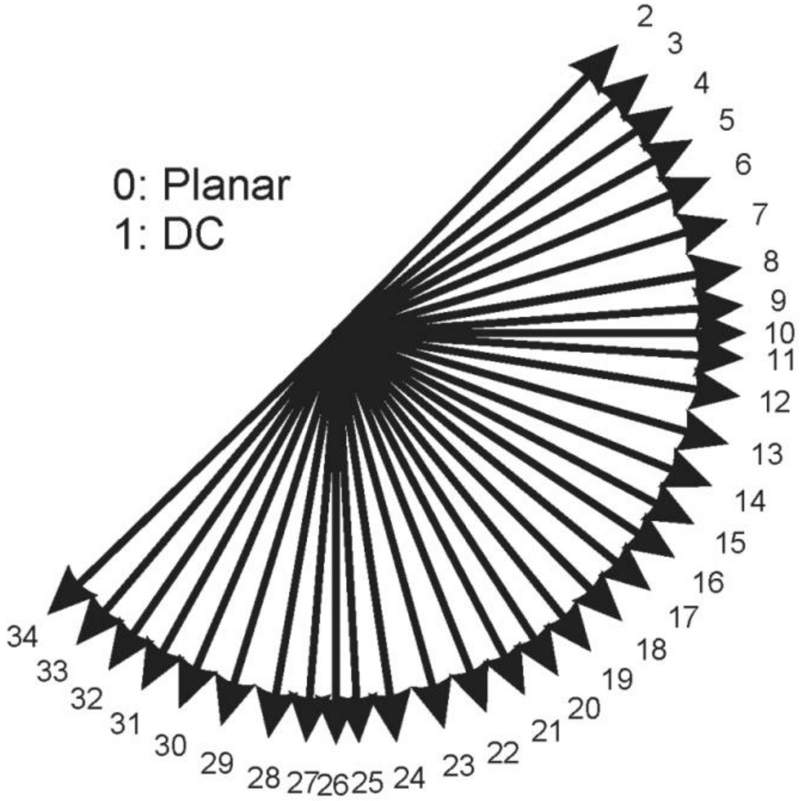


Intra Prediction

- HEVC supports :
  - ➢33 directional modes
  - ➢planar (surface fitting)
  - ➢DC prediction (flat)

Example: Directional mode 29





- Using 4N+1 spatial neighbours
- Extrapolating samples for a given direction

Motion Vector coding

- There are two methods for MV prediction:
  - Merge Mode
  - Advanced Motion Vector Prediction (AMVP)

(instead of sending the whole motion vector each time)

Motion Vector coding

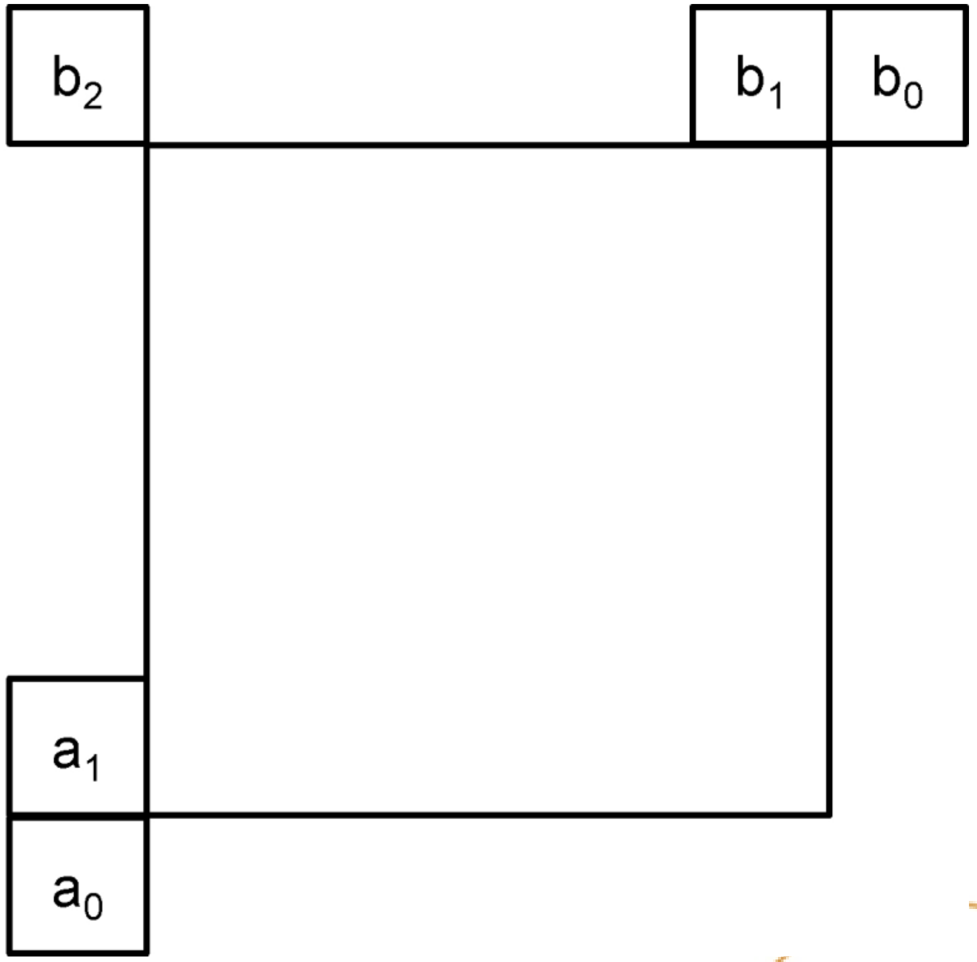
Merge Mode

- A candidate list of motion parameters is made for the corresponding PU (Using spatial and temporal neighbouring PBs)
- No motion parameters are coded, only the index information for selecting one of the candidates is transmitted
- Allows a very efficient coding for large consistently displaced picture areas. (Combined with large block sizes)

Merge Mode

- Candidates ?

Spatial →



→Availability check: {a1 , b1 , b0 , a0 , b2}

Merge Mode

- Candidates ?

Temporal → right, bottom position outside the PU

If not available→ center position

Motion Vector coding

Advanced Motion Vector Prediction (AMVP)

- AMVP is used when an inter coded CB is not coded using the merge mode
- The difference between the chosen predictor and the actual motion vector is transmitted…
- … along with the index of the chosen candidate

Advanced Motion Vector Prediction (AMVP)

- **Advanced Motion Vector Prediction (AMVP)** → defines the search window center point of a PU in the motion estimation process using the surrounding available MVs.

- Motion Vector can be calculated using →

Merge/Skip Mode

Traditional ME process that uses AMVP as a first Step.

Advanced Motion Vector Prediction (AMVP)

- **AMVP** uses two types of candidates in order to calculate the center point of the search window of a PU :
  - Spatial Candidates (Up to 2 of 5 candidates)
  - Temporal Candidates (1 of the 2 candidates)

- May be (0 or 1 or 2 or 3) for the AMVP.
  - 0 or 1 → add **ZERO MV** candidates to have 2 candidates.
  - 2 → it the target.
  - 3 → Delete the candidate with index > 1 as we need just two candidates → candidate [0], candidate [1].

Advanced Motion Vector Prediction (AMVP)

- **AMVP STEPS**:
  1. Motion Vector Candidate (MVC) set construction process.

     How to find the candidates and put them in the form to be checked !!! (<u>Spatial and Temporal Candidates</u>)

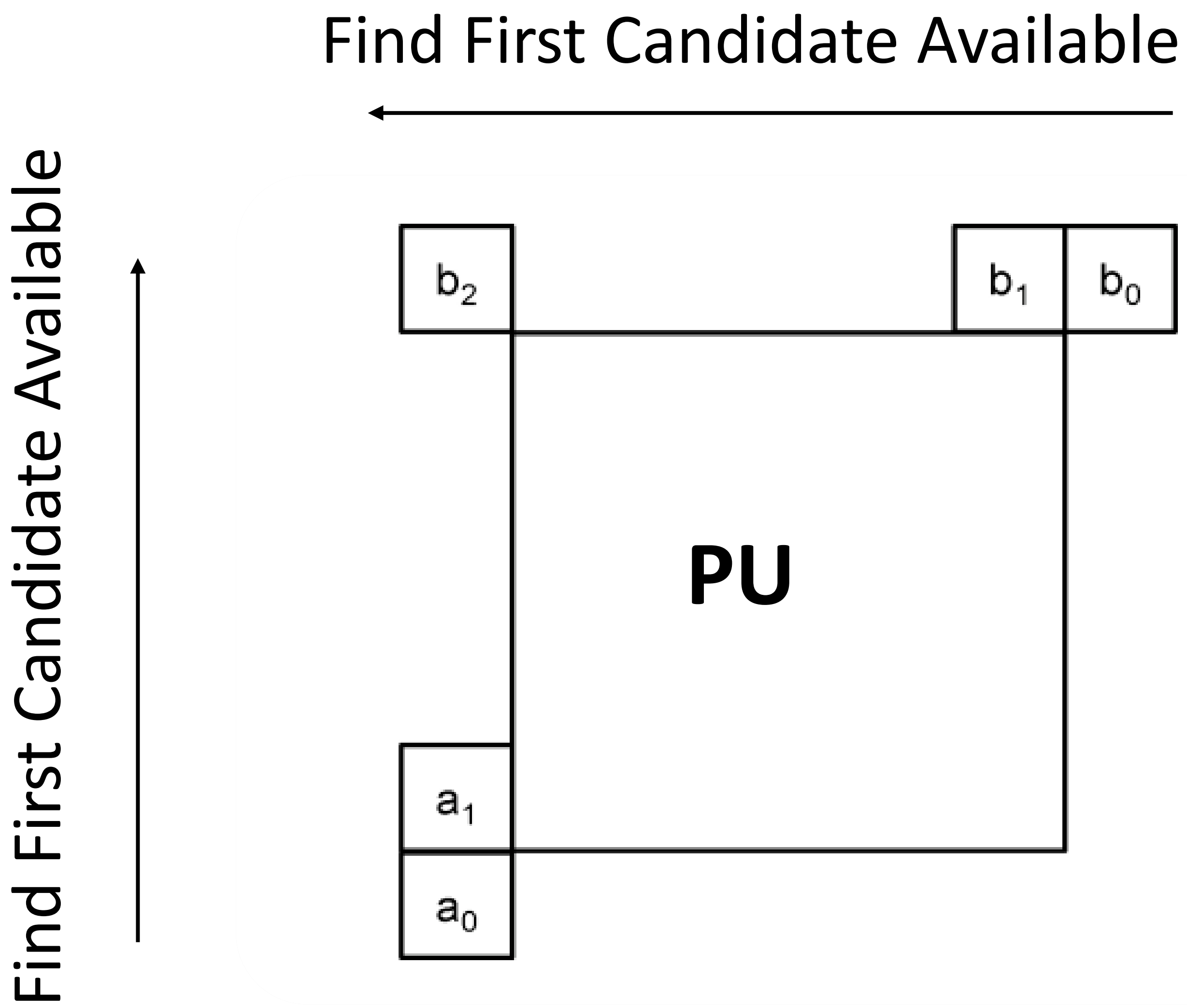
  2. Best Motion Vector Selection using the rate distortion cost function in order to choose the one with less cost value.

  - → The second step is just like the rate cost function of IME using the following equation:

$$\left(r^{*}, m^{*}\right) = \underset{r\in R, m\in M}{\arg\min}\, D_K(r,m) + \lambda_M . R_K(r,m)$$

AMVP Spatial Candidates

- **First Candidate** → is one of (**A_0**, **A_1**), **A_0** has the priority over **A_1**.
- **Second Candidate** → is one of (**B_0**, **B_1**, **B_2**), **B_0** has the priority over **B_1** over **B_2**.



Find First Candidate Available

Find First Candidate Available

AMVP Spatial Candidates

- **Candidate A is calculated as follows:**
  - **A_0** or **A_1** available and has the same reference index (frame) of the current PU → **Take the higher priority MV as it is**.
  - **A_0** and **A_1** are available, and have different reference frames from the current PU → **Take the higher priority MV and Scale it using the following equations:**

$$\mathbf{mv} = \mathrm{sign}(\mathbf{mv}_{\mathrm{cand}} \cdot \mathrm{ScaleFactor}) \cdot ((|\mathbf{mv}_{\mathrm{cand}} \cdot \mathrm{ScaleFactor}| + 2^7) \gg 8)$$

$$\mathrm{ScaleFactor} = \mathrm{clip}(-2^{12}, 2^{12} - 1, (\mathrm{tb} \cdot tx + 2^5) \gg 6)$$

$$tx = \frac{2^{14} + |\frac{\mathrm{td}}{2}|}{\mathrm{td}}$$

t_b, t_d → are the temporal distances which mean Picture Order Count (POC) difference.

t_b→ distance between current and reference pictures of the current PU
t_d→ distance between current and reference pictures of the candidate PU

AMVP Spatial Candidates

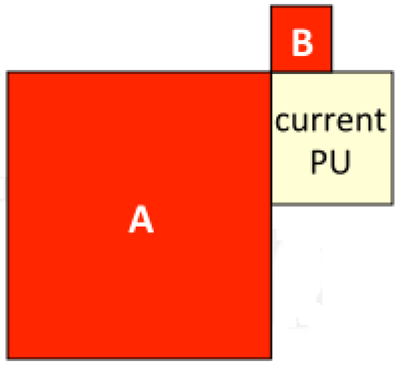
- Sign(**a**) → is a function that returns a positive one (+1) for positive numbers and a negative one (-1) for negative numbers.

- Clip(**u**, **v**, **w**) → is a clipping function that limits the value of **w** to be a value with in range of **u** → **v**.

AMVP Spatial Candidates

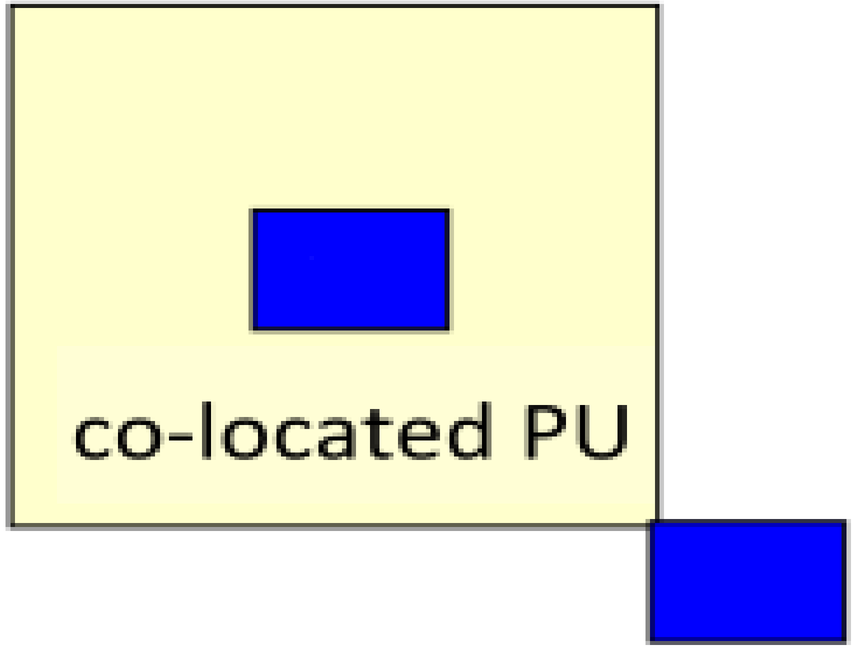
- **Candidate B is calculated as follows:**
  - Just like **A** using the references, priority and scaling equation.
  - Check if **A** is available or not ! If **A** is not available, → **A** candidate is equal to not-scaled **B** candidate if available.
  - If NOT → use **ZERO** motion vectors

All intra Prediction



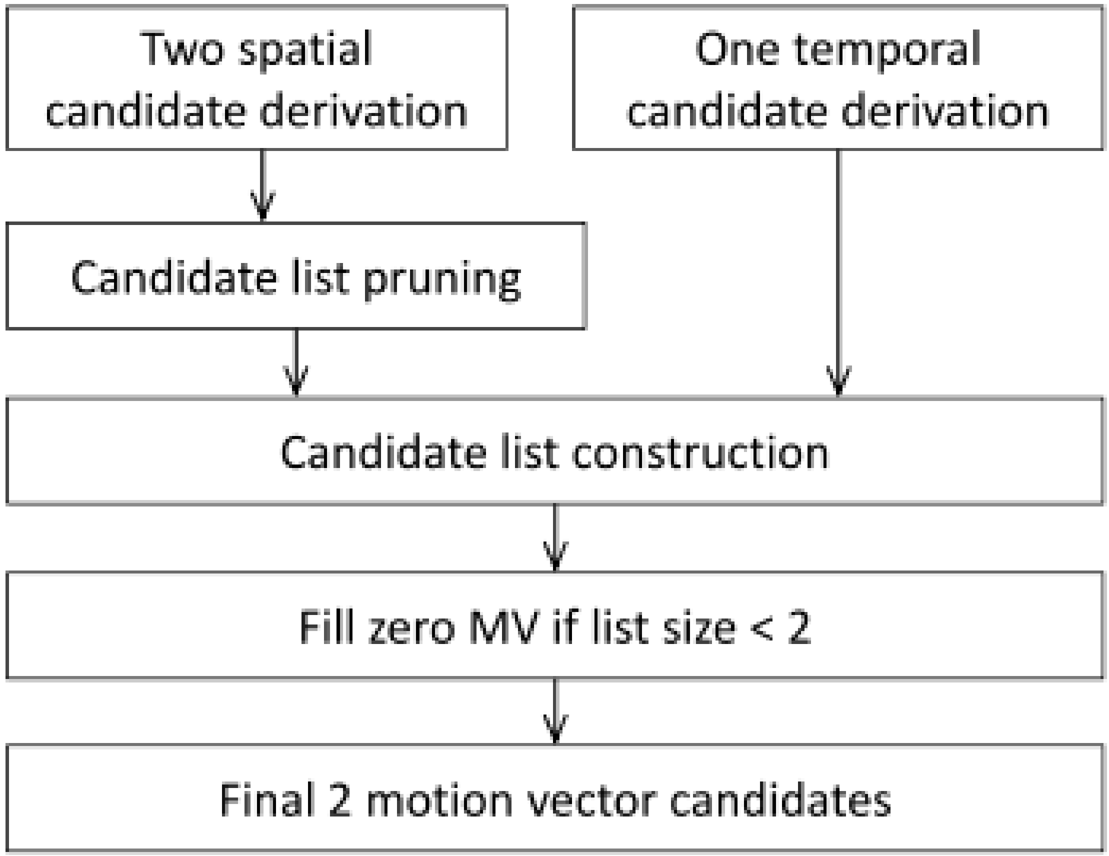
AMVP Temporal Candidates

- **Temporal Candidate is** → one of (**C_0**, **C_1**), **C_0** has the priority over **C_1**.
  - **C_0 is not considered** → when in different CTU.
  - Scaling is **MANDATORY** → Same scaling equations.
  - **TMVP** may be disabled using a flag.



AMVP Temporal Candidates

- **Two** candidates derived from **spatial** domain and **one** candidate from **temporal** domain.
- **Spatial A →** the first available of A0, A1 with priority order.
- **Spatial B →** the first available of B0, B1, B2 with priority order.
- **If A is equal to B →** drop B and take the temporal candidate.
- **If the list is > 2 →** delete the candidate with index 2.
- **If the list is < 2 →** add ZERO motion vector candidates.
- Final Candidate List is **2** (**FIXED**)



Two spatial candidate derivation

One temporal candidate derivation

Candidate list pruning

Candidate list construction

Fill zero MV if list size < 2

Final 2 motion vector candidates

In-loop Filters

- Deblocking Filter (DBF)

- Sample Adaptive Offset (SAO)

Deblocking Filter (DBF)

w/o deblocking

w/ deblocking

Deblocking Filter (DBF)

- Reduces the blocking artifacts (due to block based coding)

- Only applied to samples adjacent to PU and TU boundaries and aligned with the 8x8 sample grid

Deblocking Filter (DBF)

- 3 Strengths :
  - Strength 2: If one of the blocks is intra coded
  - Strength 1: If any of the below
    - ✓ At least one transform coefficient is non-zero
    - ✓ The references of the two blocks are not equal
    - ✓ The motion vectors are not equal
  - Strength 0: DBF not applied

Deblocking Filter (DBF)

➢ According to the strength and average quantization parameter:

- 3 cases for luma:
  - ✓ No filter
  - ✓ Weak filter
  - ✓ Strong filter

➢ 2 cases for chroma:

Normal filtering (if Strength >1) or No filtering

Deblocking Filter (DBF)

- Processing order:

1st) Horizontal filtering → For vertical edges

2nd) Vertical filtering → For horizontal edges

The filtering process can be done in parallel threads

Sample Adaptive Offset (SAO)

With SAO

Without SAO

Sample Adaptive Offset (SAO)

- New in HEVC
- After the deblocking filter
- Applies to all samples satisfying the conditions
- Performed on a region basis

Sample Adaptive Offset (SAO)

- Modifies Samples by → adding an offset
- The offset is based on → look-up table values
- Per CTB
  - Type_ID=0 No SAO
  - Type_ID=1 Band offset
  - Type_ID=2 Edge offset

Sample Adaptive Offset (SAO)

- Band offset:

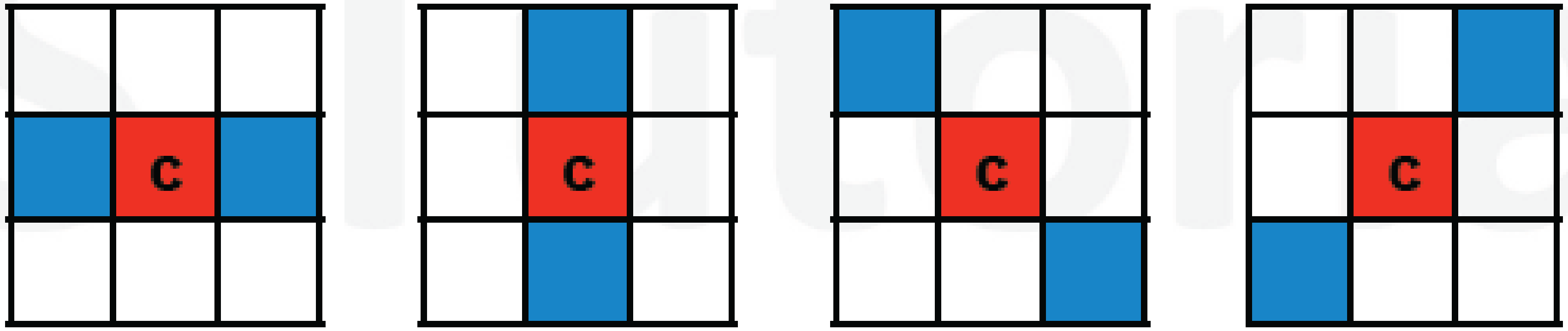
Offset value Depends on **Sample amplitude**

Full sample range → Uniformly split into 32 bands

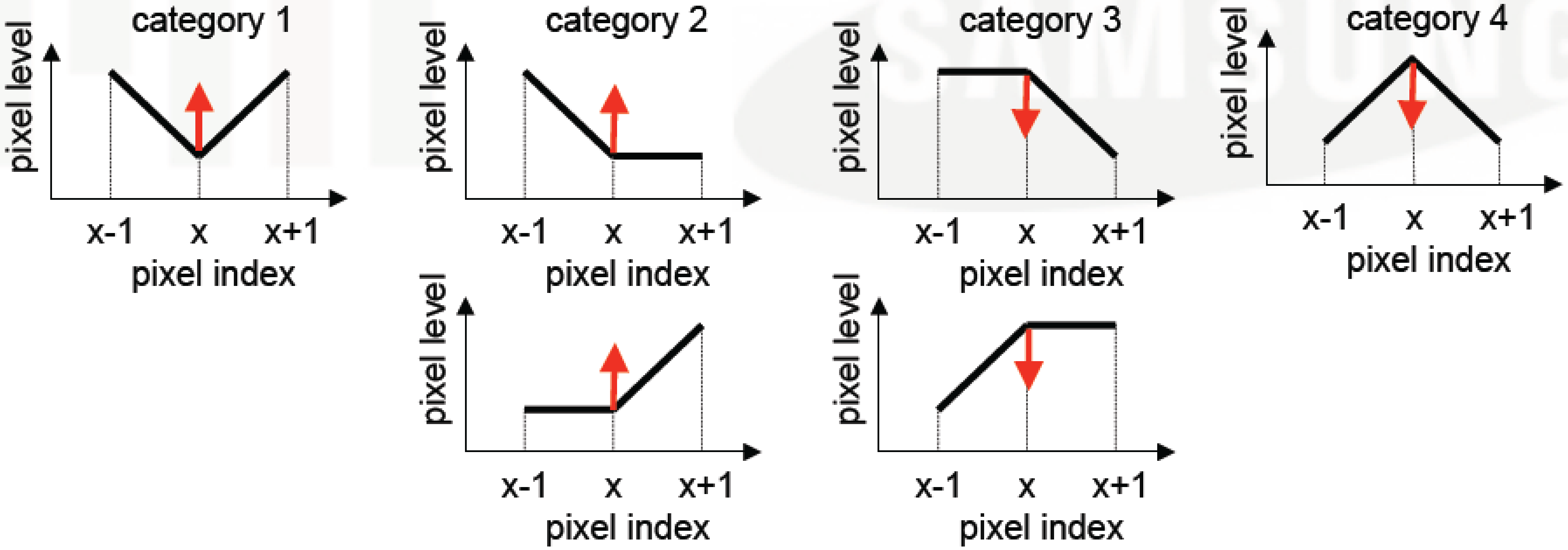
4 consecutive bands → Have a + or – band offset

Sample Adaptive Offset (SAO)

- Edge offset:
  - 4 types



  - Based on the values of the neighbors, apply one of 4 offsets



Sample Adaptive Offset (SAO)

- Edge offset:

Based on the category → A value from the look-up table

Categories 1, 2 : Negative offset

Categories 3, 4 : Positive offset

Parallel Processing Tools

Motivation:

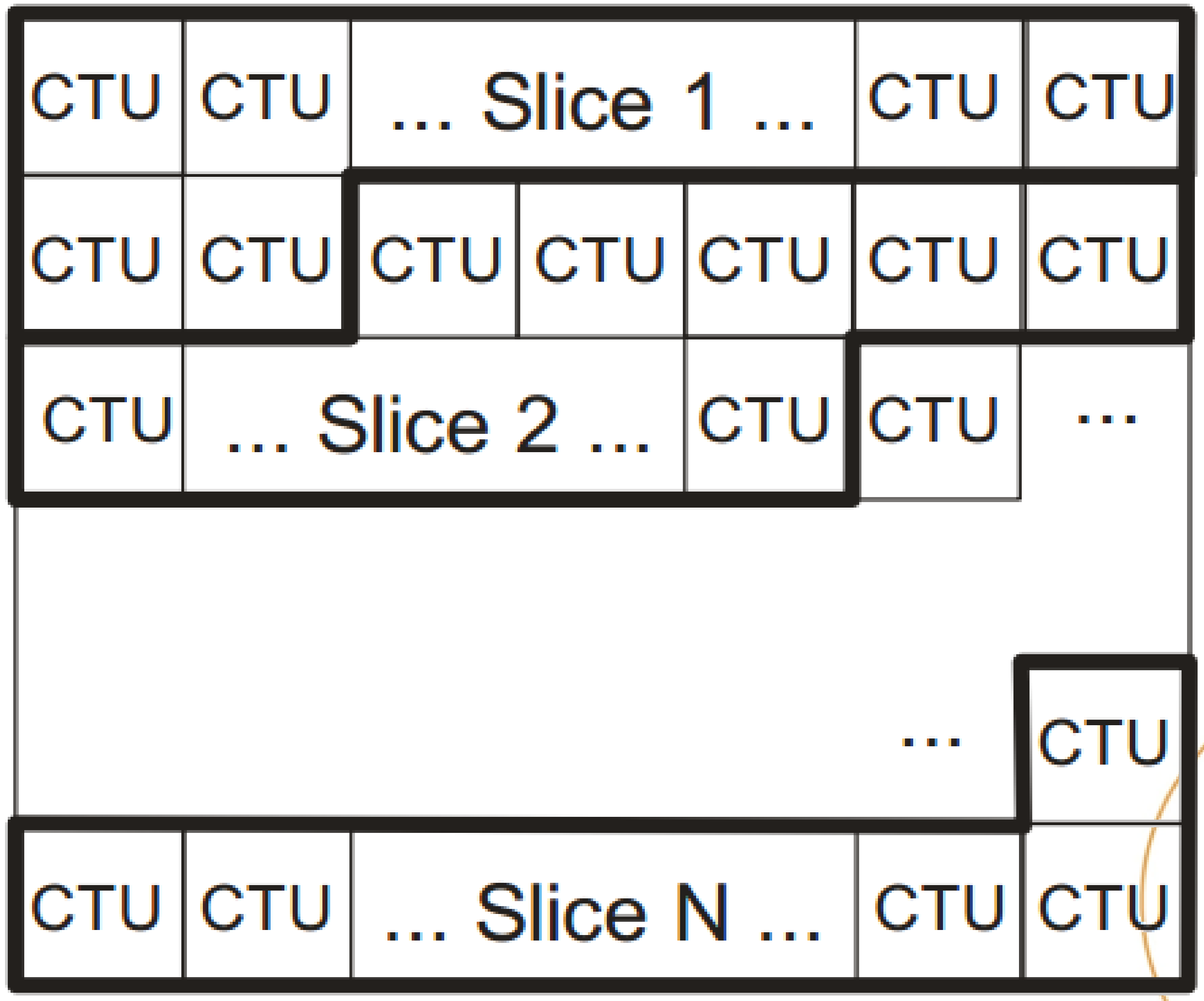
- High resolution videos
- HEVC is far more complex than its prior standards
- Since we have parallel processing architectures, why not use it !

Parallel Processing Tools

- Slices
- Tiles
- Wavefront parallel processing (WPP)

Slice

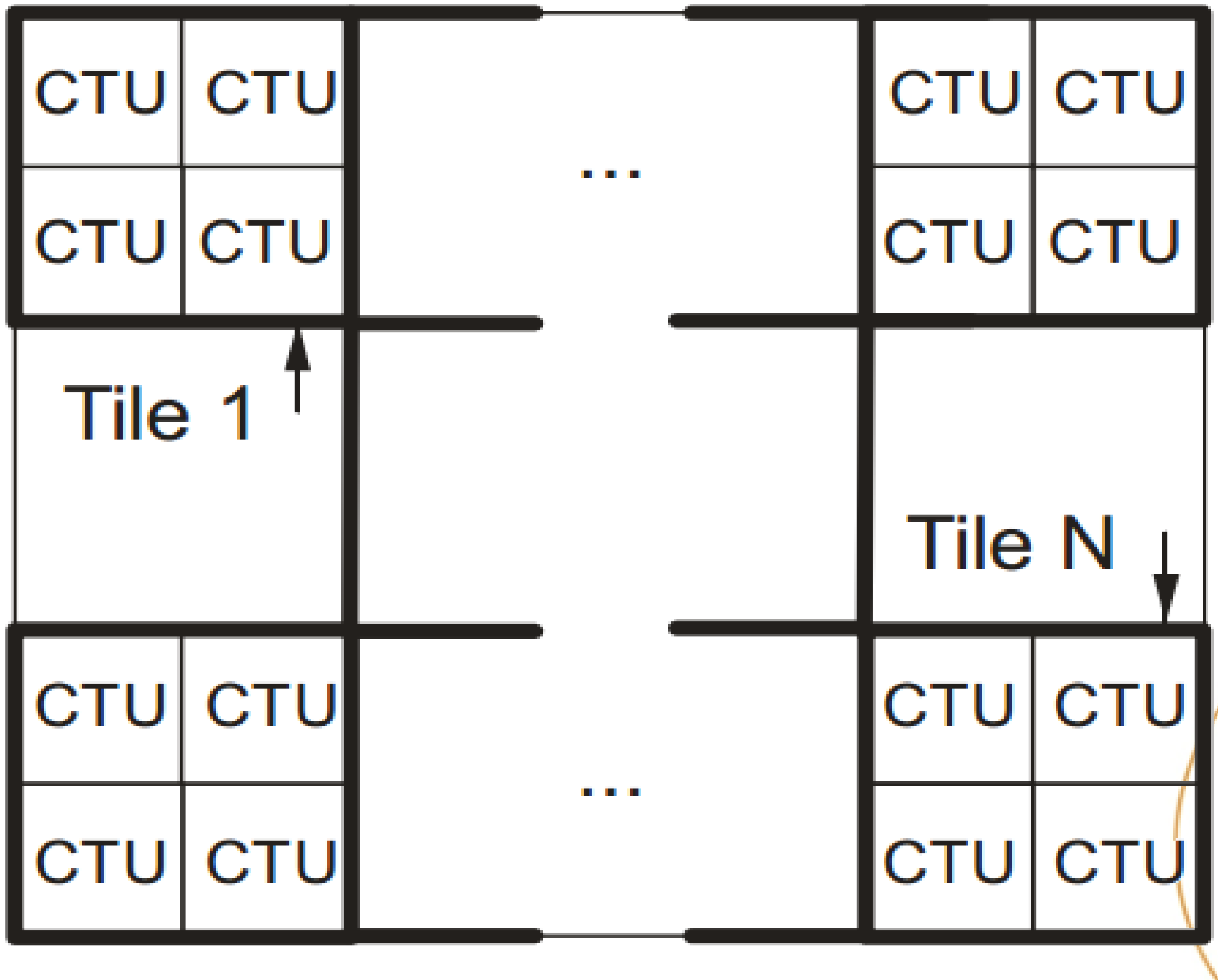
- Slices are a sequence of CTUs that are processed in the order of a raster scan. Slices are self-contained and independent.
- Each slice is encapsulated in a separate packet.



Tile

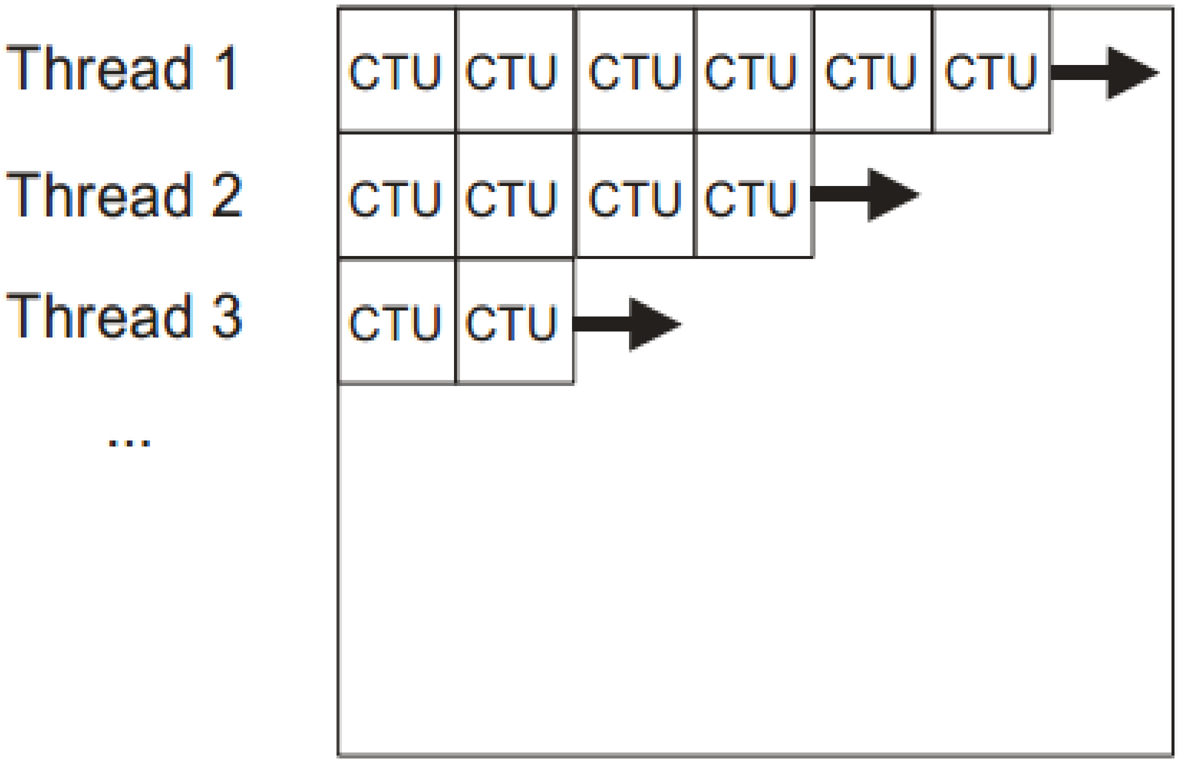
- Self-contained and independently decodable rectangular regions.
- Tiles provide parallelism at a coarse level of granularity.

Tiles more than the cores → Not efficient → Breaks dependencies



Wavefront Parallel Processing

- A slice is divided into rows of CTUs. Parallel processing of rows.
- The decoding of each row can be begun as soon a few decisions have been made in the preceding row for the adaptation of the entropy coder.

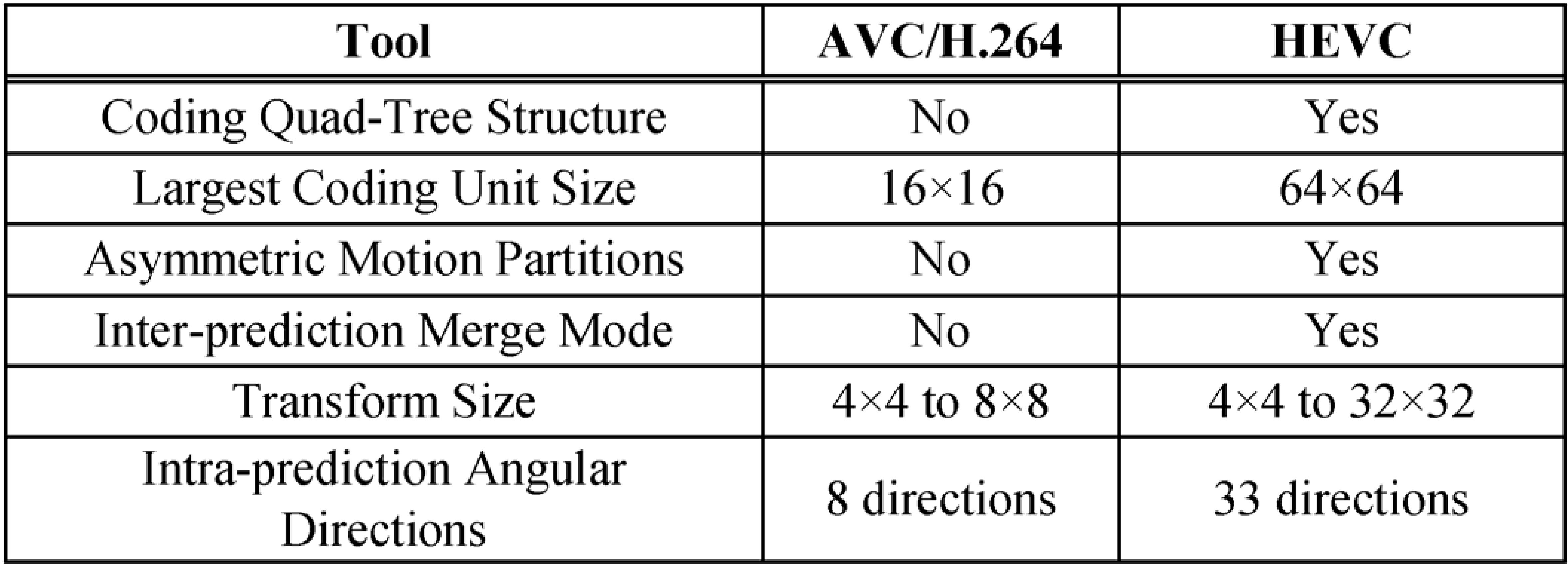

Thread 1 CTU CTU CTU CTU CTU CTU
Thread 2 CTU CTU CTU CTU
Thread 3 CTU CTU
...

No WPP with tiles !!

- Better compression than tiles. Parallel processing at a fine level of granularity.

HEVC Coding Complexity

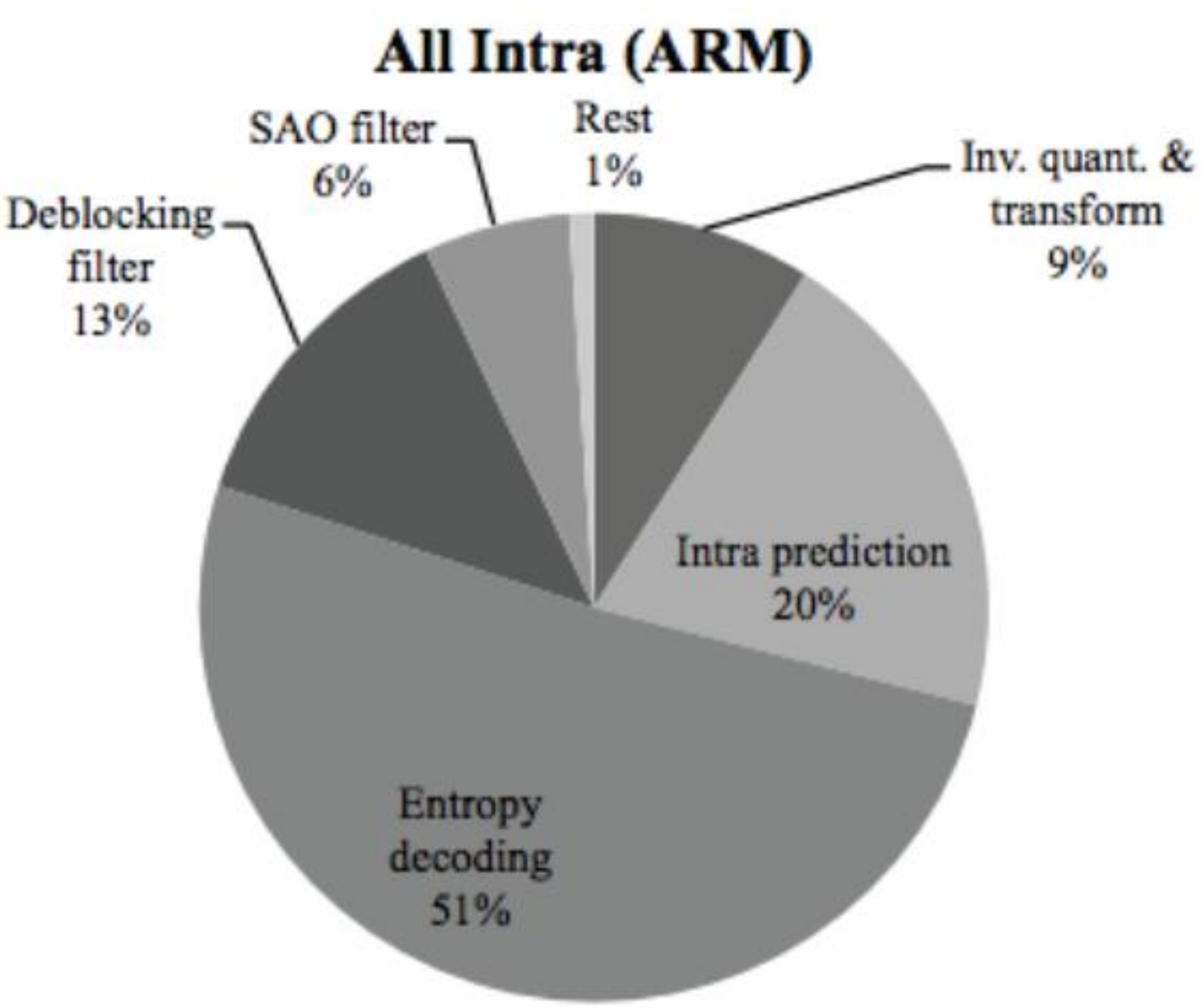
- HEVC vs H.264

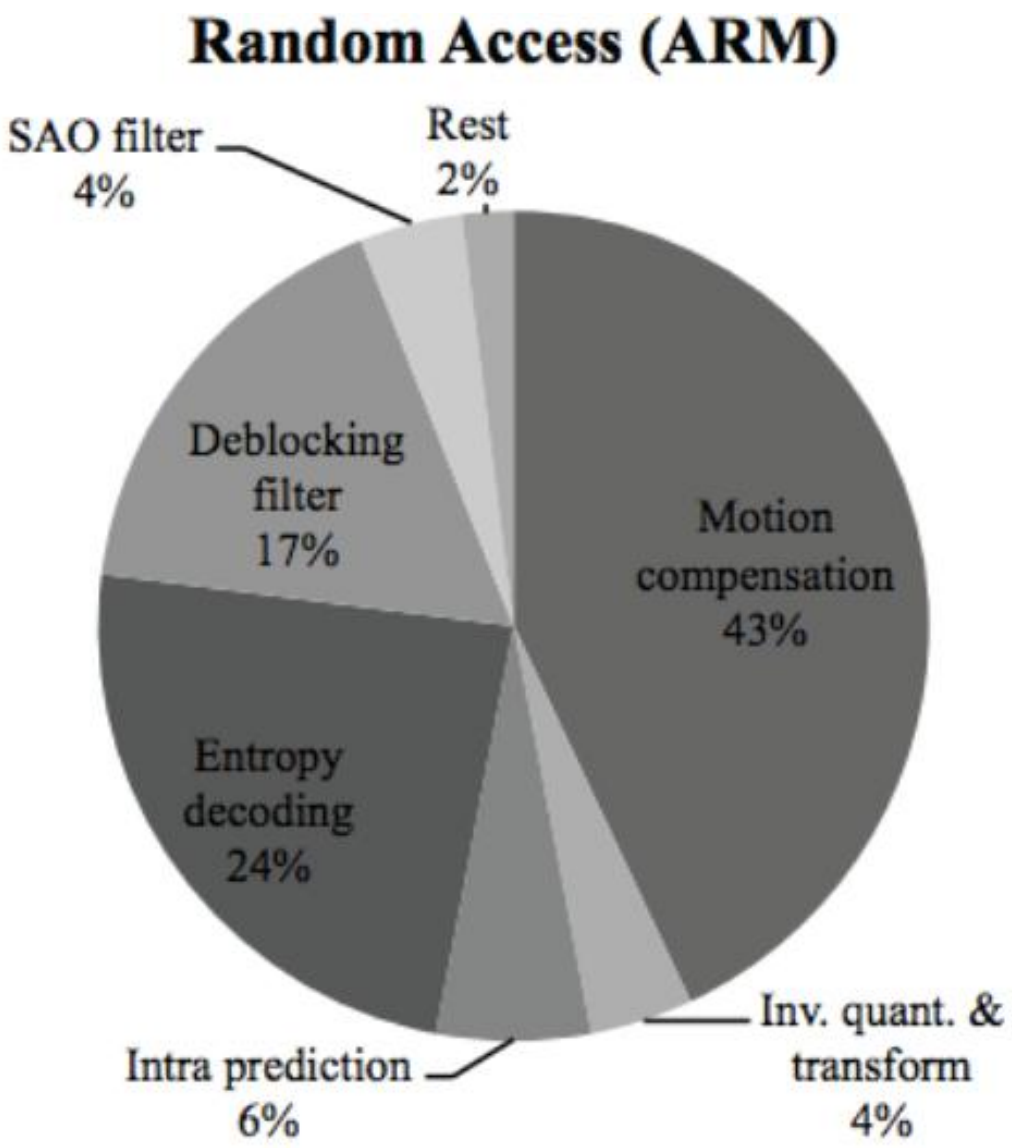


Tool	AVC/H.264	HEVC
Coding Quad-Tree Structure	No	Yes
Largest Coding Unit Size	16×16	64×64
Asymmetric Motion Partitions	No	Yes
Inter-prediction Merge Mode	No	Yes
Transform Size	4×4 to 8×8	4×4 to 32×32
Intra-prediction Angular Directions	8 directions	33 directions

HEVC Coding Complexity

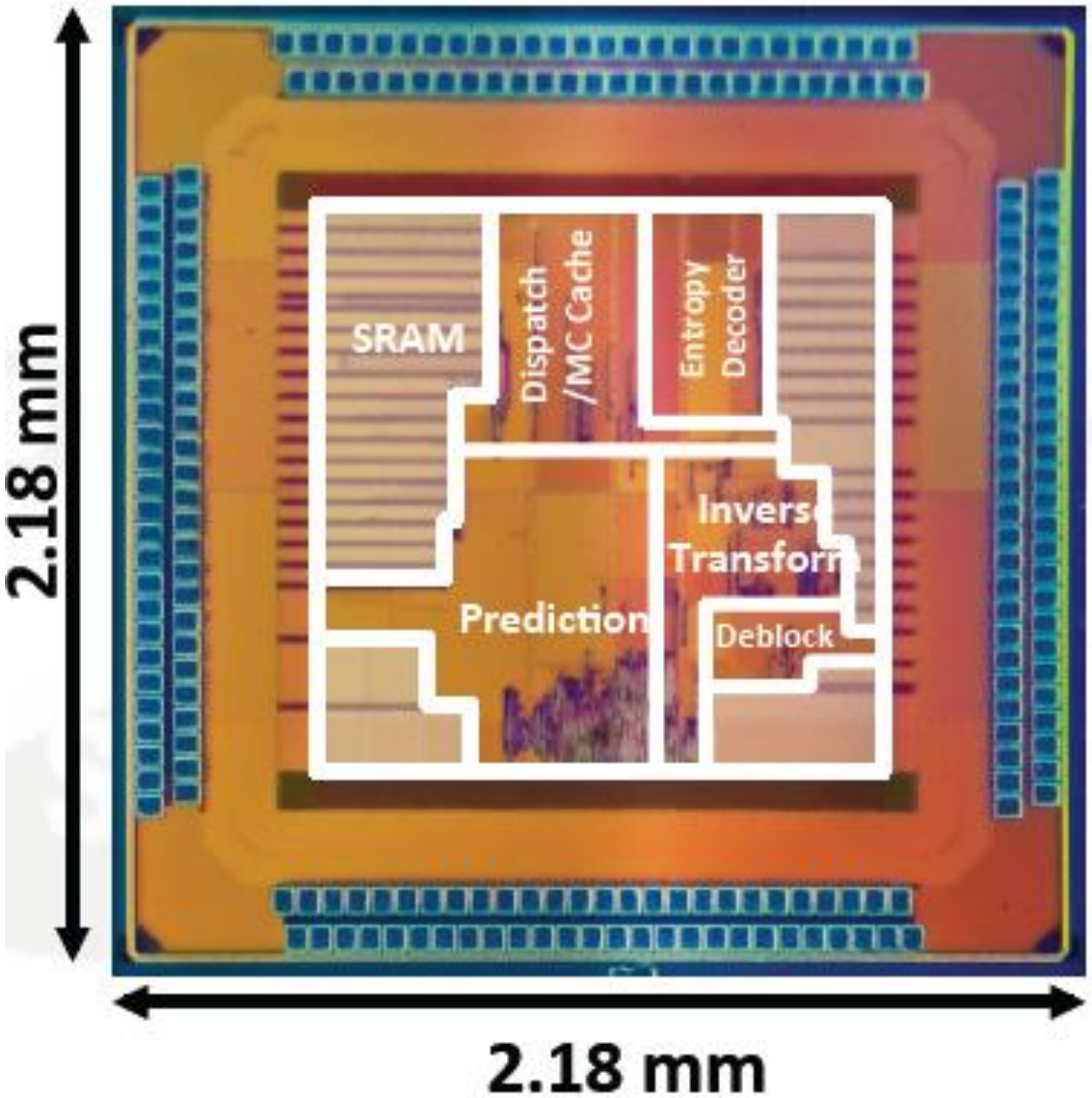
- HEVC decoder workload for different modules





Hardware HEVC Decoder

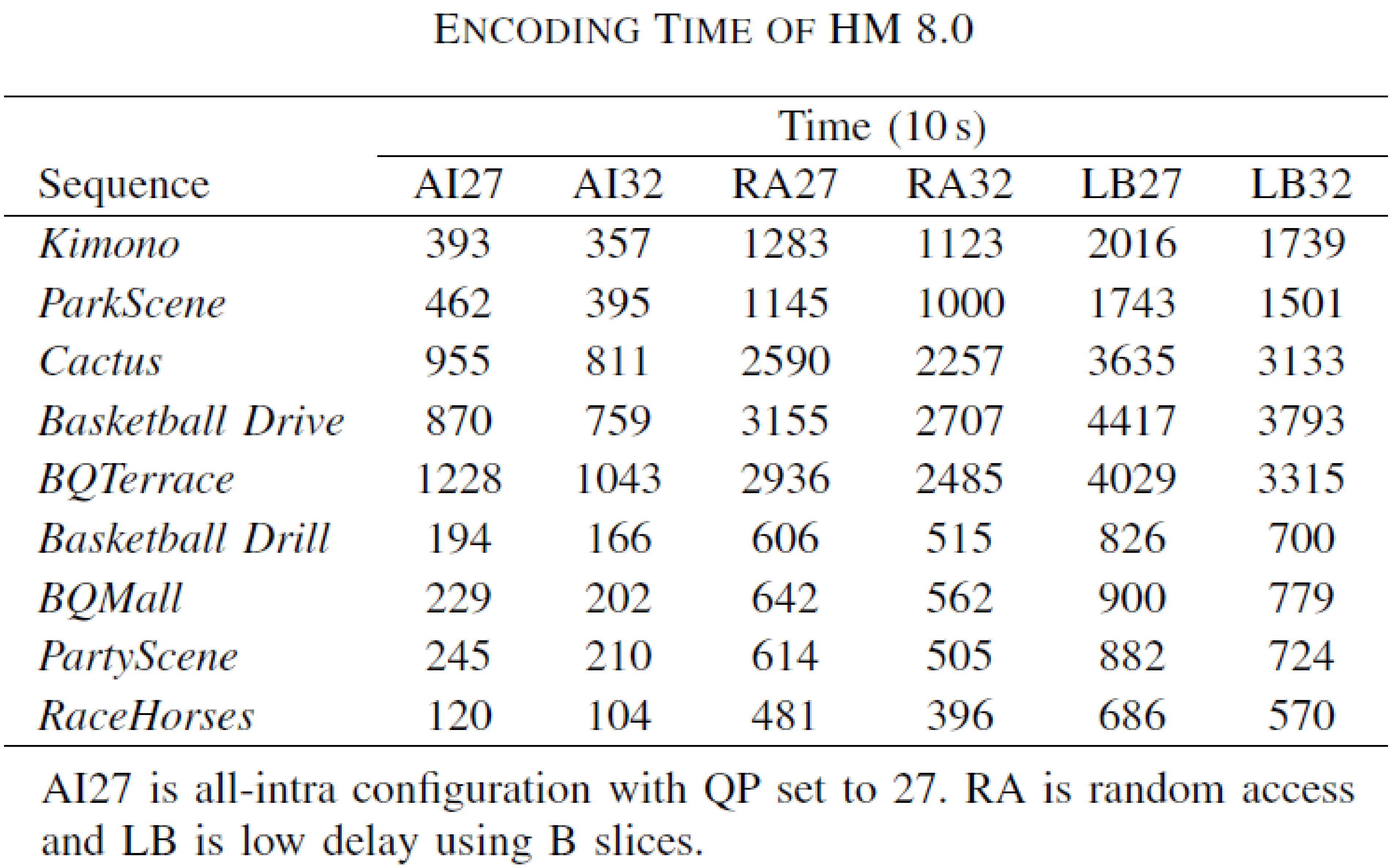
Video Coding Standard	HEVC (HM4)
Technology	TSMC 40-nm
Core Area	1.33 x 1.33 mm
Gate Count	715k
On-Chip Memory (SRAM)	124 kB
Resolution / Frame Rate	4kx2k @ 30fps (3840x2160)
Frequency	200 MHz
Core Voltage	0.9 V
Power	76 mW



C.---T. Huang et al., "A 249Mpixels/s HEVC Video Decoder Chip for Quad Full HD Applications," *IEEE ISSCC,* 2013

HEVC Coding Complexity

- Encoding times were obtained on a cluster containing Xeon-based servers (E5670 clocked at 2.93 GHz)



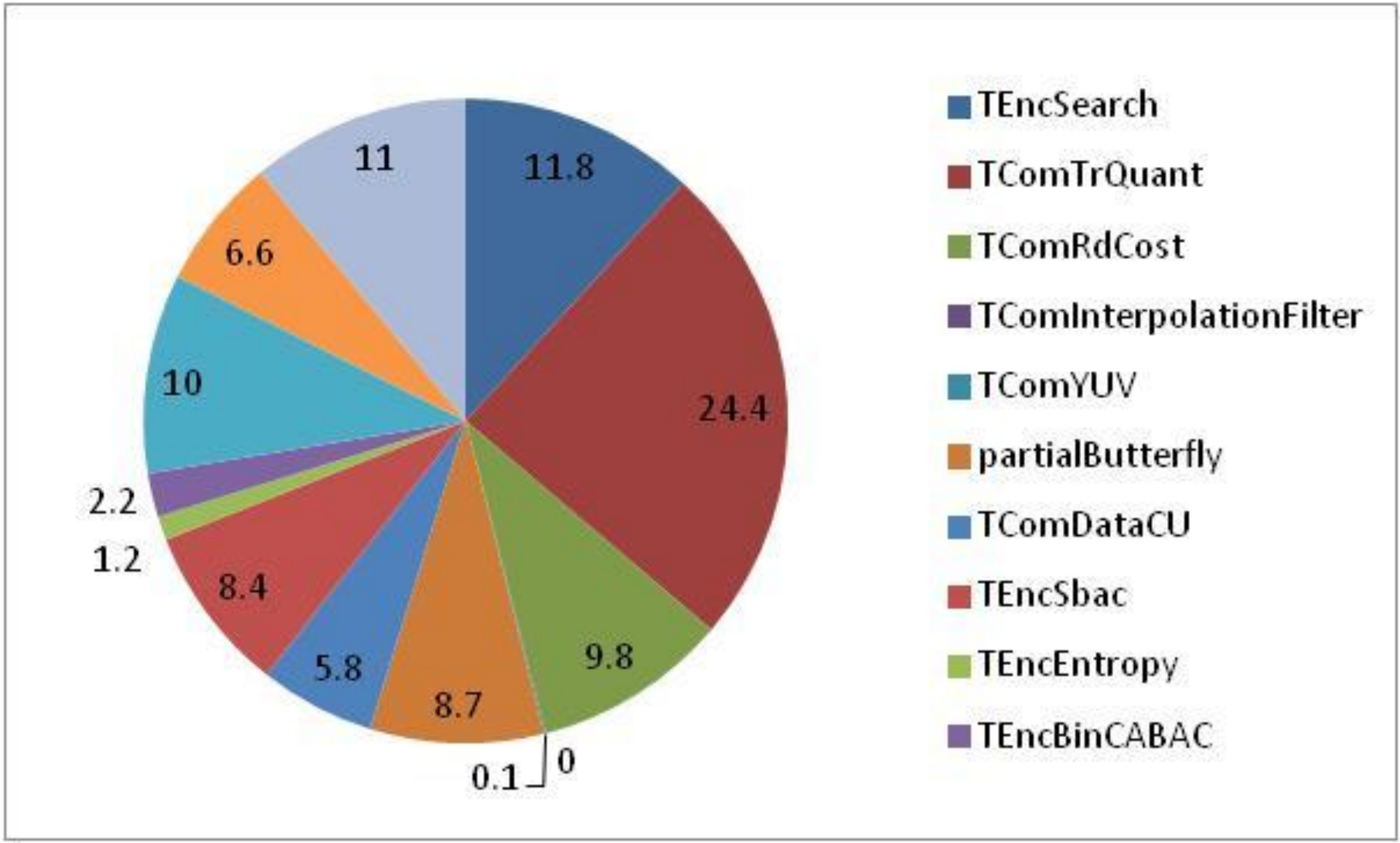
ENCODING TIME OF HM 8.0

	Time (10 s)					
Sequence	AI27	AI32	RA27	RA32	LB27	LB32
Kimono	393	357	1283	1123	2016	1739
ParkScene	462	395	1145	1000	1743	1501
Cactus	955	811	2590	2257	3635	3133
Basketball Drive	870	759	3155	2707	4417	3793
BQTerrace	1228	1043	2936	2485	4029	3315
Basketball Drill	194	166	606	515	826	700
BQMall	229	202	642	562	900	779
PartyScene	245	210	614	505	882	724
RaceHorses	120	104	481	396	686	570

AI27 is all-intra configuration with QP set to 27. RA is random access and LB is low delay using B slices.

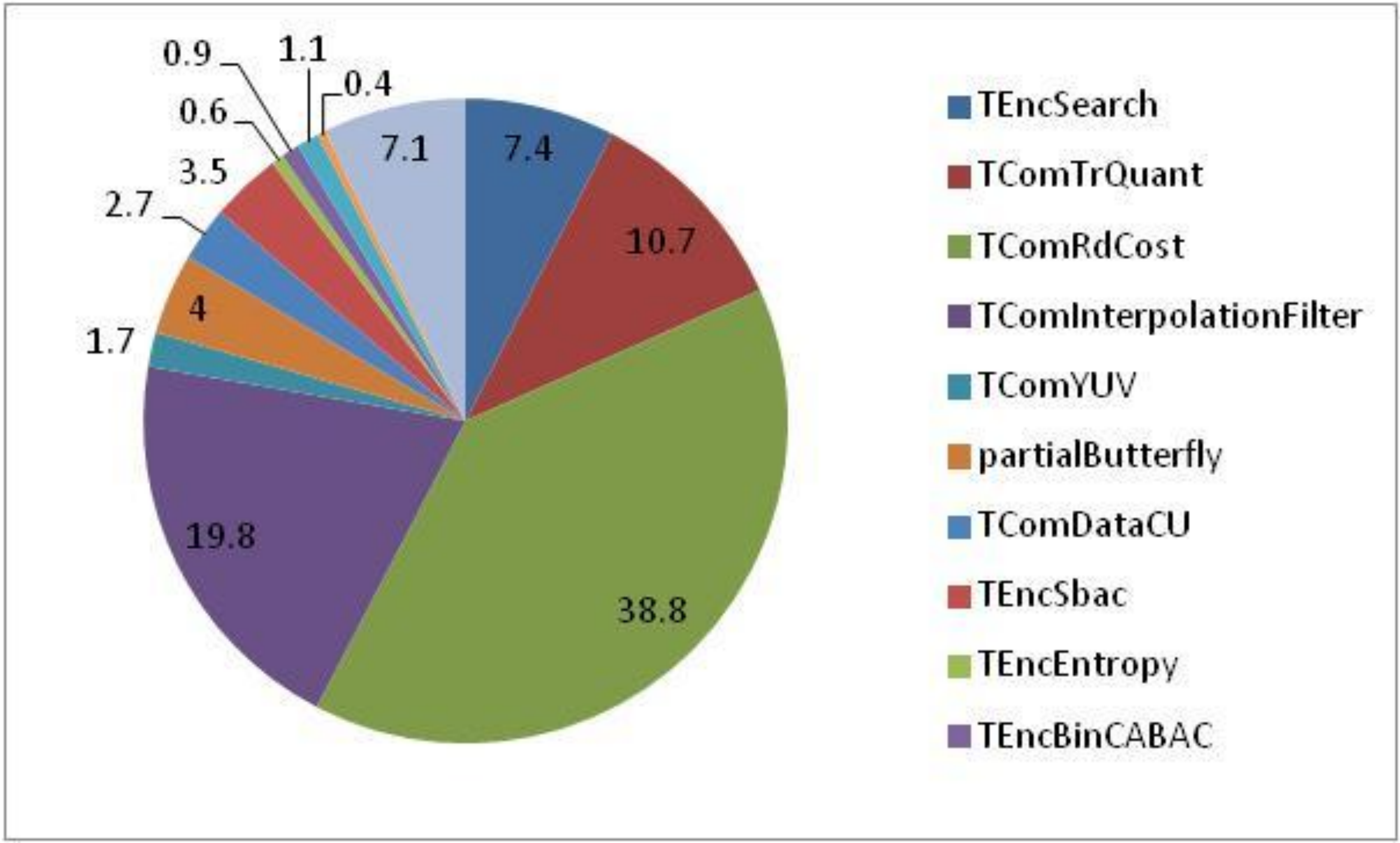
HEVC Coding Complexity

- Encoding Time Distribution by SW Classes (All Intra mode)



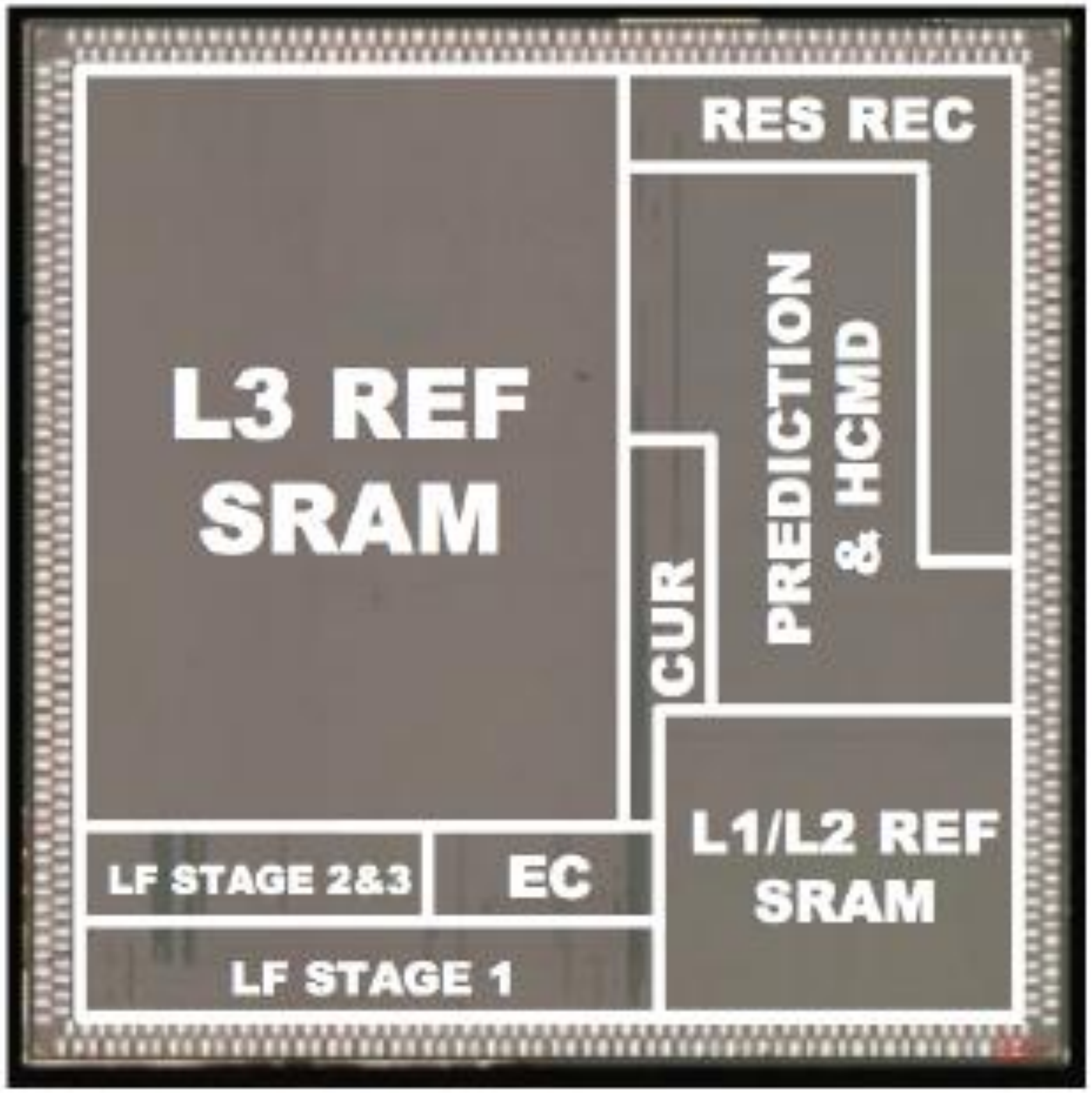
HEVC Coding Complexity

- Encoding Time Distribution by SW Classes (Random Access mode)



Hardware HEVC Encoder

Video Coding Standard	HEVC (WD4)
Technology	TSMC 28-nm HPM
Core Area	5x5mm^2
Gate Count	8350k
On-Chip Memory (SRAM)	7.14 MB
Resolution / Frame Rate	8192x4320@ 30fps
Frequency	312 MHz
Power	708 mW



S.---F. Tsai et al. , “A 1062Mpixels/s 8192×4320p High Efficiency Video Coding (H.265) encoder chip,” IEEE VLSIC, 2013

Lecture Summary

- We have discussed the following topics:
  - Video Coding standards
  - HEVC
  - Improvements in coding efficiency
    - Coding Tree Structure
    - Inter Prediction
    - Intra Prediction
    - Motion Vector coding
    - In-loop filters
  - Parallel Processing Tools
    - Slices
    - Tiles
    - Wavefront parallel processing (WPP)
  - HEVC Coding Complexity

References

- nsl.cs.sfu.ca/teaching/13/880/students/**HEVC.ppt**
- Design and Implementation of Next Generation Video Coding Systems (H.265/HEVC Tutorial), ISCAS 2014.
- Sullivan et al., “Overview of the High Efficiency Video Coding (HEVC) Standard”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 12, December 2012.
- Frojdh et al., “Next Generation Video Compression”, Ericsson Review, April 2013.
- F. Bossen et al., "HEVC Complexity and Implementation Analysis,” *IEEE Transactions on Circuits and Systems for Video Technology,* 2012.