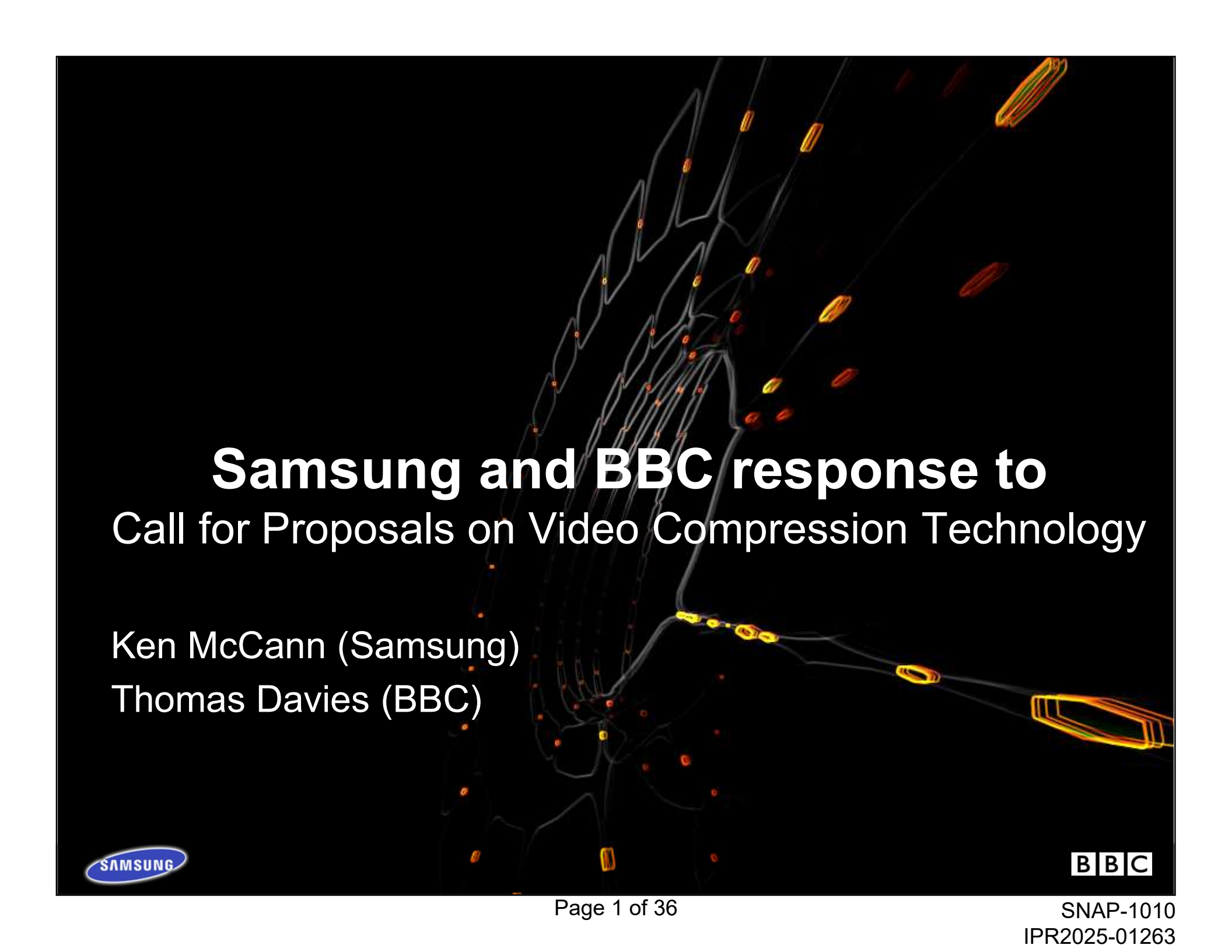




Samsung and BBC response to Call for Proposals on Video Compression Technology

Ken McCann (Samsung)
Thomas Davies (BBC)



Overview

- Introduction
- Algorithm Description
 - Unit Definition
 - Motion Representation
 - Intra-frame Prediction
 - Spatial Transforms
 - In-loop Filtering
 - Entropy Coding
- Compression Performance
- Complexity Analysis
- Conclusions

Introduction: Samsung/BBC Coding Framework

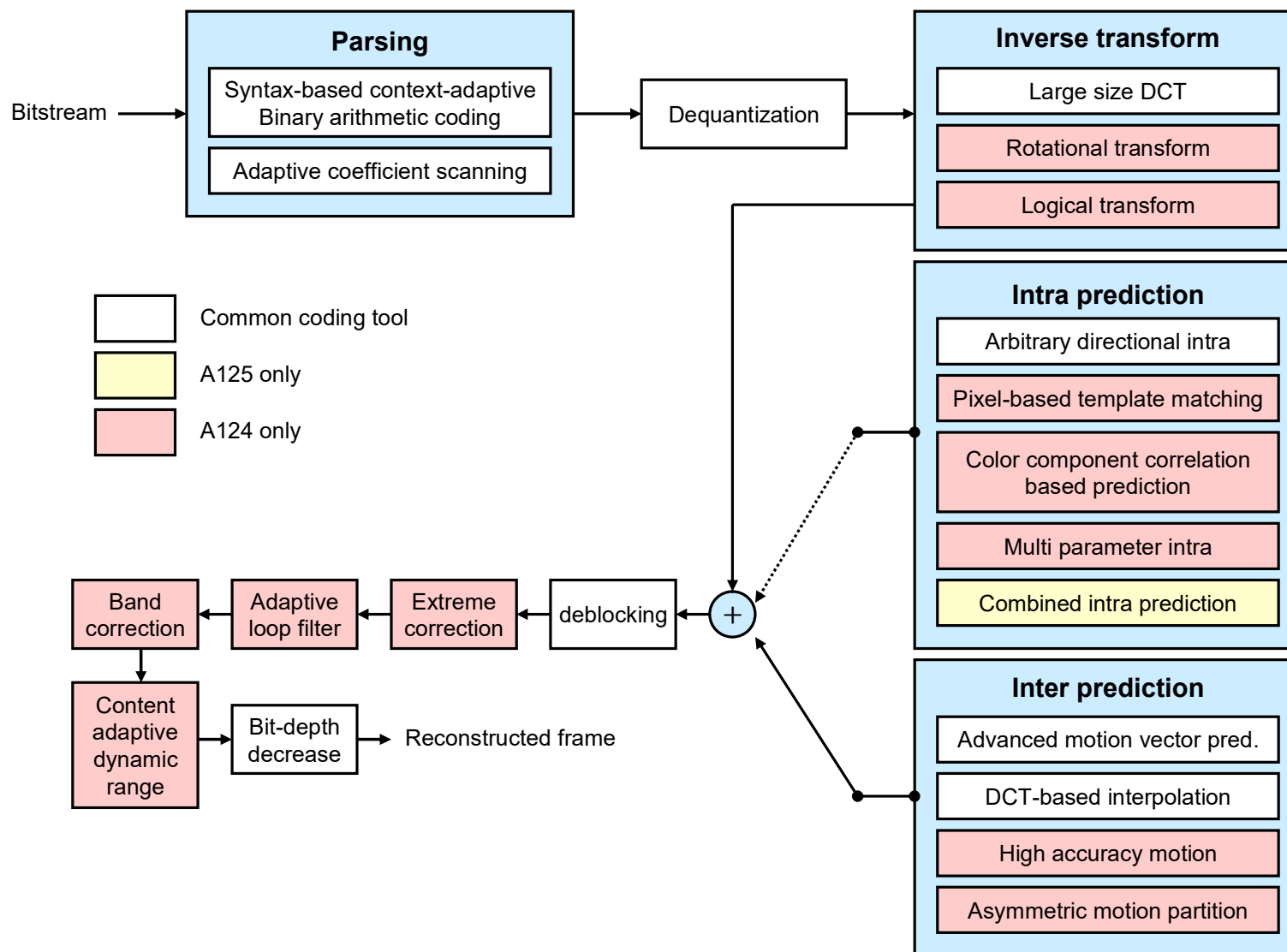
- This presentation covers
 - JCTVC-A124: Samsung Response to CfP
 - JCTVC-A125: BBC response to CfP
- The Samsung/BBC coding framework provides the ability to trade off complexity and compression efficiency
- In our responses to the CfP we demonstrate two key operating points
 - **A125**: low-complexity operating point, with comparable complexity to H.264/AVC but better compression efficiency
 - Average **efficiency about 30% better** than Alpha and Beta anchors
 - Decoding time about 0.6 to 1.3 times that of JM17.0
 - **A124**: high-performance operating point, giving even higher compression efficiency with a moderate increase in complexity over H.264/AVC
 - Average **efficiency about 40% better** than Alpha and Beta anchors
 - Decoding time about 0.9 to 2.4 times that of JM17.0



Introduction: **Key Features**

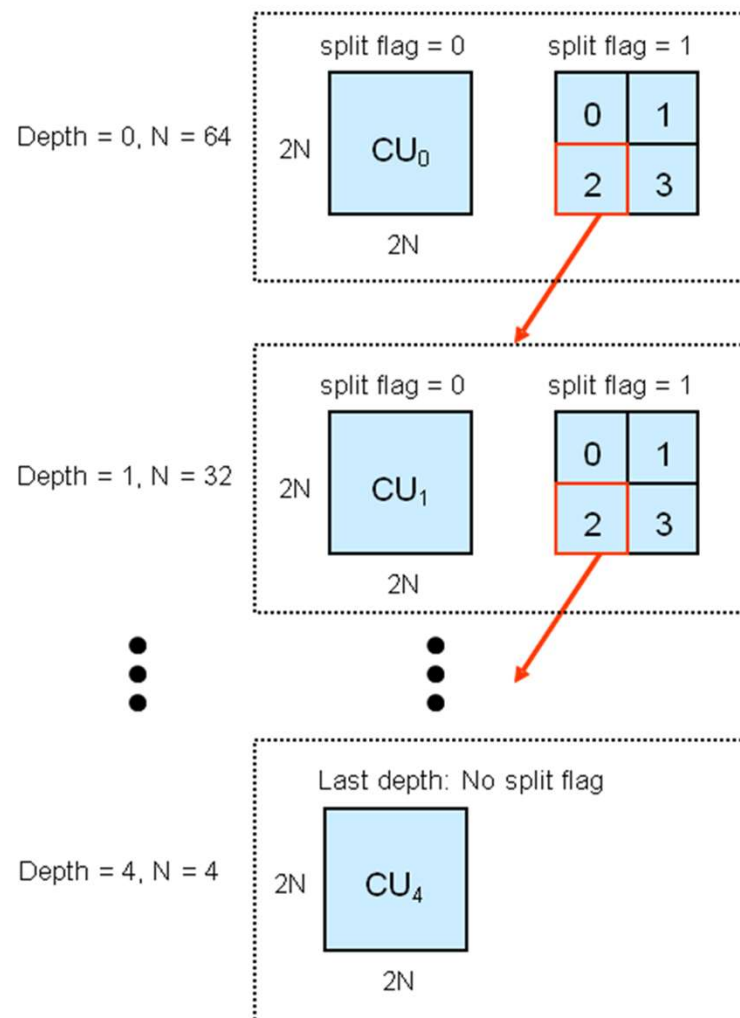
- Flexible block structure to support arbitrary min & max unit sizes
 - Coding Unit (CU)
 - Prediction Unit (PU)
 - Transform Unit (TU)
- Consistent syntax representation, independent of size
- Asymmetric motion partitions
- Greater than $\frac{1}{4}$ pixel motion accuracy with new interpolation filter
- Large integer transforms up to 64x64
- New rotational transform
- New motion vector prediction method
- New in-loop filtering methods
- New intra-coding prediction methods
- New entropy coding with explicit scan order signaling

Introduction: Building Blocks in Decoder



Unit Definition: Coding Unit (CU)

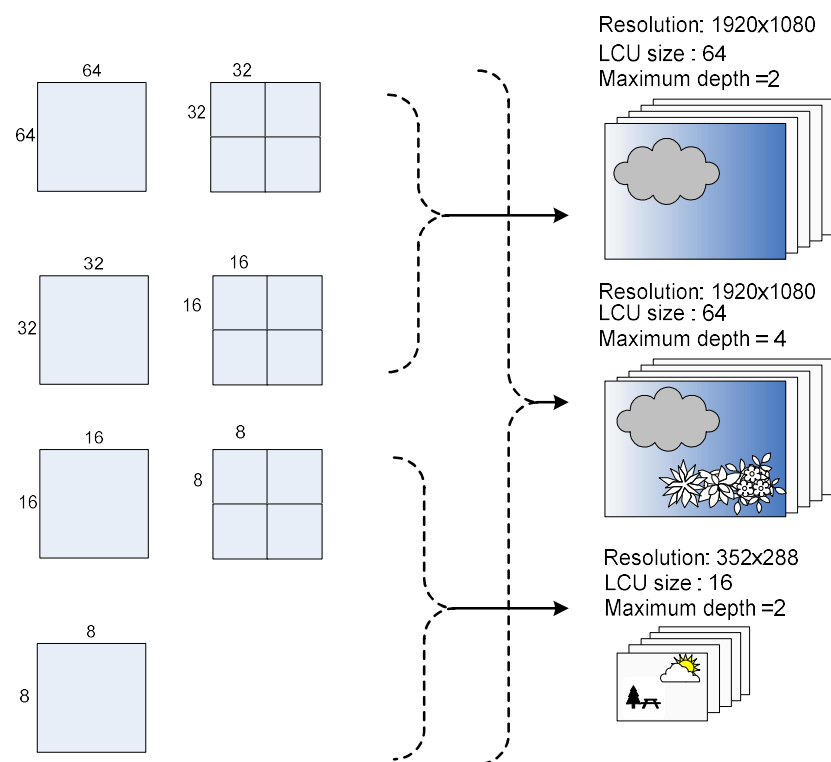
- **CU** is the basic processing block
 - Used for quad-tree based segmentation of regions
 - Plays a similar role to macroblock
 - Can take various sizes
 - Always power of 2 size
 - Always square shape
- Range of allowed sizes specified in Sequence Parameter Set
 - Largest CU (LCU)
 - Maximum hierarchical depth
 - Easily adapted for various applications
- Recursive structure with split flag
 - Single $2N \times 2N$ or four $N \times N$



LCU size = 128 ($N=64$), maximum hierarchical depth = 5

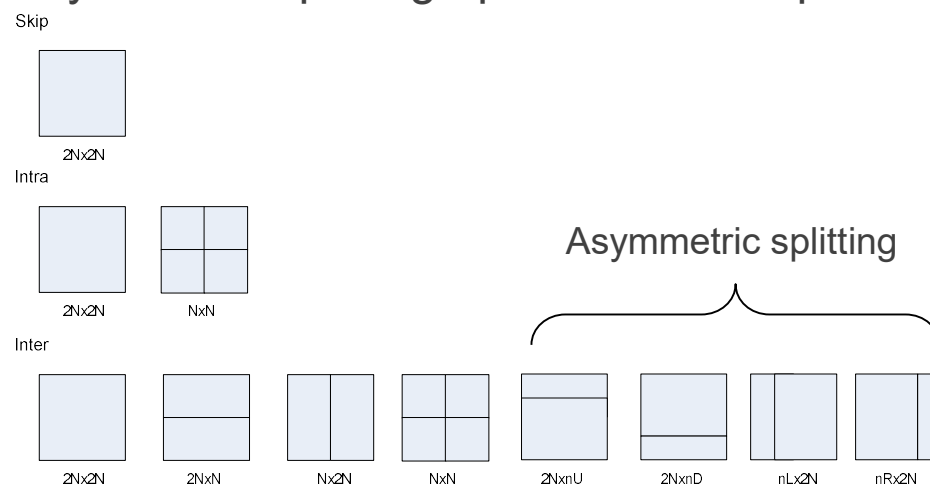
Unit Definition: Benefits of CU structure

- Supports large CU size
 - Virtually no limit to maximum size
 - Maximum of 128x128 used in CfP submissions
- Flexible structure
 - Can be optimized for content, device or application
- Size-independent syntax
 - Each CU has an identical syntax regardless of its size
 - Reduces complexity of parsing



Unit Definition: Prediction Unit (PU)

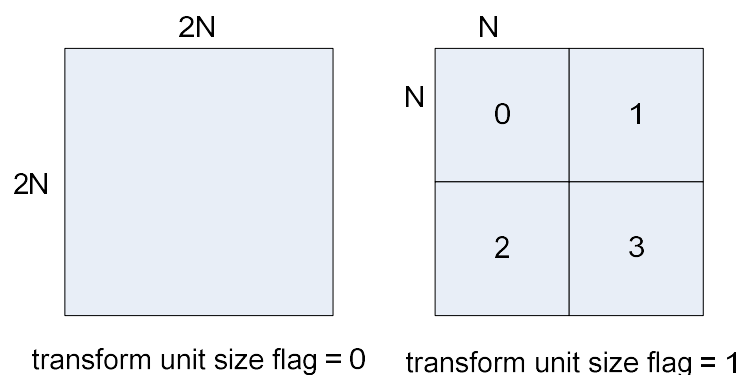
- Prediction Unit (**PU**) is the basic unit for prediction
 - Largest allowed PU size is equal to the CU size
 - Other allowed PU sizes depend on prediction type
 - Includes asymmetric splitting options for inter-prediction



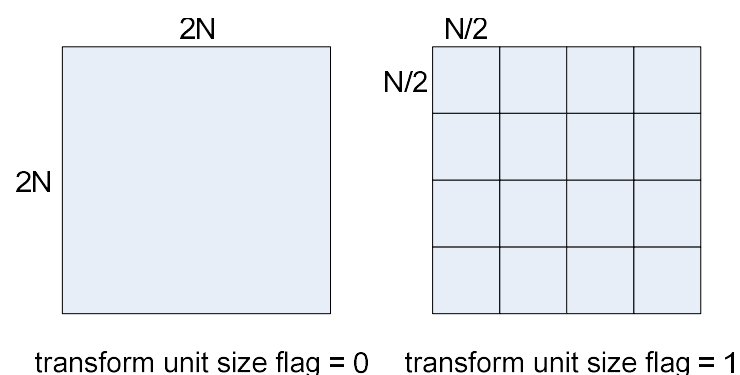
- Example of 128×128 CU
 - Skip: PU = 128×128
 - Intra: PU = 128×128 or 64×64
 - Inter: PU = 128×128 , 128×64 , 64×128 , 64×64 , 128×32 , 128×96 , 32×128 or 96×128

Unit Definition: Transform Unit (TU)

- Transform Unit (**TU**) is the basic unit for transform and quantization
 - May exceed size of PU, but not CU
- Only two TU options are allowed, signalled by transform unit size flag
 - Transform unit size flag = 0 → $2N \times 2N$ - same as CU
 - Transform unit size flag = 1 → square units of smaller size
 - $N \times N$ when PU splitting is symmetric
 - $N/2 \times N/2$ when PU splitting is asymmetric

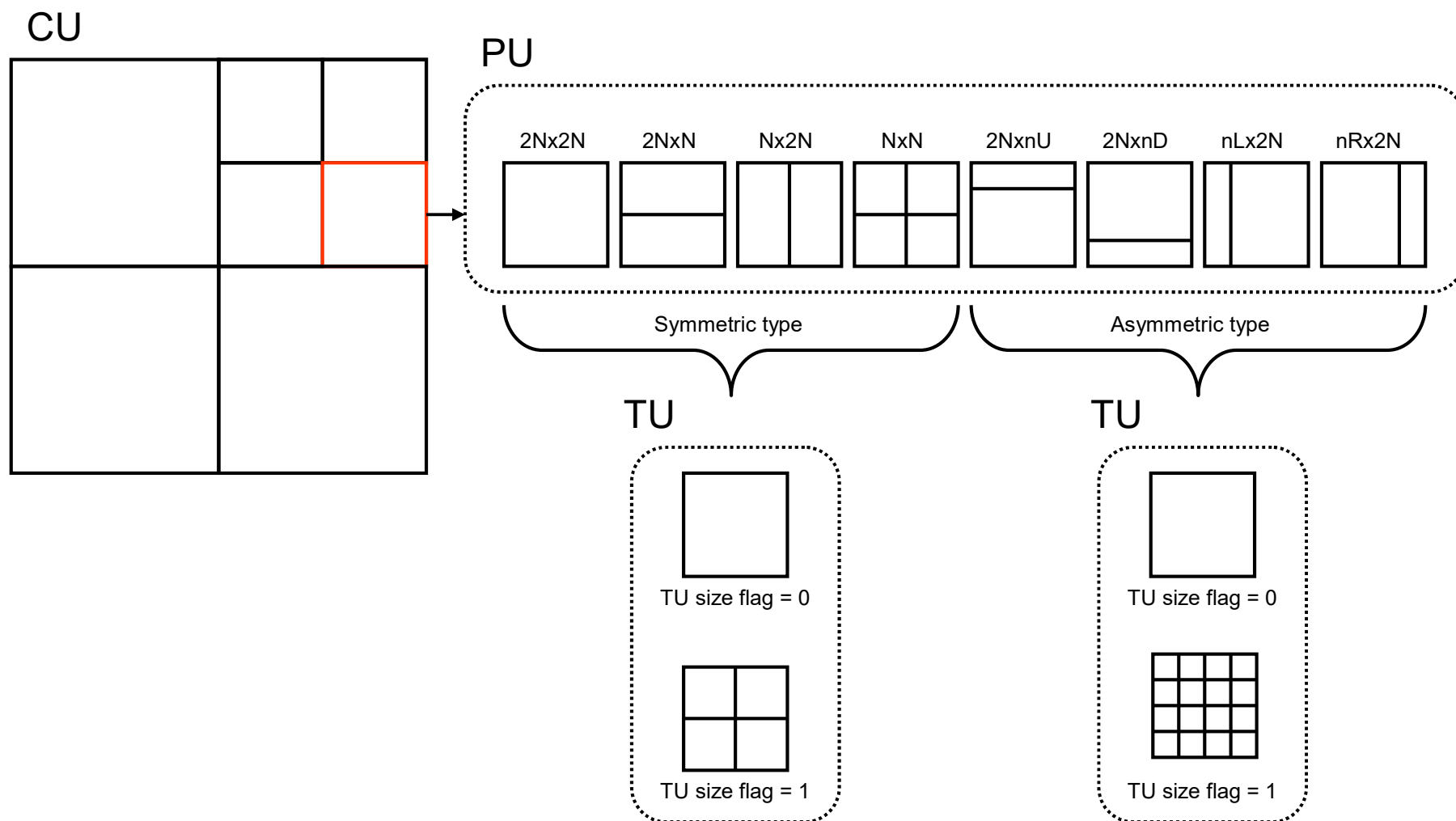


(a) $2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$ case



(b) $2N \times nU$, $2N \times nD$, $nL \times 2N$, $nR \times N$ case

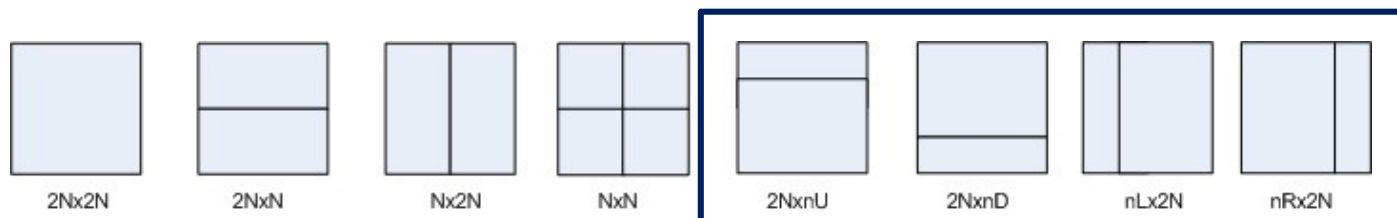
Unit Definition: Relationship of CU, PU and TU



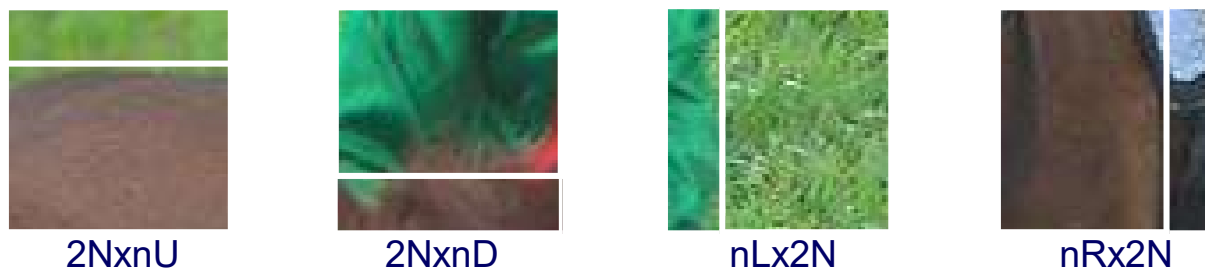
Motion: **Asymmetric Motion Partition (AMP)**

Note: Not included in A125

- Asymmetric motion partition (**AMP**)
 - Describes various object motions efficiently without further splitting
 - Computationally efficient compared to non-rectangular partitions
 - Motion estimation, motion compensation, transform, etc.



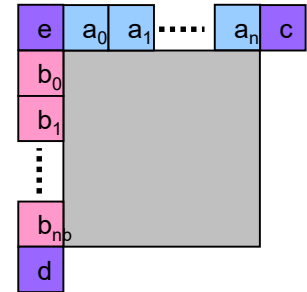
PU types for AMP



Examples of use of AMP (from RaceHorses in Class C)

Motion: **Advanced Motion Vector Prediction (AMVP)**

- Advanced Motion Vector Prediction (**AMVP**)
 - Extension of motion vector competition techniques
- Explicit motion vector predictor signaling
 - New candidate motion vectors
(motion vector candidates = {median(a' , b' , c'), a' , b' , c' , temporal predictor})
 - Three spatial motion vectors (a' , b' , c')
 - The first available one for each group (inter mode & same ref. idx)
 - Groups are the above group $\{a_0, a_1, \dots, a_{na}\}$, the left group $\{b_0, b_1, \dots, b_{nb}\}$ and the corner $\{c, d, e\}$
 - Median motion vector of three spatial motion vectors
 - Temporal motion predictor using one colocated motion vector
- Signaling overhead is minimized
 - Candidate order is adapted according to PU splitting
 - Unnecessary or duplicated motion vectors are removed

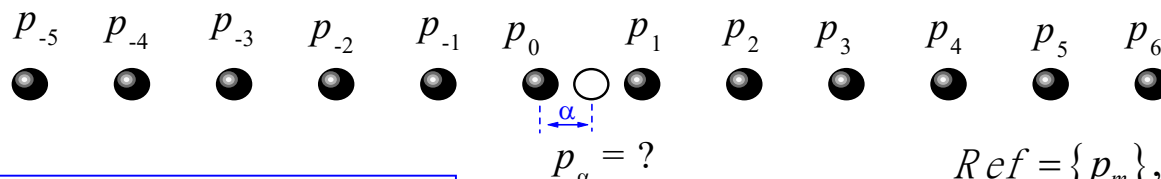


Motion: Improved Skip and Direct modes

- Improved Skip and Direct provide intermediate complexity modes
 - Skip and direct modes are enabled for both P and B slices
 - Differentiated only by whether texture information is sent or not
 - The motion of skip and direct modes is derived by AMVP
 - The motion vector prediction information is sent
 - AMVP index information is sent to determine motion predictor
 - There are three kinds of direct mode in B slice
 - Two uni-directional direct modes and a bi-directional direct mode

Motion: DCT-based interpolation filter (DIF)

- **DIF** provides an elegant method of high-accuracy interpolation
 - Direct fractional pixel generation replaces Wiener + bi-linear combination
 - Only one filtering is used to generate pixels at any accuracy
 - Mathematically, it is a forward DCT followed by inverse DCT with shifted argument of basis functions
 - Supports any accuracy & filter length
 - Implemented as a multiplication-free spatial domain filter



$$Ref = \{p_m\}, m = -(M-1), \dots, M$$

Forward DCT transform

$$C = D \times Ref$$

$$D_{km} = \frac{1}{M} \cos\left(\frac{(2m-1+2M)k\pi}{4M}\right)$$

Inverse DCT transform with shifted argument

$$p_\alpha = W(\alpha) \times C$$

$$W_0(\alpha) = \frac{1}{2}, \quad W_k(\alpha) = \cos\left(\frac{\pi(2\alpha-1+2M)k}{4M}\right)$$

Spatial domain filter

$$p_\alpha = \sum_{m=-M+1}^M Filter_m(\alpha) \times p_m$$

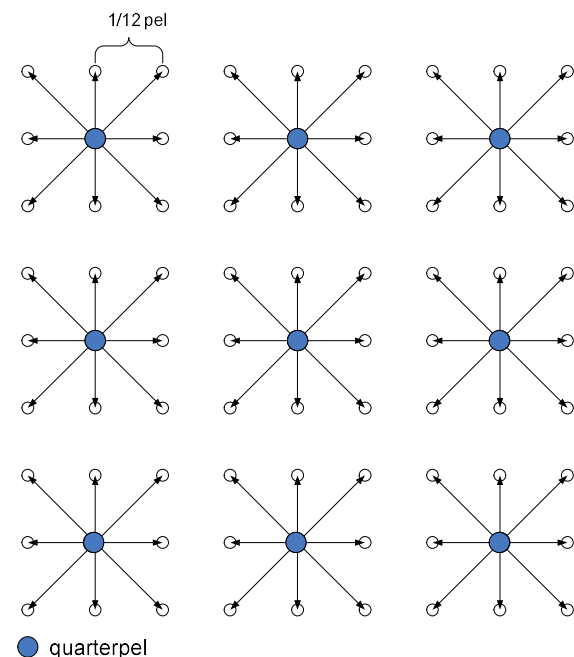
$$Filter_m(\alpha) = \sum_k W_k(\alpha) D_{km}$$

merged

Motion: **High Accuracy Motion (HAM)**

Note: Not included in A125

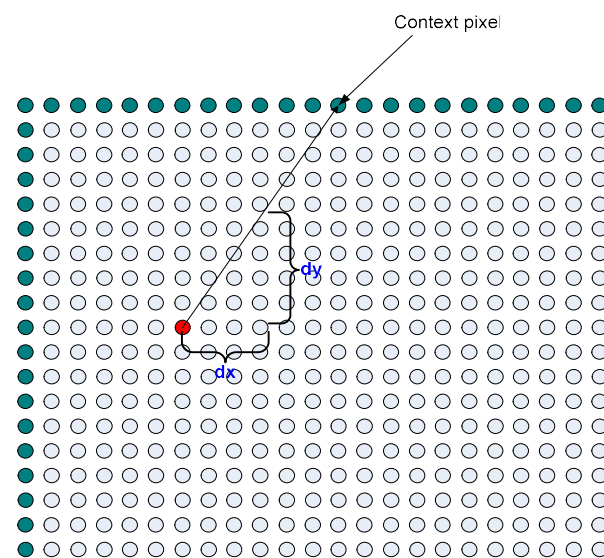
- High Accuracy Motion (**HAM**) provides
 - Higher motion accuracy than $\frac{1}{4}$ pel
- Proposal uses a refinement representation
 - Motion vector (lower accuracy, e.g. $\frac{1}{4}$ pel) + refinement (0, -1, +1)
 - 1 bit overhead when refinement not used
 - Smaller overhead to always $\frac{1}{8}$ design
 - No negative gain sequences
 - Prediction is used only for lower accuracy MV
 - To prevent randomness of MVD
 - Smaller MVD magnitude
- Current design uses $\frac{1}{12}$ pel accuracy
 - More compact coverage than $\frac{1}{8}$ pel



Intra: Arbitrary Direction Intra (ADI)

- Arbitrary Direction Intra (**ADI**) provides improved directional prediction
 - Prediction of any direction is defined by the delta value (dx, dy) from current pixel to the corresponding reference pixel: $Y[x, y] = Y[x-dx, y-dy]$
 - Left down pixels are possible reference pixels
 - Filtering of boundary reconstructed pixels before prediction
- Number of prediction modes dependent on PU size
 - Up to 33 prediction modes

PU size	Number of modes
4x4	9
8x8	9
16x16	33
32x32	33
64x64	5
128x128	5

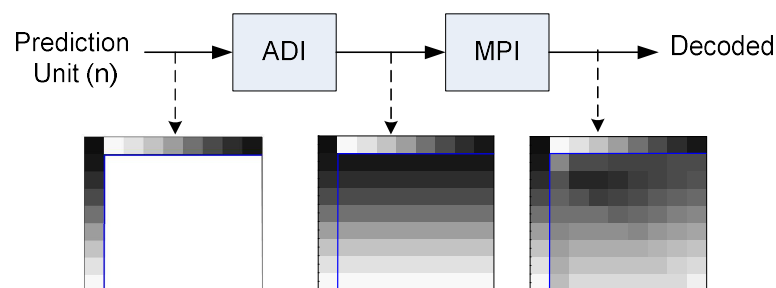


Prediction generation with arbitrary direction

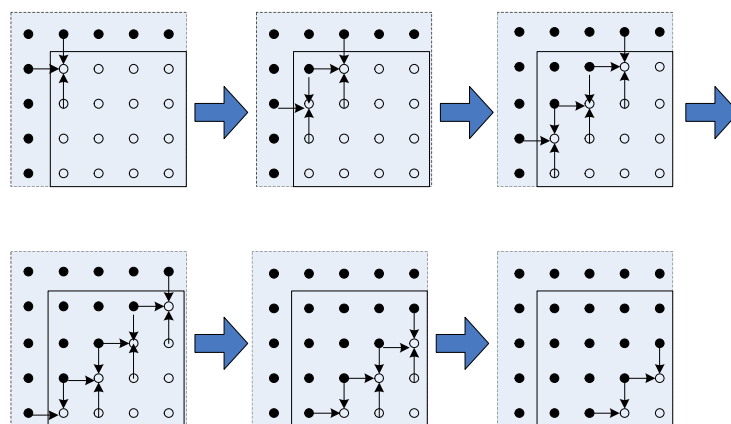
Intra: Multi-Parameter Intra (MPI)

Note: Not included in A125

- Multi-Parameter Intra (**MPI**) provides more natural prediction patterns
 - Uses a 4-point filter for each pixel inside the predicted block



$$\text{pred}'[x,y] = (\text{pred}[x,y] + \text{pred}[x-1,y] + \text{pred}[x,y-1] + \text{pred}[x,y+1] + 2) \gg 2$$



Intra: Color Component Correlation Prediction (CCCP)

Note: Not included in A125

- **CCCP** improves chroma intra prediction by using information inferred from reconstructed luma samples
 - Chroma intra prediction based on segmentation map from luma samples
 - Capable of generating complex object shapes



Original Chroma signal



Prediction of H.264/AVC

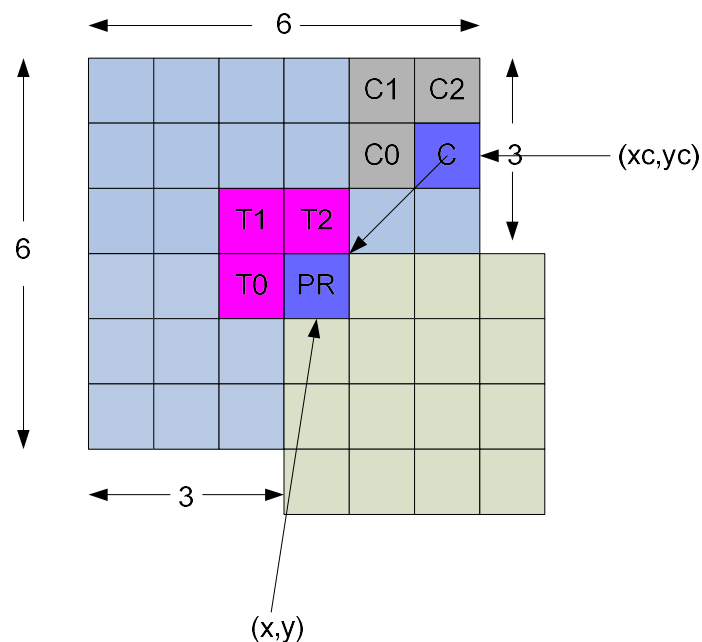


Prediction of proposed method
(CCCP replace DC mode)

Intra: Pixel based template matching (PTM)

Note: Not included in A125

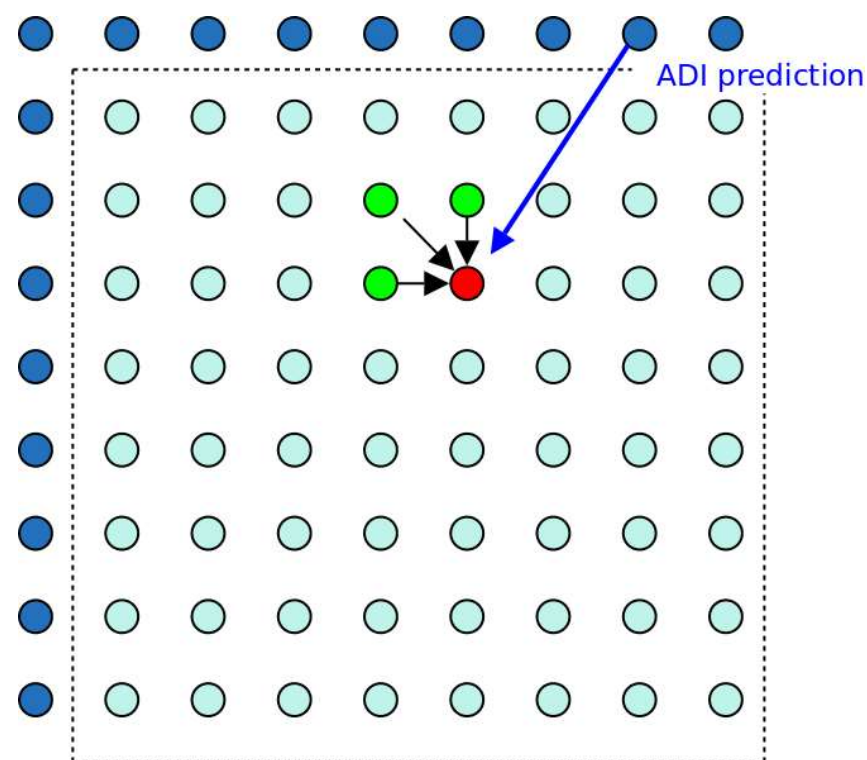
- Pixel based template matching (**PTM**) improves intra prediction in regions with repeated regular patterns
 - L-shaped search region, including already predicted samples in template
- Want to predict PR
 - Use T0, T1, T2 as template of size 6x6
 - Total 27 points are searched
 - Previously predicted pixels are reused as candidate and template
 - Choose pixel C if it gives min. SAD



Intra: Combined Intra Prediction (CIP)

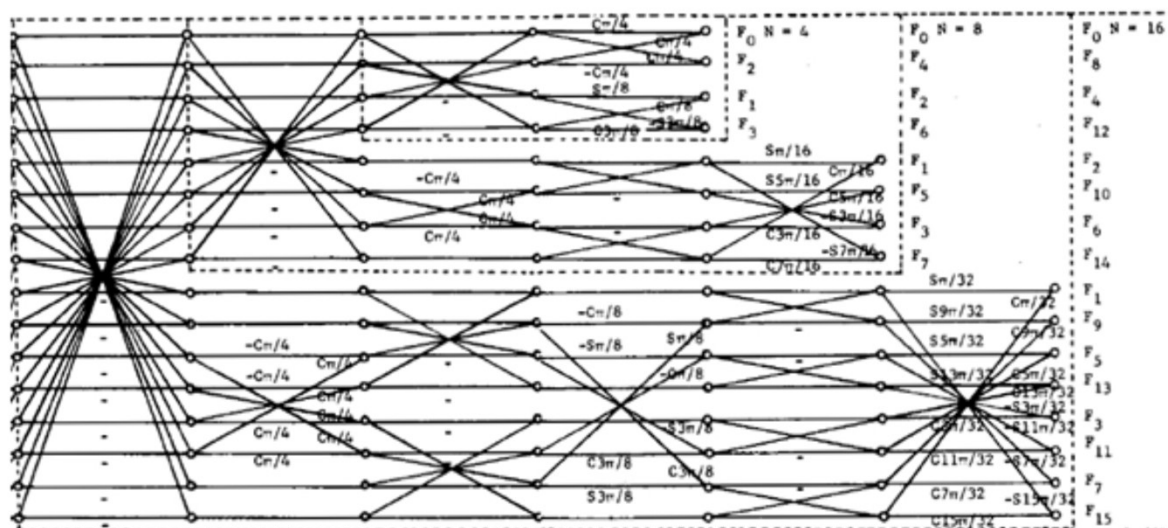
Note: Not included in A124

- Combined Intra Prediction (**CIP**) improves other prediction methods by allowing pixel-by-pixel adaptation
- In A125, ADI predictions are combined with a local mean within a block
- Forward prediction using the local mean is open-loop
 - any noise is damped by the combination factors and more than compensated by a better, adaptive prediction



Transform: Large Transform

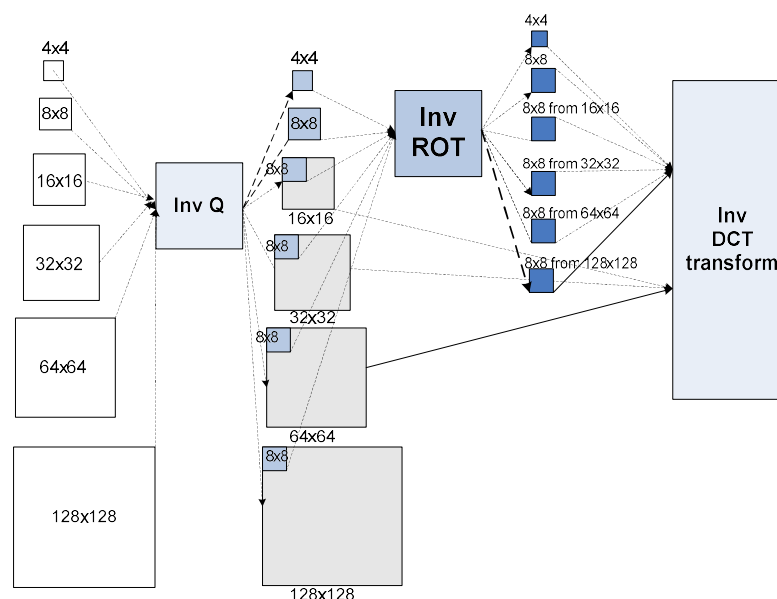
- The proposal extends transform to larger sizes
 - 16x16, 32x32 and 64x64
- Minimising complexity is important in large transform design
 - Chen's fast DCT has been chosen for this proposal
 - Reduced implementation complexity due to the regular butterfly design
 - Approximation of values from sinusoidal functions into dyadic rationals
 - Can be implemented using additions and shifts only



Transform: Rotational Transform (ROT)

Note: Not included in A125

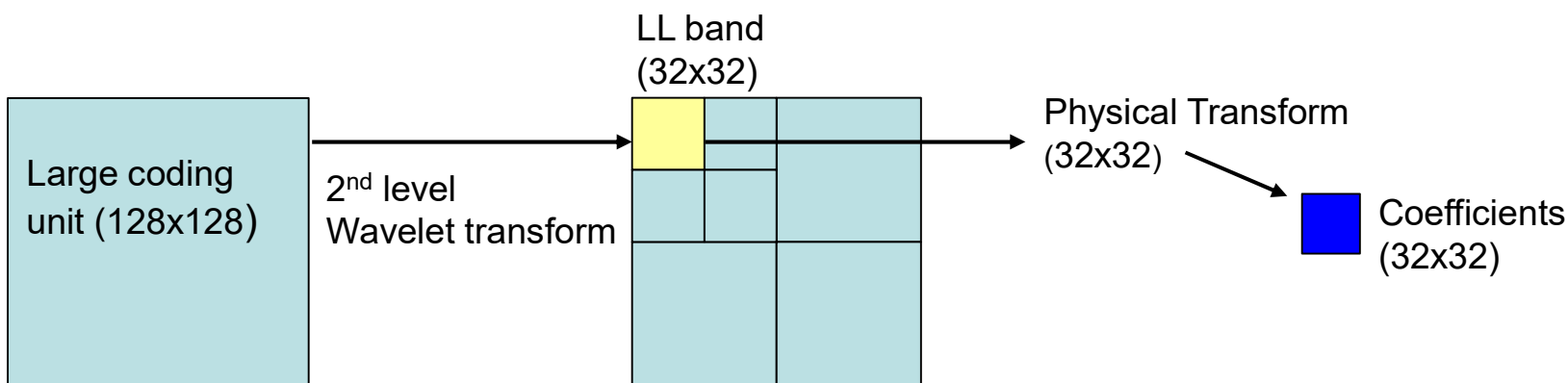
- The Rotational Transform (**ROT**) provides a way to rotate DCT basis
 - Designed as 2nd tranform after DCT: can be applied with any transforms
 - Similar to directional transform, but simpler approach
- Implementation cost is minimized in this proposal by
 - Allowing only four possible rotation angles
 - Excluding areas outside of the 8x8 low frequency area – advantages from transform domain processing



Transform: **Logical transform (LOT)**

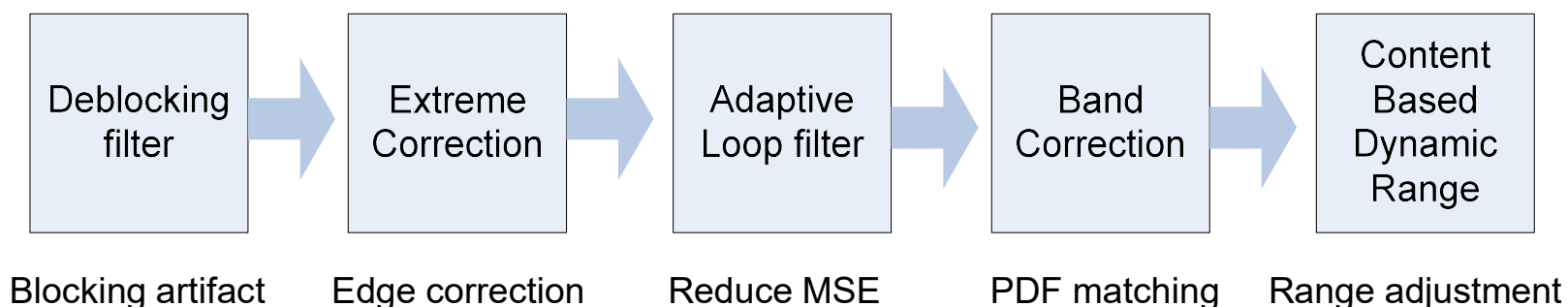
Note: Not included in A125

- The Logical Transform (**LOT**) allows the input residual size to be bigger than the maximum physical transform size
 - Roughly equivalent to taking only low-frequency components of DCT
 - Beneficial in coding smooth regions
 - Wavelet transform is followed by down-sampling and conventional transform
 - only LL-band signals are transformed by spatial transform

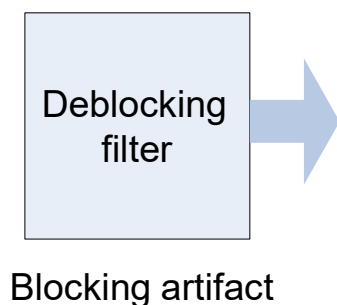


Loop Filtering: Overview of In-loop filtering

- The in-loop filter in A124 is a combination of several spatial processes



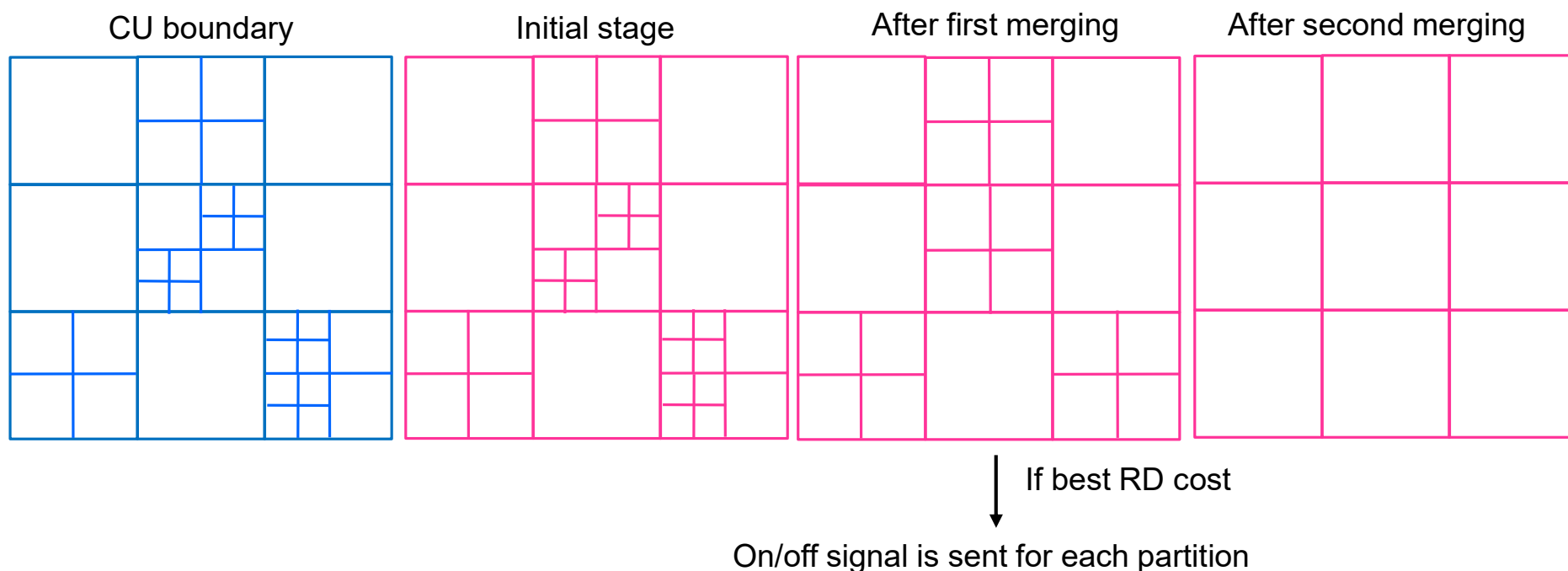
- The in-loop filter in A125 is only the Deblocking filter
 - Same filters and boundary strength decision as H.264/AVC



Loop Filtering: **CU-synchronized ALF**

Note: Not included in A125

- CU-synchronized Adaptive Loop Filter (**ALF**) further reduces distortion
 - On/off partition reuses CU boundaries - no need to transmit partition info.
 - Much simpler to estimate in encoder-side
 - Multi-level merging of CU boundary is supported
 - CU-synchronized ALF process can be implemented in decoder-side

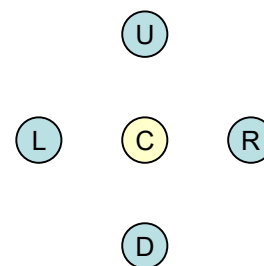


Loop Filtering: **Extreme correction (EXC)**

Note: Not included in A125

- Extreme correction (**EXC**) is useful to compensate distortion for specific pixel class, e.g. object edge
 - Extreme type is determined by comparison of current pixel value with upper, lower, left and right neighbors (for non-boundary pixels)
 - Location of points to be corrected are determined by decoder
 - Correction values are calculated for 6 types of extreme points as mean error among the frame

Category	Condition	Note
0	$P < 4$ neighbors	Local min
1	$P < 3$ neighbors && $P = 4^{\text{th}}$ neighbor	Object edge
2	$P < 3$ neighbors && $P > 4^{\text{th}}$ neighbor	Object edge
3	$P > 3$ neighbors && $P < 4^{\text{th}}$ neighbor	Object edge
4	$P > 3$ neighbors && $P = 4^{\text{th}}$ neighbor	Object edge
5	$P > 4$ neighbors	Local max



Extreme type derivation
for value of pixel P using
4 neighbours

Loop Filtering: **Band Correction (BDC)**

Note: Not included in A125

- Band Correction (**BDC**) allows the correction of systematic errors, related to specific ranges of pixel values
 - Conceptually similar to PDF matching process between two signals
 - Band may be defined by the p most significant bits of pixel value
 - Integer correction values for each band are determined while coding
 - Correction values for each band are coded in slice header

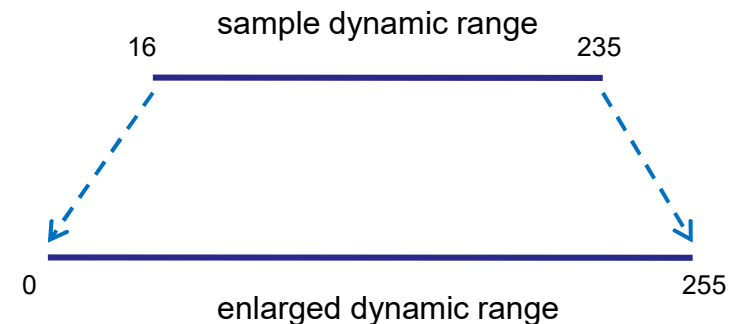
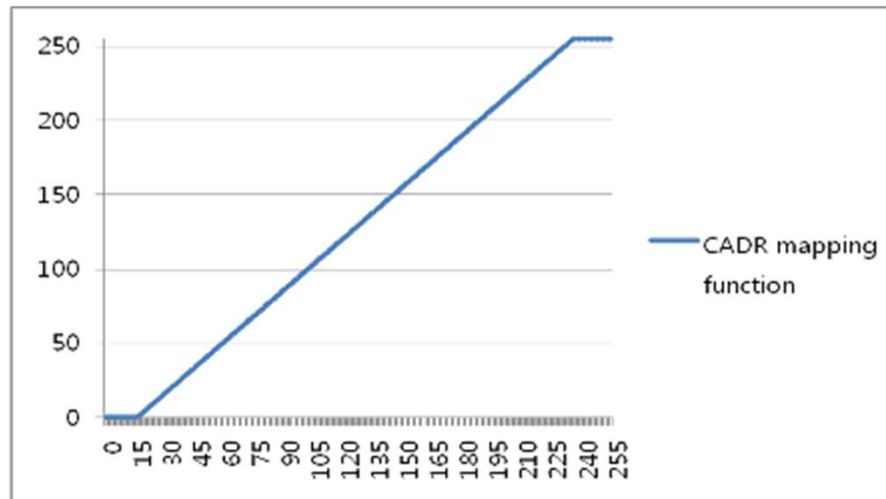
Example: Band derivation by 4 most significant bits for 12-bit depth of pixel values and correction values (PeopleOnStreet 1st frame).

Band number	0	1	...	15
Reconstructed pixel value band	0 - 255	256 - 511	...	3840 - 4095
Correction value	6	-4	...	8

Loop Filtering: **Content Adaptive Dynamic Range (CADR)**

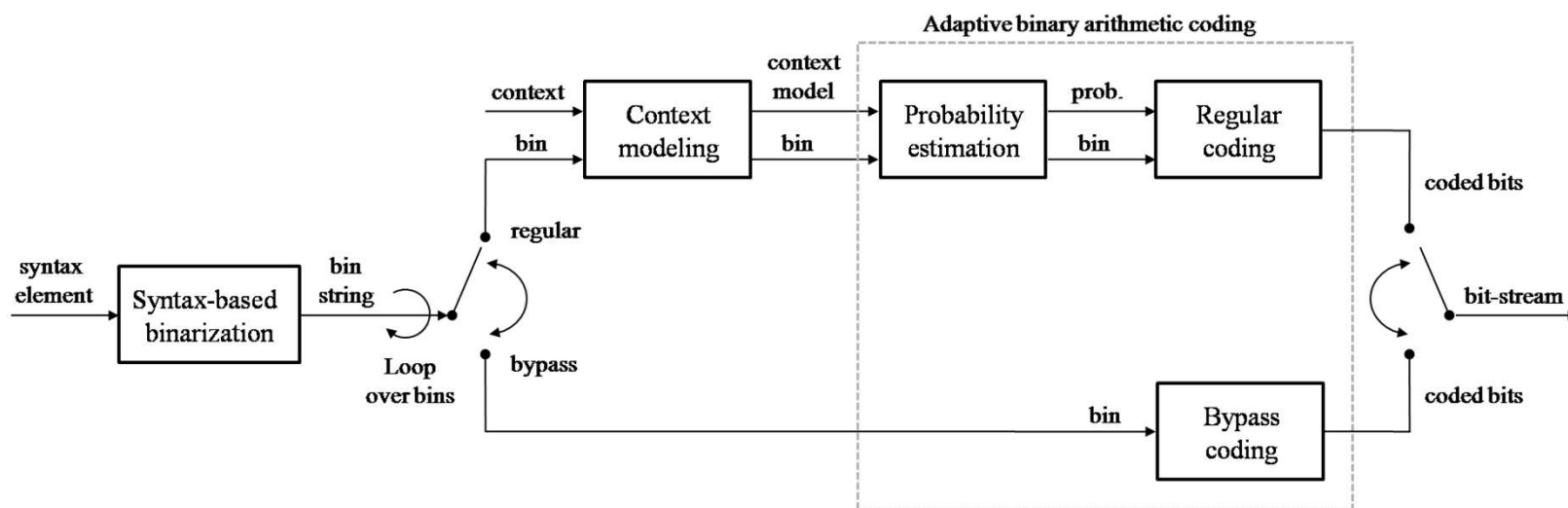
Note: Not included in A125

- Content Adaptive Dynamic Range (**CADR**) gives improved accuracy for internal calculations by exploiting known limits to luma samples
 - Without requiring increased bit depth – useful for bit-depth limited H/W
- For example, clipped BT.709 luma samples lie in the range [16, 235]
 - CADR mapping expands dynamic range to [0,255]



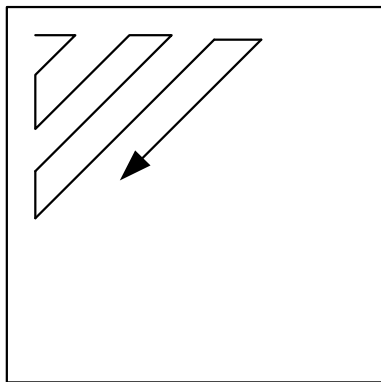
Entropy Coding: SBAC

- The proposal uses Syntax-based context-adaptive binary arithmetic coding (SBAC)
 - Coding engine is based on JPEG Annex D
 - Coding performance appears to be slightly better than H.264/AVC's CABAC
 - Overall architecture is similar to CABAC, but the details of each step are different

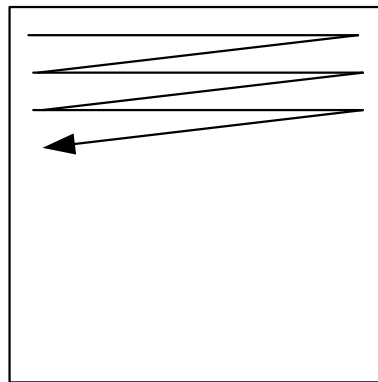


Entropy Coding: **Adaptive Coefficient Scanning (ACS)**

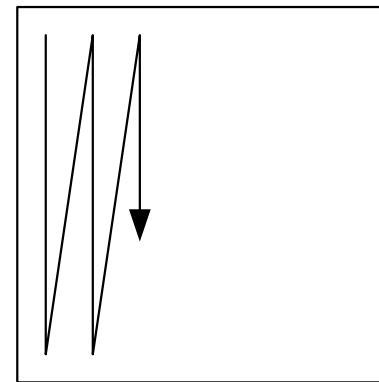
- Adaptive Coefficient Scanning (ACS) improves the coding performance when using large transform blocks
- Allows scanning pattern to be selected by encoder:
 - Conventional zig-zag
 - Horizontal scan
 - Vertical scan
- Only signalled when there are non-DC coefficients



zig-zag scan



horizontal scan

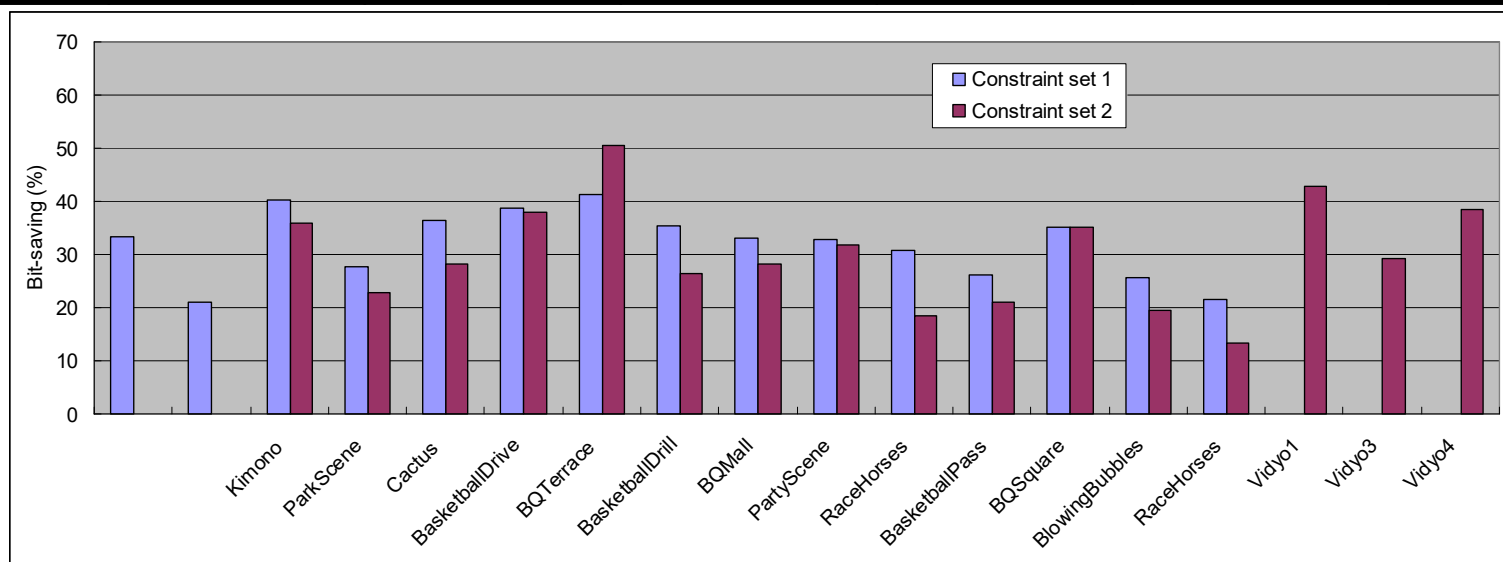


vertical scan

Compression Performance (A125)

- Average bit-saving 31.95% for CS1 and 29.97% for CS2
 - Best classes: Class B@CS1 and Class E@CS2
 - Worst class: Class D@CS2
 - Best sequence: BQTerrace@CS2 (50.55%)
 - Worst sequence: RaceHorses@CS2 (13.31%)

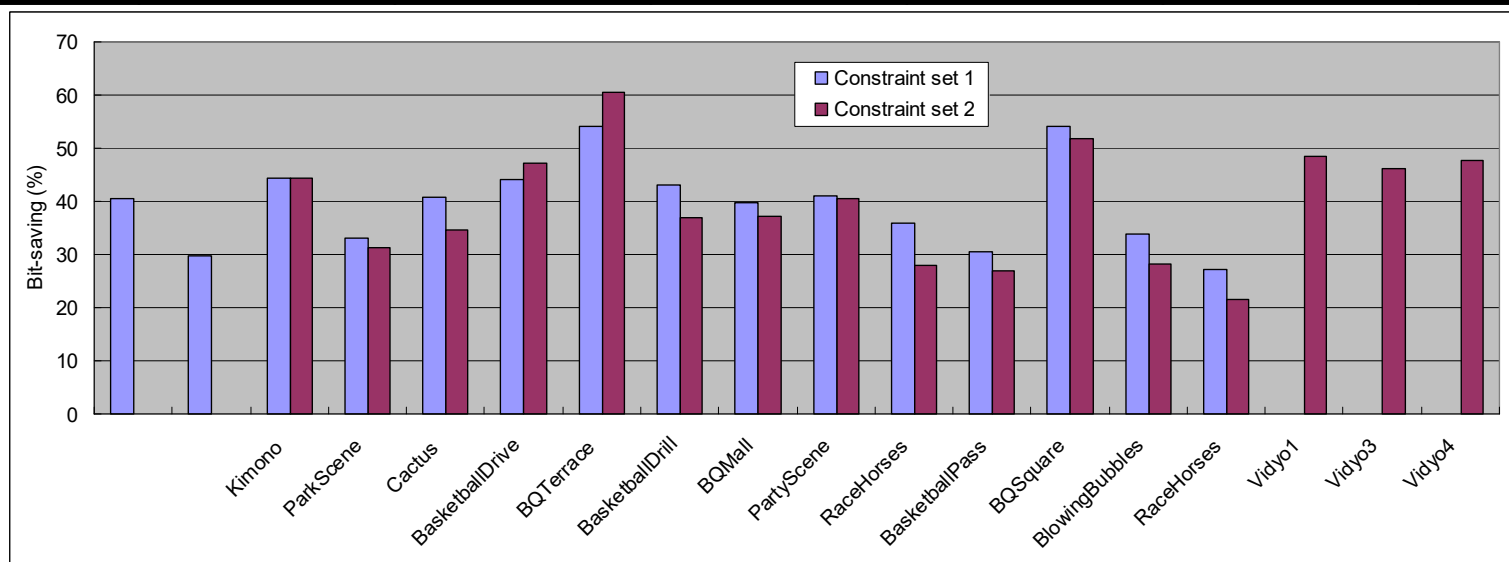
	Class A	Class B	Class C	Class D	Class E	Average
CS1	27.24	36.85	33.04	27.07		31.95
CS2		35.06	26.19	22.23	36.85	29.97



Compression Performance (A124)

- Average bit-saving 39.49% for CS1 and 39.48% for CS2
 - Best class: Class E@CS2
 - Worst class: Class D@CS2
 - Best sequence: BQTerrace@CS2 (60.62%)
 - Worst sequence: RaceHorses@CS2 (21.59%)

	Class A	Class B	Class C	Class D	Class E	Average
CS1	35.21	43.29	39.94	36.45		39.49
CS2		43.66	35.58	32.16	47.48	39.48



Complexity Analysis (A125)

- Decoding time using PC with fast SATA drive
 - Average decoding time about 1.3 times that of JM17.0

JM17.0	Class A	Class B	Class C	Class D	Class E	Average
CS1	x1.26	x0.94	x1.05	x1.38		x1.16
CS2		x1.14	x1.32	x1.58	x1.35	x1.35

- Decoding time using PC with SCSI drive
 - Average decoding time about 0.6 times that of JM17.0

JM17.0	Class A	Class B	Class C	Class D	Class E	Average
CS1	X0.30	X0.31	X0.53	X1.13		X0.57
CS2		X0.32	X0.49	X1.21	X0.39	X0.60

Complexity Analysis (A124)

- Decoding time using PC with fast SATA drive
 - Average decoding time about 2.4 times that of JM17.0

JM17.0	Class A	Class B	Class C	Class D	Class E	Average
CS1	x3.01	x2.26	x1.93	x2.45		x2.41
CS2		x2.67	x2.20	x2.63	x1.64	x2.29

- Decoding time using PC with SCSI drive
 - Average decoding time about 0.9 times that of JM17.0

JM17.0	Class A	Class B	Class C	Class D	Class E	Average
CS1	X0.64	X0.61	X0.79	X1.45		X0.87
CS2		X0.62	X0.85	x1.51	X0.48	X0.87

Further improvements after submission (A124)

- Average bit-saving is now 41.58% for CS1
 - 2.09% better than in submitted proposal

Class	Sequence	Submitted	Improved	Delta
A	Traffic	40.59	42.99	2.40
	PeopleOnStreet	29.84	31.20	1.36
B	Kimono	44.42	45.80	1.37
	ParkScene	32.96	36.09	3.13
	Cactus	40.78	43.28	2.49
	BasketballDrive	44.21	45.11	0.90
	BQTerrace	54.06	55.96	1.90
C	BasketballDrill	43.09	44.56	1.47
	BQMall	39.69	41.94	2.25
	PartyScene	41.13	43.36	2.23
	RaceHorses	35.86	37.11	1.25
D	BasketballPass	30.55	33.45	2.90
	BQSquare	54.19	56.92	2.73
	BlowingBubbles	33.97	36.95	2.98
	RaceHorses	27.08	28.95	1.86

Class	Submitted	Improved	Delta
A	35.21	37.09	1.88
B	43.29	45.25	1.96
C	39.94	41.74	1.80
D	36.45	39.07	2.62
ALL	39.49	41.58	2.09

Newly added tools

- Skip & direct mode using HAM
- New deblocking filter design
- Bi-directional prediction refinement

Conclusions

- The Samsung/BBC coding framework has been described in some detail
- In our responses to the CfP we demonstrated two key operating points
 - A low-complexity operating point, with comparable complexity to H.264/AVC and better compression efficiency
 - Average efficiency about 30% better than Alpha and Beta anchors
 - Decoding time about 0.6 to 1.3 times that of JM17.0
 - A high-performance operating point, giving even higher compression efficiency with a moderate increase in complexity over H.264/AVC
 - Average efficiency about 40% better than Alpha and Beta anchors
 - Decoding time about 0.9 to 2.4 times that of JM17.0
- The Samsung/BBC coding framework should be considered to be a strong candidate for the Test Model that will be used as the basis of the Core Experiments in the next phase of HVC standardization