

# **CE10: Fast integer transform based on modified Loeffler's factorization**

---

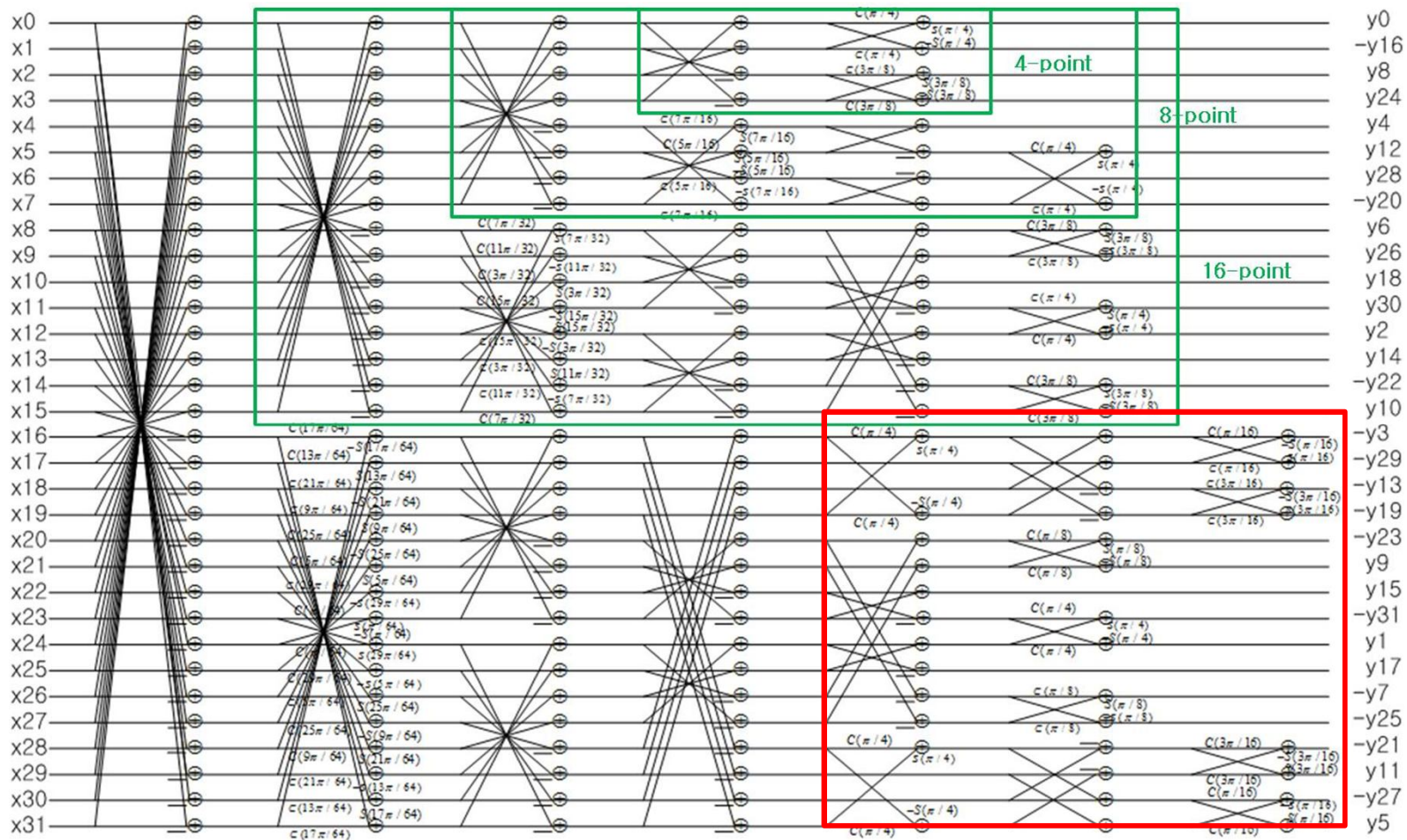
**Il-Koo Kim, Tammy Lee, Jianle Chen,  
Vadim Seregin, Yoon Mi Hong, Woo-Jin Han  
(Samsung)  
Pankaj Topiwala (FastVDO)**

# Summary

- ❖ Fast integer transform based on modified Loeffler's factorization
  - a) it can be implemented with both factorized butterfly form or matrix multiplication form
  - b) with factorized butterfly form, number of arithmetic operations is significantly less than matrix multiplication
  - c) 4x4, 8x8 and 16x16 transforms are embedded in 32x32 transform
  - d) input signal and intermediate transpose buffer for inverse transform are represented within 16 bit
  - e) six scalar values are used to replace de-quantization scaling matrix for all transform size

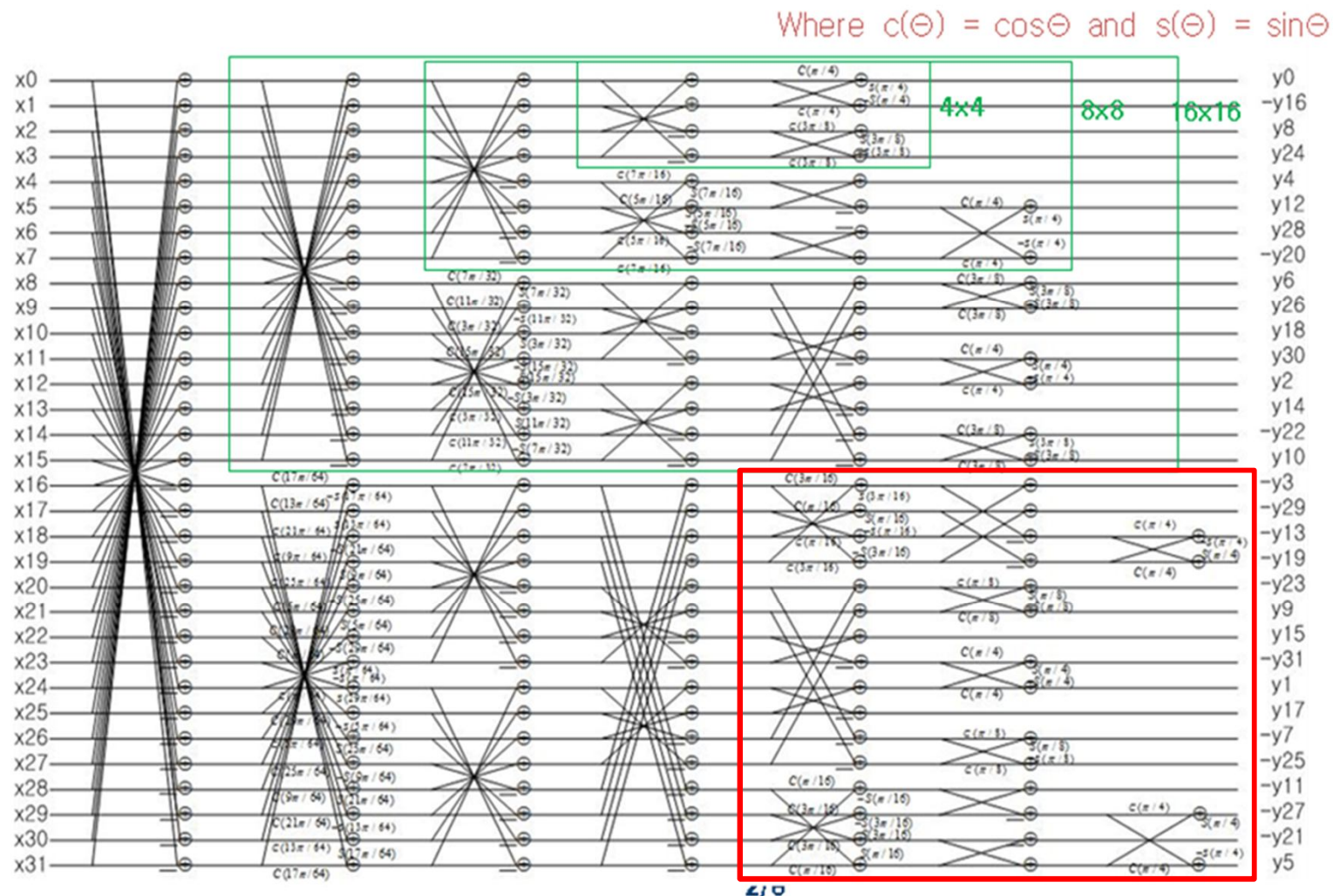
# Loeffler's factorization

- ❖ Signal flow graph of fast integer transform proposed in JCTVC-D365
  - 7 stages



# Modified factorization

- ❖ Modified signal flow graph with simplified last stages (red box)

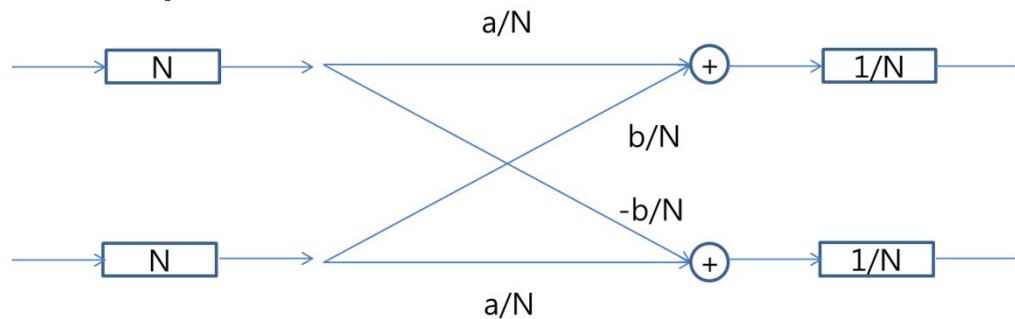


# Matrix conversion

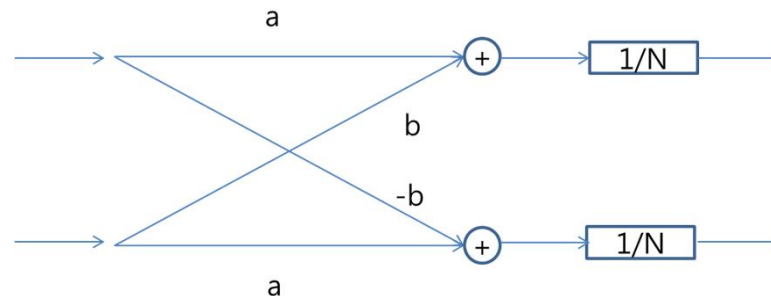
- ❖ Factorized butterfly can be converted to one matrix, however...
  - Previous implementation requires right shift in transform core
  - Lossless conversion is not possible due to rounding error
- ❖ Method 1) Pre-scaling of input
  - Pre-scale input signal to remove rounding error completely
  - Need proper post-scaling
- ❖ Method 2) Right shift removal in transform core
  - Instead of pre-scaling
  - Dyadic rational is mapped to integer by removing denominator appropriately
    - Except pre and post scaling, no right shifts are involved
  - Need proper post-scaling
- ❖ Method 2 is selected as final design presented here

# Matrix conversion

## ❖ Simple example



- Removing rounding error in butterfly with pre- and post-scaling

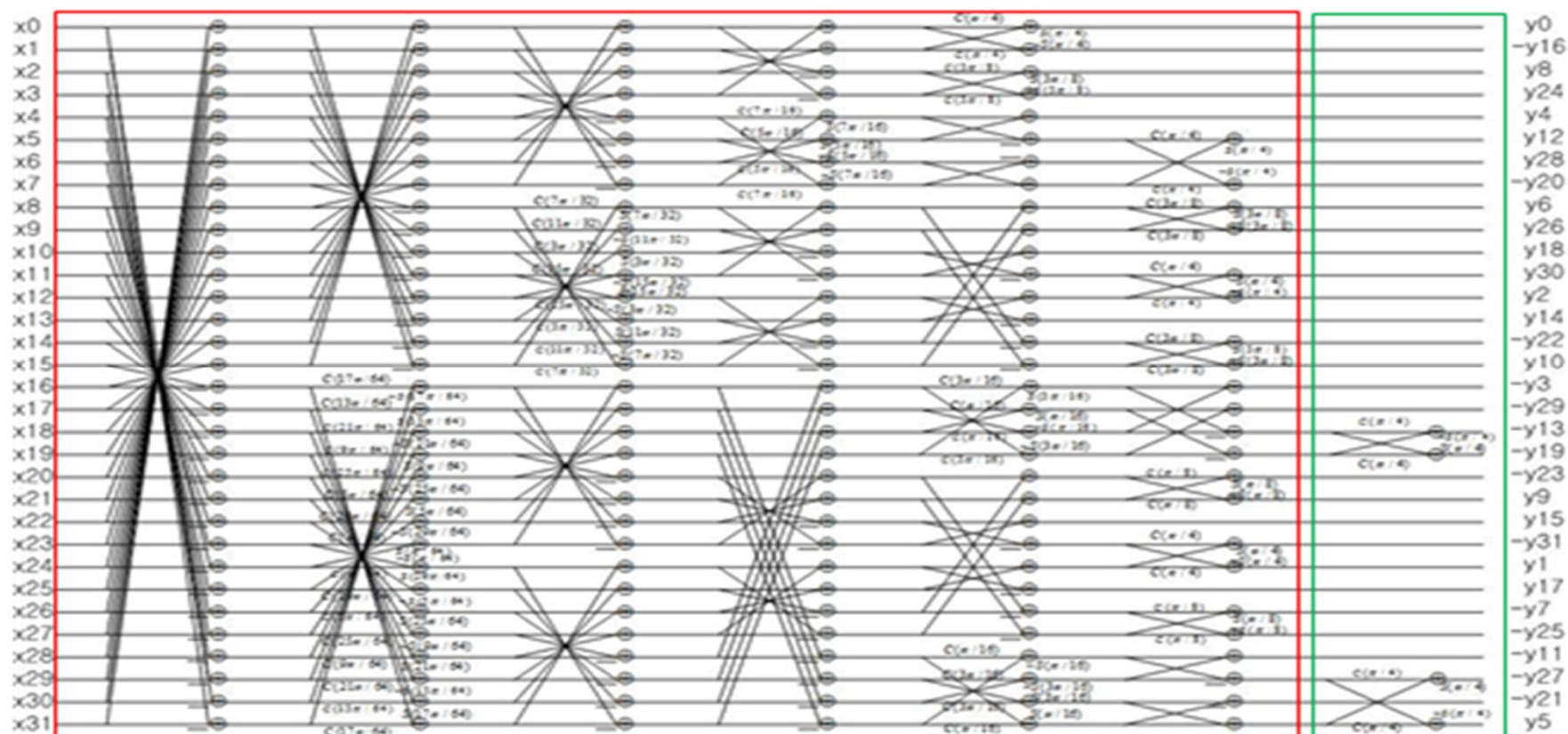


- Removing rounding error in butterfly with integer multiplication and post-scaling
- If multiple stages are used for transform design, the post-scaling can be moved to last stage. This modification can remove redundant right shift operations.



# Matrix conversion

- ❖ Split of signal flow graph for 32x32 transform
  - Only 6 stages of total 7 stages are considered to convert to matrix
  - Last stage can be incorporated into scaling/de-scaling stages



# Matrix conversion

- ❖ 16x16 transform matrix (For 32x32 transform, please see JCTVC-E277)
  - 11 bit representation is possible
  - Derived from 32x32 transform matrix directly
  - 4x4 and 8x8 transforms can be derived from this matrix

704	968	992	936	928	868	814	768	704	640	550	456	384	292	192	88
704	946	832	632	384	99	-198	-480	-704	-896	-946	-992	-928	-771	-544	-286
704	858	544	99	-384	-762	-946	-960	-704	-288	198	621	928	978	832	462
704	748	192	-459	-928	-935	-550	96	704	992	814	285	-384	-863	-992	-616
704	616	-192	-863	-928	-285	550	992	704	-96	-814	-935	-384	459	992	748
704	462	-544	-978	-384	621	946	288	-704	-960	-198	762	928	99	-832	-858
704	286	-832	-771	384	992	198	-896	-704	480	946	99	-928	-632	544	946
704	88	-992	-292	928	456	-814	-640	704	768	-550	-868	384	936	-192	-968
704	-88	-992	292	928	-456	-814	640	704	-768	-550	868	384	-936	-192	968
704	-286	-832	771	384	-992	198	896	-704	-480	946	-99	-928	632	544	-946
704	-462	-544	978	-384	-621	946	-288	-704	960	-198	-762	928	-99	-832	858
704	-616	-192	863	-928	285	550	-992	704	96	-814	935	-384	-459	992	-748
704	-748	192	459	-928	935	-550	-96	704	-992	814	-285	-384	863	-992	616
704	-858	544	-99	-384	762	-946	960	-704	288	198	-621	928	-978	832	-462
704	-946	832	-632	384	-99	-198	480	-704	896	-946	992	-928	771	-544	286
704	-968	992	-936	928	-868	814	-768	704	-640	550	-456	384	-292	192	-88



# De-quantization

## ❖ De-quantization with 6 scalar values

- LevelScale[6] = {21, 23, 28, 30, 23, 38},

$$d_{ij} = ( c_{ij} * \text{LevelScale}[ qP \% 6 ] ) \ll ( qP / 6 ), \text{ with } i, j = 0..nS-1$$

## ❖ Intermediate data bit depth restriction

- Restrict the dynamic range of inverse transform input signal with 16 bits representation

$$d'_{ij} = \text{Clip3}(-2^{15}, 2^{15}-1, ( d_{ij} + \text{offset} ) \gg \text{PreScaleBitblk}), \text{ with } i, j = 0..nS-1$$

- PreScaleBitblk is equal to 0, 1, 2, 3 respectively for transform block 4x4, 8x8, 16x16 and 32x32

# Complexity:

## ❖ Number of arithmetic operations

- For 16x16 and 32x32 transform, total number of operations is less than that of HM-2.0
- And factorization implementation has significantly less number of arithmetic operations than matrix multiplication implementation, especially for large size transform.

32x32	addition	multiplication	shift	total
HM-2.0	13440	7424	4224	25088
Matrix	66816	65792	2304	134912
Factorization	14976	6656	2944	24576

16x16	addition	multiplication	shift	total
HM-2.0	2624	1408	832	4864
matrix	8448	8192	512	17152
factorization	2560	1344	768	4672

# Complexity:

## ❖ Number of arithmetic operations

- For 4x4 and 8x8 transform, HM-2.0 has less number of operations than proposed transform
  - 4x4 and 8x8 transform in HM-2.0 are same with AVC's (multiplication free design)

8x8	addition	multiplication	shift	total
HM-2.0	576	0	160	736
matrix	1088	1024	128	2240
factorization	480	256	224	960

4x4	addition	multiplication	shift	total
HM-2.0	80	0	16	96
matrix	144	128	32	304
factorization	80	48	64	192

# Complexity:

## ❖ Bit-width analysis

- To check the bit-width of inverse transform, the derivation of bit-width at 4 intermediate stages were done using software developed for B.2.32 of JCTVC-D610.
- bit-width of all intermediate stage are restricted under 16-bit for all size transform

	IO	ITInput	ITStage1	ITOutput
4x4	12	16	16	11
8x8	13	16	16	12
16x16	14	16	16	12
32x32	15	16	16	12

# Coding efficiency:

## ❖ Under common test condition:

- Fast factorization implementation: 0.1% gain with 1% encoding and 3% decoding time increase

	Intra			Intra LoCo			Random access			Random access LoCo			Low delay			Low delay LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	-0.1	-0.5	-0.3	-0.8	-0.4	-0.3	0.0	0.1	-0.3	-0.5	0.3	0.4						
Class B	-0.1	-0.1	-0.1	-0.3	-0.4	-0.4	0.0	0.0	0.2	-0.1	0.2	0.3	-0.1	0.6	0.3	-0.1	0.3	0.5
Class C	0.1	-0.2	-0.3	-0.1	-0.4	-0.7	0.1	0.2	0.1	0.0	0.2	0.2	0.0	0.5	0.2	-0.1	0.4	0.2
Class D	0.1	-0.1	-0.2	-0.1	-0.5	-0.7	0.1	0.1	0.3	0.0	0.5	0.2	0.0	0.0	-0.1	0.1	0.4	0.5
Class E	-0.1	-0.3	-0.4	-0.5	-0.4	-0.6							-0.3	-0.8	0.6	-0.1	0.3	0.3
All	0.0	-0.2	-0.3	-0.3	-0.4	-0.5	0.0	0.1	0.1	-0.1	0.3	0.3	-0.1	0.1	0.2	-0.1	0.4	0.4
Enc Time[%]	100%			103%			101%			102%			102%			96%		
Dec Time[%]	109%			102%			102%			105%			106%			95%		

- Matrix multiplication implementation: 0.1% gain with 12% encoding and 18% decoding time increase

	Intra			Intra LoCo			Random access			Random access LoCo			Low delay			Low delay LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	-0.1	-0.5	-0.3	-0.8	-0.4	-0.3	0.0	0.1	-0.3	-0.5	0.3	0.4						
Class B	-0.1	-0.1	-0.1	-0.3	-0.4	-0.4	0.0	0.0	0.2	-0.1	0.2	0.3	-0.1	0.6	0.3	-0.1	0.3	0.5
Class C	0.1	-0.2	-0.3	-0.1	-0.4	-0.7	0.1	0.2	0.1	0.0	0.2	0.2	0.0	0.5	0.2	-0.1	0.4	0.2
Class D	0.1	-0.1	-0.2	-0.1	-0.5	-0.7	0.1	0.1	0.3	0.0	0.5	0.2	0.0	0.0	-0.1	0.1	0.4	0.5
Class E	-0.1	-0.3	-0.4	-0.5	-0.4	-0.6							-0.3	-0.8	0.6	-0.1	0.3	0.3
All	0.0	-0.2	-0.3	-0.3	-0.4	-0.5	0.0	0.1	0.1	-0.1	0.3	0.3	-0.1	0.1	0.2	-0.1	0.4	0.4
Enc Time[%]	124%			126%			107%			105%			105%			105%		
Dec Time[%]	121%			126%			110%			111%			117%			120%		

# Coding efficiency:

## ❖ At low Qp

- 0.8% gain with 9% encoding time increase and 2% decoding time decrease (time measurement is not accurate)

							Random ac cess			Random ac cess LoCo			Low delay			Low delay LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	-0.4	-1.0	-1.0	-0.8	-0.6	-0.6	-0.6	-0.5	-0.3	-1.2	-0.4	-0.4						
Class B	-0.3	-0.9	-1.0	-1.6	-0.7	-0.8	-0.7	-0.3	-0.6	-1.6	-0.3	-0.4	-0.9	-0.3	-0.5	-2.1	-0.3	-0.3
Class C	0.2	-0.2	-0.2	0.7	1.4	1.3	-0.4	0.0	0.0	-0.8	-0.1	-0.3	-0.5	-0.1	-0.1	-1.2	-0.2	-0.3
Class D	0.2	-0.2	-0.2	0.5	1.1	1.1	-0.3	0.3	0.2	-0.8	-0.2	-0.3	-0.6	0.2	0.0	-1.3	-0.4	-0.4
Class E	-1.6	-1.7	-1.7	-1.4	-1.6	-1.8							-1.0	-0.7	-0.5	-1.7	-0.7	-0.8
All	-0.3	-0.7	-0.8	-0.5	0.0	0.0	-0.5	-0.1	-0.2	-1.1	-0.2	-0.4	-0.7	-0.2	-0.3	-1.6	-0.4	-0.4
Enc Time[%]	109%			122%			105%			107%			104%			109%		
Dec Time[%]	99%			94%			101%			93%			101%			98%		

## ❖ At high Qp

- 0.0% gain with 11% encoding and 18% decoding time increase. (time measurement is not accurate)

							Random ac			Random ac								
	Intra			Intra LoCo			Random access			Random access LoCo			Low delay			Low delay LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.0	0.0	0.3	-0.1	0.4	0.3	-0.2	0.2	0.8	-0.2	0.3	-0.5						
Class B	0.0	0.2	0.4	0.0	0.4	0.4	0.2	-0.3	1.1	-0.1	-1.0	-0.3	-0.1	-0.8	-0.1	0.2	-0.3	1.1
Class C	0.0	0.0	-0.1	0.1	-0.7	-0.2	-0.2	0.5	0.6	0.0	-1.2	-0.1	0.2	0.8	-0.2	0.1	-1.4	-2.2
Class D	0.0	0.7	0.0	0.0	-0.8	-1.0	-0.2	-1.1	-1.3	0.0	-2.7	-0.5	0.5	-1.2	0.0	0.0	-2.6	-1.6
Class E	0.3	0.4	0.1	0.2	-0.2	0.3							0.0	-2.0	0.3	-0.3	-3.2	-1.3
All	0.1	0.3	0.1	0.1	-0.2	-0.1	-0.1	-0.2	0.3	0.0	-1.3	-0.4	0.1	-0.7	0.0	0.0	-1.7	-0.9
Enc Time[%]	122%			123%			102%			105%			108%			105%		
Dec Time[%]	121%			145%			99%			107%			111%			125%		



# Conclusions

- ❖ Fast integer transform based on modified Loeffler's factorization is proposed.
  - Both factorized implementation or matrix multiplication implementation are possible
  - With factorized butterfly form, number of arithmetic operations is significantly less than matrix multiplication case
  - 4x4, 8x8 and 16x16 transforms are embedded in 32x32 transform
  - Input signal and intermediate transpose buffer for inverse transform are represented within 16 bit
  - Six scalar values are used to replace de-quantization scaling matrix for all transform size
  
- ❖ We recommend adopting the proposed transform solution, which supports both of known full factorization implementation and matrix multiplication implementation, into the next version of HM.