INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Student :
Vincent GIRAUD

Tutors :
Pierre-Loup Cabarat
Wassim Hamidouche

Low level optimizations of the Future Video Coding inverse transforms

InnovR module 2017-2018
Electronics and Industrial Informatics

SUMMARY

Introduction

Current context

The *Future Video Coding* standard

State of the art

Residual data management in H.265/MPEG-4 HEVC

The Adaptive Multiple Transform (AMT)

Single Instruction Multiple Data (SIMD) computing

Proposed computations

Exploitation of spatial parallelism

Customized treatment for constant residues

Results

Technical configuration

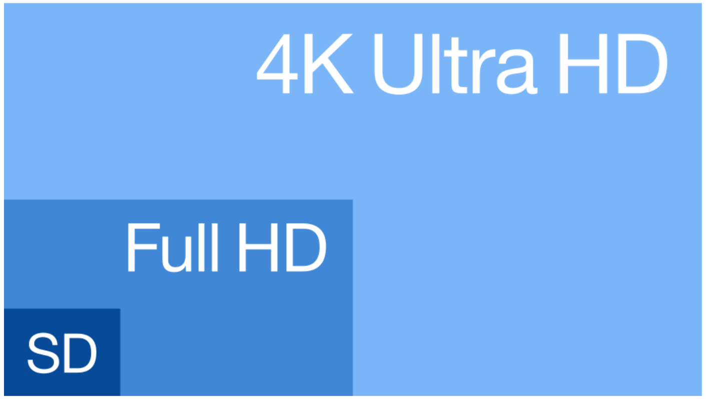
Analysis

Conclusion

H.264/MPEG-4 part 10 (2003)

H.265/MPEG-H part 2 (2013)

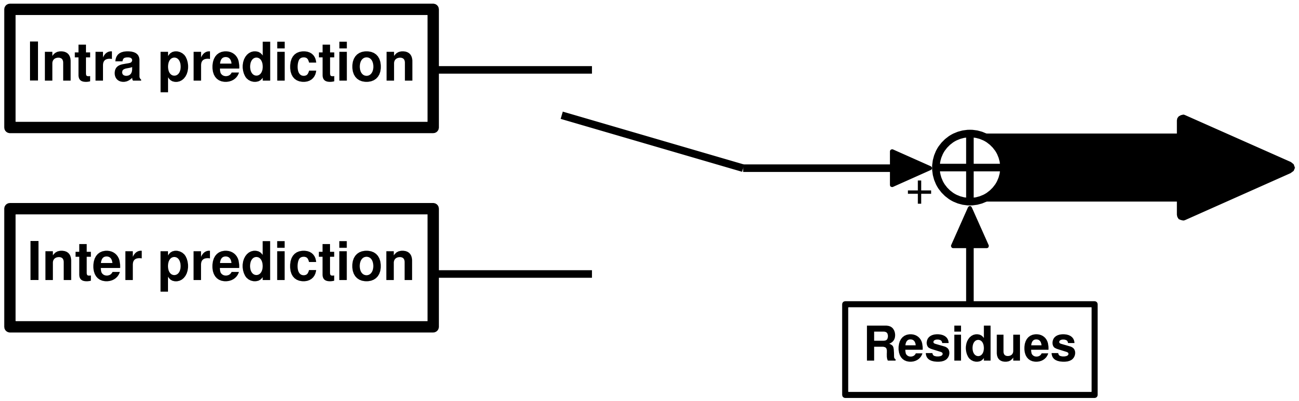
Future Video Coding (FVC)



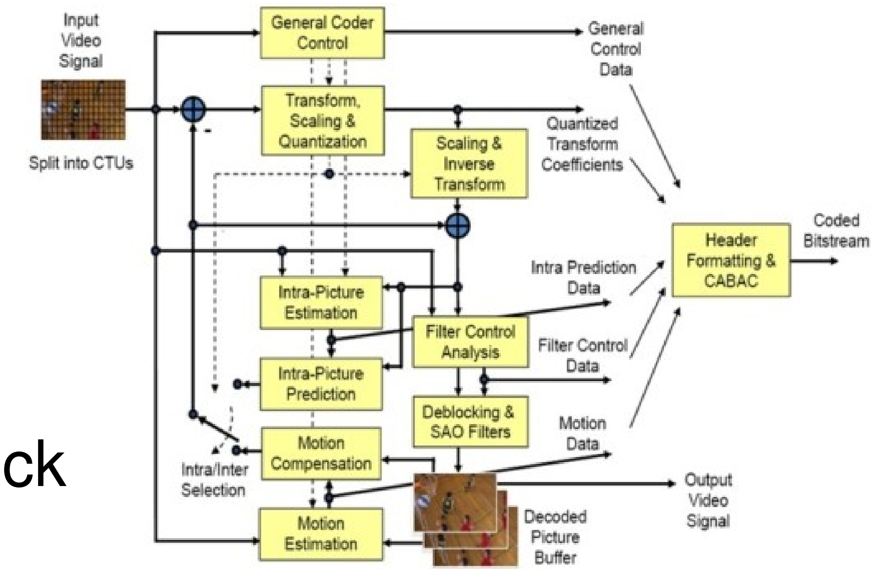
Future Video Coding

- To be released around 2021
- Bitrate reduced by 50 %
- New residual data management

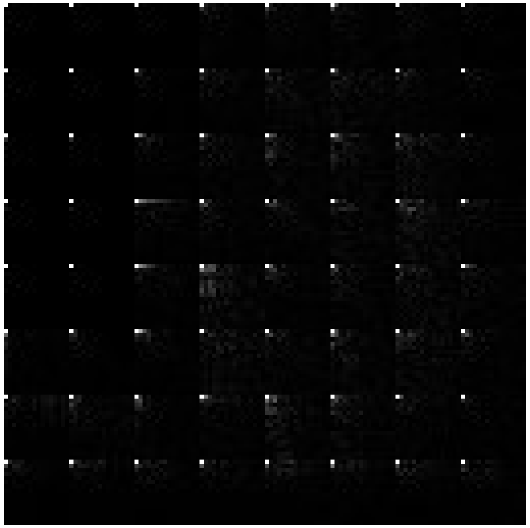
Residual data management in modern video standards (like H.265/MPEG-4 HEVC)



Prediction + Residues = Output block



Residual data management in H.265/MPEG-4 HEVC



$$X_k = \sum_{n=0}^{N-1} x_n . \cos(\frac{\pi}{N}.[n+\frac{1}{2}].k)$$

$$\begin{bmatrix} 64 & 64 & 64 & 64 \\ 84 & 35 & -35 & -84 \\ 64 & -64 & -64 & 64 \\ 35 & -84 & 84 & -35 \end{bmatrix}$$

Basically the only transform used

Residual data management in FVC

DCT-II	$T_i(j) = \omega_0 \times \sqrt{\frac{2}{N}} \times cos(\frac{\pi \times i \times (2j+1)}{2N})$ where $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } i = 0 \\ 1 & \text{if } i \neq 0 \end{cases}$

DCT-II	$T_i(j) = \omega_0 \times \sqrt{\frac{2}{N}} \times cos(\frac{\pi \times i \times (2j+1)}{2N})$ where $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } i = 0 \\ 1 & \text{if } i \neq 0 \end{cases}$
DCT-V	$T_i(j) = \omega_0 \times \omega_1 \times \sqrt{\frac{2}{2N-1}} \times cos(\frac{2 \times \pi \times i \times j}{2N-1})$ where $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } i = 0 \\ 1 & \text{if } i \neq 0 \end{cases}$ and $\omega_1 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } j = 0 \\ 1 & \text{if } j \neq 0 \end{cases}$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \times cos(\frac{\pi \times t(2i+1) \times (2j+1)}{4N+2})$
DST-I	$T_i(j) = \sqrt{\frac{2}{N+1}} \times sin(\frac{\pi \times (i+1) \times (j+1)}{N+1})$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \times sin(\frac{\pi \times (2i+1) \times (j+1)}{2N+1})$

Different transforms with different properties

Dynamic selection during encoding (RDO)

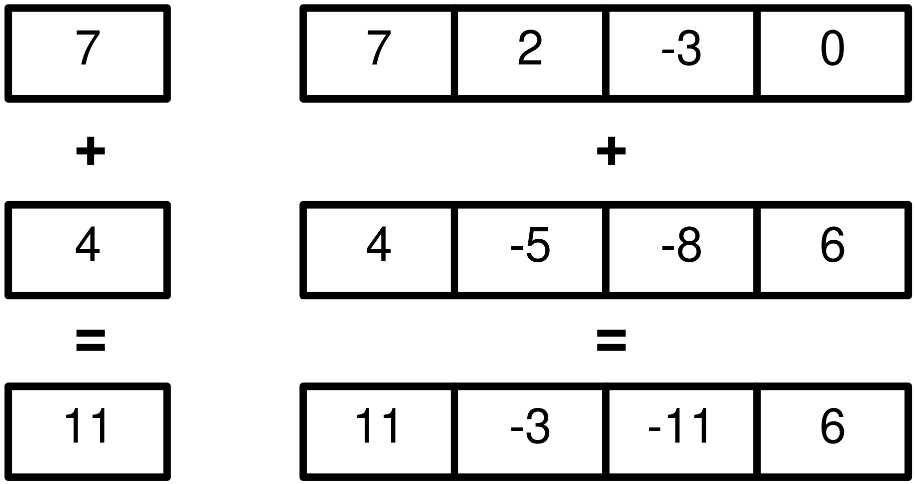
Presence of additional flags for decoding time

Adaptive
Multiple
Transform

The Adaptive Multiple Transform (AMT)

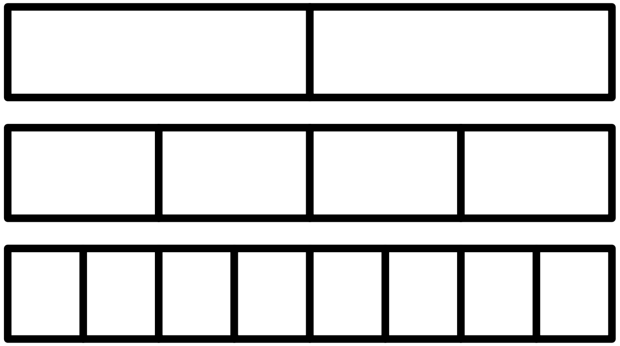
- + Provides a 5 % gain in terms of bitrate
- + Takes better into account the specificity of each TB
- - Brings a certain complexity
- - Requires more computing
- - Adds flags to each TB

Single Instruction on Multiple Data (SIMD)



- MMX
- SSE
- 3DNow!
- AVX

Different vectors' size,
different elements' size



Proposed solutions

Generic

low level

spatially parallel

optimizations

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Proposed solutions

$$(DcB)Dr^T$$

$$\left(\begin{bmatrix} Dc_0 & Dc_1 & Dc_2 & Dc_3 \\ Dc_4 & Dc_5 & Dc_6 & Dc_7 \\ Dc_8 & Dc_9 & Dc_{10} & Dc_{11} \\ Dc_{12} & Dc_{13} & Dc_{14} & Dc_{15} \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 & B_7 \\ B_8 & B_9 & B_{10} & B_{11} \\ B_{12} & B_{13} & B_{14} & B_{15} \end{bmatrix}\right) \times \begin{bmatrix} Dr_0 & Dr_4 & Dr_8 & Dr_{12} \\ Dr_1 & Dr_5 & Dr_9 & Dr_{13} \\ Dr_2 & Dr_6 & Dr_{10} & Dr_{14} \\ Dr_3 & Dr_7 & Dr_{11} & Dr_{15} \end{bmatrix}$$

Optimizing this calculation is providing improvements adapted to the whole AMT set

A matrix product requires :

- m^3 multiplications
- $m^2 * (m – 1)$ additions
- $m^2 * (2m – 1)$ operations

The SSE instruction set will be used

First algorithm

Dc_0	Dc_0	Dc_0	Dc_0
×	×	×	×
B_3	B_2	B_1	B_0
+	+	+	+
Dc_1	Dc_1	Dc_1	Dc_1
×	×	×	×
B_7	B_6	B_5	B_4
+	+	+	+
Dc_2	Dc_2	Dc_2	Dc_2
×	×	×	×
B_{11}	B_{10}	B_9	B_8
+	+	+	+
Dc_3	Dc_3	Dc_3	Dc_3
×	×	×	×
B_{15}	B_{14}	B_{13}	B_{12}

Almost no vector reorganization

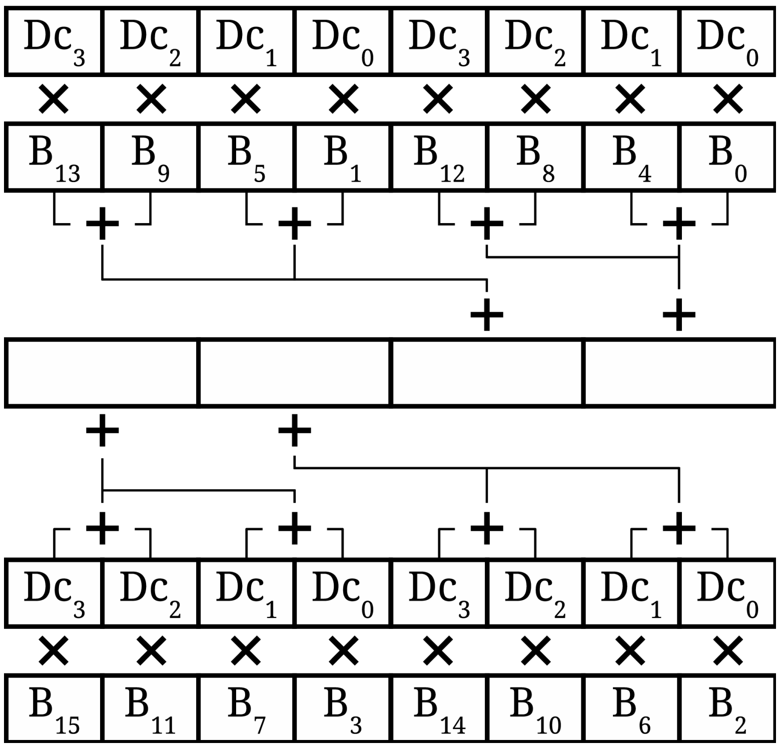
Heavy use of memory

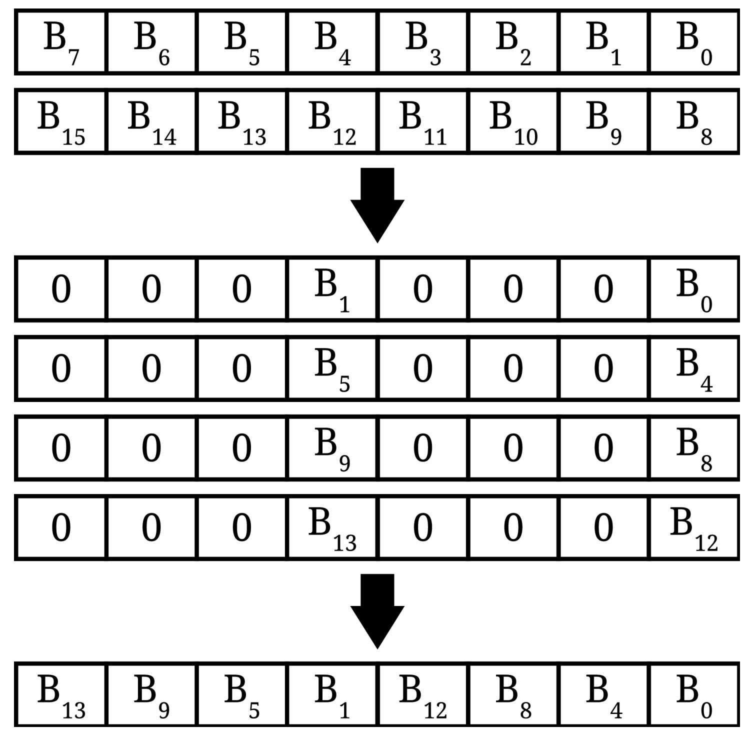
$$\left(\begin{bmatrix} Dc_0 & Dc_1 & Dc_2 & Dc_3 \\ Dc_4 & Dc_5 & Dc_6 & Dc_7 \\ Dc_8 & Dc_9 & Dc_{10} & Dc_{11} \\ Dc_{12} & Dc_{13} & Dc_{14} & Dc_{15} \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 & B_7 \\ B_8 & B_9 & B_{10} & B_{11} \\ B_{12} & B_{13} & B_{14} & B_{15} \end{bmatrix}\right) \times \begin{bmatrix} Dr_0 & Dr_4 & Dr_8 & Dr_{12} \\ Dr_1 & Dr_5 & Dr_9 & Dr_{13} \\ Dr_2 & Dr_6 & Dr_{10} & Dr_{14} \\ Dr_3 & Dr_7 & Dr_{11} & Dr_{15} \end{bmatrix}$$

Second algorithm

Lots of reorganization

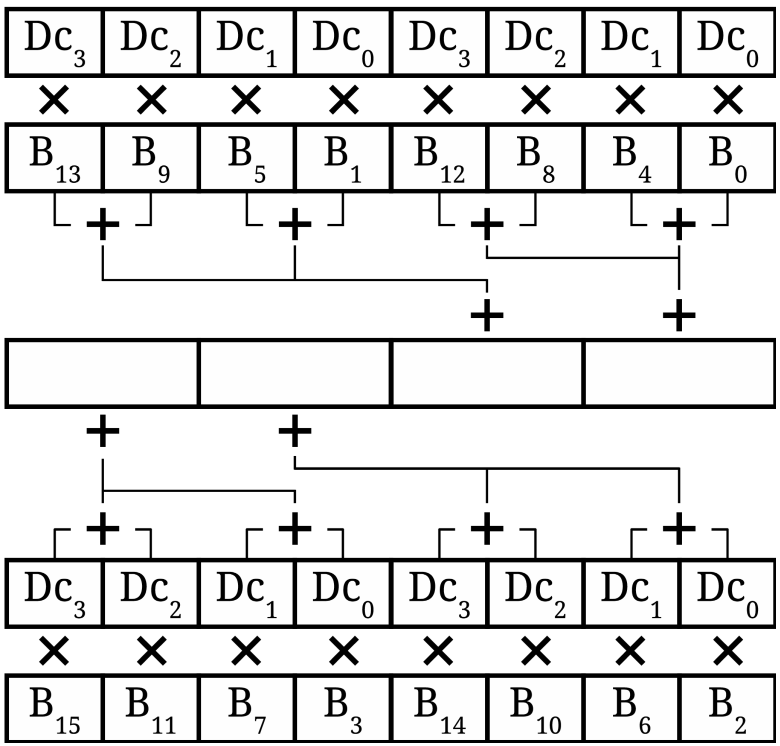
Fewer memory accesses





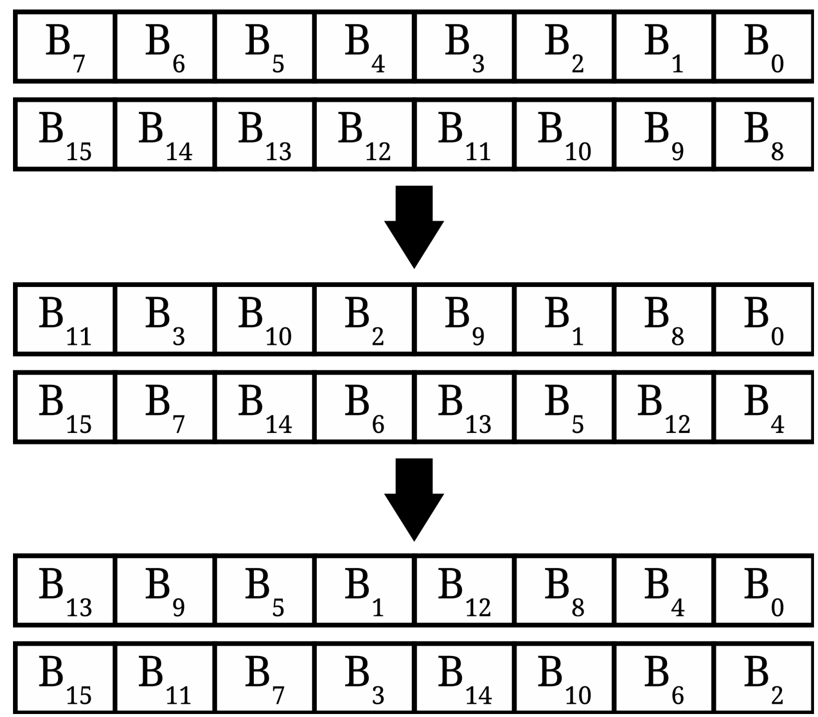
$$\left(\begin{bmatrix} Dc_0 & Dc_1 & Dc_2 & Dc_3 \\ Dc_4 & Dc_5 & Dc_6 & Dc_7 \\ Dc_8 & Dc_9 & Dc_{10} & Dc_{11} \\ Dc_{12} & Dc_{13} & Dc_{14} & Dc_{15} \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 & B_7 \\ B_8 & B_9 & B_{10} & B_{11} \\ B_{12} & B_{13} & B_{14} & B_{15} \end{bmatrix}\right) \times \begin{bmatrix} Dr_0 & Dr_4 & Dr_8 & Dr_{12} \\ Dr_1 & Dr_5 & Dr_9 & Dr_{13} \\ Dr_2 & Dr_6 & Dr_{10} & Dr_{14} \\ Dr_3 & Dr_7 & Dr_{11} & Dr_{15} \end{bmatrix}$$

Third algorithm



Still lots of reorganization

Still fewer memory accesses



$$\left(\begin{bmatrix} Dc_0 & Dc_1 & Dc_2 & Dc_3 \\ Dc_4 & Dc_5 & Dc_6 & Dc_7 \\ Dc_8 & Dc_9 & Dc_{10} & Dc_{11} \\ Dc_{12} & Dc_{13} & Dc_{14} & Dc_{15} \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 & B_7 \\ B_8 & B_9 & B_{10} & B_{11} \\ B_{12} & B_{13} & B_{14} & B_{15} \end{bmatrix}\right) \times \begin{bmatrix} Dr_0 & Dr_4 & Dr_8 & Dr_{12} \\ Dr_1 & Dr_5 & Dr_9 & Dr_{13} \\ Dr_2 & Dr_6 & Dr_{10} & Dr_{14} \\ Dr_3 & Dr_7 & Dr_{11} & Dr_{15} \end{bmatrix}$$

DC algorithm

During video coding, a significant number of TBs end up being constant

$$\begin{bmatrix} B_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

B_0	B_0	B_0	B_0
×	×	×	×
Dc_{12}	Dc_8	Dc_4	Dc_0
×	×	×	×
Dr_{12}	Dr_8	Dr_4	Dr_0

No vector rearrangement

Minimal memory access

$$\left(\begin{bmatrix} Dc_0 & Dc_1 & Dc_2 & Dc_3 \\ Dc_4 & Dc_5 & Dc_6 & Dc_7 \\ Dc_8 & Dc_9 & Dc_{10} & Dc_{11} \\ Dc_{12} & Dc_{13} & Dc_{14} & Dc_{15} \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 & B_7 \\ B_8 & B_9 & B_{10} & B_{11} \\ B_{12} & B_{13} & B_{14} & B_{15} \end{bmatrix} \right) \times \begin{bmatrix} Dr_0 & Dr_4 & Dr_8 & Dr_{12} \\ Dr_1 & Dr_5 & Dr_9 & Dr_{13} \\ Dr_2 & Dr_6 & Dr_{10} & Dr_{14} \\ Dr_3 & Dr_7 & Dr_{11} & Dr_{15} \end{bmatrix}$$

Shifting

$$\begin{bmatrix} -32768 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -32768 & 0 & 0 & 0 \\ -32768 & 0 & 0 & 0 \\ -32768 & 0 & 0 & 0 \\ -32768 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -2097152 & 0 & 0 & 0 \\ -2097152 & 0 & 0 & 0 \\ -2097152 & 0 & 0 & 0 \\ -2097152 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -2097152 & -2097152 & -2097152 & -2097152 \\ -2097152 & -2097152 & -2097152 & -2097152 \\ -2097152 & -2097152 & -2097152 & -2097152 \\ -2097152 & -2097152 & -2097152 & -2097152 \end{bmatrix}$$

$$\begin{bmatrix} -256 & -256 & -256 & -256 \\ -256 & -256 & -256 & -256 \\ -256 & -256 & -256 & -256 \\ -256 & -256 & -256 & -256 \end{bmatrix}$$

To comply with bit depth requirements, shiftings are necessary

Technical configuration for experimenting

GNU/Linux 4.9.92-1

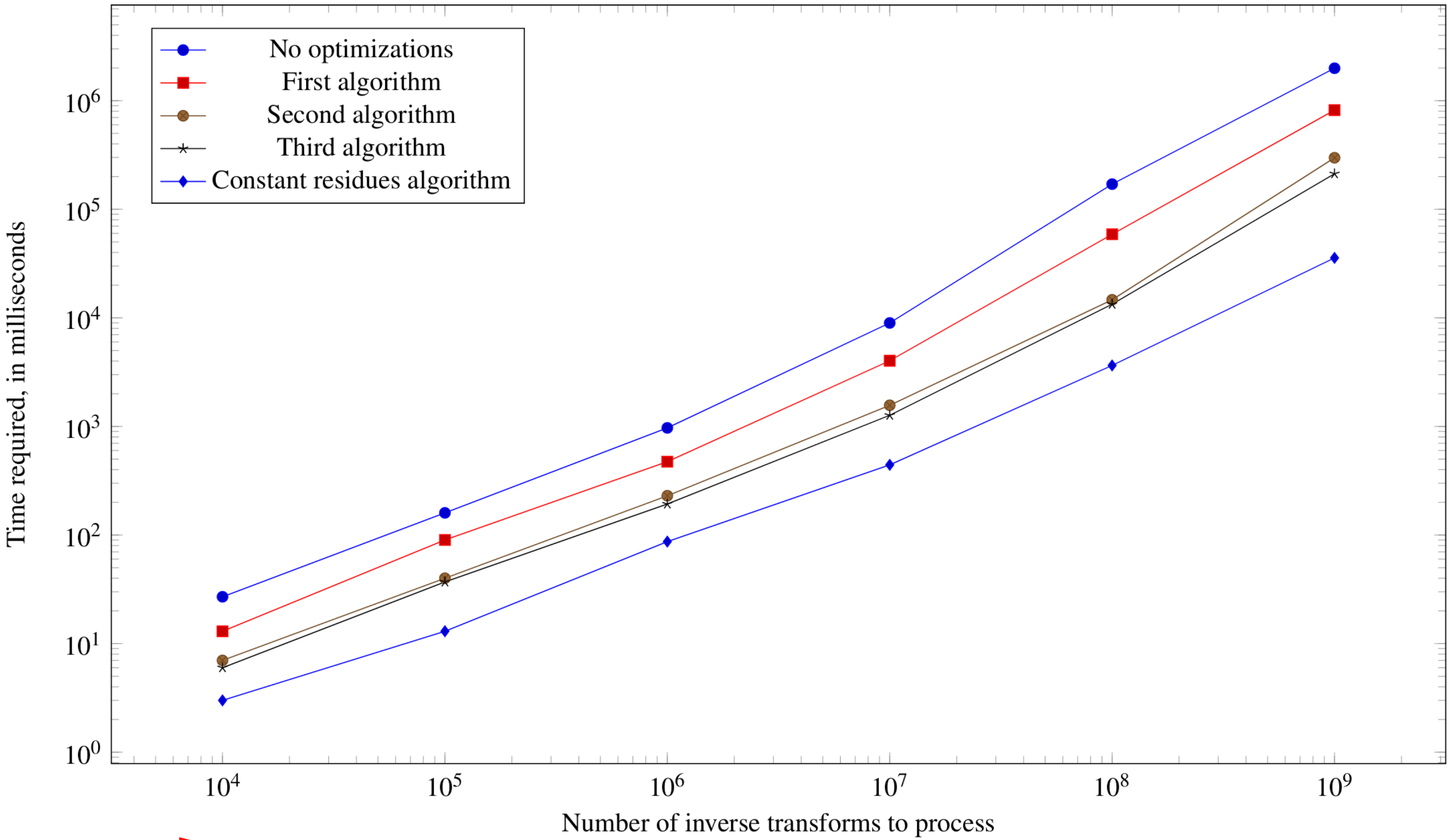
Intel Core i5-2410M at 2,30 GHz

GNU Compiler Collection (gcc)

```
-O0
-msse -msse2 -msse3 -msse4 -msse4.1 -msse4.2
```

Experiment
Results

First algorithm	224 %
Second algorithm	601 %
Third algorithm	717 %
Constant residues algorithm	2588 %



Conclusion

Spatial parallelism is efficient in the context of residues decoding

These optimizations allow for better performance without adding a lot of extra complexity

Such an implementation of the AMT in FVC is credible

References

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on Circuits and Systems for Video Technology, 22(12):1649–1668, Dec 2012.

[2] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, and M. Sadafale. Core transform design in the high effi- ciency video coding (hevc) standard. IEEE Journal of Selected Topics in Signal Processing, 7(6):1029–1041, Dec 2013.

[3] F. Loras and J. Fournier. H.264/mpeg-4 avc, un nou- veau standard de compression vidéo. Technical report, CORESA and France Télecom R&D, 2003.

[4] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand. Transform coding tech- niques in hevc. IEEE Journal of Selected Topics in Signal Processing, 7(6):978–989, Dec 2013.

[5] Chia-Wei Chang, Hao-Fan Hsu, Chih-Peng Fan, Chung- Bin Wu, and Robert Chen-Hao Chang. A fast algorithm- based cost-effective and hardware-efficient unified ar- chitecture design of 4 × 4, 8 × 8, 16 × 16, and 32 × 32 inverse core transforms for hevc. Journal of Signal Processing Systems, 82(1):69–89, Jan 2016.

[6] Pierrick Philippe, Thibaud Biatek, and Victorien Lorcy. Improvement of hevc inter-coding mode using multiple transforms, Aug 2017.

[7] Naty Sidaty, Wassim Hamidouche, Olivier Déforges, and Pierrick Philippe. Compression efficiency of the emerging video coding tools, Sep 2017.

[8] Ahmed Kammoun, Wassim Hamidouche, Fatma Bel- ghith, Jean-François Nezan, and Nouri Masmoudi. A unified 2d hardware architecture of the future video cod- ing adaptive multiple transforms on soc platform. IEEE Transactions on Consumer Electronics, 2018.

[9] Saurabh Puri, Sebastien Lasserre, and Patrick Le Cal- let. Cnn-based transform index prediction in multiple transforms framework to assist entropy coding, Aug 2017.