Highly parallel GPU accelerator for HEVC transform and quantization

Mate Cobrnic, Alen Duspara, Leon Dragic, Igor Piljic, Mario Kovac
University of Zagreb, Faculty of electrical engineering and computing;
mate.cobrnic@fer.hr

1. Introduction

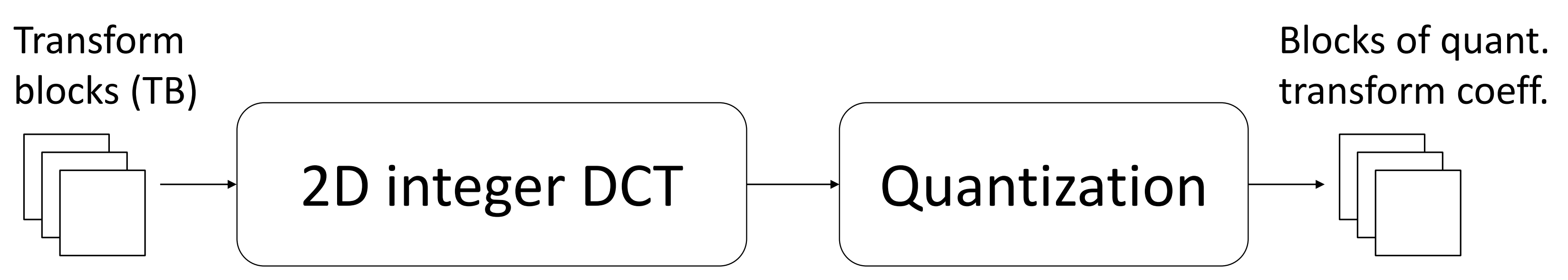
When analysing Internet traffic today it can be found that digital video content prevails. Its domination will continue to grow in the upcoming years and reach 82% of all traffic by 2022. Providing and supporting improved compression capability is therefore expected from video processing devices. This will relieve the pressure on storage systems and communication networks while creating preconditions for further development of video services. Transform and quantization (TQ) is one of the most compute-intensive parts of modern hybrid video coding systems where the coding algorithm itself is commonly standardized. HEVC is a state-of-the-art video coding standard which achieves high compression efficiency at the cost of high computational complexity.

2. Problem description

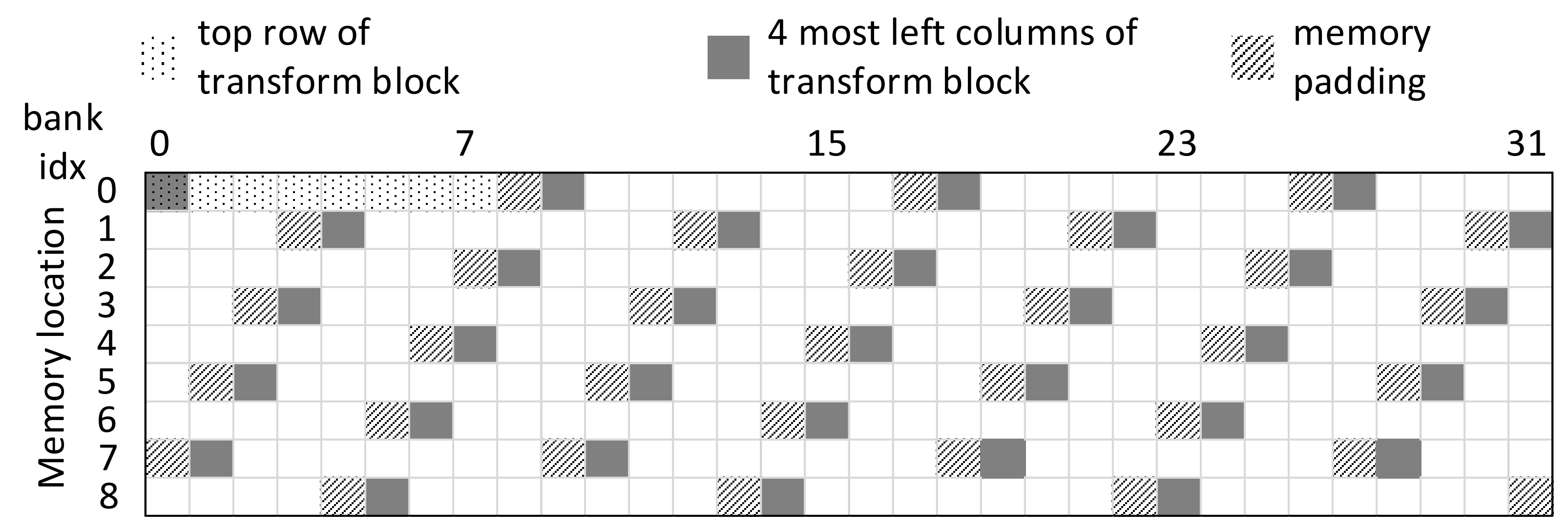
Related works which target CPU+GPU, the most common heterogeneous computing system arch. nowadays, do not discuss TQ kernel design and low-level optimizations in detail or parallelization is not carried out on the frame-level to exploit GPU parallelism.

3. Methodology

Simplified HEVC TQ functional block is examined for efficient implementation. Leaving processing of the control flags out doesn't disrupt the evaluation since most of the HEVC TQ workload relates to considered functions.

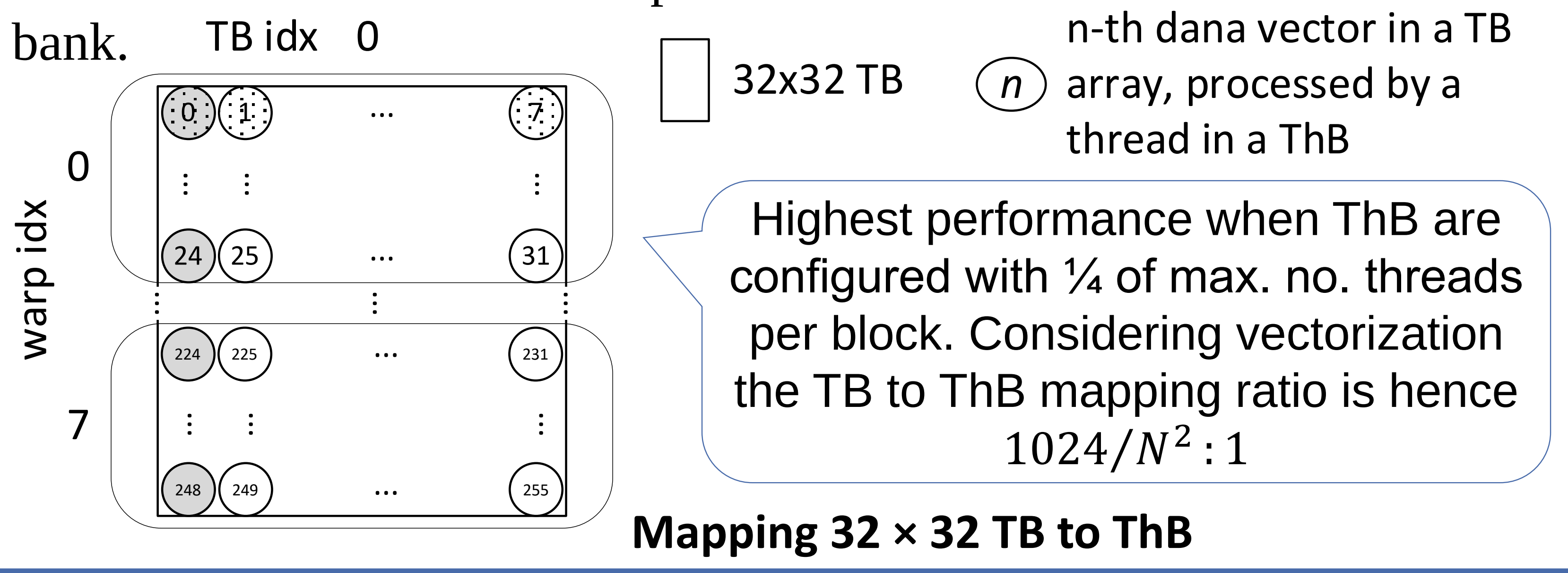


The first subblock involves double matrix multiplication $Y = D \times X \times D^T$, where D is the HEVC transform matrix with known values and X is the prediction error matrix. Matrices are of size $N \times N$, $N \in \{4, 8, 16, 32\}$. Quantization is a scalar operation.


top row of transform block
4 most left columns of transform block
memory padding
bank idx
0
7
15
23
31
Memory location
0
1
2
3
4
5
6
7
8

Distribution of data structured in shared memory using `short4` data type and processed by the top left thread in a thread-block (ThB) for 32 × 32 TB

Performance of the accelerator is primarily affected by the access efficiency to the shared memory. It is therefore configured for the 64-bit addressing mode and four 16-bit prediction error values are grouped into a data vector for max. bank bandwidth. Bank conflicts are avoided by padding the arrays with an additional column. It has been shown that memory padding method is not efficient for 4- and 8-point transforms as more bank conflicts occur due to the added column than due to the warp retrieves data vectors from the same bank.


TB idx 0
32x32 TB
n
n-th dana vector in a TB array, processed by a thread in a ThB
warp idx
0
7
0
1
7
24
25
31
224
225
231
248
249
255
...
Highest performance when ThB are configured with ¼ of max. no. threads per block. Considering vectorization the TB to ThB mapping ratio is hence $1024/N^2 : 1$

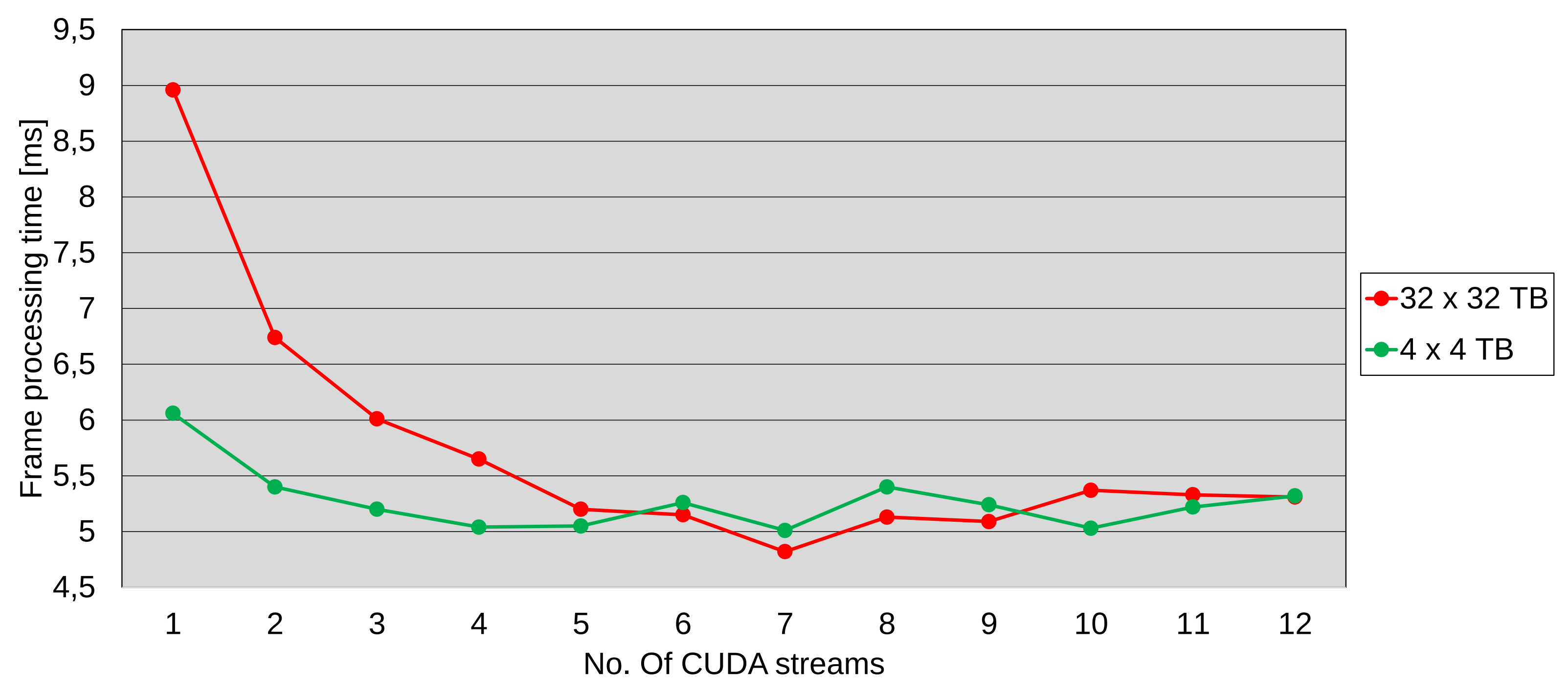
Mapping 32 × 32 TB to ThB

4. Results

The Development environment consists of Intel Core i5-3570K CPU @ 3.40 GHz, NVIDIA GPU Tesla K40c @ 745 MHz. CUDA programming model was used to support heterogeneous computation. Three different HEVC TQ implementation developed: CPU as referent impl., proposed GPU impl. and CUBLAS impl. which targets GPU as well but employs NVIDIA cuBLAS API `cublasSgemmStridedBatched(…)` and own rounding kernel to compute HEVC TQ of batch of TBs. The CUBLAS impl. operates with FP32 data type. In a DCI 4K prediction error frame, two TB distributions, edge cases considering computational complexity, are evaluated.

TB distribution	Frame processing time [ms]				
	CPU	GPU overall	GPU kernel	CUBLAS overall	CUBLAS kernel
32 × 32	705.95	8.77	3.65	19.25	8.97
4 × 4	234.59	5.9	0.78	117.24	101.75

GPU impl. outperforms other two impl.. CUBLAS impl. exhibits significantly lower performance for small size TBs. Analysis using profiling tool showed excessive transform kernel invocation (significant overhead in task initiation), low ThB utilization, global load/store efficiency <50% and shared efficiency <30%.



Impact of no. of CUDA streams on performance when processing a DCI 4K video frame

For an even more efficient GPU implementation, overlapping kernel execution and data transfer was exercised. As number of CUDA streams increases, processing time decreases. Overlapping capability, and hence the performance gain, is higher as duration of data transfer and kernel execution get closer to each other in value.

5. Conclusion

The highly parallel GPU accelerator for HEVC TQ combines page-locked data transfer, vectorized memory access with conflict-free access pattern and efficient TB to ThB mapping to provide high performance TQ across all supported transform sizes. The proposed GPU implementation outperformed CPU and CUBLAS implementation at worst 39.73 and 2.46 times respectively and revealed significant performance degradation when the latter is used for smallest size TBs. With overlapping, data transfers and kernel execution frame processing time was as low as 4.82 ms.

6. Acknowledgments

The work in this paper was partially financed by the MEEP project that has received funding from the EuroHPC JU under grant agreement No. 946002 and by the TETRAMAX project that has received funding from the EU's Horizon 2020 research and innovation programme under grant agreement No. 761349.