DECODING HIGH-THROUGHPUT JPEG2000 (HTJ2K) ON A GPU

Aous Naman and David Taubman

School of Electrical Engineering and Telecommunications, The University of New South Wales (UNSW), Sydney, Australia

INTRODUCTION

- HTJ2K, file extension .jph, is the latest addition to JPEG2000 Suite of image coding standards. It is also known as JPEG2000 Part 15, ISO/IEC 15444-15, and ITU-T.814.
- The most demanding step in conventional JPEG2000 (J2K-1) is the block coder – visiting data multiple times, and the serial nature of the context adaptive arithmetic coder.
- HTJ2K describes a "fast" block coder – codes many bitplanes at once and is highly parallelizable.
- Retains J2K-1 features, capabilities, and is compatible (losslessly-transcodable) with J2K-1 – supports limited quality scalability.
- Block coding speedup of ~10x (lossy) to ~40x (lossless).
- Slightly lower coding efficiency compared to J2K-1, ~9% ≈ -0.7dB.
- Kakadu JPEG2000 SDK v8.0 supports it – released in October.
- Open source implementation at https://github.com/aous72/OpenJPH
- Here, we reports on a GPU implementation.

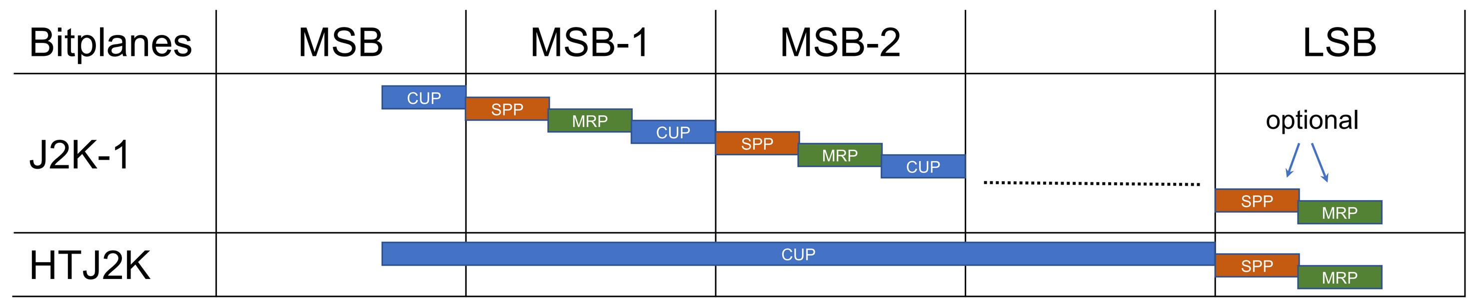

Figure 1: J2K-1 employs three passes to code a bitplane: Cleanup Pass (CUP), Significance Propagation Pass (SPP), and Magnitude Refinement Pass (MRP). In HTJ2K the first CUP encodes multiple bitplanes.

THE HTJ2K BLOCK CODESTREAM

- The codestream comprises up to 3 three coding passes: a Cleanup Pass (CUP), followed by an optional Propagation Pass (SPP), and a yet optional Magnitude Refinement Pass (MRP).
- The CUP has three byte-streams: MagSgn (forward), MEL (forward), and VLC (backward).
- Backward-forward exposes more parallelism. Here, we decode MEL and VLC, and SPP together. Later, we decode MagSgn, add SPP results, and decode MRP at the same time.

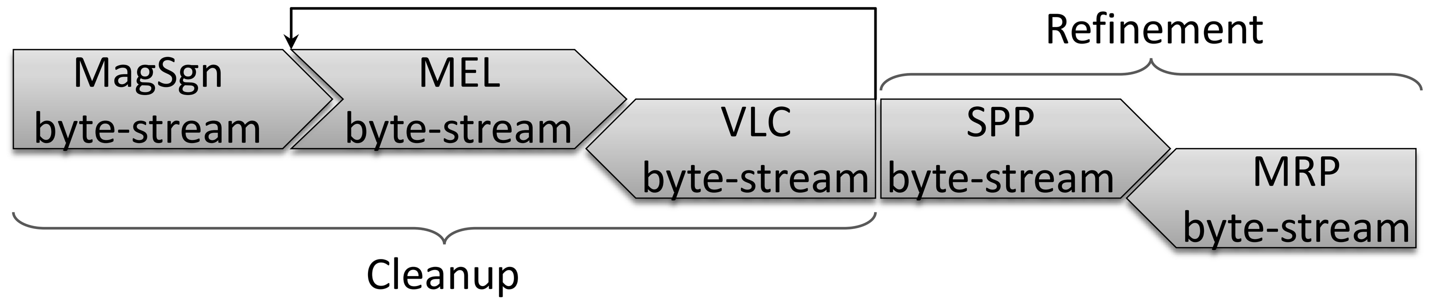

Figure 2: The segments of a HTJ2K codeblock. The last two bytes of the cleanup pass contain a pointer to the start of the MEL segment.

- Coding efficiency comes from efficiently encoding the location of significant coefficients (i.e., non-zero after quantization) & the number of bits needed to represent them.
- The MEL and the VLC byte-streams provides this efficiency.
- The MEL is an adaptive run-length encoder – efficiently codes runs of 0.
- Context-adaptive VLC encodes locations of significant samples and their number of bits.
- The MagSgn byte-stream stores the values of coefficients.

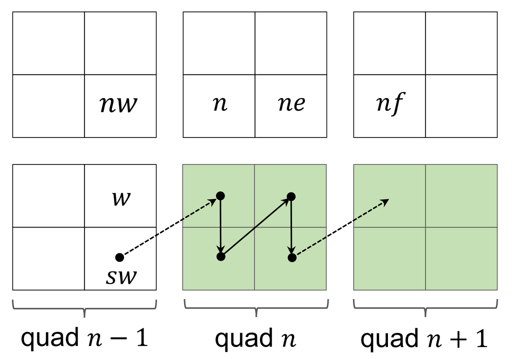

Figure 3: The HTJ2K block coder encodes samples in quads. Samples of significance sw, w, nw, n, ne, and nf are VLC context of quad n.

A GPU-BASED DECODER FOR HTJ2K

The CPU parses a JPH file, generating lists of code-block information (data locations, segment sizes, … etc.), which are transferred to GPU. In this work, the GPU operates in one of two scenarios:

Kernels with No Refinements (NR) – decoding CUP Only

- A practical decoder can always discard SPP, MRP – lower quality
- KCUPS1 (serial per code-block): decodes MEL & VLC bytes streams, stores decoded info $(\rho_q, \epsilon_q^1, \epsilon_q^k, u_q)$ in global memory. VLC tables are transferred to shared memory by first warp. Uses one thread per codeblock, and 45 registers.
- KCUPS2 (parallel per codeblock): retrieves $(\rho_q, \epsilon_q^1, \epsilon_q^k, u_q)$ from global memory and decodes MagSgn byte-stream, generating decoded coefficients. Employs one wrap per 64x64 codeblock, and 64 registers.
- WSYN: performs wavelet synthesis on all resolution except the last. Also receives info from CPU about all-zero code-blocks in order to skip data retrieval for these blocks.
- WSYN+Color: similar to the above, but performs color transform at the end. It stores the data ready for transfer to CPU. Uses 125 registers.
- All processing is performed using 32 bit floats.

Kernel with Refinements (R) – decoding CUP, SPP, and MRP

- KCUPS1+SPP: similar to KCUPS1, but also decodes SPP, for which it stores 2bits/sample in global memory. Uses 77 registers and 144 bytes shared memory as a scratchpad.
- KCUPS2+MRP: similar to KCUPS2, but also decodes MRP. It also retrieves and combines decoded SPP information. Uses 82 registers.

EXPERIMENTAL RESULTS

- Results are for 4K 4:4:4 12bit video test sequence ARRI AlexaDrums.
- 64x64 code-blocks, irreversible CDF97 wavelet, 5 levels of decomps
- No overlap in frame decoding is employed.
- Compressed image are uploaded while earlier fames are decoded.
- Frame decode rates are obtained decoding 1000 HTJ2K frames.
- 3 GPU are test: low-end GT1030 with GDDR5, mid-range GTX 1060, and "enthusiast" GTX1080.

Table 1: Decoding performance for a variety of GPUs, with and without refinements. Alternate approaches are also shown. † test conditions are not clear. * interpolated from results in [5].

	GT1030 GDDR5			GTX1060			GTX1080		
Kernel	1b	4b	ls	1b	4b	ls	1b	4b	ls
Decoding without refinement (CUP only)									
KCUPS1 (ms)	.560	1.77	4.43	.408	.485	1.48	.385	.420	.520
KCUPS2 (ms)	.727	2.00	4.88	.212	.538	1.36	.128	.310	.729
WSYN+Color (ms)	3.96	4.57	6.15	1.11	1.29	1.66	.750	.886	1.19
Frames per Second	180	118	62	550	420	220	770	588	402
Decoding with refinement (CUP, SPP, and MRP)									
KCUPS1-SPP (ms)	1.12	2.81	-	.856	1.00	-	.807	.895	-
KCUPS2-MRP (ms)	1.04	2.82	-	.300	.806	-	.172	.430	-
WSYN+Color (ms)	3.96	4.57	-	1.11	1.29	-	.750	.886	-
Frames per Second	160	96	-	425	317	-	560	440	-
Alternate Approaches									
JPEG-XS [5]	NA			NA			233*	194*	NA
JPEG2000 [7]	NA			95†			NA		

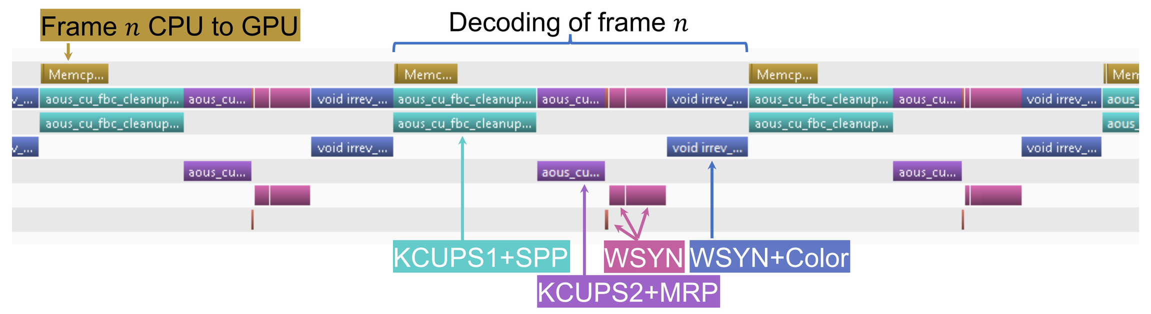

Figure 4 (left): Timeline for decoding one HTJ2K frame at 4bits/pixel on the GTX1080 when the frame is not transferred back to the host (CPU).

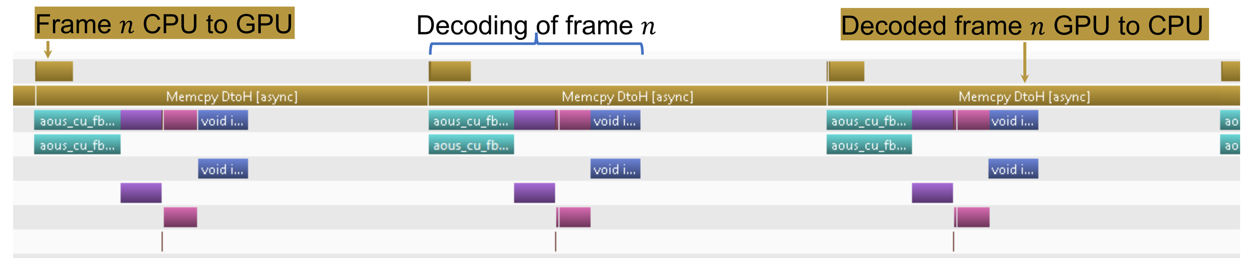

Figure 5 (left): Same as Figure 4, but when the decoded frames are transferred back to the host (CPU).

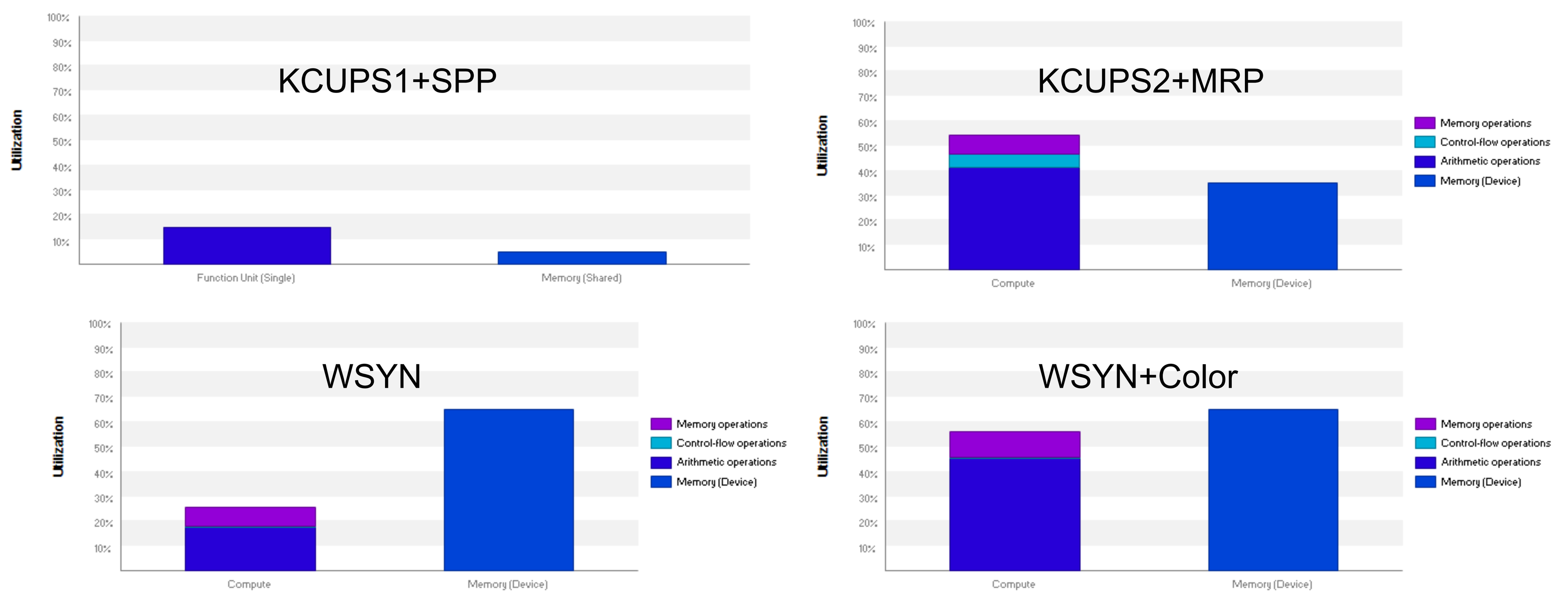

Figure 6 (above): GPU utilization for different kernels when decoding a frame coded at 4bits/pixel on the GTX1080.

CONCLUSIONS

Decoding HTJ2K files on a GPU is feasible and can achieve very high frame rates, even on low-end GPUs; it is many folds faster than JPEG2000. Decoding 8K 4:4:4 HDR at 120 fps is possible on a GTX1080. Next, we will explore GPU encoding.