Fast and Parallel Computation of the Discrete Periodic Radon Transform on GPUs, multi-core CPUs and FPGAs

Cesar Carranza*, Daniel Llamocca†, and Marios Pattichis‡

*Sección Electricidad y Electrónica, Pontificia Universidad Católica del Perú, Lima-32, Perú
†Electrical and Computer Engineering Department, Oakland University, Rochester, MI, 48309, USA
‡Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM, USA
Emails: acarran@pucp.edu.pe, llamocca@oakland.edu, pattichis@ece.unm.edu

Abstract

The current paper presents new and efficient algorithms for computing the DPRT and its inverse on multi-core CPUs (*FastDirDPRT*) and GPUs (*FastRayDPRT*). The results provide significant evidence of the success of the new algorithms. On an 8-core CPU, with support for two threads per core, a speedup of approximately 10× (up to 12.83×) is achieved. On a 2048-core GPU, a speedup in the range of 526 to 873 is achieved. The DPRT can be computed exactly and in real-time (30 frames per second) for 1471 × 1471 images on the GPU. Furthermore, the GPU algorithms approximate the performance of an efficient FPGA implementation using $2N$ parallel cores at 100MHz.

Background

The DPRT of $f(i,j)$, is defined using [1]:

$$R(m,d) = \begin{cases} \sum_{i=0}^{N-1} f(i, \langle d + mi \rangle_N), & 0 \le m < N, \\ \sum_{j=0}^{N-1} f(d,j), & m = N. \end{cases}$$

And for the inverse DPRT:

$$f(i,j) = \frac{1}{N}\left[\sum_{m=0}^{N-1} R(m, \langle j - mi \rangle_N) - S + R(N,i)\right].$$

where $S = \sum_{d=0}^{d=N-1} R(0,d)$.

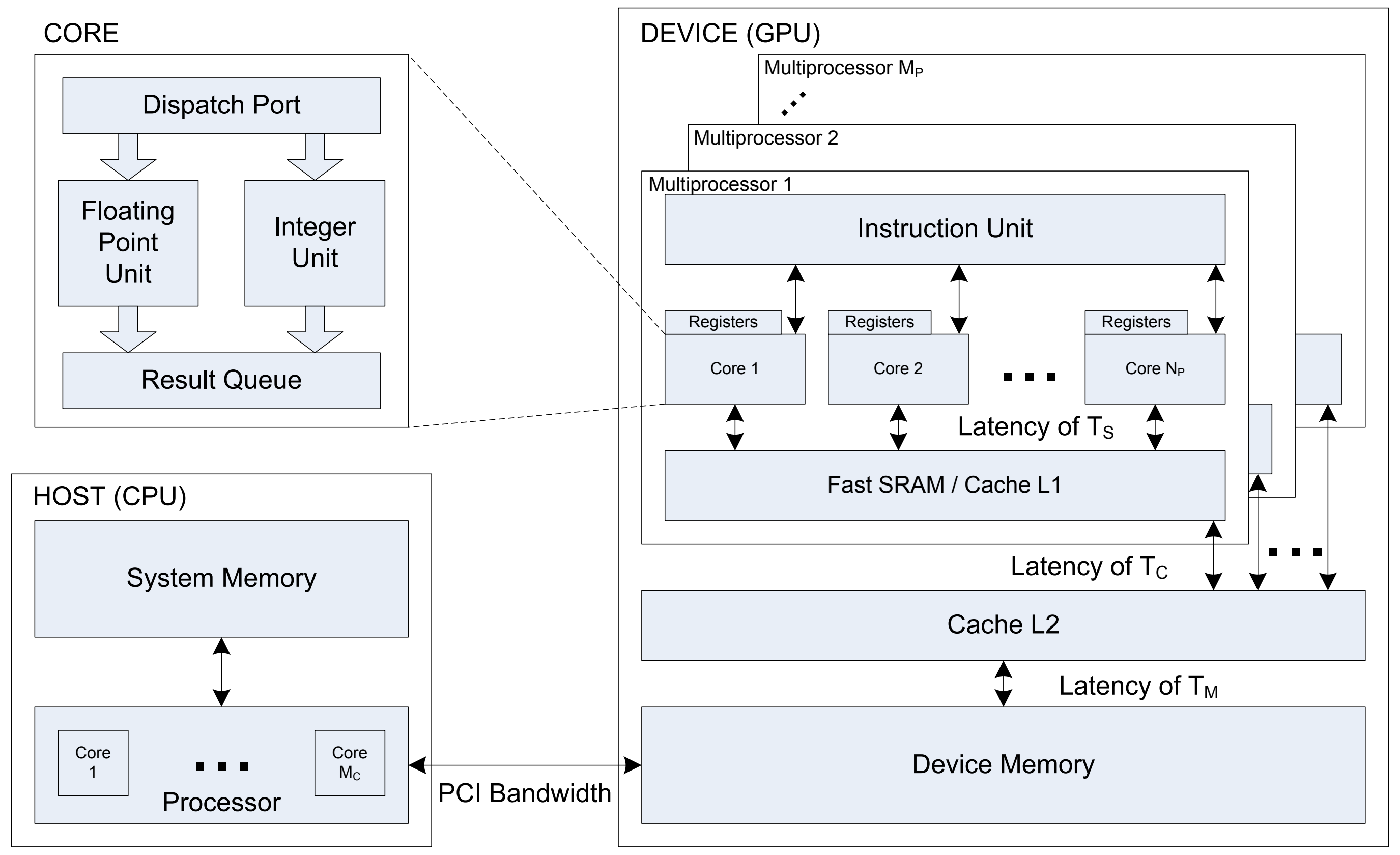
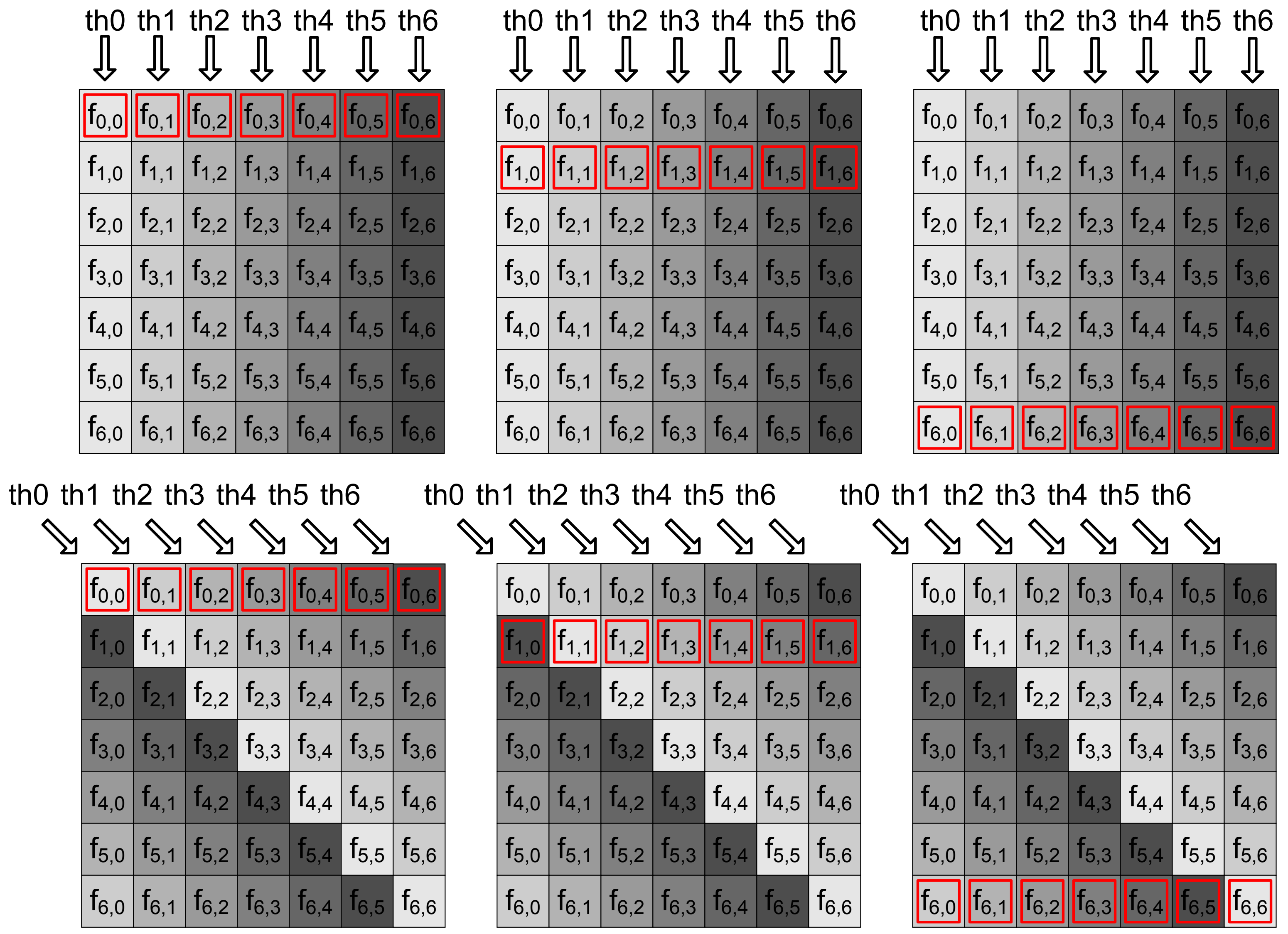


Fig. 1: A top level block diagram for the basic architecture. The CPU is the Host and the GPU the Device.

Methods



For the *FastRayDPRT* and *FastRayInvDPRT*, the main idea behind the fast computation of the rays is to use Row-major memory accesses for synchronized, parallel thread execution. In this example, the input image is assumed to be of size 7 × 7. For each step, a red square highlights the pixel being accessed. For each thread, we use distinct grayscale shade to identify the pixels that need to be accessed. The FPGA implementation can be found in [1].

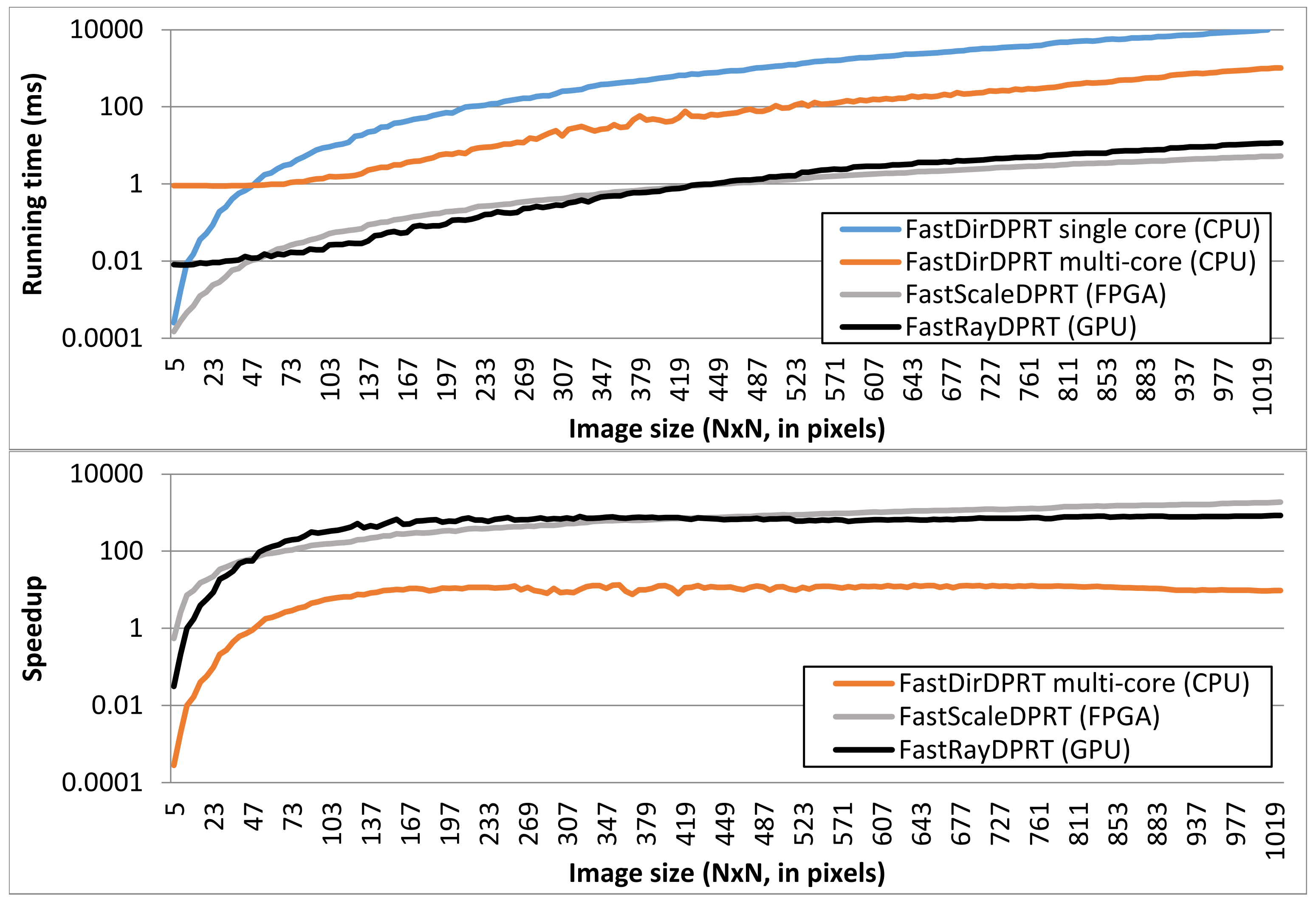
Computational complexity (GPU)

Let p be the number of processors ($p = M_P \cdot N_P$). Then, we have:

- ***Linear execution time*** $O(N)$**:** Clearly, if $p \ge N \cdot (N+1)$, we have more processors than required rays to compute. Since each thread requires $O(N)$ additions, we can compute the entire DPRT in $O(N)$ time.
- ***Quadratic execution time*** $O(N^2)$**:** Similarly, if $p = N$, each core must compute $N+1$ threads at a complexity of $O(N^2)$. Asymptotically, if p grows as $O(N)$, each processor needs to compute $O(N)$ rays with complexity of $O(N^2)$.
- ***Cubic execution time*** $O(N^3)$**:** For a single processor, we have $p = 1$ that needs to compute all of the directions. Furthermore, if p is $O(1)$, computational complexity still grows as $O(N^3)$.

Results

The host CPU included an Intel Xeon CPU E5-2630 v3 @3.2GHz with 8 hardware cores which implemented 16 logical processors via hyper-threading. For the device, we used an Nvidia GPU (GeForce GTX 980, Maxwell architecture GM204). The card was configured using: 16 MP. 128 CUDA cores per MP, for a total of 2048 cores@1367 MHz. For sizes up to 1471 × 1471, the DPRT and inverse DPRT implementations were exact using 32-bit fixed point arithmetic.



Performance comparisons between CPU, GPU and FPGA implementations [1].

Conclusion

The success of the proposed algorithms has been demonstrated in the results. Applications of the DPRT for convolutions [2] is also presented at ICIP 2018.

References

[1] C. Carranza, D. Llamocca, and M. Pattichis, "Fast and scalable computation of the forward and inverse discrete periodic radon transform," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 119–133, Jan 2016.

[2] ——, "Fast 2d convolutions and cross-correlations using scalable architectures," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2230–2245, May 2017.