# Evaluating Latency and Throughput bound Acceleration of FPGAs and GPUs for Adaptive Optics Algorithms

Vivek Venugopalan
United Technologies Research Center
411 Silver Lane, E Hartford, CT USA
Email:venugov@utrc.utc.com

***Abstract*—General purpose extremely large aperture optical/infrared telescopes are instrumental in vastly advancing the astrophysical knowledge on a variety of subjects such as star formation, super-massive black holes, solar magnetic atmosphere, exoplanets and protoplanetary systems. The computational requirements for data processing and real-time control are not feasible using conventional computer and cluster architectures. This paper investigates the latency and throughput bound hardware acceleration of wavefront reconstruction and real-time control algorithms using GPUs and FPGAs. Two different correlation methods are studied and targeted on the hardware accelerators for optimal performance. The GPU-based implementations exhibit lower latency due to its superior floating point capability and supporting libraries. The FPGA-based implementation is slower and requires more fine-tuning to yield more throughput than the GPU implementation.**

***Index Terms*—Adaptive optics systems, wavefront correction, field programmable gate arrays, parallel processing, graphics processing units, real-time systems**

## I. Introduction

Ground-based telescopes have been responsible for many of the great discoveries in astronomy for the last two decades. Some of the breakthroughs include the the nature and distribution of the dark matter and dark energy which dominate the Universe [1]. Most of these discoveries resulted from processing of data obtained from these telescopes. Hence, it is important to collect accurate data that can be later post-processed to extract the required information. The accuracy of the collected data is ensured by applying real-time correction if necessary. Real-time corrections are expensive in terms of computational complexity, algorithm latency and data movement between processing architectures.

Hardware accelerators have made a significant impact for providing computational acceleration along with emphasis on latency. The most commonly available accelerators are the fine-grain graphics processing units (GPU) and the coarse-grained reconfigurable field programmable gate arrays (FPGA). Both GPUs and FPGA-based systems are capable of providing excellent application speedup. However, both the platforms are different in terms of their programming model and underlying architecture. GPUs consist of a fixed pipeline with about 200-2500 computational cores clocked at low GHz range. As opposed to GPUs, FPGAs consist of reconfigurable fabric that can be configured to build a custom circuit and has clock rate of upto a maximum of 600 MHz. Research efforts have focused on comparing FPGAs with GPUs for a variety of applications ranging from astrophysics, optical flow and video processing [2], [3].

GPUs and FPGAs possess varying computational characteristics and hence cannot be compared only on the basis of floating point operations or fast pipelined datapaths. In most cases, the algorithm has to be fine-tuned and optimized before targeting a GPU, whereas the FPGA implementation has to be optimized to target the specific FPGA device (e.g. number of available logic cells, DSP slices, etc.). This paper compares and contrasts the experiences creating GPU and FPGA implementations of real-time image processing algorithms for adaptive optics (AO). The research questions addressed are: *(1) What performance parameters define the selection of hardware accelerators for a specific algorithm? (2) How does parallelism and pipelining translate to overall application performance on both GPUs and FPGAs?*

The paper is structured as follows: Section II describes the AO wavefront reconstruction algorithm and the system used in ground-based telescopes; Section III describes the initial image acquisition process and the different stages of the AO algorithm along with the cross-correlation methods; the GPU implementation is explained in Section IV and the FPGA implementation is described in Section V; the results and conclusion are presented in Sections VI and VII respectively.

## II. Adaptive Optics

Atmospheric turbulence distorts the wavefront by generating phase variations in the incoming light and limits the resolution of large ground-based telescopes. The adaptive optics (AO) system senses the wavefront aberrations and improves the resolution of the telescope by correcting the phase differences originating from atmospheric disturbance [4], [5]. The atmospheric turbulence can be represented as a state space model of the form,

$$\xi_{k+1} = A_d \xi_k + B_d g_k \tag{1}$$

$$\phi_k = C_d \xi_k + g_k \tag{2}$$

978-1-4799-6233-4/14/$31.00 

where $\phi$ is the phase difference, $k$ is the time index, $g$ is the Gaussian distributed noise, $A_d$ is the state transition matrix, $B_d$ is the Kalman gain and $C_d$ is the output matrix. The wavefront sensor (WFS) measures the phase distortion in the form of gradients $\varphi_x(i,j)$ and $\varphi_y(i,j)$ shown in equations 3 and 4,

$$\varphi_x(i,j) = \phi(i+1,j) - \phi(i,j) \tag{3}$$

$$\varphi_y(i,j) = \phi(i,j+1) - \phi(i,j) \tag{4}$$

The Shack-Hartmann sensor is commonly used as a WFS and the sensor measurement can be modeled as a stochastic approximation given in equation 5, where $W$ is the phase-to-WFS influence matrix and $n$ is the WFS measurement noise.

$$\varphi_k = W\phi_k + n_k \tag{5}$$

The WFS measurements are used to determine the optimal positions of the deformable mirror (DM) actuators. The DM compensates for the measured distortion by changing shape. The DM actuator correction is given in equation 6, where $D$ is the DM influence matrix, $v_k$ is the control input and $z^{-1}$ is the discrete shift operator.

$$\phi_{dm,k} = z^{-1}Dv_k \tag{6}$$

The objective of the AO system is to minimize the residual error given in equation 7, resulting in reconstruction of the wavefront and improved resolution of the telescope.

$$\phi_{\epsilon,k} = \phi_k + \phi_{dm,k} \tag{7}$$

The wavefront reconstruction consists of the estimated wavefront phases $\hat{\phi}$ being derived from the measurements $s_k$. The key components of the AO system are shown in Figure 1. The tip-tilt mirror (TTM) moves fast to remove the image shift and the Shack-Hartmann lenslet array focuses the sub-aperture images on the sensor of the high speed camera. The processing system calculates the phase shift and the resulting actuator commands for controlling the DM and TTM.

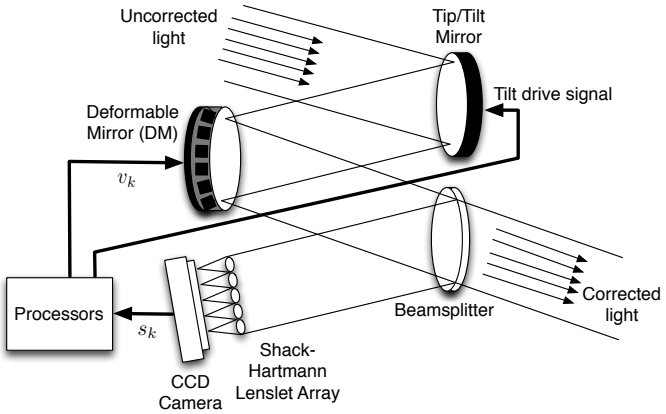


Fig. 1: The Adaptive Optics system

## III. Real-time Processing of the AO System

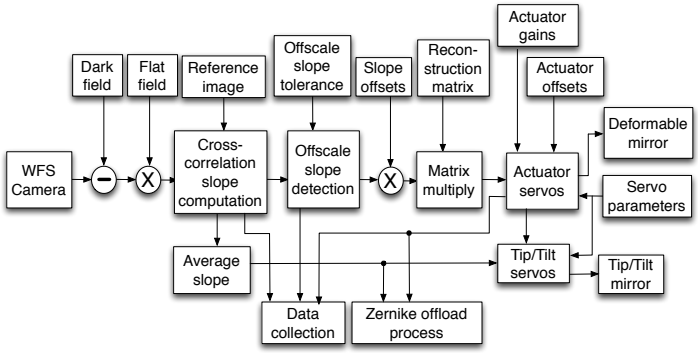


Fig. 2: Real-time processing for the Adaptive Optics system

The real-time processing of the AO system is shown in Figure 2. The high speed camera sends the 1856 20×20 pixel raw sub-aperture images to the processing system. The sub-apertures undergo a dark field correction followed by a flat field correction, which is the equivalent to correcting the images for zero level and gain equalization. The resulting flat field corrected image is 2D correlated with a reference image captured approximately once every hour. The 2D cross-correlation step determines the shift in the $x$ and $y$ direction of each sub-aperture, as compared to the reference image. The wavefront reconstruction step consists of a precomputed 3712×1920 reconstruction matrix, which is multiplied with the $\vec{x}$ and $\vec{y}$ shifts. The matrix multiplication products are summed after all the sub-aperture images in that particular frame have been processed and is then transmitted to the DM and TTM actuators after applying a servo loop algorithm.

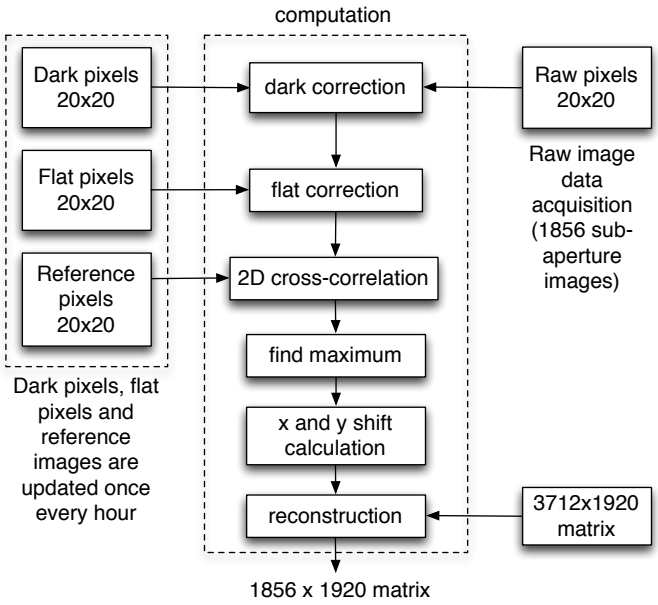


Fig. 3: AO system showing the data acquisition and the computation scope for targeted implementations

The real-time processing requirements are determined by

the frame rate of the camera and the number of actuators used for the DM and TTM. The frame rate of the camera is equal to or greater than 2 KHz, so that it can output 1856 sub-apertures of 20 $\times$ 20 pixels each. The AO system with the data acquisition of the 1856 sub-aperture images and the pixel correction images as inputs to the different stages of the algorithm is shown in Figure 3. Two different methods have been suggested for determining the 2D cross-correlation. The first method involves taking the Fast Fourier Transform, while the second method involves a 7 $\times$ 7 correlation.

### A. FFT-based correlation

The FFT-based correlation theorem states that the Fourier Transform of the two images' cross-correlation is the complex conjugate multiplication of their Fourier transforms. Therefore, $Corr(i,j)$ represents the correlated image, $Ref(i,j)$ represents the reference image, $CRaw(i,j)$ represents the dark and flat corrected raw image, and $\mathbf{F}\{\}$ represents the Fourier Transform in equation 8.

$$\mathbf{F}\{Corr(i,j)\} = \mathbf{F}\{Ref(i,j)\} \cdot \mathbf{F}\{CRaw(i,j)\} \quad (8)$$

The complex conjugate multiplication is as follows: Given two operands $a = a_r + ja_i$ and $b = b_r + jb_i$, the output equations can be defined as,

$$p_r = (a_r \cdot b_r) + (a_i \cdot b_i) \quad (9)$$
$$p_i = (a_r \cdot b_i) - (a_i \cdot b_r) \quad (10)$$

The Fast Fourier Transform (FFT) based correlation routine is shown in Figure 4. This method produces an array of 20×20 values after the Inverse FFT stage.

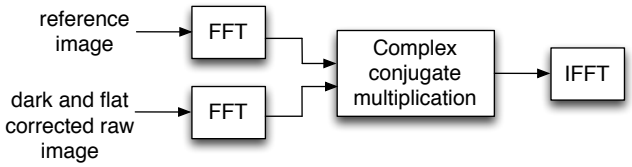


Fig. 4: FFT-based correlation routine

### B. 7×7 correlation

Although the FFT-based correlation is preferred, the computational complexity may be reduced by performing a 7×7 correlation. Due to the constant update interval in generating the reference images and acquiring the raw sub-aperture images, it is seen that the difference in the shift between the sub-aperture image and the reference image is rarely greater than plus or minus 3 pixels. Hence, an output cross-correlation array of 7×7 values is enough for computing a partial 2D cross-correlation. To account for this cross-correlation routine, the new reference image consists of 26×26 pixels. This reference image is used to generate 49 different reference images with overlap as shown in Figure 5 as pre-computation step.

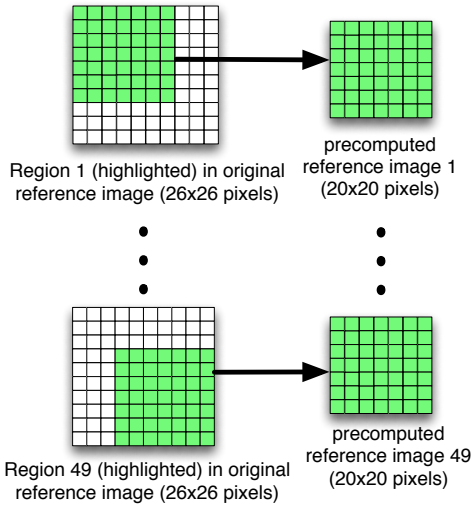


Fig. 5: Pre-computation step for the 7×7 correlation

Let $O(i,j)$ denote the original raw image where $i = 1 : nx$ and $j = 1 : ny$. Also, $nx = 20$ and $ny = 20$. Let $Ref_k(i,j)$ denote the reference image, where $k = 1 : nref$ and $nref = 49$. Therefore, the cross-correlation vector $\dot{X}_k$ can be calculated using,

$$X_k(i,j) = Ref_k(i,j) \cdot O(i,j) \quad (11)$$

$$\dot{X}_k = \sum_{i=1}^{nx} \sum_{j=1}^{ny} X_k(i,j) \quad (12)$$

### C. Interpolation routine

The maximum value and its index is calculated from either the 400 or the 49 cross-correlated values per sub-aperture. The maximum value and the neighboring pixels are then used to calculate the $\vec{x}$ and $\vec{y}$ shifts using the interpolation equations.

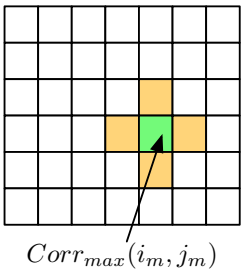


Fig. 6: Maximum correlation value and its neighboring pixels

Let $Corr_{max}(i_m, j_m)$ represent the maximum value of the correlation array $Corr(i,j)$ $\forall i = 1 : nx$ and $\forall j = 1 : ny$, $i_m$ and $j_m$ represent the indices at which the maximum value occurs as shown in Figure 6. For the FFT based correlation routine, the interpolation equations shown in 13 and 14 include an extra subtraction by 7, since the reference pixel was not corrected before correlation.

$$\vec{x} = i_m - 0.5 + \frac{Corr_{max}(i_m, j_m) - Corr(i_m - 1, j_m)}{2 * Corr_{max}(i_m, j_m) - Corr(i_m - 1, j_m) - Corr(i_m + 1, j_m)} - 7 \tag{13}$$

$$\vec{y} = j_m - 0.5 + \frac{Corr_{max}(i_m, j_m) - Corr(i_m, j_m - 1)}{2 * Corr_{max}(i_m, j_m) - Corr(i_m, j_m - 1) - Corr(i_m, j_m + 1)} - 7 \tag{14}$$

$$\vec{x} = i_m - 0.5 + \frac{Corr_{max}(i_m, j_m) - Corr(i_m - 1, j_m)}{2 * Corr_{max}(i_m, j_m) - Corr(i_m - 1, j_m) - Corr(i_m + 1, j_m)} \tag{15}$$

$$\vec{y} = j_m - 0.5 + \frac{Corr_{max}(i_m, j_m) - Corr(i_m, j_m - 1)}{2 * Corr_{max}(i_m, j_m) - Corr(i_m, j_m - 1) - Corr(i_m, j_m + 1)} \tag{16}$$

### *D. Reconstruction routine*

Once the AO algorithm determines the $\vec{x}$ and $\vec{y}$ shifts for the 1856 sub-aperture images, these values are used to compute the 1920 actuator values for driving the mirrors as described in Algorithm 1. The reconstruction routine involves the multiplication of the 1856 sub-aperture images with a 3712×1920 reconstruction matrix and the resulting products are then accumulated to give a single array of 1920 actuator values.

Let $M_{xy}(2 * p, q)$ denote the reconstruction matrix of size $p = 1856$ and $q = 1920$. Therefore, the resulting wavefront reconstruction values for driving the actuators $rac(q)$ can be calculated as follows:

**Algorithm 1** Pseudo-code for reconstruction procedure

**for** p = 1 to n, where n is the number of sub-apertures **do**

**for** q = 1 to nac, where nac is the number of actuators **do**

$xac(p, q) = \vec{x}(p) * M_{xy}(2 * p, q)$

$yac(p, q) = \vec{y}(p) * M_{xy}(2 * p + 1, q)$

$rac(q) += xac(p, q) + yac(p, q)$

**end for**

**end for**

## IV. GPU Implementation

GPUs are inexpensive, commonly available off-the-shelf (COTS) devices used to accelerate a large number of applications in [6], [7], [8]. However, GPUs are perceived as discrete power hungry HPC devices that do not not fit within embedded system computing. The NVIDIA Kepler series GPUs are more FLOPS/Watt efficient and are being used to drive real-time image processing capabilities [9].

NVIDIA's Kepler series GPU [10] consists of a maximum of 15 Streaming Execution (SMX) units and up to six 64-bit memory controllers. Each SMX unit has 192 single-precision CUDA cores as shown in Figure 7 and each core comprises of fully pipelined floating-point and integer arithmetic logic units. The SIMD execution model is one of the main differences between CPU and GPU. However, the SIMD model imposes constraints on the GPU when executing threads with task-level parallelism. Each SMX can execute only one task-level thread on a single core, while the remaining cores remain idle. The idling of a dominant number of cores in a SMX when executing task-level threads leads to poor utilization of the GPU's processing capability. The execution of divergent flow threads reduces the instruction throughput, as the multiple flow control threads are executed sequentially. Therefore, not all algorithms can run as it is on GPUs and still obtain speedup. In most cases, existing algorithms need to be re-written and optimized to increase core utilization with minimum divergent threads.

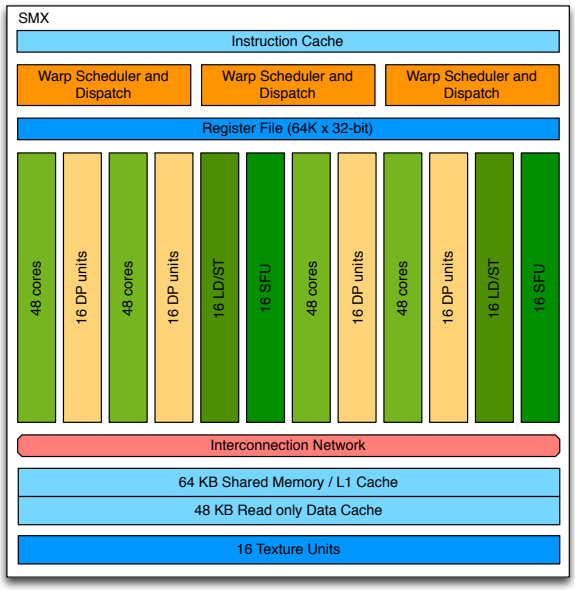


Fig. 7: Internal Architecture of Streaming Execution Processor in NVIDIA's Kepler GPU

The FFT-based correlation was implemented using NVIDIA's cufft library, which is more latency bound as compared to the $7 \times 7$ correlation. The thread partition also differs for both implementations. In the FFT correlation, the maximum value (main kernel) is determined using a reduction operation over 400 elements per sub-aperture image, which means that the number of blocks is same as the number

of images. In the $7 \times 7$ correlation method, the correlation step involves multiply accumulate operation to generate 49 values for each sub-aperture image, where the number of blocks = 49 $\times$ number of images. The maximum value is then found from these 49 correlated values/image where the number of blocks is same as the number of images. The GPU implementation of the reconstruction matrix is straightforward, where the number of blocks launched on the GPU is equal to the number of actuators. The $\vec{x}$ and $\vec{y}$ shifts calculated in the cross-correlation routine are cached into shared memory. The odd and even rows are multiplied with the $\vec{y}$ and $\vec{x}$ values and are accumulated per thread resulting in a final array of 1920 values.

## V. FPGA Implementation

FPGAs have been targeted for mapping AO algorithms as part of the European Extremely Large Telescope (E-ELT) in [11]. However, all such systems are very specific with respect to the algorithms used and in some cases also the instrumentation. The choice of the camera and the data streaming from the camera to the processing back-end plays an important role in the choice of the hardware accelerator. The Xilinx Zynq platform [12] consists of a dual core ARM processor referred to as the Processing System (PS) and a Xilinx Artix FPGA known as the Programmable Logic (PL) fused on the same die. The data transfer between the PS and PL section is through the high speed AXI bus.

Before implementing both methods on a FPGA, it is necessary to estimate the resource utilization in terms of DSP slices and logic cells to make sure that the design can fit on the FPGA. Since FFT blocks are available as part of the FPGA vendor's core library, it is not very difficult to instantiate and configure a FFT core. As opposed to the FFT method, the $7\times7$ routine is compute intensive, as it requires at least 400 multiply accumulate blocks for implementing a streaming datapath. The datapath can be pipelined to accommodate smaller multiply accumulate blocks, but it involves additional logic to keep track of the 400 pixels for each of the 49 reference images. Therefore, due to ease of implementation, the FFT based correlation method is targeted on the FPGA. Figure 8 shows the partition of the computational processes between the PS and PL sections of the Zynq platform. The FFT correlation block implementation is similar to Figure 4, where the FFT and the inverse FFT are calculated using Xilinx's cores.

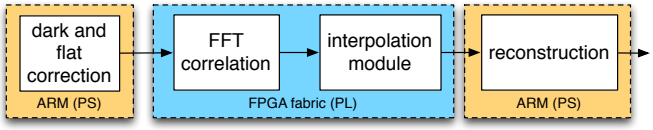


Fig. 8: Partitioning the computation between the PS and the PL sections of the Zynq

The maximum value is obtained from the inverse FFT correlated values as shown in Figure 9. The *find_max* module is implemented using comparators and the maximum value is determined. The index of the maximum value is also calculated and is used to lookup the adjacent values from the correlated sub-aperture value. The final $\vec{x}$ and $\vec{y}$ shifts are calculated as per the equations in 13 and 14.

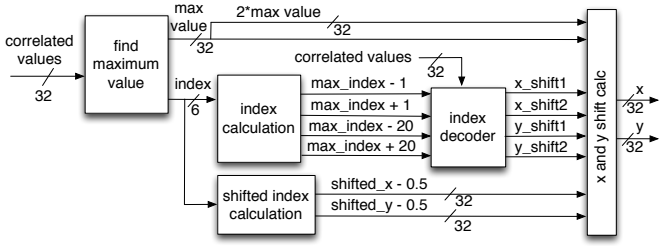


Fig. 9: Block diagram of the interpolation module

The reconstruction routine is not targeted on the FPGA as it involves a lot of floating point calculations. Although the floating point cores can be pipelined to implement the multiply accumulate operations, enough resources are not available to completely map all the calculations on the available logic fabric. Hence, the reconstruction routine is targeted on the PS section of the Zynq platform.

## VI. Results

Two different GPUs are considered for prototyping the AO algorithms: NVIDIA's GeoForce GT650M and the K20. Table I shows a comparison of the main characteristics between the both GPUs. The algorithms are compiled using NVIDIA's CUDA 5.5 SDK. The FPGA implementation is prototyped on Xilinx's ZC7020 development board [13], which consists of a dual core ARM processor and a Xilinx Artix FPGA with 85K logic cells.

TABLE I: GPU characteristics

| Characteristics | GT650M | K20 |
|---|---|---|
| Clock rate (MHz) | 700 | 900 |
| No. of cores | 384 | 2496 |
| Global memory (GB) | 1 | 5 |
| L2 Cache (KB) | 256 | 1536 |

### A. GPU Results

The FFT correlation based AO algorithm was implemented on both the GPUs for varying number of sub-aperture images as shown in Figure 10. The $7\times7$ correlation based AO algorithm results are shown in Figure 11.

It is evident from Figures 10 and 11 that the $7\times7$ correlation method is faster than the FFT correlation method. The lowest timing numbers indicate best performance in both figures and the NVIDIA K20 GPU outperforms the GT650M by more than 10x. For the reconstruction algorithm, both the GPUs provide similar speedup, the GT650M took 1.8017 ms as compared to 1.0406 ms for the K20. The K20 GPU provides excellent speedup over the GT650M GPU implementation, due to the increase in the number of cores.

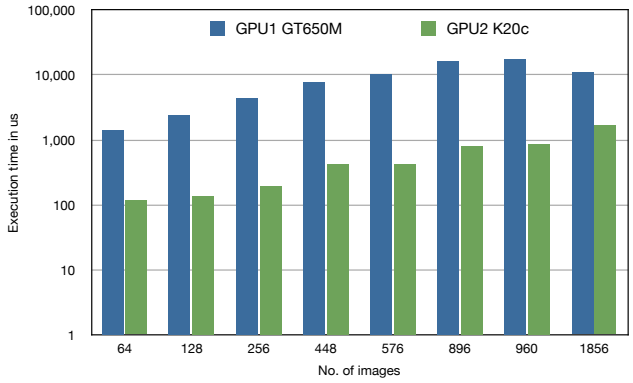


Fig. 10: GPU results for FFT based correlation

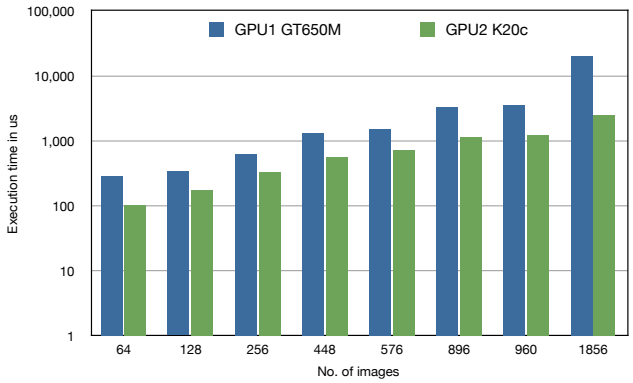


Fig. 11: GPU results for $7 \times 7$ correlation

### B. FPGA Results

The FPGA latency calculation and resource utilization is obtained using the Xilinx Vivado tool. The PL section on the Zynq platform is clocked at 250 MHz and the total latency for 1 image is $43.66\mu s$. These values are observed in real-time using the JTAG AXI monitor, which is very similar to Xilinx's Chipscope tool. Without any pipelining, the total latency for 1856 images results in 80 ms, making it 8x slower than the GT650M.

### C. Performance Analysis

Both the algorithms map better on a SIMD architecture as shown in the results in Figures 10 and 11. The real-time processing for AO systems belongs to the "Spectral" dwarf problem [14], where GPUs provide superior performance due to its native GPU CUFFT library. Hence, in this case parallelism yields better performance. At the same time, the FPGA solution can be fine-tuned to provide higher throughput. However, this fine-tuning results in higher logic resource utilization due to the introduction of state-machines and custom logic to keep track of individual images and corresponding intermediate values.

The selection of hardware accelerators is also a challenging problem, with regards to the effort required for implementation. The GPU implementations build upon the preliminary IDL, Matlab, C implementation and NVIDIA provides efficient libraries for accelerating the application. In comparison, the FPGA implementation requires a lot of time and effort setting up the computational datapath, synchronizing the movement of data between the PS and PL sections and debugging the individual blocks using the Xilinx tools. In terms of power consumption, FPGAs are more energy efficient as compared to the GPUs.

## VII. Conclusion

The GPU implementation of the AO algorithm provides best overall performance as compared to the FPGA implementation. However, the GPU's performance is relevant only when the application is compute-bound instead of being memory-bound. Also, the GPU is dependent on a master CPU for managing the data transfer. The fine-tuned FPGA implementation may provide better throughput as compared to the GPU.

## References

[1] D. N. Spergel, L. Verde, H. V. Peiris, E. Komatsu, M. Nolta, C. Bennett, M. Halpern, G. Hinshaw, N. Jarosik, A. Kogut, *et al.*, "First-year Wilkinson Microwave Anisotropy Probe (WMAP) observations: determination of cosmological parameters," *The Astrophysical Journal Supplement Series*, vol. 148, no. 1, p. 175, 2003.

[2] J. Bodily, B. Nelson, Z. Wei, D.-J. Lee, and J. Chase, "A Comparison Study on Implementing Optical Flow and Digital Communications on FPGAs and GPUs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 3, no. 2, pp. 6:1–6:22, May 2010.

[3] B. Cope, P. Y. K. Cheung, W. Luk, and L. Howes, "Performance comparison of graphics processors to reconfigurable logic: A case study," *Computers, IEEE Transactions on*, vol. 59, no. 4, pp. 433–448, April 2010.

[4] R. Tyson, *Principles of adaptive optics*. CRC Press, 2010.

[5] W. Klop, "Development of an efficient parallel wavefront reconstructor - with implementation on a GPU," Master's thesis, Delft University of Technology, June 2011.

[6] B. R. Barsdell, D. G. Barnes, and C. J. Fluke, "Analysing astronomy algorithms for graphics processing units and beyond," *Monthly Notices of the Royal Astronomical Society*, vol. 408, no. 3, pp. 1936–1944, 2010.

[7] V. Venugopal, C. D. Patterson, and K. A. Shinpaugh, "Accelerating Particle Image Velocimetry Using Hybrid Architectures," in *Symposium on Application Accelerators in High-Performance Computing (SAAHPC)*, July 2009.

[8] V. Venugopal, K. Richards, S. Barden, T. Rimmele, S. Gregory, and L. Johnson, "Accelerating Real-time processing of the ATST Adaptive Optics System using Coarse-grained Parallel Hardware Architectures," in *International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, July 2011, pp. 296–301.

[9] V. Venugopal and S. Kannan, "Accelerating real-time lidar data processing using gpus," in *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on*, August 2013, pp. 1168–1171.

[10] NVIDIA Inc., "NVIDIA's Next Generation CUDA Compute Architecture: Kepler TM GK110," Whitepaper, May 2012.

[11] D. Gratadour, A. Sevin, J. Brulé, É. Gendron, and G. Rousset, "GPUs for adaptive optics: simulations and real-time control," pp. 84 475R–84 475R–7, 2012.

[12] Xilinx Inc., "Zynq-7000 All Programmable SoC." [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm

[13] Xilinx Inc. (2012) Xilinx Zynq-7000 SoC ZC702 Evaluation kit.

[14] R. Inta, D. J. Bowman, and S. M. Scott, ""The "Chimera": An Off-The-Shelf CPU/GPGPU/FPGA Hybrid Computing Platform," *International Journal of Reconfigurable Computing*, vol. 2012, no. 241439, p. 10, 2012.