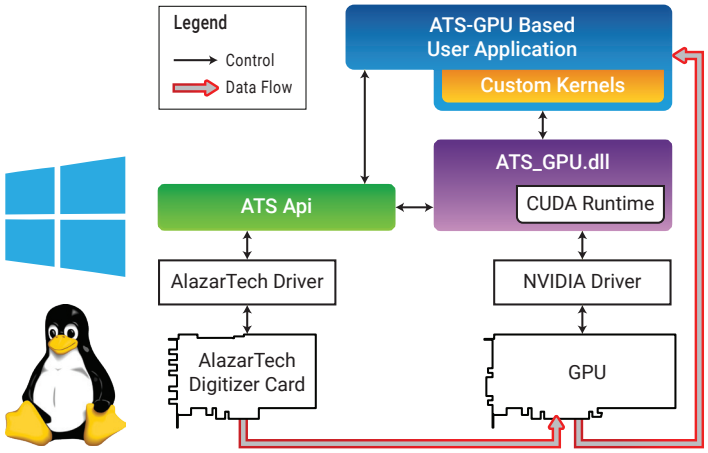
ATS-GPU-BASE
Real Time Signal Processing Software

- Transfer A/D data to GPU at high speed
- Up to 6.9 GB/s transfer rate for PCIe Gen 3 digitizer boards
- Supports CUDA® compute capability 3.0+
- Designed to work with AlazarTech® PCI Express waveform digitizers
- Availability of an optional OCT Signal Processing Library that includes:
  - ◇ Very High-Speed Floating Point FFT
  - ◇ Dispersion Compensation and Windowing Functions
- Compatible with Windows® & Linux®



Product	GPU Compatibility	Operating System	Kernel Programming Language	Throughput to GPU
ATS-GPU-BASE version 4.0	CUDA compute capability 3.0+	64-bit Windows & 64-bit Linux	CUDA	Up to 6.9 GB/s

Overview

ATS-GPU-BASE is a software library developed by AlazarTech that transfers data acquired by its family of PCI Express waveform digitizers to a CUDA-enabled Graphical Processing unit (GPU) at sustained transfer rates as high as 6.9 GB/s.

Modern GPUs include very powerful processing units and a very high speed graphical memory bus. This combination makes them perfectly suited for signal processing applications.

Unfortunately, it is not easy for other hardware devices, such as waveform digitizers, to DMA data directly to the GPU's on-board memory. This forces users to manually copy data from the buffer returned by the waveform digitizer to the GPU. This copying process is relatively slow and causes the overall data throughput to be drastically reduced.

ATS-GPU-BASE solves this problem by transferring data acquired by an AlazarTech PCI Express waveform digitizer to a CUDA-enabled GPU at sustained transfer rates as high as 6.9 GB/s. Data is presented in GPU memory as a buffer queue. Expert-level GPU programmers can create very high-performance custom kernels to manipulate this data using an easy-to-use application programming interface (API).

ATS-GPU-BASE includes an example program that demonstrates how to use the ATS-GPU library to transfer data from a waveform digitizer to a GPU. The example also shows how to do simple data processing on the GPU using CUDA kernels, and how to transfer the processed data back to host memory (RAM). Users can use this example program as a starting point to create their own custom kernels to do GPU-based DSP.

An example of a high-performance kernel is the optional ATS-GPU-OCT library (sold separately), which acquires data using ATS-GPU-BASE and then implements full OCT signal processing algorithm using CUDA kernels. ATS-GPU-OCT was benchmarked at up to 900,000 4K FFTs per second, demonstrating the power and efficiency of the ATS-GPU-BASE platform. Kernel code running on the GPU can do zero-padding, apply a windowing function, do a floating point FFT, calculate the amplitude, and convert the result to a log scale. Please refer to the ATS-GPU-OCT datasheet for more information.

GPU-Based Signal Processing

Graphical Processing Units (GPUs) were originally designed for rendering high-quality video for gaming applications, which required being able to perform massive amount of real-time calculations. The highly parallel architecture of modern GPUs also makes them an ideal platform for digital signal processing (DSP) and high performance computing (HPC) systems.

In the past, complex real-time signal processing, such as FFT, correlation, FIR filtering etc., could only be achieved using dedicated DSP processors or by implementing the algorithms inside and FPGA or an ASIC. All these methods are non-trivial, expensive, time consuming and require highly specialized engineering skills.

Using GPUs, users can implement any algorithm that can be parallelized in a GPU using well known software techniques and gain a better than 10-fold improvement over CPU based signal processing. The reason why GPUs perform so well for DSP applications is that they contain hundreds of processing cores (kernels) running in parallel, while sharing a very high speed graphical memory bank.

Data Throughput Benchmarks

Benchmarks were done on AlazarTech ATS®9373 in an Intel i9-7900X 10-Core @ 3.3 GHz system with an ASUS® x299 motherboard, 32 GB DDR4, and NVIDIA® Quadro® P5000 GPU. The following throughput was measured in continuous streaming mode:

Buffer Size	Throughput*
1 MB	6.8 GB/s
2 MB	6.8 GB/s
4 MB	6.9 GB/s
8 MB	6.9 GB/s

* Based on benchmarks done in November 2018

Tests in an Intel i7 5930k 6-core @ 3.5 GHz system with an ASUS x99 Deluxe motherboard, 64 GB DDR4, and the same NVIDIA Quadro P5000 GPU produced very similar results.

A Typical ATS-GPU-BASE Application

A typical user application that uses ATS-GPU-BASE consists of the following minimum sections:

1) User application sets up waveform digitizer hardware (sample rate, input range, trigger parameters etc.).
2) User application allocates buffers and sets up the GPU.
3) User application starts data capture.

   ATS-GPU-BASE starts streaming data to GPU, one buffer at a time.
4) User-written GPU kernels do the following:
   - Process a buffer;
   - Copy result buffer to user memory;
   - Get next buffer, and repeat.
5) User application running on CPU consumes result buffer.

   For highest performance, make sure data consumption is faster than the rate at which result buffers are supplied by GPU kernels.
6) This continues until the application has to be closed.

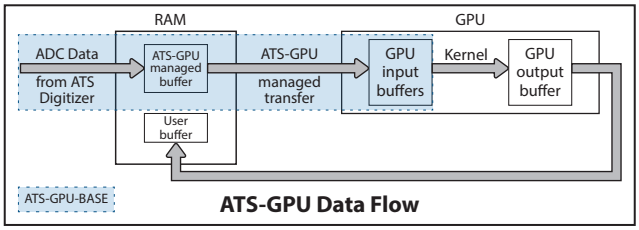
ATS-GPU-BASE and CUDA Runtime Library

ATS-GPU-BASE is shipped with a specific version of CUDA runtime library and links statically to it.

Programmers are allowed to use a different version of CUDA runtime library for their custom kernel code. NVIDIA guarantees that the two versions of CUDA runtime libraries will be interoperable.

Note: ATS-GPU only supports Windows versions and Linux distributions that are supported by NVIDIA's CUDA Toolkit. 32-bit operating system support is also similarly limited by NVIDIA. In particular, the ATS-GPU-OCT Signal Processing library cannot be built as a 32-bit library. We currently use CUDA toolkit 10.0, older versions are untested.

ATS-GPU Data Flow



ATS-GPU-BASE is supplied with an example user application in source code. The application includes GPU kernels that use ATS-GPU-BASE to receive data, do very simple signal processing (data inversion), and copy the processed (inverted) data back to a user buffer. All this is done at the highest possible data transfer rate.

Programmers can replace the data inversion code with application-specific signal processing kernels to develop custom applications.

Performance Dependencies

Since the host CPU is involved in moving data to and from the GPU and in scheduling GPU kernels, CPU speed and motherboard's memory bandwidth can have a significant impact on the overall performance.

The optional ATS-GPU-OCT Signal Processing Library was used to benchmark performance.

On an ASUS X299 motherboard that uses an Intel i9-7900X 3.3 GHz CPU and DDR4 memory (32 GB RAM), a combination of the ATS9373 and NVIDIA Quadro P5000 (Pascal) GPU was able to do a 2048 point FFT at a rate of 1900 kHz.

An older DDR3-based machine performed significantly slower.

Complexity of the kernel code running on the GPU can have a significant impact on the overall performance. Users should optimize their code to take advantage of the GPU's high speed memory.

Computer Power Supply

GPUs are power hungry. Even consumer-grade models such as ASUS GTX980 require a power supply that can provide at least 500 Watts of power. As such, users must make sure their computer's power supply has sufficient capacity.

Compatible GPUs

ATS-GPU is designed to be compatible with all compute capability 3.0 or higher CUDA-enabled GPUs. Testing was done using NVIDIA Quadro P5000.

It should be noted that ATS-GPU supports only one GPU at a time. If you have multiple GPUs installed in your computer, ATS-GPU will let you select one of them for use.

Data Throughput to GPU

The data transfer rate to GPU is dependent on the generation of PCI Express digitizer board used:

PCIe Link Speed	Transfer Rate
Gen 3: ATS9373, ATS9371	Up to 6.9 GB/s
Gen 2: ATS9360, ATS9416	Up to 3.5 GB/s
Gen 1: ATS9870, ATS9350, ATS9351, ATS9120, ATS9625, ATS9626, ATS9440, ATS9462	Up to 1.6 GB/s (Exact rate is limited by digitizer sample rate)

Compatible Waveform Digitizers

All AlazarTech PCI Express waveform digitizers are compatible with ATS-GPU. Only single-board configurations are supported at this time.

AlazarTech's PCI bus waveform digitizers are not supported, as the host CPU is more than capable of handling data rates generated by PCI bus boards.

ATS-GPU cannot directly be interfaced with non-AlazarTech waveform digitizers.

Software Licensing Policy

Users are allowed to freely distribute the ATS-GPU-BASE library as long as they have purchased one ATS-GPU-BASE license and there is an AlazarTech PCI Express waveform digitizer present in the same computer. If an AlazarTech PCI Express waveform digitizer is not present in the computer, users must purchase a separate license for each computer on which ATS-GPU is installed.

In no case is the user allowed to distribute or share the source code of ATS-GPU with other users.

ATS-GPU-BASE
Real Time Signal Processing Software

Annual Subscriptions
The purchase of an ATS-GPU-BASE license includes a subscription that provides customers with the following benefits for a period of 1 year on ATS-GPU-BASE:

- Download ATS-GPU-BASE updates from the AlazarTech website;
- Receive new example programs as they become available;
- Receive technical support on ATS-GPU-BASE.

Additional add-on libraries for ATS-GPU-BASE, such as the *ATS-GPU-OCT Signal Processing Library* are not covered by the annual subscription, i.e. holders of an annual subscription will have to purchase subscriptions for additional libraries separately.

It is recommended that customers renew their ATS-GPU-BASE subscriptions in order to maintain their product up-to-date, and receive new features and technical support.

Note that support is provided for product bugs, and not for writing custom GPU kernels or for learning GPU programming.

Writing Custom GPU Kernels
ATS-GPU-BASE includes an example program in C/C++ source code, which implements very simple GPU kernels that invert data and write it back to a buffer in computer memory.

Expert-level GPU programmers who need to write their own kernels should start with the included source code, add CUDA code in the appropriate place, and compile their libraries.

The example program is provided with a Visual Studio project and a CMake build file. We use more recent C++ features, and Visual Studio 2015 and later is required. On Linux, a C++11 compiler is required and can be accessed on older distributions via a devtoolset (RHEL and CentOS 6 for example).

Writing, testing, and debugging modified kernels will be the sole responsibility of the user and AlazarTech will not be responsible for assisting the user with such custom modifications.

Users must have expert programming knowledge of CUDA development in order to customize ATS-GPU kernels.

ATS-GPU main API functions
```
ATS_GPU_AbortCapture
ATS_GPU_AllocBuffer
ATS_GPU_FreeBuffer
ATS_GPU_GetBuffer
ATS_GPU_PostBuffer
ATS_GPU_QueryCUDADeviceCount
ATS_GPU_QueryCUDADeviceName
ATS_GPU_SetCUDAComputeDevice
ATS_GPU_Setup
ATS_GPU_StartCapture
```

ORDERING INFORMATION

ATS-GPU-BASE: GPU Streaming Library License + 1 Year Subscription	ATSGPU-001
ATS-GPU-BASE-1YR: 1 year extended support & maintenance for ATS-GPU-BASE	ATSGPU-002
ATS-GPU-OCT: Signal Processing Library License + 1 Year Subscription (requires ATSGPU-001)	ATSGPU-101
ATS-GPU-OCT-1YR: 1 year extended support & maintenance for ATS-GPU-OCT	ATSGPU-102

Manufactured By:

Alazar Technologies, Inc.

6600 TRANS-CANADA HIGHWAY, SUITE 310
POINTE-CLAIRE, QC, CANADA H9R 4S2

TOLL FREE: 1-877-7-ALAZAR OR 1-877-725-2927
TEL: (514) 426-4899 FAX: (514) 426-2723

E-MAIL: sales@alazartech.com