Application Note

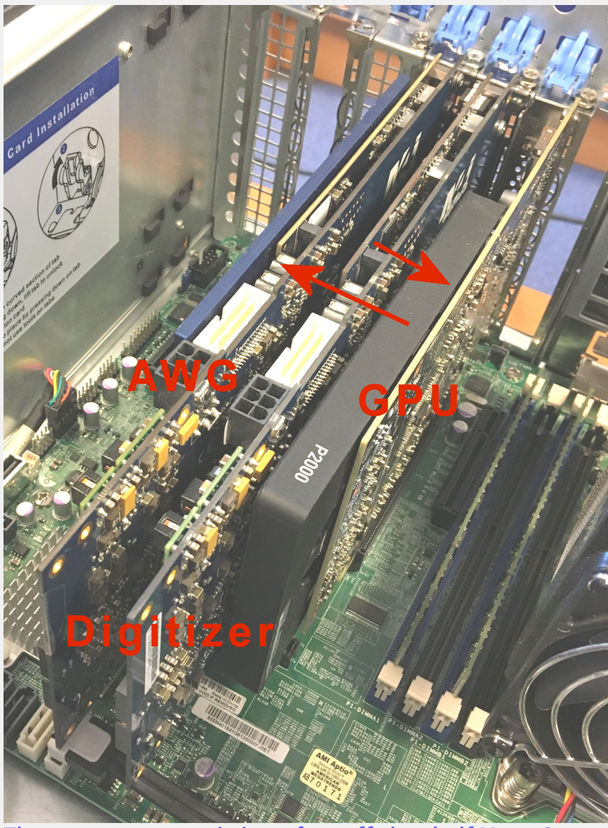
Closed Loop Test using a PC Digitizer and AWG

This application note shows the setup of a PC-based closed-loop using general purpose PCIe Digitizers and AWGs in combination with plain C++ programming. Different setups have been tested using Windows and Linux environment. The article also shows the differences in latency when adding a GPU for processing power to the loop.

Target applications

There are different potential applications on the market that need a low latency to make fast decisions or to make fast modifications. These applications are normally covered under the name "real-time" applications. This name originally referred to hardware based real-time applications, where you need a defined and guaranteed response time to be sure that your controlling setup is reacting in time to an external stimulus. Some examples are found in automotive or aerospace systems. However, the label "real-time" is also often used as a synonym for "fast answer". In such cases the interesting figure is the latency between input and output (or the decision and response). This article should be of interest for anyone involved in the following target applications:

- Control loops: a controller type application where the input is used as a base for the output to control some external device or devices.
- Decision making: reacting on an external event by creating some kind of changed output.
- Manipulation loops: acquiring a signal, modify the signal and then outputting it again.
- Stimulus response: generating a signal, acquiring a response, analyzing it and making a decision



The test system consisting of an off-the-shelf Xeon Server board, a Spectrum M4i.4451-x8 500 MS/s 14 Bit Digitizer, a M4i.6622-x8 625 MS/s 16 Bit AWG and a Nvidia Quadro P2000 GPU

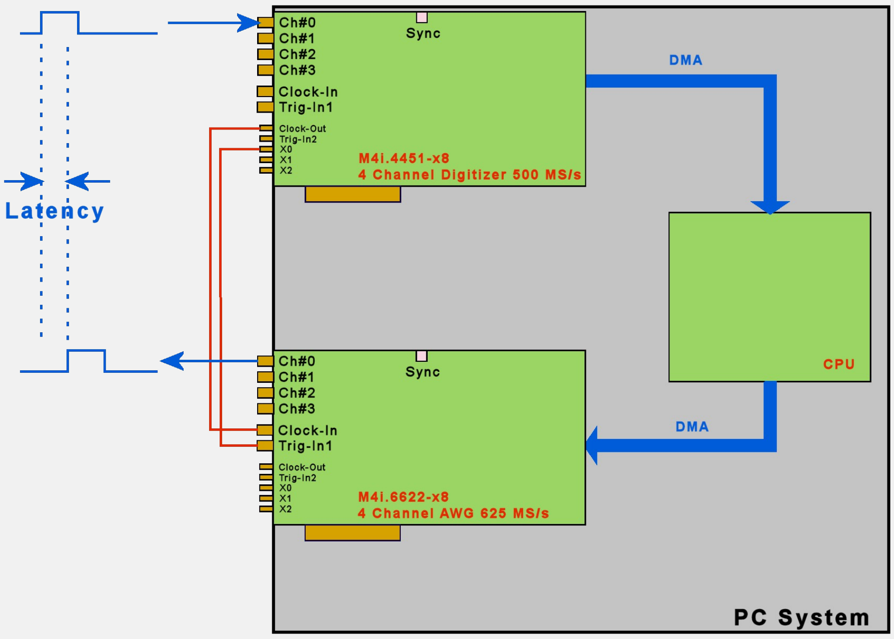
Technical background

All Spectrum products are designed for fast FIFO mode data transfer, be it a Digitizer or an AWG. A fast FIFO mode is based on optimized drivers that control the scatter-gather DMA (direct memory access) process. Unfortunately, the demand for high throughput always generates the need for the buffering of data. First, a fast DMA is based on large buffer size transfers. This optimizes the relationship between the data transfer and the control command overhead. Second, any external delay in the answering or response time can be compensated by the large buffers. If the buffers are too small, the result is usually an overrun or underrun of the FIFO buffers, which stops the FIFO transfer.

The classical approach for these applications is to have a real-time operating system with

defined answer times and priority settings, matching real-time drivers and hardware that is optimized for short latency (and not for highest throughput). An even faster solution would be to have the complete process implemented in hardware (FPGA) without interaction by the host system. However, sometimes these setups are just programming or cost overkill. FPGA programming requires specific knowledge and the hardware often has limitations that restrict a systems functionality and its data processing capability. Similarly, real-time systems add the expense of a license for a real-time operating system and may restrict the access to useful third-party software tools. In many cases setting up a closed-loop system that's based on Windows (or Linux) may prove to be more economical as well as being faster to implement.

Test results for PC-based software (Windows)



To provide an indication of what can be achieved for a closed loop type system, running under Windows, a test system was configured using a Spectrum M4i.4451-x8 digitizer and an M4i.6622-x8 AWG. Both cards are externally connected. The reference clock output of the digitizer is used to feed the reference clock input of the AWG, providing clock synchronization, and the trigger output of the digitizer is used to start the output of the AWG. The AWG is pre-loaded with some zero-data which is replayed until the data acquired digitizer data is received. The size of the pre-loaded data defines the latency between acquired data and the looped data. The right hand block diagram shows the setup.

Tests have been done on a Spectrum streaming system SPcB6-E6 running Windows 7 Professional

Speed	Channels	FIFO transfer	software buffer	notify size	latency
500 MS/s	1 (14/16 Bit)	2 x 1 GByte/s	768 kByte	256 kByte	790 µs
250 MS/s	1 (14/16 Bit)	2 x 500 MByte/s	360 kByte	120 kByte	740 µs
125 MS/s	1 (14/16 Bit)	2 x 250 MByte/s	240 kByte	80 kByte	980 µs
62.5 MS/s	1 (14/16 Bit)	2 x 125 MByte/s	240 kByte	60 kByte	1,97 ms
500 MS/s	2 (14/16 Bit)	2 x 2 GByte/s	1.5 MByte	256 kByte	790 µs
250 MS/s	4 (14/16 Bit)	2 x 2 GByte/s	1.5 MByte	256 kByte	790 µs

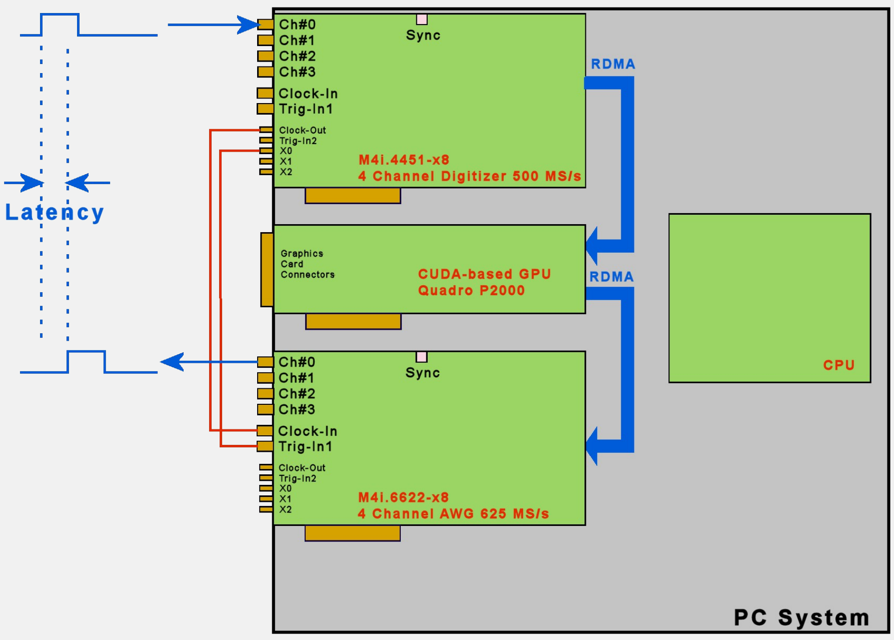
As can be seen from the table, the best results have been achieved with a notify size for the DMA transfer that splits the software buffer into 3 or 4 parts. The minimum notify size that works stably is around 64 kByte. That limitation comes from the internal hardware buffers whose size needs to be compensated by the software buffers. The results shown here are the best performance that was reached for run-times of a minute.

Test results for PC-based software (Linux)

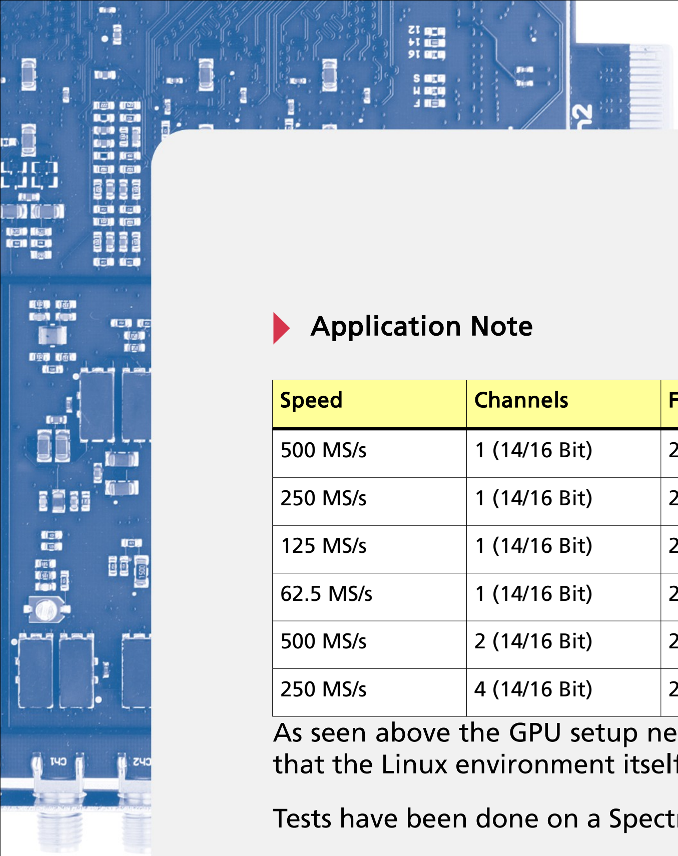
Same setup as above but using a Linux operating system.

Speed	Channels	FIFO transfer	software buffer	notify size	latency
500 MS/s	1 (14/16 Bit)	2 x 1 GByte/s	2 MByte	256 kByte	2.1 ms
250 MS/s	1 (14/16 Bit)	2 x 500 MByte/s	1 MByte	128 kByte	2.1 ms
125 MS/s	1 (14/16 Bit)	2 x 250 MByte/s	512 kByte	64 kByte	2.1 ms
62.5 MS/s	1 (14/16 Bit)	2 x 125 MByte/s	512 kByte	32 kByte	2.1 ms
500 MS/s	2 (14/16 Bit)	2 x 2 GByte/s	4 MByte	256 kByte	2.1 ms
250 MS/s	4 (14/16 Bit)	2 x 2 GByte/s	4 MByte	256 kByte	2.1 ms

Results for GPU-based software (SCAPP option) under Linux



In a second test setup using an M4i.4451-x8 digitizer and an M4i.6622-x8 AWG a GPU is added to the system. Again, both cards are externally connected with the reference clock output of the digitizer feeding the reference clock input of the AWG and the trigger output of the digitizer starting the output of the AWG. In this setup the GPU directly receives and transmits data using RDMA. The GPU only copies the data once from digitizer buffer to AWG buffer with no manipulation. The AWG is pre-loaded with some zero-data which is replayed until the acquired digitizer data is received from the GPU. The size of the pre-loaded data defines the latency between acquired data and looped data



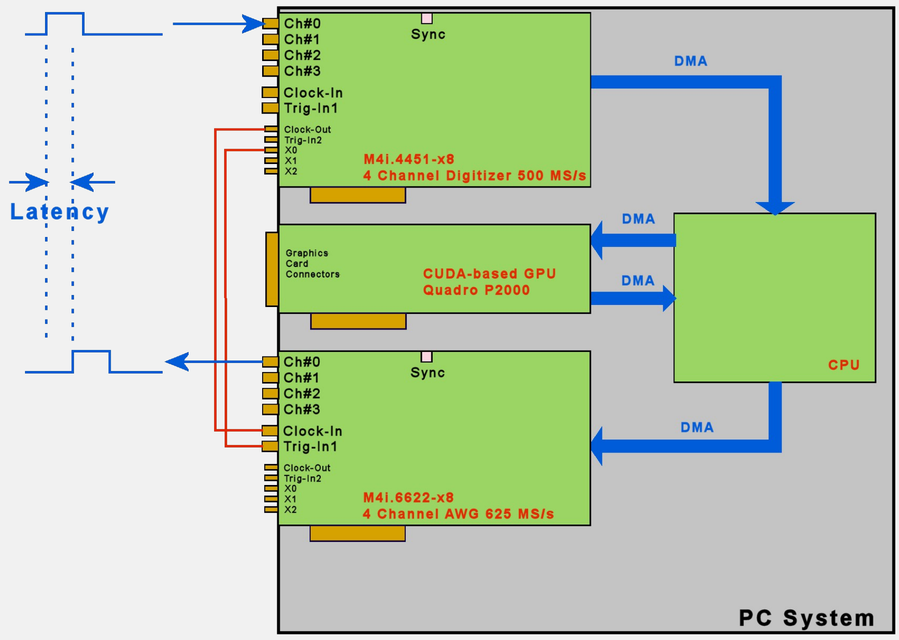
Speed	Channels	FIFO transfer	software buffer	notify size	latency
500 MS/s	1 (14/16 Bit)	2 x 1 GByte/s	2.6 MByte	256 kByte	2.6 ms
250 MS/s	1 (14/16 Bit)	2 x 500 MByte/s	1.3 MByte	128 kByte	2.6 ms
125 MS/s	1 (14/16 Bit)	2 x 250 MByte/s	640 kByte	64 kByte	2.6 ms
62.5 MS/s	1 (14/16 Bit)	2 x 125 MByte/s	640 kByte	64 kByte	5.2 ms
500 MS/s	2 (14/16 Bit)	2 x 2 GByte/s	5.1 MByte	256 kByte	2.6 ms
250 MS/s	4 (14/16 Bit)	2 x 2 GByte/s	5.1 MByte	256 kByte	2.6 ms

As seen above the GPU setup needs more buffers to run stable. This is mainly due to the fact that the Linux environment itself needs more buffers to compensate for background tasks.

Tests have been done on a Spectrum streaming system SPcB6-E6 running Linux

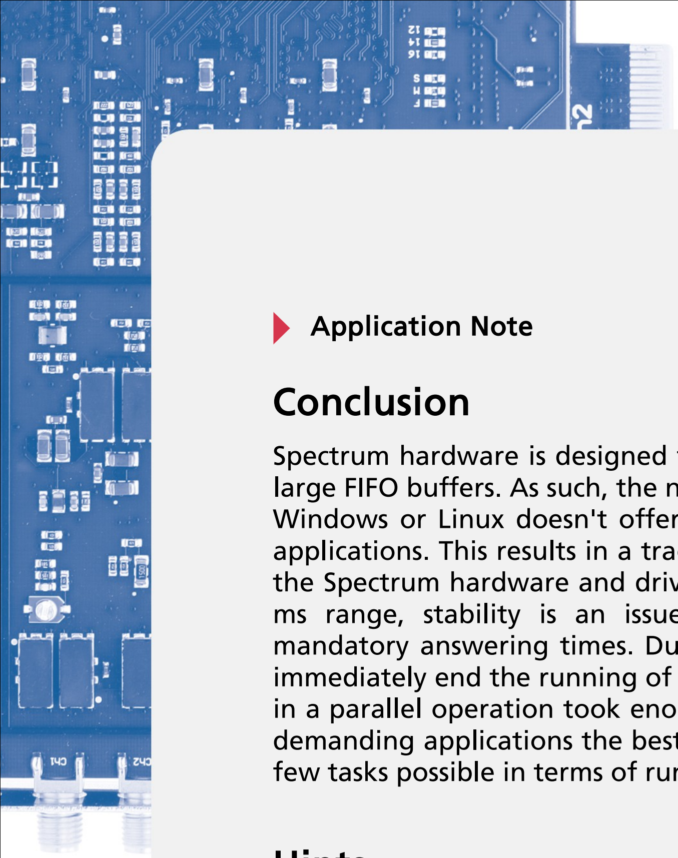
Results for GPU-based software under Windows

Under Windows there is no direct data transfer (RDMA) between digitizer/AWG and GPU possible as the Nvidia GPU driver doesn't support this feature. Therefore data has to be transferred to CPU memory first, copied there and then transferred to the GPU. The manipulated data has to take the same way back. In general this adds some latency and also adds some more risks for overrun/underrun as in total four DMA transfers are running for each block.



Speed	Channels	FIFO transfer	software buffer	notify size	latency
500 MS/s	1 (14/16 Bit)	2 x 1 GByte/s	3 MByte	1 MByte	3.1 ms
250 MS/s	1 (14/16 Bit)	2 x 500 MByte/s	1.5 MByte	512 kByte	3.1 ms
125 MS/s	1 (14/16 Bit)	2 x 250 MByte/s	750 kByte	256 kByte	3.1 ms

As expected the closed-loop test results and stability are worst in that scenario. Doubling the DMA transfers adds latency and adds the need for larger buffers to maintain stability and avoid underruns or overruns.



Conclusion

Spectrum hardware is designed to achieve high data throughput, meaning the products use large FIFO buffers. As such, the non-deterministic behavior of standard operating systems like Windows or Linux doesn't offer perfect system performance when it comes to closed-loop applications. This results in a trade-off between stability and latency performance. Although the Spectrum hardware and driver allows very fast reaction times, with a latency in the sub-ms range, stability is an issue when the operating system layer can’t guarantee the mandatory answering times. During testing we encountered many system flaws that could immediately end the running of a closed loop process. For example, just opening the browser in a parallel operation took enough processor time from our loop to stop it running. So, for demanding applications the best approach is to insure the operating system is performing as few tasks possible in terms of running programs and background jobs.

Hints

The test programs are available as part of the examples package. Please note the following:

- Best performance is received with the release version of the program only, the debug version will always have much worse results
- Any driver logging need to be de-activated
- The software development system needs to be closed to run the loop. The software development GUI always monitors programs, even the release version, and therefore degrades performance
- Close all other programs and services that may run in the background
- Please note that all these test results have been achieved with simple data copy only. There was no calculation or manipulation on the data. It is expected that real-world applications will need longer latency to run stable due to the additional calculation time
- Please note that the results shown above are just the limit what can be achieved on best circumstances. That is not a guaranteed performance. For critical applications where an interruption would cause problems a large safety margin and extensive tests would be needed