Evaluation of the ITER Real-Time Framework for Data Acquisition and Processing from Pulsed Gigasample Digitizers

M. Kadziela¹ · B. Jablonski¹ · P. Perek¹ · D. Makowski¹

Accepted: 3 October 2020 / Published online: 3 November 2020
© The Author(s) 2020

Abstract

Plasma diagnostics systems are becoming progressively more advanced. Contemporarily, researchers strive to achieve longer plasma pulses, and therefore, appropriate hardware is required. Analogue-to-Digital Converters are applied for data acquisition in many plasma diagnostic systems. Some diagnostic systems need data acquisition with gigahertz sampling frequency. However, gigasample digitizers working in continuous mode generate an enormous stream of data that requires suitable, high-performance processing systems. This becomes even more complicated and expensive for complex multi-channel systems. Nonetheless, numerous plasma diagnostic systems operate in a pulse mode. Thomson scattering (TS) diagnostics is a good example of a multi-channel system that does not require continuous data acquisition. Taking this into consideration, the authors decided to evaluate the CAEN DT5742 gigasample digitizer as a more cost-effective solution that would utilize the pulsed nature of the TS diagnostic system. The paper presents a complete data acquisition and processing system dedicated for plasma diagnostics based on the ITER real-time framework (RTF). Integration of RTF with real hardware is discussed. The authors of the paper have developed software including RTF function block for the CAEN DT5742 digitizer, example data processing algorithms, data archiving and publishing for plasma control system.

Keywords Data acquisition · Data processing · High-speed digitizer · Real-time framework · Thomson scattering diagnostics

Introduction

Research facilities, such as ITER [1–4], W7-X [5–7], KSTAR [8–10], ASDEX-U [11, 12] or JET [13, 14] work on plasma diagnostic systems, which use analogue-to-digital converters (ADCs) sampling with GS/s speed. However, the application of gigasample ADCs could be expensive in such systems, especially when they require large quantities of ADC channels. Thomson scattering (TS) diagnostics is an example of such a system where high-speed digitizers are required [1, 10, 14, 15]. The operating principle behind a Thomson Scattering system is that a laser beam is injected into the plasma with a specific frequency. Subsequently, the scattered light is collected and sent to a polychromator where it is split to isolate and

characterize subranges of the spectrum. Then the analogue signal is digitized and further processed.

Because of the significant price of gigasample digitizers, the authors proposed and tested a more cost-effective solution that would take advantage of the pulsed nature of the Thomson Scattering diagnostic system. The investigated high-speed digitizer working in pulsed mode is the CAEN DT5742 which is shown in Fig. 1. The 12-bit digitizer can work up to 5 GS/s sampling frequency which is well suited for the high frequency of operation with short data acquisition windows.

Since the Thomson Scattering diagnostic system can operate with a 50 Hz repetition rate, the time window for data acquisition and processing is 20 ms. The authors decided to use the ITER Real-Time Framework for executing a chain of selected algorithms.

RTF is a powerful software environment for the implementation of real-time algorithms. Moreover, the framework speeds up the development process, since various software and library components are included in the

✉ M. Kadziela
mkadzi@dmc.s.pl

¹ Department of Microelectronics and Computer Science, Lodz University of Technology, Lodz, Poland

Fig. 1 CAEN DT5742 digitizer [16]

framework. The ITER Real-Time Framework has been developed to allow for a modular way of developing applications so that the entire process is simplified, and the resulting algorithms are easily reusable and adaptable. Real-Time applications developed with the use of ITER RTF are created with Function Blocks, which are the basic building components of the application and they represent operations and algorithms. Creating an application consists of connecting input and output ports of the different Function Blocks to form a complete application. Moreover, Function Blocks can also be used to encapsulate other Function Blocks which allows the application to have an apparent and hierarchical structure that is easy to comprehend. Real-Time Framework also contains services that provide site-specific facilities orthogonal to the Function Blocks. Services can be used for data exchange, data archiving, profiling, etc. The Real-Time Framework comes with a limited number of predefined Function Blocks that can be used by any application developer, however, there exists the possibility to develop user-defined Function Blocks with custom functionalities. Usually, those are developed using C++. Nonetheless, ITER RTF offers support for the development of models in Simulink, that can be converted into RTF Function Blocks. However, there have not been any real applications developed with the use of Simulink and ITER RTF yet.

This is the first application of the framework as it has not been tested for the purpose of diagnostics with real hardware according to the authors' knowledge. This paper focuses on the evaluation of the ITER Real-Time Framework and the functional aspects of the development process connected with it. It includes preliminary conclusions based on the authors' experience with the framework and the limitations that come with the use of Simulink as the tool for the development of user-defined Function Blocks. The integration of RTF with hardware as well as a complete data acquisition and processing system based on RTF is discussed. The optimization and detailed latency measurements are out of the scope of the paper. However, the authors plan to investigate the real-time aspects and performance of the framework in the future.

Architecture of Data Acquisition and Processing System

A typical data acquisition and processing system is composed of a sensor connected to a digitizer, data acquisition software, processing algorithms, archivization, system configuration, and control.

The pulsed nature of diagnostic systems such as Thomson Scattering allows the use of a pulsed mode data acquisition system where the sampling can be precisely triggered in relation to the probe signal. To digitize the pulses that are in the range of tens of nanoseconds, a GS/s sampling frequency is required.

Pulsed High-Speed Digitizers

For this purpose, a CAEN DT5742—a Switched Capacitor Digitizer with up to 5 GS/s sampling frequency was used. The Switched Capacitor technology is based on the domino ring sampler, version 4 (DRS4) chip. It has 1024 capacitor cells per each of its 16 analog input channels which equates to a 200 ns acquisition window at 5 GS/s [16].

Figure 2 shows a data acquisition and processing system based on the CAEN digitizer. The system comprises the digitizer, a fast controller with an interface card, a central processing unit (CPU), and a network card, data archiver, and a host for data visualization. The CAEN DT5742 digitizer is connected using a bidirectional optical link to an interface card inside the fast controller. This connection enables the fast controller to configure the digitizer and receive data. The data can then be processed with the use of dedicated applications working in real-time under the RT Linux kernel. One of the dedicated solutions for real-time data processing is the ITER Real-Time Framework which will be described later in this paper. The diagram depicts that data processing is done mainly with the use of CPU. After data is processed, the network card sends it via synchronous databus network (SDN) and data acquisition network (DAN). With SDN the measurements are transferred in real-time to be used by various systems such as the plasma control system (PCS). On the other hand, DAN is used for large, raw data stream archivization, so time is not a factor. Finally, the archived data can be visualized for further analysis and investigation.

Software Architecture of Data Acquisition and Processing System

The generic software part for data acquisition and processing is shown in Fig. 3. Firmware in FPGA controls the device and transmits data via the optical link. Linux device drivers are programs that run in kernel mode and can

Fig. 2 Data acquisition and processing system based on CAEN digitizer

Fig. 3 Software layers of data acquisition and processing system

interact with the hardware of the respective device. They provide support for the data stream. The digitizer library enables the configuration and control of the digitizer. Data acquisition software is used for controlling the data acquisition process. On the other hand, the data processing part refers to the part that performs some operations on the digitized data. Finally, the data visualization part is responsible for displaying the data.

Framework for Real-Time Data Processing

Real-Time frameworks generally operate with a specific clock rate, which means that, for instance, a control cycle is executed at 1 kHz frequency resulting in a 1-ms window for all tasks to be completed in. Thomson Scattering diagnostics is a soft real-time system [17–19] working with a 50 Hz repetition rate. The data acquisition and processing must be completed promptly to deliver measurements to

other systems such as the plasma control system (PCS). However, a missed deadline will not have catastrophic consequences as this diagnostic system is not a part of machine protection.

The generic structure of a real-time data acquisition and processing system is shown in Fig. 4. The following three main components can be distinguished: data acquisition, data processing, and data archiving. They can be implemented in the Real-Time Framework. Data acquisition is responsible for digitizing and collecting data, data processing can be a chain of different algorithms that are used to process data as needed. Lastly, the data can be sent via SDN and DAN, for plasma control and archivization purposes. All components in RTF are developed as separate Function Blocks or encapsulated chains of multiple blocks working sequentially.

Thomson Scattering diagnostics is an example of a system where soft real-time is needed. The difficulty with real-time data processing in such a system relates to the fact that the execution time window allocated for the entire data processing path is around a few milliseconds. This usually means that multiple time-consuming and complex algorithms will have to complete promptly.

Real-Time Framework

ITER Real-Time framework has been developed as a new solution that would help to unify the process of algorithm development and system implementation. RTF is a powerful environment for the implementation of functions that derive physics measurements in real-time. Moreover, it separates algorithms, hardware interaction, and system configuration. The RTF application contains processing logic that can be distributed and run on different threads, processes, or computer nodes to speed up the computations. Real-Time Framework has a built-in mechanism for sending data called gateways. Additionally, an advantage of RTF is that a lot of work connected with the application development process has already been done by the developer which allows more time to be allocated strictly for

Fig. 4 Real-Time Framework data acquisition and processing schematic

algorithm development. Real-Time Framework gives the possibility to precisely configure the deployment of an application by changing input and output data sources as well as parameters without recompiling the application. ITER RTF allows for a modular way of developing applications that simplifies the whole process and makes it more adaptable and the applications reusable. Real-Time applications are created with Function Blocks which represent an operation together with its inputs and outputs (see Fig. 5).

Data Processing Block

Function Blocks are basic building components for RTF applications. Composing the application is done by connecting input and output ports of consecutive blocks. Moreover, Function Blocks can also be used to encapsulate other FBs giving the application a more hierarchical structure that is easier to comprehend.

The ITER Real-Time Framework comes with a library of basic blocks that correspond to simple operations. Those blocks can be used by application developers. However, there also exists the possibility to develop user-defined Function Blocks with custom functionalities. RTF allows for the custom blocks to be installed as a plugin on the

system and utilized in future applications without the need for recreating them. Custom Function Blocks can be developed in C/C++ programming languages, however, there also exists a way to transform data processing algorithms developed in Simulink to be compatible and usable with RTF.

Pulse Fitting in Real-Time

Pulse Fitting Algorithm

Thomson Scattering diagnostics requires various data processing algorithms to obtain the desired measurements. One of the basic algorithms is pulse fitting, which has been identified as the most time-consuming and complex part of data processing. Thus, the authors have chosen to develop it first. The algorithm uses the raw signal to iteratively find a set of four parameters that characterize the curve that best fits the data. The output of this algorithm can be used to generate the final curve which is described by the resulting parameters. The formula used in the algorithm to generate the approximation curve is [11]:

$$P(t) = P_L \int_0^t e^{-\frac{(t'-t_L)^2}{\tau_L^2}} e^{-\frac{t-t'}{\tau_{amp}}} dt' \quad (1)$$

Where $P(t)$ is the approximation function shape, P_L is the signal amplitude, t_L is the time at which the laser pulse attains a maximum, τ_L is the time duration of the laser pulse (Full-Width Half Maximum of the pulse), τ_{amp} is the characteristic time of the amplifier system, t' is the integration step for numerical integration, t is the total time duration of the pulse.

Pulse Fitting Model Implemented in C++

Pulse fitting was developed as a Function Block in the ITER Real-Time Framework with the use of the C++ programming language. The algorithm is called

Fig. 5 Function block schematic [20]

convolution fitting and it iteratively approximates the raw data with the curve described by Eq. (1). It uses a non-linear least square method to determine the best set of parameters. The program loads the raw data and prepares it for analysis by eliminating the background noise. Then the lower and upper bounds for all four fitting parameters are set after which the convolution fitting begins until the best parameters are found. The main steps of the algorithm can be seen in Fig. 6.

An example of the resulting approximation curve is shown in Fig. 7. The red line represents the approximation curve. The raw data has two peaks and the second one is related to stray light-like effects which should not be considered when approximating the data.

Pulse Fitting Model Implemented in Simulink

ITER Real-Time Framework offers support for Simulink for the development of data processing algorithms. The authors have been asked to test this support. Therefore, the main focus was to evaluate the functional aspects of this feature. To verify whether Simulink can be used as a platform for the development of real-time data processing algorithms, the pulse fitting algorithm was also implemented as a Simulink model.

While developing the Simulink model, developers must be aware of the limitations if they wish to use this model to create a Real-Time Framework Function Block. For a Simulink model to be compatible with RTF it must be possible to generate C/C++ code from the model. The code can be generated with the use of Simulink Coder. Furthermore, the Simulink model should be configured according to the guidelines provided by the ITER RTF User Manual [20]. The developers must know the entire system and the algorithm they would like to create. The RTF Function Blocks have some corresponding design paradigms in Simulink which must be used to obtain the desired functionality. For instance, input and output ports of the Function Block must be defined as ports in Simulink, whereas parameters are represented by constant blocks with variable names assigned to them. After successful code generation from the Simulink model, the resulting archive containing the code must be converted into a dynamic library usable by the Real-Time Framework using a dedicated script. Finally, the Simulink model can be used as a part of the RTF application by defining a dedicated

Function Block that will utilize the generated dynamic library.

During the development process of the convolution fitting algorithm in Simulink, several obstacles were encountered. The development process was accompanied by constant testing and improving the RTF-to-Simulink interface which did not foresee all the possible algorithms one might want to develop. Moreover, the code generation limitations of Simulink forced the necessity to implement some parts of the algorithm differently. Due to the fact, that MATLAB's function for least square fitting does not support code generation, an alternative solution had to be found. A brute-force implementation of the least square fitting algorithm was chosen. This approach relates to iterating through all possible combinations of values for the parameters, constrained by upper and lower bounds as well as the allowable iteration steps for each of them. This is a very simple and time-consuming solution. However, this algorithm is justified for testing the RTF excluding its performance.

Example Application and Initial Results

Structure of Test Application

The test application consisted of a high-speed digitizer and a Real-Time Framework application (see Fig. 8). The RTF

Fig. 7 Plot of the resulting curve fitted to raw signal

Fig. 6 Structure of the convolution fitting algorithm

application included two main parts. First, there was a function block responsible for data acquisition with the CAEN DT5742 digitizer. This device support block was implemented to support real hardware. Additionally, an alternative implementation for data emulation was developed. The block was configured to imitate the data coming from a system in real-time by repeatedly sending five channels of real data. The resulting digitized signal was connected to the data processing algorithms that were implemented in C++ and Simulink and executed in real-time with the Real-Time Framework. Finally, the intermediate and final results were sent over SDN and DAN. The archived data was then available for further analysis and visualization.

Performance Evaluation and Discussion

The authors were successful in using the ITER Real-Time Framework for the development of a complete data acquisition and processing system. Two Function Blocks for the CAEN DT5742 digitizer were developed to support work in a real system as well as testing with emulated data. The C++ and Simulink implementation of the exemplary data processing algorithm both accurately approximated the raw data emulated by the CAEN Function Block which showed the functional correctness of the data processing algorithms. Consequently, the entire data processing path was investigated to obtain an overview of the system's most time-consuming part: the curve fitting algorithm,

which in the authors' tests proved to be tens or even hundreds times more time consuming than the rest of the data acquisition and processing path.

The tests conducted at the Lodz University of Technology, Department of Microelectronics and Computer Science used the data acquisition block for emulating real data. The performance tests were conducted for five sets of parameters with increasing ranges for the signal amplitude (P_L) and the time of pulse maximum (t_L). The bounds for the two remaining parameters: time duration of the pulse (τ_L) and the characteristic time of the amplifier (τ_{amp}) remained unchanged throughout the performance tests, as those parameters are configured only once per system and do not change during the operation of a given Thomson Scattering diagnostic system. Choosing varying ranges for the parameters is crucial to the performance of the algorithms as they directly correspond to the number of iterations that need to be performed in a brute-force search. The different performance tests roughly corresponded to 250, 500, 750, 1000, and 1250 thousands of iterations that had to be performed by the algorithm.

The first algorithm is the Simulink model of the brute-force algorithm, which was implemented as a Function Block in the Real-Time Framework and the computations were done in real-time on a CPU. The second algorithm was implemented using C++ and its performance was tested purely on a processor. The Simulink implementation of the curve fitting algorithm was noticeably slower than the other methods. The algorithm developed in Simulink is functionally correct and provides accurate results, however, the performance tests unveiled that the Simulink implementation is significantly slower than any of the other methods that were investigated.

The chart shows in Fig. 9 the performance comparison between Simulink—red line and CPU—blue line. Both implementations have linearly increasing execution times as the number of brute-force iterations increases. However, the Simulink implementation has a steeper slope, which

Fig. 8 Structure diagram for the test application

Fig. 9 Performance comparison Simulink vs CPU

suggests that the code generated from a Simulink model creates some overhead which is more visible as the number of iterations increases. A thorough investigation of this behaviour will be part of the future development of the system.

GPU Evaluation

Additionally, during the tests, a C++ /CUDA implementation of the convolution fitting algorithm running on a GPU was investigated. The first C++ and Simulink implementations running on CPU were developed as Function Blocks in the ITER Real-Time Framework. However, the algorithms tested on a GPU were disconnected from the ITER Real-Time Framework and run as standalone programs. The authors have planned to develop an interface Function Block to integrate the RTF with GPU if the preliminary tests showed that the GPU can increase the computational power. The idea of those experiments was to measure whether the GPU is capable of providing a significant computational improvement over the CPU [21]. The test platform for CUDA-accelerated algorithms was NVIDIA Jetson TX2 board equipped with NVIDIA Pascal Architecture GPU with 256 CUDA Cores @ 1.30 GHz and 8 GB L128 bit DD4 Memory. CPU algorithms were run on Intel(R) Core(TM) i7-4790 K @ 4.00 GHz processor. Since a brute-force algorithm searches through all possible combinations of the curve fitting parameters, one must impose constraints on the search range and the step size. Reducing the step or extending the bounds for the parameters has a significant impact on the procedure runtime.

Table 1 summarizes the performance measurement results averaged over 10 independent runs.

The first version of the GPU algorithm parallelized the most inner loop together with the integral value approximation, which resulted in a nearly 1.7-fold speed-up compared to the CPU alternative. However, in this implementation, the kernel was launched an excessive number of times. Further performance gains arrived from reducing launches to the single kernel call. The optimized kernel also has coalesced memory access and more efficiently leverages shared memory, as each block of threads computes the best parameters in a given subset [22]. Moreover, double-precision floating-point numbers were replaced

with single-precision floating-point numbers to reduce Register Spilling. The aforementioned improvements required the adjustment of the sequential CPU programming paradigm so that additional optimization techniques, unique for highly parallel GPU programming model, were included.

Data migration between host—CPU and device—GPU was benchmarked to investigate whether it is not a limiting factor of performance. The measurements concerned five sets of data, 1024 double-precision floating-point numbers each and five transfer methods [23]. The tested procedure included time necessary for memory allocation, multiplication of each number by a constant, and returning data to the host. Results averaged over 30 independent runs are visualized in Fig. 10.

It is visible that certain methods are faster than others, nonetheless one should select the method having regard to their characteristics to reduce the overhead related to migrating data between heterogeneous host and device memory spaces.

Overall, the involvement of a GPU allowed obtaining nearly 15-fold speed-up with minimal data migration penalty. It is noteworthy that the test platform is an embedded computing system, therefore the performance would be enhanced on a dedicated GPU or computation-oriented GPUs from NVIDIA Tesla or Quadro series. However, the brute-force methods cannot compete in terms of efficiency with the optimized curve-fitting algorithms due to the superior algorithm time complexity type. For further performance improvements, one might incorporate a GPU together with an efficient curve-fitting algorithm for faster computations [24].

Conclusions

Real-time data acquisition and processing is an essential part of any diagnostic system. The choice of proper hardware and software is crucial to the overall success. The

Table 1 Performance measurement results

Algorithm	Time (s)
Simulink CPU	169.8
C++ CPU	77.1
C++ /CUDA GPU	45.9
C++ /CUDA GPU optimized	5.2

Fig. 10 Data transfer benchmark

paper focused on the evaluation of the ITER Real-Time Framework and assessment of the functional aspects, ease of use, and usefulness of the framework. The paper also governed the development of the first, to the authors' knowledge, data acquisition, and processing system based on real hardware using ITER Real-Time Framework for diagnostic system applications. The software support for high-speed CAEN DT5742 digitizer was developed and evaluated. Function Blocks supporting real device operation and signal emulation were implemented. Moreover, the curve fitting algorithm was successfully implemented using Simulink and C++ and integrated with the Real-Time Framework. Evaluation of Simulink support offered by the ITER Real-Time Framework proved that the Simulink support offered by the framework it works correctly. This means that Simulink can be used as an alternative development platform, however, resulting performance and existing limitations should be taken into consideration. The results obtained by both implementations of the data processing algorithms coincided which meant that the functional aspect is fulfilled. However, as the preliminary test showed, the unconstrained iterative algorithm is not suited for real-time applications. There are plans to conduct a more insightful and thorough performance assessment of the system as well as the RTF. Different implementations of the convolution fitting algorithm will be investigated. Furthermore, the authors plan to continue the work on the Thomson Scattering diagnostics system and develop subsequent data processing algorithms.

Plans for the Future

The evaluation of the functional aspects of the ITER Real-Time Framework proved that it is a useful tool for the development of real-time applications. The authors plan to investigate the framework's performance. Moreover, as the preliminary performance tests showed, the iterative, brute-force curve fitting algorithm is not suited for real-time applications. Thus, in the future, different implementation of convolution fitting will be investigated. The main target of future research will be to thoroughly measure and improve the overall latency and performance of the system. For this purpose, various libraries providing an optimized version of non-linear least square fitting will be investigated. An example of such a library could be ALGLIB [25], which is compatible with C++ , and therefore, could be used with the Real-Time Framework. Additionally, the results obtained during the performance tests showed that utilizing GPU for computations can accelerate the computation. Thus, future research will include further investigation of the GPUs as well as the development of an RTF

Function Block which would allow integration of Real-Time Framework with GPU.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. M. Bassan, P. Andrew, G. Kurskiev, E. Mukhin, T. Hatae, G. Vayakis, E. Yatsuka, M. Walsh, Thomson scattering diagnostic systems in ITER. *J. Instrum.* **11**(01), C01052–C01052 (2016)
2. E. Yatsuka et al., Development of laser beam injection system for the edge Thomson scattering (ETS) in ITER. *J. Instrum.* **11**(01), C01006–C01006 (2016). <https://doi.org/10.1088/1748-0221/11/01/c01006>
3. E. Yatsuka et al., Technical innovations for ITER edge Thomson scattering measurement system. *Fusion Eng. Des.* **136**, 1068–1072 (2018). <https://doi.org/10.1016/j.fusengdes.2018.04.071>
4. S. Ivanenko et al., Prototype of data acquisition systems for ITER divertor Thomson scattering diagnostic. *IEEE Trans. Nucl. Sci.* **62**(3), 1181–1186 (2015). <https://doi.org/10.1109/tns.2015.2428195>
5. E. Pasch, M. Beurskens, S. Bozhnikov, G. Fuchert, J. Knauer, R. Wolf, The Thomson scattering system at Wendelstein 7-X. *Rev. Sci. Instrum.* **87**(11), 11E729 (2016). <https://doi.org/10.1063/1.4962248>
6. S. Bozhnikov et al., The Thomson scattering diagnostic at Wendelstein 7-X and its performance in the first operation phase. *J. Instrum.* **12**(10), P10004–P10004 (2017). <https://doi.org/10.1088/1748-0221/12/10/p10004>
7. I. Abramovic et al., Collective Thomson scattering data analysis for Wendelstein 7-X. *J. Instrum.* **12**(08), C08015–C08015 (2017). <https://doi.org/10.1088/1748-0221/12/08/c08015>
8. J. Lee, H. Kim, I. Yamada, H. Funaba, Y. Kim, D. Kim, Research of fast DAQ system in KSTAR Thomson scattering diagnostic. *J. Instrum.* **12**(12), C12035–C12035 (2017). <https://doi.org/10.1088/1748-0221/12/12/c12035>
9. J. Lee, S. Oh, H. Wi, Development of KSTAR Thomson scattering system. *Rev. Sci. Instrum.* **81**(10), 10D528 (2010). <https://doi.org/10.1063/1.3494275>
10. J. Lee et al., Tangential Thomson scattering diagnostic for the KSTAR tokamak. *J. Instrum.* **7**(02), C02026–C02026 (2012). <https://doi.org/10.1088/1748-0221/7/02/c02026>
11. B. Kurzan, M. Jakobi, H. Murmann, Signal processing of Thomson scattering data in a noisy environment in ASDEX Upgrade. *Plasma Phys. Contr. Fusion* **46**(1), 229–317 (2004)
12. B. Kurzan, H. Murmann, Edge and core Thomson scattering systems and their calibration on the ASDEX Upgrade tokamak. *Rev. Sci. Instrum.* **82**(10), 103501 (2011). <https://doi.org/10.1063/1.3643771>

13. L. Frassinetti et al., Spatial resolution of the JET Thomson scattering system. *Rev. Sci. Instrum.* **83**(1), 013506 (2012). <https://doi.org/10.1063/1.3673467>
14. R. Pasqualotto et al., High resolution Thomson scattering for joint European torus (JET). *Rev. Sci. Instrum.* **75**(10), 3891–3893 (2004). <https://doi.org/10.1063/1.1787922>
15. M. Bowden, Y. Goto, H. Yanaga, P. Howarth, K. Uchino, K. Muraoka, A Thomson scattering diagnostic system for measurement of electron properties of processing plasmas. *Plasma Sources Sci. Technol.* **8**(2), 203–209 (1999). <https://doi.org/10.1088/0963-0252/8/2/002>
16. DT5742 - 16+1 Channel 12 bit 5 GS/s Switched capacitor digitizer n.d. Available: <https://www.caen.it/products/dt5742/>. Accessed 21 Jun 2020
17. G. Buttazzo, G. Lipari, L. Abeni, *Soft Real-Time Systems* (Springer-Verlag, New York Inc., Dordrecht, 2006)
18. C. Liu, J. Anderson, Task scheduling with self-suspensions in soft real-time multiprocessor systems. 2009 30th IEEE Real-Time Syst. Symp. (2009). <https://doi.org/10.1109/RTSS.2009.10>
19. G. Lipari, L. Palopoli, Real-Time scheduling: from hard to soft real-time systems (Computing Research Repository, 2015). <https://arxiv.org/pdf/1512.01978.pdf>. Accessed 2 Nov 2020
20. L. Woong-Ryol, Real-time framework (RTF) software user manual, ITER internal document, IDM UID: WBZDRJ, (2019)
21. S. Lee, J. Lee, T. Tak, T. Lee, J. Hong, Design of GPU-based parallel computation architecture of Thomson scattering diagnostic in KSTAR. *Fusion Eng. Des.* **158**, 111624 (2020). <https://doi.org/10.1016/j.fusengdes.2020.111624>
22. I. Hendarto, Y. Kurniawan, in Performance factors of a CUDA GPU parallel program: a case study on a PDF password cracking brute-force algorithm, 2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA), (2017). doi: <https://doi.org/10.1109/ic3ina.2017.8251736>.
23. R. Santos, D. Eler, R. Garcia, Performance evaluation of data migration methods between the host and the device in CUDA-based programming, in *Information Technology: New Generation, Advances in Intelligent System and Computing*. ed. by S. Latifi (Springer International Publishing, NewYork, 2016), pp. 689–700
24. A. Przybylski, B. Thiel, J. Keller-Findeisen, B. Stock, M. Bates, Gpufit: An open-source toolkit for GPU-accelerated curve fitting. *Sci. Rep.* (2017). <https://doi.org/10.1038/s41598-017-15313-9>
25. ALGLIB n.d. Available: <https://www.alglib.net/>. Accessed 21 Jun 2020

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.