UNIVERSIDAD POLITÉCNICA DE MADRID

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y
SISTEMAS DE TELECOMUNICACIÓN

**METHODOLOGY FOR THE INTEGRATION OF HIGH-
THROUGHPUT DATA AND IMAGE ACQUISITION SYSTEMS IN
EPICS**

DOCTORAL DISSERTATION

Sergio Esquembri Martínez

Computer Science Engineer

ADVISORS

Eduardo Barrera López de Turiso

Ph.D. by the Technical University of Madrid

Mariano Ruiz González

Ph.D. in Telecommunications Engineering by the Technical University of Madrid

2017

TRIBUNAL

Tribunal nombrado por el Magfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día de de 2017

Presidente: Dr. EDUARDO JUAREZ MARTÍNEZ

Vocales: Dr. ADRIANO LUCHETTA

Dr. DANIEL MOZOS MUÑOZ

Dr. JOSU JUGO GARCÍA

Secretario: Dr. JUAN MANUEL LÓPEZ NAVARRO

Suplentes: Dr. VINCENT MARTIN

Dr. CARLOS GONZÁLEZ CALVO

Realizado el acto de lectura y defensa de la Tesis Doctoral el de de 2017 en la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación de la Universidad Politécnica de Madrid.

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

*Dedicada a mis padres,
por su inagotable esfuerzo y apoyo*

Agradecimientos

Ha sido un largo camino el que me ha llevado hasta donde estoy ahora. Hace unos años no hubiera pensado jamás que llegaría presentar una tesis, y no hubiera sido así de no contar con el gran número de apoyos recibidos durante este tiempo, tanto personales como profesionales. Si no fuera por ellos esta meta permanecería inalcanzable. Por ello debo agradecer:

A mi director de tesis Eduardo Barrera, por su constante búsqueda de la calidad y la excelencia en cada uno de los trabajos, y la minuciosidad de sus constantemente necesitadas revisiones. Por ayudarme a no perder el norte y guiar mis primeros pasos en el mundo de la investigación. Gracias por tu paciencia y por enseñarme que siempre se puede hacer un poco mejor.

A mi director de tesis Mariano Ruiz, quién le debo haber comenzado este camino. Su gran trabajo y dedicación han sido mis referentes, e intentar seguir su ejemplo ha sido tan exigente como gratificante. Brindarme la posibilidad de trabajar con él ha sido uno de los mayores favores que podré recibir nunca. Gracias por enseñarme a valorar el esfuerzo propio pero siempre manteniendo la humildad.

A los profesores Antonio Carpeño y Julián Nieto, y al doctor Diego Sanz, porque nunca se empieza desde cero y sus trabajos se ven reflejados en esta tesis de una manera o de otra. Gracias por dejarme aprender de vosotros y apoyarme en vuestro trabajo para dar un paso más.

También quiero agradecer a los compañeros y excompañeros del grupo de Investigación en Instrumentación y Acústica Aplicada, porque el ambiente de trabajo tan agradable y lleno de recursos se debe a ellos. Especialmente a Enrique, Alberto, Miguel, Álvaro, Raúl, Carlos, Néstor, Paul y Guillermo a quienes considero más amigos que compañeros, sin ellos estos casi cuatro años sin duda hubieran pasado mucho más lento. Gracias por vuestro apoyo dentro y fuera del laboratorio.

A mis amigos, quienes me apoyaron durante este viaje, tanto comprendiendo las ausencias y como en ocasiones soportando la presencia. Especialmente a Chus, María, Kat y Saúl, quienes siempre tendrán un hueco en mis pensamientos aunque no les pueda dedicar el tiempo que merecen.

A mi familia, quienes me acompañaron todo el camino y sus ánimos me han ayudado a llegar hasta el final. Especialmente a mis padres, mi hermano y Tamara, quienes se llevaron la peor parte pero no por ello dejaron de regalarme el mejor apoyo que pude tener. Espero poder ser capaz de retornaros todo lo que me habéis ofrecido.

Por último, quisiera dejar constancia que la elaboración de este trabajo ha sido parcialmente financiada por el Ministerio de Economía y Competitividad bajo los proyectos del Plan Nacional de I+D+i ENE2012-38970-C04-04 y ENE2015-64914-C3-3-R, la ayuda para contratos predoctorales para la formación de doctores BES-2013-064875, y las ayudas a la movilidad predoctoral la realización de estancias breves en centros de I+D españoles y extranjeros del año 2014. También ha sido financiado por las ayudas del Consejo Social de la Universidad Politécnica de Madrid para el fomento de la formación y la internacionalización de doctorandos para el curso 2015-2016.

INDEX

1	INTRODUCTION.....	1
1.1	Scope and goals	2
1.2	Outline	3
2	BACKGROUND AND STATE OF THE ART.....	5
2.1	Introduction	6
2.2	Technologies and form factors used in data acquisition systems for big physics	7
2.2.1	Reference system model.....	7
2.2.2	Acquisition Hardware: FPGA	10
2.2.3	Processing hardware: GPU	11
2.2.4	System controller: CPU.....	14
2.2.5	Form Factor	14
2.2.5.1	PXI/PXIe.....	15
2.2.5.2	xTCA	17
2.2.6	Data and Image sources	19
2.3	EPICS.....	19
2.3.1	EPICS Device Drivers	20
2.3.1.1	asynDriver.....	21
2.3.1.2	areaDetector	21
2.3.1.3	Nominal Device Support	22
2.4	Data and Image acquisition system in big physics experiments.....	23
2.4.1	JET.....	26
2.4.1.1	JET CODAS	27
2.4.1.2	JETs RT System.....	28
2.4.2	W7-X.....	30
2.4.2.1	W 7-X CoDaC.....	31
2.4.3	ITER.....	33

2.4.3.1	ITER CODAC.....	34
2.4.4	KSTAR.....	35
2.4.4.1	KSTAR Control System.....	36
2.4.5	ESS.....	38
2.4.5.1	ESS ICS.....	40
2.5	Conclusions.....	43
3	METHODOLOGY FOR THE DEVELOPMENT OF DATA AND IMAGE ACQUISITION SYSTEMS	47
3.1	Introduction	48
3.2	Commercial Products	48
3.2.1	Hardware Components.....	48
3.2.1.1	FPGA-based devices.....	50
3.2.1.1.1	FlexRIO devices.....	50
3.2.1.1.2	CompactRIO devices	53
3.2.1.2	GPUs	56
3.2.1.3	Host Computer.....	57
3.2.2	Development Tools.....	57
3.2.2.1	Graphical Hardware Description Tool: LabVIEW FPGA	58
3.2.2.2	GPU Development Environment: CUDA	61
3.2.2.3	Human Machine Interface Development Tool: Control System Studio.....	62
3.3	Products developed for the methodology	63
3.3.1	IRIO Software tools.....	63
3.3.1.1	IRIOCore.....	66
3.3.1.2	IRIO EPICS Interfaces.....	72
3.3.1.2.1	IRIOAsyn.....	74
3.3.1.2.2	IRIOAD.....	75
3.3.1.2.3	IRIONDS	76

3.3.2	Supplementary Developments.....	77
3.3.2.1	NDS DAN plugin.....	77
3.3.2.2	FPGA PTP-Synchronized Clock.....	78
3.3.2.3	SPAD Disruption Predictor	81
3.4	Development and testing cycle	83
3.4.1	Choosing the target	83
3.4.2	FPGA bitfile development and testing.....	85
3.4.3	Integration in application.....	86
3.5	Conclusions	88
4	EVALUATION AND RESULTS.....	91
4.1	Evaluation.....	93
4.1.1	Hardware architecture.....	93
4.1.2	Software architecture	96
4.1.3	Functionality	97
4.1.3.1	Automatic resource recognition and management.....	97
4.1.3.1.1	Resource Recognition	98
4.1.3.1.2	FlexRIO data acquisition mandatory and auxiliary resources	98
4.1.3.1.3	FlexRIO Image acquisition profile.....	99
4.1.3.1.4	FlexRIO data acquisition profile with analog signals.....	101
4.1.3.1.5	FlexRIO digital I/O management.....	102
4.1.3.1.6	CompactRIO profiles.....	102
4.1.3.2	EPICS integration.....	103
4.1.3.2.1	IRIOAsyn reasons and interfaces for mandatory resources	104
4.1.3.2.2	IRIOAsyn reasons and interfaces for analog input and analog output resources	104

4.1.3.2.3	IRIOAsyn reasons and interfaces for image acquisition and CameraLink configuration	105
4.1.3.2.4	IRIOAsyn driver misconfiguration and error management	106
4.1.3.2.5	IRIONDS for FlexRIO data and image acquisition	106
4.1.3.2.6	IRIOAD for FlexRIO image acquisition with hardware timestamping.....	107
4.1.4	Performance	108
4.1.5	Integrity for IRIO Library.....	111
4.2	Application Results	113
4.2.1	ITER PXIe Image Acquisition and Processing Reference System	113
4.2.2	KSTAR Application.....	117
4.2.3	IRIO-3DLab	119
5	CONCLUSIONS, CONTRIBUTIONS, AND FUTURE WORK.....	125
5.1	Conclusions	126
5.2	Contributions	127
5.3	Technology transfer and mock-up implementations.....	129
5.4	Academic production and dissemination.....	129
5.4.1	Articles in indexed journals	129
5.4.2	International Conferences and Meetings	132
5.4.3	Short Courses for International Organizations.....	135
5.5	Future Work.....	135
6	REFERENCES.....	139

FIGURE INDEX

Fig. 1 Reference Model of the data acquisition and processing system	9
Fig. 2 Schematic of FPGA architecture.	10
Fig. 3 Tesla GPU Architecture.....	12
Fig. 4 Schema of data transfer using GPU in data acquisition and processing systems without RDMA (a) and with RDMA (b).	13
Fig. 5 Elements of a PXIe system	16
Fig. 6 The ATCA standard.....	17
Fig. 7 Picture of an MTCA chassis.....	18
Fig. 8 Schema of MTCA architecture.	18
Fig. 9 Diagram of hardware device handling within an EPICS Device Driver ..	20
Fig. 10 Comparison between tokamak and stellarator main magnet systems and plasma shape.	25
Fig. 11 The JET tokamak.	26
Fig. 12 JET's Real-time measurement and control scheme.	27
Fig. 13 Schema of typical MARTe application and its integration into JETs RTDN and persistence system.	29
Fig. 14 Graphic of the Wendelstein 7-X and the superconducting magnetic coil system.	30
Fig. 15 Overview of Wendelstein 7-X Control, Data Acquisition and Communication system	32
Fig. 16 Representation of ITER tokamak and size comparison with JET	33
Fig. 17 Schematic of ITER CODAC System	34
Fig. 18 Picture and model of KSTAR tokamak.....	36
Fig. 19 Structure of KSTAR Control System	37
Fig. 20 Comparison of beam brightness and duration among ESS and current neutron sources.....	38
Fig. 21 Representation of ESS experiment.....	39
Fig. 22 Overview of ESS ICS and its main components.....	41
Fig. 23 Schematic example of ESS Control box components.....	42
Fig. 24 Technologies considered for ITER Fast Plant System Controllers.....	44
Fig. 25 Schema of the hardware components used in the methodology and its connections.	49
Fig. 26 Schema of possible data acquisition systems using FlexRIO devices.	51

Fig. 27 Schema of different systems based on CompactRIO	54
Fig. 28 Schema of the development tools used in the methodology	58
Fig. 29 Example of LabVIEW Front Panel, Block Diagram, and the drag and drop palettes.....	59
Fig. 30 Development schema for FlexRIO and CompactRIO devices using LabVIEW FPGA.	61
Fig. 31 Example of BOY OPI developed with CSS.....	63
Fig. 32 Diagram of IRIO Software Tools for the integration of RIO Devices and GPUs in EPICS.	64
Fig. 33 Typical development cycle for systems including reconfigurable FPGA devices	67
Fig. 34 Overview of the relationship among the IRIO Library calls, the NI-RIO Linux Device Driver API, and the LabVIEW FPGA implementation	68
Fig. 35 Schema of the connection between the EPICS PVs defined in the templates and the resources implanted in the FPGA through the IRIO EPICS Interfaces.....	73
Fig. 36 Representation of the timestamp embedded in an image using the PTP- synchronized hardware clock implemented in the FPGA	80
Fig. 37 FPGA synchronized clock offset distribution with respect to NI 6683 clock after 2,000 measurements (one measure per second).....	81
Fig. 38 Development cycle using IRIO Software Tools.....	83
Fig. 39 Schema of an IRIO Library-based application for data acquisition.....	87
Fig. 40 CompactRIO connection schema for testing.....	94
Fig. 41 ITER PXIe Image acquisition and processing system architecture.....	113
Fig. 42 Logical representation of the ITER reference image acquisition and processing system.....	114
Fig. 43 ITER image acquisition system OPI panel.	115
Fig. 44 Schematic of the third-party VHDL-based IP Core for zero crossing detection and phase shift calculation used in KSTAR	118
Fig. 45 LabVIEW code for zero crossing detection and phase shift calculation used in KSTAR.....	118
Fig. 46 Result comparison of density control in KSTAR implemented using data from ITER's PXIe architecture (left) and KSTAR mTCA architecture (right) for the KSTAR 15869 shot.	119

Fig. 47 Facilities of IRIO-3DLab: (a) Laboratory Building, (b) Exposition Room, (c) Meeting Room, and (d) Technical Room	120
Fig. 48 IRIO-3DLab architecture	121
Fig. 49 CSS OPI panel available inside the virtual world for the control of the neutron diagnostics	123

TABLE INDEX

Table 1 PCI/PCIe standards maximum throughputs	16
Table 2 Comparison of resources available at the different FlexRIO devices	52
Table 3 Specification of different CompactRIO chassis	55
Table 4 CompactRIO Device Profiles.....	69
Table 5 FlexRIO Device Profiles	69
Table 6 Reproduction of part of a table from the IRIO Design Rules document defining the FPGA resource names, types, and functionalities.....	70
Table 7 Resume of the characteristics of the FPGA hardware devices supported by IRIO Software tools.....	84
Table 8 Description of connections among CompactRIO IO modules for testing	95
Table 9 Examples of FPGA resource utilization for different data processes...	110
Table 10 GPU Process Time for Calibration, Thresholding and Find-Max.....	111
Table 11 GPU CCL process time for the YACLAB [73] image databases.....	111
Table 12 FPGA Resource Utilization in ITER IAS Implementation	116
Table 13 Comparison of image processing times on CPU and GPU	116
Table 14 Comparison of CCL processing times on CPU and GPU using different image databases	117
Table 15 List of FlexRIO Devices supported by the latest version of IRIO Library (V1.2.0).....	153
Table 16 List of FlexRIO Adapter Modules supported by the latest version of IRIO Library (V1.2.0).....	154
Table 17 List of CompactRIO chassis and modules supported by the latest version of IRIO Library (V1.2.0)	155

Abstract

Data acquisition and processing systems are essential parts of today's world. The purpose of these systems is to take measurements from real world characteristics in such a way that they can be processed to obtain information. The scale of these systems varies enormously from the smallest examples found in portable and wearable technology (such as smartphones and tablets) to big systems based on industrial communication buses that acquire and process several gigabytes of information per second.

Special cases of these systems are those used in big physics experiments. The nature and purpose of these experiments change considerably from one to another, but all of them need to extract as much information as possible and with the utmost precision from the physic phenomena under study. This leads to a large amount of data to be processed and archived. Moreover, some of the acquired or processed data may be needed for the control of the experiments, so it is necessary to calculate them in real-time and with the lowest possible latency. Additionally, the reliability and availability of these systems must be guaranteed for the correct operation of the experiments, especially for the control and safety systems of the experiment.

Another common characteristic in big physics experiments is their organization as a Supervisory Control And Data Acquisition (SCADA) system. The size and complexity of these experiments make necessary the use of these systems that allow dividing the functionality among the different systems while they are communicated for the coordination and synchronization of the actions taken by each one of them.

This doctoral dissertation proposes a generic model as reference for the real-time data acquisition and processing solutions in big physics experiments, especially for those used in magnetic confinement fusion experiments. The proposed model tries to address the common requirements for this kind of systems. The model is designed using the technologies present in current experiments, as reconfigurable logic devices based on Field Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs), Central Processing Units (CPUs), and architectures based on the industrial version of Peripheral Component Interconnect Express (PCIe) known as PCI eXtensions for Instrumentation Express (PXIe). The SCADA system used for the integration of the data acquisition systems is the Experimental Physics and Industrial Control System (EPICS), chosen for being one most used distributed control system in big physics experiments.

However, there are some problems derived from the use of these technologies. The integration of the systems in EPICS requires the development of an intermediate application that interfaces between the hardware devices and EPICS. In the case of systems involving FPGA-based devices, the development of these applications becomes more challenging. This is because FPGAs devices can be configured in many different ways, and the development of one application for each possible configuration will increase the maintainability cost of the experiment. It is also necessary to consider that the application must integrate together devices very different from each other, as FPGAs, GPUs, and CPUs, and this must be done in such a way that they can work in collaboration for the data process. However, also important is that the systems developed are easily customizable, as they will be used for playing very different roles in the experiment, and it will be necessary to add custom functionalities, as specific data process or control actions taken by the acquisition device.

In response to the problems described, this thesis is focused on the development of a methodology for the implementation of high throughput data acquisition and processing systems and its integration in EPICS. The methodology takes the proposed model system as a reference, and it tries to address the issues that the architecture presents pursuing to ease the implementation of these systems.

The proposed methodology is based on the use of a set of software tools specifically designed for this purpose. The developed tools are publicly available and at the disposal of the scientific community under an open software license.

Given the above, the main topics of this thesis are the following:

- Study of data acquisition and processing systems requirements for big physics experiments as an example of an application where a high throughput is required.
- Analysis of the hardware and software architecture of a data acquisition and processing system based on the technologies mentioned previously.
- Definition and specification of the methodology and the related development cycle for the implementation of those systems and their integration in EPICS.
- Description of the products developed in support of the methodology and for easing the use of the different technologies involved in the system.
- Evaluation and validation of the proposed methodology, including actual use cases where the methodology is being applied.

Resumen

Los sistemas de adquisición y procesamiento de datos son fundamentales hoy en día. El fundamento de un sistema de adquisición es tomar muestras del mundo real de manera que puedan ser procesadas para obtener información. Estos sistemas están presentes en muchas escalas, desde los sistemas más pequeños, como los utilizados en la tecnología portable (smartphones, tablets,...), hasta grandes sistemas basados en buses industriales que adquieren y procesan gigabytes de información por segundo.

Un caso especial de estos sistemas son los utilizados en grandes experimentos físicos. La naturaleza o propósito de estos experimentos varía enormemente de unos a otros, pero todos tienen en común la necesidad de extraer la mayor cantidad de información posible del fenómeno bajo estudio, y con la mayor precisión que se pueda alcanzar. Esto da lugar a gran volumen de datos generados que deben ser procesados y almacenados. Además algunos de los datos adquiridos o resultado de procesar los mismos son requeridos para operar el experimento, por lo que deben ser calculados en tiempo real y con la menor latencia posible. Como requisito adicional, estos sistemas necesitan garantizar su disponibilidad y fiabilidad para poder operar correctamente la máquina. Estas características se hacen críticas para ciertos sistemas, como los encargados de los sistemas de seguridad y control del experimento.

Otra característica común en los experimentos físicos es que siguen un modelo de adquisición y control (SCADA) distribuido. La envergadura de estos experimentos hace necesario el uso de estos modelos que permiten separar la funcionalidad de los distintos sistemas mientras que la comunicación entre los mismos permite coordinar y sincronizar las acciones de cada uno de ellos.

En esta tesis doctoral se propone un modelo genérico como solución de referencia para los sistemas de adquisición y procesamiento de datos en grandes experimentos físicos, con enfoque especial en dispositivos de fusión nuclear por confinamiento magnético. El modelo diseñado intenta satisfacer los requisitos comunes de este tipo de sistemas utilizando tecnologías que se encuentran actualmente en uso en diferentes instalaciones, como dispositivos basados en hardware de lógica configurable del tipo FPGA, dispositivos de procesamiento de gráficos (GPUs), unidades de procesamiento central (CPUs), y arquitecturas basadas en una versión industrial del bus PCIe conocida como PXIe. Como sistema SCADA para la integración del sistema de adquisición se ha elegido EPICS, por ser uno de los sistemas de control distribuido más usados en grandes experimentos físicos.

No obstante, el uso de estas tecnologías no está exenta de problemas. Integrar los sistemas en EPICS requiere el desarrollo de una aplicación intermedia que actúe de interfaz entre los dispositivos usados y EPICS. En el caso del uso de FPGAs el desarrollo de esta aplicación se dificulta, debido a que en la FPGA se pueden implementar un gran número de soluciones distintas, y el desarrollo de una aplicación distinta para cada solución dificultaría mucho la mantenibilidad del experimento. También es necesario tener en cuenta que la aplicación debe integrar dispositivos muy dispares, como FPGAs, GPUs y CPUs, de manera que contribuyan de manera conjunta al procesamiento de los datos. Por otro lado, debido a los distintos usos que pueden tener estos sistemas, es necesario que los mismos se puedan personalizar de manera fácil permitiendo la inclusión de nuevas funcionalidades, como procesamiento de datos específico o mecanismos de control integrados en la plataforma de adquisición.

Por todo esto, esta tesis se centra en el desarrollo de una metodología para la implementación de sistemas de adquisición y procesamiento de datos de alto rendimiento y su integración en EPICS. La metodología toma como sistema objetivo el modelo propuesto anteriormente, y trata de ofrecer solución los problemas de la arquitectura y facilitar la tarea de implementar estos sistemas.

Esta metodología se basa en el uso de unas herramientas software que han sido específicamente diseñadas para la misma y que han sido puestas a disposición de la comunidad científica bajo licencia de software libre.

Teniendo en cuenta lo anterior, los puntos abordados en esta tesis doctoral son los siguientes:

- Estudio de los requisitos de los sistemas de adquisición y procesamiento de datos en el contexto de los grandes experimentos físicos como ejemplo de aplicaciones que requieren gran rendimiento.
- Análisis de la arquitectura hardware y software de un sistema de adquisición de datos basado en las tecnologías citadas anteriormente.
- Definición y especificación de la metodología y el ciclo de desarrollo establecido por la misma para la implementación de los sistemas y su integración en EPICS.
- Explicación de los productos desarrollados para soportar la metodología y facilitar el uso de las diferentes tecnologías que componen el sistema.
- La evaluación y validación de la metodología propuesta, incluyendo casos de uso en los que actualmente se está utilizando la metodología desarrollada.

Chapter 1

1 INTRODUCTION

1.1 Scope and goals

The scope of this thesis is focused on the search for a solution for the data and image acquisition and processing with hardware architectures based on the industrial version of the widely used Peripheral Component Interconnect Express (PCIe) communication bus, called PCI eXtensions for Instrumentation Express (PXIe). This architecture has been selected for implementing the diagnostic systems and the instrumentation and control systems of the ITER thermonuclear fusion experiment. ITER also opted for using the Experimental Physics and Industrial Control System (EPICS) as Supervisory Control And Data Acquisition (SCADA) tool. In this thesis is presented a solution integrated into this SCADA system. The proposed solution allows the implementation of data acquisition and real-time processing of the data acquired using reconfigurable hardware devices based on Field Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs), and Central Processing Units (CPUs).

Considering this, the goals of this thesis are defined as follows:

1. The development of a methodology for the integration of data acquisition and processing systems in EPICS.
2. To cover the widest range of possible applications, allowing the user to adapt the system to their needs, but at the same time trying to reduce as much as possible the effort needed to implement the systems and integrate them in EPICS.
3. To define the system architecture required for the use of FPGAs, GPUs, and CPUs in a data acquisition and processing system.
4. To develop solutions that can be used by the scientific community, and that the community itself can improve and extend using a collaborative model.
5. To guarantee the quality level of the solution provided to be successfully used in highly demanding applications.
6. To ease the maintainability of the developed solutions, their upgrade, and their reuse by several users.
7. To achieve state-of-art throughputs of data acquisition and processing.
8. To provide the synchronization required for the rest of the systems under the distributed control system.

1.2 Outline

The thesis is organized into 5 chapters, including this first introductory chapter, plus a chapter dedicated to the reference documents, and the appendices containing a list of the acronym used and tables listing the hardware devices supported by the methodology and their characteristics. Apart from the content presented in this chapter, the rest of the thesis is organized as follows:

- **Chapter 2, Background and State of the Art**, contains the contextualization of the thesis in the field of big physics experiments. The chapter includes a description of typical requirements for data acquisition and processing systems in this field, the reference system model that guided the architecture proposed in this thesis, and an overview of the technologies involved in the reference system, including EPICS. The chapter ends with a review of some big physics experiments relevant for the background of the thesis.
- **Chapter 3, Methodology for the Development of Data and Image Acquisition Systems**, defines the proposed methodology, including the details about commercial and developed products in which the methodology is based, and the defined steps that must be followed to develop the solutions.
- **Chapter 4, Evaluation and Results**, reports the procedures followed for the validation of the methodology and the outcome of this process, also including some examples of solutions where the proposed methodology was applied.
- **Chapter 5, Conclusions, Contributions, and Future Work**, presents a summary of the conclusions and contributions derived from the work described in this thesis, including the academic production and dissemination and the enumeration of some potential research lines that can continue with the presented work

Chapter 2

2 BACKGROUND AND STATE OF THE ART

2.1 Introduction

This chapter introduces the use of data acquisition and processing systems in the context of big physics experiments. These experiments present a very demanding scenario for data acquisition and processing systems. The complexity of these experiments requires precise instrumentation and measurements of diverse characteristics to be operated properly and extract all the information possible about the physic phenomena. These reasons make the big physics experiments very interesting for the research in data acquisition and processing systems.

This chapter presents the technologies used in current experiments. These technologies are explained in the context of a reference model for high throughput data acquisition and processing system. The model presents an architecture trying to comply with the requirements needed for these systems in big physics. Moreover, specific technologies that can be used to implement the different parts of this ideal system are presented. The model presented has been elaborated using the architectures and solutions published by the researchers and scientist of the big physics community.

Special mention is done about EPICS. This set of software tools for developing distributed control systems is one of the most commonly used in big physics experiments. EPICS has been chosen for some of the most important experiments currently ongoing or under development and thus becoming a reference for distributed control systems. For this reason, the methodology presented here is focused on the integration of high throughput data acquisition systems with EPICS.

The technologies presented will be placed in the context of some big physics experiments, including information about the control and data acquisition system selected for the experiment. The experiments chosed cover different application and purposes, as well as different technologies and architectures used for the implementation of the data acquisition and processing system.

Finally, the conclusions of this chapter will present a rationale of the technology chosen for the implementation of data acquisition and processing systems in the proposed methodology, including its limitations and drawbacks.

2.2 Technologies and form factors used in data acquisition systems for big physics

2.2.1 Reference system model

In this section, we present the definition of a reference data acquisition and processing system model and its characteristics to understand the functionality, advantages, and drawbacks of the hardware and software elements used in advanced data acquisition systems for big physics experiments. This model is guided by the general requirements of data acquisition and processing systems in big physics experiments, which will be sketched before the definition of the model itself. However, the data acquisition systems used in big physics are very heterogeneous, and so it is their purpose for the experiment. Not all the requirements will apply to all the data acquisition systems of an experiment, but it will give a general idea of what can be expected from these systems.

One of the main requirements for the data acquisition systems in big physics is the high throughput and performance. Due to the complexity of the phenomena under study in these experiments, the acquisition systems usually are required to measure a characteristic with all the possible resolution and at the maximum sampling rate. These conditions result in the generation of massive amounts of data that must be handled by the system.

The process needed for the data depends on the type of data acquired and its purpose. A common procedure with all the data acquired is the archiving with fast data availability, which is usually done by transferring the data to another system that handles the permanent storage. In other cases, the data will be needed for control purposes. For this, the capability to close the control loop in the acquisition device can be interesting, especially if it can be done with low and deterministic latency. However, other control actions may need to be done considering values from multiple systems. Also is possible that a control action may be done in coordination with other systems. This situation leads to the necessity of precise timestamping of the data acquired. Also, the data may need some process in order extract useful information. If the information is needed for the operation of the experiment, the process must be done in real-time with the least possible latency. The magnitude of the process differ from one case to

another, but it may involve complex algorithms or massive numeric operations in floating point.

Nonetheless, the specifications for a certain system may change during the experiment lifetime. The nature of these experiments will require changes in the operation of the machine or its diagnostics. Due to this, the systems used in this kind of experiments need to be flexible to support different configurations, easing as much as possible the development and deployment of new solutions.

Other requirements of the system correspond to the availability and reliability of the system. Usually, the hardware used is expected to last for 20-25 years at least, and the availability and reliability of systems dedicated to the monitoring of control relevant characteristics are critical. However, due to the nature of the experiment part of the hardware may need to be placed in harsh conditions, e.g. strong electromagnetic fields, radiation, or vibration. This can be the case of the acquisition hardware. For this reasons, the availability and reliability of the hardware used should also be evaluated considering the environment of the hardware in the final deployment.

Considering these requirements, a reference model for a data acquisition and processing system is proposed. The reference model is depicted in Fig. 1. In this model the data acquisition and processing system is composed of three differentiated elements, the system controller, the processing hardware, and the acquisition hardware, all of them interconnected by a high bandwidth data bus. The acquisition hardware is connected to the appropriate sensors, but could also be connected to some actuators for closing the control loop. The system controller is connected to other systems and the central unit through the distributed control system network. The other connection of the system is to a timing network that is managed by a master clock that provides an absolute time reference for the whole experiment. The functionality of each part of the acquisition system will be explained below.

The system controller is the responsible for configuring the acquisition and processing hardware according to the parameters received from the distributed control network. It will handle the acquisition and processing and serve as the

Fig. 1 Reference Model of the data acquisition and processing system

interface between the system and the central unit or other systems, like the archiving system.

The processing hardware will perform the necessary computations over the acquired data. The data process can take place in a single piece of hardware, but it is also commonly distributed among different processing elements. Depending on the characteristics or type of data acquired the data process may not be necessary or vary from a simple conversion from analog or digital values to engineering units, data compression, feature extraction, or in some cases complex matrix operations like the ones needed for image processing.

The acquisition hardware is the part of the system collecting the data from the sensors according to the configuration provided by the system controller. Typical parameters configured in the data acquisition are the sampling rate, decimation, acquisition mode (free-run, periodic, triggered by an external signal or due to some criteria over the sampled data, pretrigger, posttrigger). In certain configurations, it may also be connected to some actuators for closing the loop control in the data acquisition system and thus reducing the latency of the control actions.

As can be seen in Fig. 1, different technologies are used for the elements of the system. These technologies will be explained in the following sections, as well as the rationale for its use in the system.

2.2.2 Acquisition Hardware: FPGA

The Field Programmable Gate Arrays (FPGAs) are reconfigurable hardware devices, which means that the behavior of the device is defined after its manufacturing. In the case of the FPGAs, it is possible to change this configuration without limit. The internal architecture of an FPGA can be seen in Fig. 2. It consists of a matrix of configurable logic blocks, a hierarchy of reconfigurable interconnections, and input/output (I/O) cells. The configuration of an FPGA is done by downloading a bitfile that configures the function performed by each logic block and how they are connected using the reconfigurable interconnections. The FPGA bitfile is obtained by synthesizing the functional design which is typically specified using hardware description languages (HDLs) like VHDL or Verilog. However, the use of hardware description languages is considered to be very complex, and the difficulty in the development grows exponentially with the complexity of the system to be implemented. However, there are other alternatives as utilities that transcript restricted sentences of programming languages as C or Python to HDL, or as will

Fig. 2 Schematic of FPGA architecture.

Reproduced from <http://www.vision.caltech.edu/CNS248/Fpga/fpga1a.gif>

be explained later, graphic programming tools like LabVIEW FPGA. These tools proved to ease the implementation of systems involving FPGAs, reducing the development effort.

Nowadays, the FPGA devices are widely used for a big range of application for its deterministic behavior and high performance while providing great flexibility. The reconfiguration capabilities make them ideal for hardware prototyping or emulation, but also as hardware accelerators. Some vendors also provide a mechanism for protecting the FPGA to be reconfigured or to clone the actual configuration, making of FPGAs a valid solution also for a final product. Regarding data acquisition devices based on FPGA, they usually include additional elements as Random-Access Memories (RAMs), Digital Signal Processors (DSPs), Digital-to-Analog Converters (DACs), or Analog-to-Digital Converters (ADCs). The FPGA designer can make use of these elements to improve the performance of the design or reduce the logic blocks used.

In the reference model, an FPGA device is proposed to implement the acquisition hardware. The high performance and high throughput allow high-speed data acquisition (up to MS/s), while the deterministic processing capabilities of the FPGA allow to perform a first process of the samples or to close the control loop. Also, the reconfiguration capability eases the procedure for the implementation and deployment of changes to the data acquisition system.

2.2.3 Processing hardware: GPU

The model proposed uses Graphic Processor Units (GPUs) for processing the data acquired. The GPUs are devices originally designed to accelerate image computing and graphics handling for displays. GPUs become practical for general purpose computing in 2001, when GPU manufacturers added support for programmable shaders and floating point operations. The scientific community exploited these new features to perform matrix and vector operations in the GPU, which were then natively supported. However, the use of GPUs for purposes required the reformulation of the computational problems in terms of graphic operations. In 2007 NVIDIA launched a software and hardware solution for general purpose computing on GPUs called Compute Unified Device Architecture (CUDA) [1]. Since then, other GPU manufacturers also offer GPUs for general purpose computing, and also some hardware vendor independent

frameworks, like OpenCL [2], and even alternatives for parallel programming in heterogeneous systems integrating CPUs, GPUs, and FPGAs, like OpenACC [3] or OpenMP [4].

The architecture of the first NVIDIA Tesla is depicted in Fig. 3. The main elements of architecture are the Streaming Multiprocessors (SM). The SMs are composed by several Streaming Processors (SP), also referred as CUDA Cores. The SP are the basic operating unit of the GPU, where the execution of the GPU code will take place. The SM controls and coordinates the execution of its SPs and manages the data transfers between the global DRAM memory located outside the SMs and the local SDRAM shared memory and cache memory located inside the SMs. The SMs are seen by the programmer as a set of processors where the same instructions can be executed over multiple data, usually called Single Instruction, Multiple Data (SIMD). The internal architecture of the GPUs changes between different manufacturers and generations, but they follow a similar organization. The great number of cores allow for the execution of massive parallel operations, making the GPUs ideal for image processing.

Fig. 3 Tesla GPU Architecture.

Reproduced from [68].

In a computer-based architecture, the GPUs cannot be used as independent devices. They need a controller, normally a CPU, which programs the GPU functions to be executed at a certain moment. The controller may also need to transfer the data from the CPU memory to the GPU memory and, once the process is done, it will retrieve the process results back to the CPU memory. In the case of using GPUs in data acquisition and processing system like the one presented before, this implies that the CPU must get the data from the data acquisition device to the CPU memory, then transfer this data to the GPU, and finally get the results (Fig. 4 (a)). However, certain GPU devices also allow data transfers using Direct Memory Access (DMA) from other devices than the CPU as long as they are connected to the same PCI/PCIe bus. In this case, the data can go directly from the data acquisition device to the GPU (Fig. 4 (b)). This feature called Remote DMA (RDMA) can effectively reduce the latency of the process, avoiding unnecessary data copies and improving bandwidth use.

Fig. 4 Schema of data transfer using GPU in data acquisition and processing systems without RDMA (a) and with RDMA (b).

In the context of big physics experiments, they can be used as hardware accelerators for multiple algorithms applied to the experiment vision system, like compression, event recognition, or feature extraction [5]. Also, it is forecast the need of using GPUs for other than image-based diagnostics, like real-time plasma equilibrium reconstruction [6].

2.2.4 System controller: CPU

The Central Processing Unit (CPU) is used in the model to control the system. As system controller, the CPU must configure the acquisition hardware and processing hardware according to the instructions received by the central unit. It is also the responsible for managing the communications with the external systems. In a distributed system like the one proposed in the model, this is done by implementing software programs that interface the hardware manufacturer drivers with the distributed control system.

The Operating System (OS) running on the CPU is the software interfacing among the user programs and the hardware. Nowadays, CPUs count with several cores, each one capable of running several parallel threads, but some operations may introduce undesired latency in the system. The scheduling of the threads among the different cores and the priority of the threads is also managed by the OS, and it is a matter of big relevance in systems with strong timing constraints. This scheduling can be tuned and configured by the user in Real-Time OS (RTOS). VxWorks [7] is an example of proprietary RTOS for embedded systems widely used in the scientific community. In the case of ITER, the OS chosen is CODAC Core System (CSS), which is based on Red Hat Enterprise Linux (RHEL). RHEL counts with extensions for real-time called Message, Real-time and Grid (MRG). These extensions are also considered for its use in the experiment.

2.2.5 Form Factor

One important characteristic to consider when designing a data acquisition system for a real experiment is the form factor. The form factor determines the size, shape, and interface of the hardware devices used. The form factor chosen also has influence in the bus or buses that interconnect the different elements of the system.

In this section two form factors are presented: PXI/PXIe and the ones derived from the Advanced Telecommunications Computing Architecture (ATCA), called xTCA. These form factors are analyzed here for been the most used for the development of current experiments, although older solutions like the ones based on VERSA Module Eurocard (VME) and VME eXtension for Instrumentation (VXI) [8] are still in use in some experiments.

2.2.5.1 PXI/PXIe

The PCI eXtension for Instrumentation (PXI) is a platform for instrumentation proposed by National Instruments (NI) in 1997 and promoted by the 69 members of the PXI Systems Alliance (PXISA). PXI is based on the PCI architecture, adding a ruggedized mechanical form based on CompactPCI. PXI adds timing and synchronization capabilities using specially designed routes for triggers and clock in the chassis backplane.

A PXI system is composed of a chassis and the different cards connected to it. In the case of the proposed system, the different cards connected to the chassis could be the FPGA devices used for data acquisition. The PXI timing capabilities can be enhanced by connecting a timing card, a card that is synchronized with an external source and can work as a time reference of the PXI system. The system can be controlled using a PXI controller connected directly to the chassis or using a Multiplatform eXtension for Instrumentation (MXI) link that connects the chassis to a PCI interface of a host computer and working as a PCI/PXI bridge. In 2005 it was developed the PXI Express (PXIe) and MXI Express (MXIe) standards, the adaptations of PXI and MXI standards to the PCI Express. Examples of different elements of a PXIe system can be observed in Fig. 5. The maximum throughput achievable with these standards depends on the version of PCI/PCIe on which they are based. These limits can be observed in Table 1. In the case of PCIe, the important factor is the aggregate bandwidth that can manage the complete PXIe system.

Fig. 5 Elements of a PXIe system

Table 1 PCI/PCIe standards maximum throughputs

Standard (Year)	Max. Throughput
PCI 32-bit, 33 MHz (1992)	133.3 MB/s
PCI 32-bit, 66 MHz (1995)	266.7 MB/s
PCI 64-bit, 33 MHz (1993)	
PCI 64-bit, 66 MHz (1995)	533.3 MB/s
PCIe 1.0 (2004)	250 MB/s (x1) – 4 GB/s (x16)
PCIe 2.0 (2007)	500 MB/s (x1) – 8 GB/s (x16)
PCIe 3.0 (2011)	984.6 MB/s (x1) – 15.8 GB/s (x16)

2.2.5.2 xTCA

The term xTCA is used to denominate all the variants derived from the Advanced Telecommunication Computing Architecture (ATCA) technology. This technology was originally designed for high demand applications requiring high performance, as telecommunication data centers. The ATCA standard was developed by the PCI Industrial Computer Manufacturer Group (PICMG) and referenced as PICMG 3.x. As depicted in Fig. 6, the ATCA standard defines a big card size that allows the implementation of resourceful, high-performance acquisition devices. The main inconvenient for the use of this technology is the high investment needed to set up a system and develop applications.

Fig. 6 The ATCA standard

In (a), a picture of an ATCA crate. In (b), schematics of ATCA front and extended boards and its connections with the crate backplane.

One of the variants of the ATCA technology is MicroTCA (MTCA) (Fig. 7). This standard was developed to apply the ATCA technology for physics experiments, using more compact solutions. The MTCA architecture is represented in Fig. 8. In this architecture, the main elements are the Advanced Mezzanine Cards (AMC), which are communicated among them using the PCIe busses or other networks implemented in the chassis backplane.

Fig. 7 Picture of an MTCA chassis.

Fig. 8 Schema of MTCA architecture.

2.2.6 Data and Image sources

The analysis of the possible data sources is important to design the system. However, this analysis is very specific to the purpose of the system, and out of the scope of the thesis. For the sake of the simplicity, the sensors will be classified according to the type of data acquired from them. Here the categories considered are digital, analog, or image sensors.

When acquiring digital data, the value read from each channel at a certain moment can be represented by a zero or a one. Therefore, this data will be read in the acquisition device as a bit.

When acquiring analog data, the representation is important. The value read in the acquisition device for each channel at a certain moment is represented by a collection of bits. The number of bits is determined by the resolution of the ADC. The typical resolutions can vary from 8-bit to 16-bit for high sampling rate devices. The interpretation of the analog data acquired needs of a conversion function, which will depend on the characteristics of the ADC used.

The image sensors can have a digital or analog interface, but what is relevant is the interpretation. In image acquisition what is acquired is a collection of values representing the pixels of the image. The resolution of the pixel can vary between one or several bytes. The value of a pixel may contain information about one color (monochrome) or several colors depending on the sensor.

2.3 EPICS

The Experimental Physics and Industrial Control System (EPICS) [9], [10] is a set of software tools and applications which provide a software infrastructure for building distributed control systems. EPICS is based on the Client/Server and Publisher/Subscriber models. EPICS servers, called Input/Output Controllers (IOCs), are managed and monitored through the Process Variables (PVs). The PVs are controlled and accessed by the clients in the network using a specific network protocol called Channel Access (CA) based on TCP/IP or UDP/IP. Each user can create and program the behavior of the required PVs, but this requires deep knowledge of the EPICS core functionalities. However, there is an extensive set of already defined PVs developed by the EPICS community that can be configured to work as needed. The connection between the PVs and the device

Fig. 9 Diagram of hardware device handling within an EPICS Device Driver

to be controlled is done by implementing an EPICS Device Driver. In the EPICS Device Driver the developer can define how a PV is related with the hardware associating, for example, PV read or write operations with calls to the device manufacturer driver as depicted in Fig. 9. Another possibility commonly used in EPICS device drivers is that the EPICS Device Driver is publishing the data acquired from the device in a PV or set of PVs.

2.3.1 EPICS Device Drivers

There are several approaches for implementing an EPICS Device Driver. The first approach is directly implementing the functionality of the records. In this approach, the user must implement almost individually the functionality of each PV used in the driver. This approach generated a lot of duplicated effort, as several drivers with similar characteristic implemented the PV functionality differently. This reason motivated the development of EPICS Driver frameworks. The frameworks try to solve the most common problems when developing EPICS Device Drivers. These ease the maintainability of the drivers while spare the developer from redoing the work done already for other drivers. Examples of these are presented in the following paragraphs.

2.3.1.1 asynDriver

asynDriver was developed to solve the interface between the most commonly used PVs and datatypes. To implement an EPICS Device Driver based on asynDriver, the user registers the data interfaces supported by the driver, the methods to attend the request depending on the data type, and the reasons, a name to identify the purpose of the PV (e.g. SamplingRate, or DAQStart). Read and write actions on PVs will trigger calls to the defined methods, where the user can define the behavior depending on the reason of the PV that triggered the call.

asynDriver supposed the first approach to standardize EPICS Device Driver implementation, but there was still some functionality replied across different drivers. Also, different devices were handled differently, even for the same functionality, using different sets of PVs with different values for representing the same behavior. This situation complicated the development of the control functions in a heterogeneous system. Currently, there are two driver frameworks trying to solve this problem. These two frameworks are presented below, areaDetector and Nominal Device Support (NDS), both of them are built over asynDriver.

2.3.1.2 areaDetector

areaDetector [11] is an EPICS driver framework for controlling detectors with data over two or more dimensions. areaDetector offers a predefined set of PVs and a driver structure to implement the EPICS Device Driver. areaDetector considers the most common configuration parameters of image acquisition systems, for example, exposure time, binning, and image size. To include support for a new detector, the developer just needs to add the appropriate calls to the camera or frame grabber manufacturer driver in the defined functions, and the control will be automatically available through the predefined areaDetector PVs. The developer can also add more PVs and handling methods to extend the functionality, and therefore adding support for other features of a concrete imaging system not covered by areaDetector. The use of a predefined set of PVs is particularly interesting for complex systems using different cameras of different types and manufacturers. With areaDetector, the PVs to control the main characteristics of all the cameras will share the type and name, except for the camera identifier, instead of having each one a custom set of PVs. This implies

that the control system only needs to add support for the camera specific functionalities, while the rest of the acquisition parameters are controlled uniformly. The EPICS community has already developed more than twenty device drivers based on areaDetector, including Firewire cameras, Pilatus and Pixirad pixel-array detectors, and mar345 Marresearch GmbH cameras.

The other advantage of areaDetector is the use of plugins for image processing. Plugins receive the data from the areaDetector Drivers or other plugins. The data structure used at this effects is the areaDetector N-Dimensional Array (NDArray). This structure is used as input and output for all drivers and plugins, so all the possible combination of plugins and drivers is possible. This structure helps to configure a custom processing chain that can also be changed at runtime. As it happens with the drivers, the EPICS community has already developed several processing plugins, covering some of the most common functions required in an image acquisition and processing system, as Region-Of-Interest (ROI) extraction, geometric transforms, and conversion to JPEG, HDF5, EPICS standard waveform PV, or EPICS v4 PV. These last two converters allow the connection of plugins and drivers running in different IOCs. This feature can be used to configure parallel and distributed processing chain using multiple IOCs running on different computers.

The Graphical User Interfaces (GUIs) for areaDetector drivers and plugins follow the same policy than the drivers and plugins themselves. As some of the PVs to use are shared among all the drivers and plugins, the GUIs follow a hierarchical model where the base display have the common set of PVs and it is extended with the custom PVs added to control the device driver or plugin features. All the areaDetector GUIs used to be implemented using Motif Editor and Display Manager (MEDM), but this tool to create and visualize control screens has become obsolete. Newer versions of areaDetector come with Operator Panel Interfaces (OPIs) implemented using Control System Studio (CSS) and Best OPI Yet (BOY).

2.3.1.3 Nominal Device Support

The Nominal Device Support (NDS) [12] was developed by the company Cosylab for ITER, and it is meant to be used mainly for ITER Diagnostics. NDS propose a driver structure and a standard set of PVs to control the most common

functionalities of a data acquisition system, similar to `areaDetector` but not only focused on multidimensional sensors but also for analog or digital acquisition devices. The NDS driver follows a hierarchical structure using the concepts of device, channel group, and channel. A NDS device is formed by one or more channel groups. Each channel group has one or more channels sharing some characteristics or configurations. In NDS each part of the data acquisition system has its own set of PVs to manage its specific set of functions. The functionality of the acquisition system is organized hierarchically. The device functions are those that will affect the entire acquisition, configurations of a channel group will affect only to its channels, and changes in the parameters of a channel only take effect on the channel itself. Additionally, the device, the channel groups, and the channel are managed by state machines. These state machines are also organized hierarchically, so one must first configure the device and set it to ready, and then continue with the channel groups and the channels.

NDS is implemented using C++ classes. Therefore, the implementation of a new EPICS Device Driver only implies the creation of the device, channel group, and channel classes extending the NDS base classes, and to adapt the basic functionality to the particular system used. This adaptation includes the addition of calls to the manufacturer driver in the NDS methods and the definition of new methods to add support to new functionality specific to the system used.

Like in `areaDetector`, in NDS is also possible to create plugins and create a processing chain based on the plugin architecture, but there aren't as much processing plugins as there are for `areaDetector`, and the collection of available plugins is limited to a few examples and proofs of concepts. However, NDS have support for `areaDetector` NDArrays, therefore is possible to use the `areaDetector` plugins for NDS.

2.4 Data and Image acquisition system in big physics experiments

The aim of the research carried out in this thesis was the design and development of a methodology for implementing high throughput data and image acquisition systems. This kind of systems are used in wide variety of fields, but this work focuses on big physics experiments, with special emphasis on the application to ITER. However, the proposed methodology can be applied to other systems totally or partially, depending on the target architecture.

In this section, some big physics experiments will be introduced. It is clearly out of the scope of this thesis to offer an overview of the current status of all big physics experiments. Only a few selected examples are presented here, the ones that the author considered more relevant for the methodology for being the newest, the most important in the field, or for using part of the hardware or software elements included in the methodology. The reader must also consider that there is no possibility or intention to present a complete list of experiments using specific technologies or even a complete and detailed description of the systems used in the selected experiments. The details of the implementation of these experiments are commonly divulged only through journal articles and conference presentations. These communications usually never cover the complete specifications of all systems used in the experiments, rather than general descriptions of some of the systems and sometimes details about the implementation of a specific system or subsystem.

Two types of experiments are presented here four nuclear fusion devices, and a neutron source. Here are presented a few notes about the purpose and main characteristics of these experiments.

- Nuclear fusion devices are experiments trying to demonstrate the possibility of achieving net energy from a controlled fusion process, with the ultimate objective of constructing nuclear fusion power plants. Here net energy is interpreted as producing more energy from the fusion process than the energy used to start and maintain the process. The research in fusion energy started in 1950 and today is still a hot topic, with hundreds of million dollars spent yearly on publicly funded research. There are several approaches under research, but the most developed nowadays is the magnetic confinement, and among the possibilities of magnetic confinement, the principals are the tokamak and the stellarator. In tokamaks, the plasma magnetic confinement is achieved via strong toroidal fields. In the other hand, stellarators confine plasma using a magnetic field that bends over itself continuously. A schema of the plasma shape and main magnet system of both approaches is depicted in Fig. 10. Both have their advantages and disadvantages, for example, the strong toroidal magnetic field used in tokamaks makes the plasma non-stable,

Fig. 10 Comparison between tokamak and stellarator main magnet systems and plasma shape.

Reproduced from www.economist.com

and therefore suffering from some instabilities not present in stellarators. These instabilities difficult the operation in steady-state needed to obtain net energy. In contrast, building the tokamak with its torus shape is easier than the Mobius-like shape of the stellarator, as well as the design of the divertor. A more exhaustive comparison of stellarator and tokamak approaches can be found in [13] and [14]. In the following subsections, four examples of fusion devices will be presented: the Korea Superconducting Tokamak Advanced Research (KSTAR), one of the first tokamak using superconducting magnets; the Joint European Torus (JET), the largest magnetic confinement device; ITER, the first fusion device under construction designed to produce net energy; and Wendelstein 7-X (W 7-X), the world's largest stellarator.

- Neutron sources are devices designed for the study of materials using neutron beams. The applications of the material research carried out with neutral beams go from the discovery of new materials for pharmaceutical drugs, aerospace or plastics, to better understanding of fundamental and particle physics. The detailed studies depend on the quantity of neutrons produced by the source. The classical approach used to produce neutrons was based on nuclear reactors, but there is a limitation on the number of neutrons produced using this method. This limitation is overcome by the neutron spallation sources based on particle accelerators. These neutron sources generate the neutron beam by impacting highly accelerated protons into a special target. The European Spallation Source (ESS) will be

presented below, a neutron spallation source currently under construction that will generate a neutron beams 100 times brighter that any current neutron source.

2.4.1 JET

The Joint European Torus (JET) is a fusion research facility located at Culham Centre for Fusion Energy in Oxfordshire, UK. JET is a joint European project carried out under the fusion program established by the Council of the European in 1970. Its purpose, as defined original design proposal of 1975 [15], is to obtain and study a plasma in conditions and dimension approaching those needed in a thermonuclear power plant. The device is in operation since 1983, and currently is the world's largest operational tokamak. In Fig. 11 can be seen a representation of JET size in scale with a human together with a photography of JET during maintenance operations in 1985. JET is also the first and only fusion experiment capable of operating with the deuterium-tritium fuel mix that will be used in the future commercial fusion reactors [16]. JET achieved in 1997 16 MW the record of produced fusion power using that fuel mix, a record that is still unbeaten. JET operation is done in short pulses of approximately 40 s of duration every 45 minutes. During this pulses, an average of 40 GB of data is generated, and only the 10% of it are processed data. Nowadays, the main objective of JET is to prepare and test the technologies and plasma operating scenarios that will be

Fig. 11 The JET tokamak.

In the left side, representation JET tokamak in comparison with a human. At the right side, photography of the JET tokamak during maintenance in 1985.

needed for ITER. At this purpose, JET has evolved during this years, introducing several changes in the machine to progress with the advancements in fusion research. Two of the major changes in the machine matches with design decisions for ITER. These changes are the modification of the machine to include a divertor [17] and the substitution of the carbon-based first wall for the so-called ITER-like wall made of beryllium and tungsten [18]. According to the roadmap to the realization of fusion energy [19], JET will end the operations on 2019-2020. The plans include a set of experimental campaigns covering the entire range of hydrogen isotopes (hydrogen, deuterium, and tritium) to prepare for a final extensive campaign using deuterium and tritium [20].

2.4.1.1 JET CODAS

JET's COntrol and Data Acquisition Systems (CODAS) [21], [22] is implemented as a distributed control system following a hierarchical and modular architecture. The three levels of the hierarchical control structure are identified as the JET-wide supervisory control (Level 1), several subsystem supervisory control (Level 2), and local control systems for the different units and components (Level 3). The plasma control function is carried out by the cooperation of different hardware systems performing the necessary I/O and processing functions. In JET, the systems performing these functions are the various diagnostics, analysis, local actuator managers, controllers, and plasma protection systems. These systems are connected using an Asynchronous Transfer Mode (ATM) network, as shown

Fig. 12 JET's Real-time measurement and control scheme.

Reproduced from [24].

in Fig. 12 [23]. The distributed organization of the system provides of great maintainability and robustness. The smaller are the systems, the easier are to be managed and maintained, as the recommissioning and upgrade effort is proportional to the system size. Also, the failure of a system only affects the function implemented by that system, and this problem can be solved adding redundancy in the functionality suppliers or with fail-safe strategies to terminate the plasma. Another advantage of having a distributed system is the possibility to add a new function without the recommissioning of the whole system, as it only requires the integration of a new system.

The systems used at JET are very heterogeneous and use different technologies. For example, the motional Stark effect (MSE) diagnostic [24] uses a crate hosting several PowerPCs and ADCs. The crate interconnects all the modules using VME, a backplane bus originally developed in 1979 for the Motorola 68000 microprocessor that was later standardized as IEEE 1014-1987 [8]. The PowerPCs in the VME crate run the RTOS VxWorks [7]. Another example of a JET system using completely different technology is the Plasma Vertical Stabilization (VS) system [25]. The VS system is based on ATCA technology [26]. The VS ATCA system is entirely in-house developed and comprises the ATCA shelf, a controller board with a quad-core processor running a Linux OS with Real-Time Application Interface (RTAI), and six FPGA-based Multiple Input Multiple Output (MIMO) boards. The ATCA backplane uses two connection topologies simultaneously, a PCIe with star topology connecting the controller with the MIMO boards, and an Aurora with full mesh topology connecting the MIMO boards.

2.4.1.2 JETs RT System

The distributed control philosophy and the heterogeneity of the systems present in JET influenced in the creation of the Multithreaded Application Real-Time executor (MARTE) [27], [28]. MARTE is a C++ multi-platform framework for the development and execution of real-time control systems. The main objective of MARTE is to establish a clear separation between the algorithm implementation and the hardware and software interfaces with the plant input/output system. It also provides tools for real-time thread scheduling, offline and online validation, and an architecture allowing code reuse and easy maintainability.

MARTE applications consist of a set of Generic Application Modules (GAMs) connected using the Dynamic Data Buffer (DDB). Each GAM can read and write an arbitrary number of data from the DDB. Special GAMs might be used to perform I/O tasks, as sample signals from the Real-Time Data Network (RTDN) or send the alarm value to the RTDN. GAMs are executed inside threads managed by the real-time scheduler. Communication and synchronization between the GAMs executed in different threads are also possible. An example of a MARTE application structure and how are they integrated with JETs RTDN and persistence system is depicted in Fig. 13. The number of threads, their timing, or the distribution of GAMs among the threads is completely configurable. In runtime, the real-time executor will run the GAMs of each thread periodically in the order specified.

MARTE architecture allows the modularization of an algorithm in a set of GAMs, each one of them can be easily reused in other application. Modifications of a specific GAM (e.g. to improve the performance) are also possible and can be done without modifying the rest of the GAMs as long as the interface (the DDB data read and write) remains invariable.

Fig. 13 Schema of typical MARTE application and its integration into JETs RTDN and persistence system.

2.4.2 W7-X

The Wendelstein 7-X (W7-X) (Fig. 14) is the biggest fusion reactor of the stellarator type and it is located in Greifswald, Germany. The project is funded mainly by Germany with the participation of Europe and, in minor grade, the United States of America. The project started in 1994, and it was designed to be the successor of the first stellarator belonging to the “Advanced Stellarator” generation, the Wendelstein 7-AS. The W7-X was designed to evaluate the main components of a future fusion power plant based on stellarator technology. For this, the W 7-X aims to achieve plasma equilibrium and confinement similar to the ones achieved in a tokamak of the same size. It is expected to obtain these conditions in steady-state operation, with pulses lasting up to 30 minutes. Nonetheless, the W7-X is not aiming for energy-yielding plasmas using tritium fuel, as it is expected that the research carried out at ITER at this effects can be largely transferred to stellarator devices. Its construction finished on 2014, and after the first helium plasmas in late-2015 the first experimental hydrogen campaign finished in March 2016.

Fig. 14 Graphic of the Wendelstein 7-X and the superconducting magnetic coil system.

The coil system is composed by 50 non-planar coils (in red) and 20 planar coils (in orange). The shape of the generated magnetic field is depicted in blue.

One of the main challenges of using the stellarator technology comes from the magnetic field required for its operation. The magnetic field in stellarators bends over itself to ease the plasma confinement. The shape of the magnetic field is modeled using complex codes in the high-performance computer, as well as the shape of the non-planar magnetic coils needed to achieve the required shape. For the W 7-X, a set of 50 non-planar superconducting coils were designed, and 20 supplementary planar coils were installed for auxiliary magnetic field shape correction. The magnetic coil system and magnetic field of W 7-X are depicted in Fig. 14. The improvements in the coil manufacturing process allowed the W 7-X to generate the desired magnetic field with a discrepancy of less than $1/100.000$, an accuracy without precedents in a fusion device. Another challenge of this experiment is related with the data management derived from the steady-state operation as it is expected to generate more than 20 GB/s during the 30 minutes of operation. This big amount of information must be properly handled and stored. Moreover, this must be done in real-time during the operation, as data to handle is too large to be buffered and wait for the interpulse time for the storage.

2.4.2.1 W 7-X CoDaC

The W7-X Control, Data Acquisition and Communication (CoDaC) [29]-[31] system is based on custom developed software. The CoDaC system is designed hierarchically with the central control system on the top of the hierarchy, as depicted in Fig. 15. The system is based on the idea of segment. A segment is defined as an aggregation of constant parameters being valid for a certain time interval. At the beginning of the experiment, the W7-X will be operated in pulse mode, with short plasmas consisting of a single segment. The successive pulses will be managed as successive segments with the pulse configuration with stop segments for the interpulse time. In a second stage of the operation, the W7-X will be operated in steady state. This mode will be represented by a succession of segments. The other possible operation mode designed for the W7-X is a long-term discharge composed by several experiments. Each experiment can be a segment or a set of segments, and between the experiments there will be an intermediate segment for making a controlled transition between experiments. The software for the design of the segments and the configuration of the pulse with the defined segments is custom made. In operation, the local controllers will receive the segment information from the plasma control system, but they can

Fig. 15 Overview of Wendelstein 7-X Control, Data Acquisition and Communication system

Reproduced from [33]

also run autonomously for commissioning and test. The control system is divided into three groups: safety systems, slow controllers, and fast controllers. The safety system is mainly implemented with PLCs and connected to the central control system using secure buses as Profisafe and a dedicated signal network. Slow controllers are also implemented using Programmable Logic Controllers (PLCs), ruggedized computers designed for the control of industrial processes. The interface between slow controllers and the central control system is based on regular Ethernet using TCP or UDP. The fast controllers are implemented with regular PCs. Those controllers with real-time requirements run VxWorks, and the rest use Linux or Windows OSs. The interface with the control system is in both cases based on Ethernet, but the real-time data network is separated physically from the standard network and uses different protocols to ensure low latency. The analysis algorithms are implemented in Java to ensure the portability among the different platforms.

2.4.3 ITER

ITER (Fig. 16) is an international nuclear fusion project to design and build an experimental fusion tokamak reactor in southern France. It is one of the most ambitious energy projects in the world today with an estimated cost of more than 12 billion euros (more than 14 billion dollars). The project started in 1986 as an agreement among the European Union, Japan, the former Soviet Union, and the United States of America. This agreement has been extended over the years to include also the People's Republic of China, the Republic of Korea, and India. ITER construction started in 2007, and the construction is still ongoing in Saint-Paul-lès-Durance (France). Once finished, ITER will be the world's biggest tokamak. Its main goal is to demonstrate the feasibility of nuclear fusion as a source of energy. With this intention, ITER was designed to fulfill a series of goals [32], [33]. ITER is planned to be the first fusion device producing net energy, i.e. producing more energy than the energy invested in the plasma heating. The amplification factor varies from $Q=5$ when operating the tokamak in steady-state, to $Q \geq 10$ for long-pulse operation (> 1000 s). ITER must also demonstrate the technological viability of a fusion power plant, and it will serve for the research and test of all the systems required for the future fusion power plants, including

Fig. 16 Representation of ITER tokamak and size comparison with JET

the tritium breeding system required to produce the hydrogen isotope needed to fuel the fusion reactor. The results obtained from the design, development, and research carried out in ITER will serve to the development of the DEMOnstration Power Station (DEMO) that will be the first attempt to construct a tokamak-based fusion power plant.

2.4.3.1 ITER CODAC

ITER's Control, Data Access and Communication (CODAC) is depicted in Fig. 17. CODAC is a distributed and hierarchized control system. The different CODAC subsystems are interconnected using several dedicated networks, all of them based on Ethernet. The implementation of ITER's CODAC is very challenging as one of the unique characteristics of ITER is that 90% of the components will be supplied in-kind by the ITER members through their Domestic Agencies. This implies the integration of numerous systems developed by a variety of manufacturers. These systems are estimated to generate and exchange about one million signals and up to 50 Gb/s. The ITER approach for managing in a consistent way these plant systems is based on a set of standards for the plant instrumentation and control systems named Plant System

Fig. 17 Schematic of ITER CODAC System

Handbook [34] and the use of a software suite called CODAC Core System (CCS) [35]. The OS chosen for CCS is RHEL with the possibility of using the MRG extensions for real-time in Fast Controllers. The other big component of CCS is EPICS. For the configuration of the systems and simplification of the development using CCS tools, ITER has developed the Self-Description Data (SDD) toolkit. Using this tools, the system configuration can be described and the toolkit can generate the configuration files for the different applications of the environment, namely the EPICS applications and IOCs, alarm and archiving system, and BOY Human-Machine Interfaces (HMIs) among others.

The hardware that can be used for the instrumentation and control systems of the plants is defined in the ITER Instrumentation and Control (I&C) Catalogs [36], [37]. According to these catalogs, the slow controllers will be implemented using PLCs. PLCs are meant to be used for processes with slow regulation loops ($>10\text{ms}$ in the case of ITER) that are expected to be stable and will not change much during the existence of the experiment. The rationale for this is that the added robustness of the PLCs makes them slower compared to other systems. Also, the modification, deployment, and maintainability of a PLC application is difficult and must be done individually. For Fast Controllers the catalog is based on a host controller with CCS and FPGA-based devices in four form factors: PXI, PXIe, MTCA, and ATCA. These technologies were chosen for the easing the modification and deployment of Fast Controllers, as they are intended to be used for functions directly implied in the experiment, and therefore frequent changes are expected.

2.4.4 KSTAR

The Korea Superconducting Tokamak Advanced Research (KSTAR) [38], depicted in Fig. 18, is the major effort of the national fusion program of the Republic of Korea. The tokamak, designed in 1995 and built in 2007, is placed in Daejeon and managed by the National Fusion Research Institute (NFRI). The device is the first tokamak to feature a fully superconducting magnet system [39] and is designed to support steady-state operation of high-performance plasma. After the first phase of the device dedicated to advanced tokamak physics study with a pulse length of 20 s, now the steady-state operation is under study. For now, the KSTAR holds the world's record of high-performance plasma pulse

Fig. 18 Picture and model of KSTAR tokamak.

Model reproduced from [66].

length, with 70 s achieved in December 2016. It is expected that KSTAR will be able to maintain the high-performance plasma condition up to 300 s. The operation of the KSTAR experiment is contributing to the development of ITER and future tokamaks. More specifically, KSTAR is providing knowledge in the fabrication of Nb_3Sn superconducting coils, the assembly of the advanced tokamak components, and the operation of a superconducting tokamak. Moreover, KSTAR is also testing some of the developments and prototypes implemented for ITER, proving its suitability for the operation of a real machine already constructed. This is the case of the Interlock system and the CODAC Tools, including the IRIO Software developed under the scope of this thesis.

2.4.4.1 KSTAR Control System

The KSTAR Control System ([40], [41]) is organized as a network-based distributed control system. It is composed by the Plasma Control System, the Machine Control System, the Diagnostic Control System, the Time Synchronization System, and the Interlock System. These systems are connected using different networks, as depicted in Fig. 19. The control and supervision of the different subsystems are done using EPICS and a dedicated machine control

Fig. 19 Structure of KSTAR Control System

Reproduced from [40]

network based on Ethernet. A gateway is used to provide access control for the EPICS clients. The gateway also allows to generate virtual circuits between the clients and the subsystems for reducing the number of CA connections on the IOC side, and thus reducing the CPU usage. This network is also used for archiving of machine operation data using the CA Archiver. This archiver will manage a low throughput due to the low data/event rate used in machine operation. However, this archiver needs to work constantly without interruption, as some plants like the vacuum, the helium cryogenic system, and the superconducting magnets need supervision 24 hours a day. The other part of the archiving, dedicated to diagnostic and experimental data, is done through a dedicated GigabitEthernet network and using MDSPlus [42]. This data is only produced during the pulses, but the throughput is much higher (almost 6 GB/s) than the one required for machine operation. The rest of the networks are optical networks. The optical timing network distributes a 100 MHz timing clock and event information for all the subsystems. The interlock signals distributed a separate interlock network based on the ControlNet open industrial protocol. The EPICS systems access to this network using EtherIP communication modules for the conversion of between ControlNet and UDP. The last network is a real-time ring network based on reflective memory modules with very low latency ($< 0.4 \mu\text{s}/\text{node}$) and high data throughput (more than 174 MB/s). This network is used for communication of the plasma control system with the diagnostic systems and the actuators.

The different subsystems are implemented using several technologies. Interlock systems will be implemented with PLCs integrated into EPICS with the EtherIP modules mentioned above. For other critical systems requiring robustness and real-time behavior with small data footprint, two alternatives are used. Some of them are VME systems with the VxWorks RTOS, which can run EPICS natively. The others use CompactFieldPoint devices, the predecessors of CompactRIO devices, running PharLab RTOS and RT-LabVIEW programs. These devices are integrated with EPICS through a custom designed TCP/IP protocol. The high throughput data acquisition systems without real-time requirements are implemented using PXI devices from National Instruments. These devices are integrated into EPICS with a custom EPICS Device Driver called genericPXI. The genericPXI driver allows the integration of all NI PXI devices using the NIDAQ and NIDAQmx hardware in EPICS, similarly to what the IRIO Software explained in the next chapter does with NI RIO devices. The hardware used in KSTAR reveals a great similarity with the hardware architecture used in ITER. ITER architecture is almost the same than the architecture in KSTAR but using the latest improvements in the technology.

2.4.5 ESS

The European Spallation Source (ESS) [43] is the neutron spallation source under construction in Lund (Sweden). The ESS is expected to be the world's most

Fig. 20 Comparison of beam brightness and duration among ESS and current neutron sources

powerful neutron source, outperforming current neutron sources by both pulse length and beam brightness (Fig. 20), delivering a peak flux five times higher than any accelerator-based spallation source and 30 times than any reactor-based neutron source. The unprecedented pulse brightness and length will allow obtaining a deeper knowledge of materials, and under new conditions currently impossible. Nonetheless, these characteristics can be configured for each individual experiment in contrast with the rest of neutron sources which fix this parameters in the design of the source and instruments. This flexibility will allow the users to adjust the experiment to the features of interest and vary the resolution to match to the time and spatial correlations desired.

Its construction begun on 2014 and it is expected to start its operation on 2019. From 2010 to 2015 it was owned jointly by the Swedish and Danish Government, but in 2015 the European commission established it as a European Research Infrastructure Consortium. The design of the ESS experiment is represented in Fig. 21. An ion source products a proton beam that is accelerated in a linear accelerator up to 2 GeV. The accelerated proton beam hits a 4-tonne helium-cooled rotating wheel made of tungsten. The impact of the protons with the nuclei of the target produces a collection of high-energy neutrons that are redirected towards the 22 instrument stations where the neutrons will interact with the material or tissue sample, generating more than 20 GB/s. Each station is calibrated to observe a different set of interactions. The use of this stations will be offered as a service to both industry and academia, who will determine the kind of samples and interactions that will be studied. The instruments and

Fig. 21 Representation of ESS experiment

condition of the experiments in ESS are designed to support studies soft condensed matter, life science, magnetic and electronic phenomena, chemistry of materials, energy research, engineering materials and geosciences, archaeology and heritage conservation, and fundamental and particle physics.

ESS is contributing to the scientific development in other ways apart from the characteristics of the beam and instruments. One of these contributions is in the development of a new converter material for neutron detectors. Most instruments in current neutron sources use detectors based on Helium-3, a helium isotope with very limited availability. At this effect, ESS is collaborating in the development of the technology for neutron detectors based on enriched boron. ESS will also offer a complete infrastructure to allow the users benefit of the most productive use of the instruments, including sample preparation and characterization, and detector and sample environment systems among others. It will also collaborate with the Data Management and Software Centre of the University of Copenhagen for completing the user experience with advanced data analysis tools and computational support for modeling and simulation.

2.4.5.1 ESS ICS

The ESS Integrated Control System (ICS) [44] is depicted in Fig. 22. It is designed as an object-oriented, decentralized, and distributed system based on EPICS. It is composed by five main parts, namely the Control System Core, the Control Boxes, the Beam Line Element Databases (BLED), the HMIs and Development Environment (DevEnv).

The Control System Core is the coordinator of all the different parts and the responsible to offer the services needed continuously, as the archiving, alarm monitoring, timing distribution, slow feedback controls. It also includes the machine and personnel protection systems. These services are implemented with EPICS IOCs.

The Control Boxes are the units dealing with the physical connections to the equipment and integrate them in the ICS using EPICS. They perform the real-time control of the equipment while offering the abstract representation of this equipment to the higher control layers (i.e. the Control System Core and the HMIs).

The BLED is made by the tools and databases used for the central storage, the machine information and configuration management and access control. This includes the IOC and HMI versioning, storage, configuration, and deployment. It will use ITER SDD toolkit for relating the EPICS and BLED databases.

Fig. 22 Overview of ESS ICS and its main components

The HMIs is the collection of devices and software tools used for the interface with the users. It includes graphical and non-graphical interfaces, and it needs to cover the different requirements of the different users, from engineers to operators or physicist.

The development environment provides the services required for the proper development of solutions for ESS. It includes the mechanisms for granting software consistency, artifact sharing, and code storage among others. It also provides a safe procedure for the upgrades development.

The different parts will be connected one of the seven dedicated networks. These networks are the control network used for the EPICS CA connections, the timing network used for time synchronization and timestamping, the machine protection system network, the personnel protection network, the safety/interlock network, the video streaming network, and the diagnostic archiving network. These networks will use standard computer network or optic fiber, and UDP/TCP protocols will be used whenever is possible.

The hardware architecture of the control boxes is depicted in Fig. 23. The architecture of ESS control boxes follows the philosophy followed by ITER and other experiments. The control box is composed by one or more computers running an IOC each one, and the different devices used for control and data acquisition. The devices could be connected to the computes via an Ethernet switch, as is the case of the PLCs used for safety and interlock or other intelligent controllers used for data acquisition, or connected directly to the computer. For

Fig. 23 Schematic example of ESS Control box components

the latter case, the case of control boxes will be based on MTCA or compactPCI PlusIO (cPCI PlusIO) technologies. MTCA will be used for systems requiring high throughput possibly with real-time requirements, while cPCI PlusIO will be used as a more cost-effective solution for systems with lower throughput requirements.

2.5 Conclusions

This chapter has presented a reference data acquisition model and the possible technologies that can be used to implement it. This model tries to integrate all the functionality usually required for the use in big physics experiments. The application to big physics experiments provides an interesting framework where a compromise must be found between the performance offered by novel technologies and the robustness and reliability of the well-established ones.

The big physics experiments presented in this chapter show how in each experiment different alternatives are chosen. In these experiments we can find examples of custom made solutions and adaptations of existing products, both found at the hardware and software levels. However, as the complexity of the experiment grows usually the custom made solutions are discouraged in benefit of the commercial products that certify the needed levels of reliability and stability.

The logical and physical distribution of the systems involved in the experiments requires the use of a distributed control system. From the different alternatives, EPICS has become a de-facto standard for the distributed control system implementation in big physics experiments. It has been used by different experiments since 1994, what provides guarantees of the stability of the solution, while the active community and its open source license allow keeping it updated with the latest improvements in the technology. It is easier to customize EPICS to a particular experiment than implement a new control system software. These facts make EPICS a valuable tool to consider for the methodology presented in this thesis.

Regarding the hardware elements, the ITER experiment was taken as a reference target for the technology to use in this work. One of the reasons for this is that ITER is one of the most ambitious projects currently ongoing, and also one of the

most complex and expensive ones. ITER is currently in the construction phase, but much work is being done for designing and prototyping the systems that will be used in the final system. For the implementation of ITER systems, ITER provides a set of hardware catalog and software tools. All the elements in ITER catalog must fulfill the reliability and robustness requirements, including the necessary radiation, shocking, and vibration test. The elements in the hardware must also guarantee their availability until the end of the experiment, which is in more than 25 years. Also, the technologies used in ITER will serve as a reference for the future fusion devices, including DEMO, the first prototype of a nuclear fusion power plant.

Considering this, ITER provides an excellent framework for developing a methodology such as the one proposed in this thesis. From the data acquisition and processing systems used in ITER, the ones with higher throughput and processing requirements are the Fast Plant System Controllers. For the implementation of these systems ITER proposes hardware solutions, as shown in Fig. 24. The solutions are based on PCI/PCIe, PXI/PXIe, and xTCA. The work presented here is focused on the use of PCI/PCIe and PXI/PXIe. The reason for this is that, while offering a good and powerful alternative to PXI/PXIe, the xTCA technology is newer and it was less established at the beginnings of this thesis.

Fig. 24 Technologies considered for ITER Fast Plant System Controllers

The price and complexity of this solution are also higher, and PXI/PXIe offered a wider variety of options for implementing the systems target of this work. In any case, PXIe or xTCA solutions are both based on the use of PCIe technology. All previous big physics experiments were based on communication buses supporting a throughput lower than the 1 Gb/s provided by a PCIe Gen 1 link. Next generation of instrumentation systems are based on PCIe, and the performance and maximum throughput will be linked to the evolution of this technology.

However, the status of the PXI/PXIe technology also presented some issues to be solved to adjust it to the reference system proposed. First of all, there was no available any integration of the PXI/PXIe technology with EPICS. The use of FPGA gives flexibility to the system, making easy to change how the acquisition hardware is implemented and allowing the implementation of data process in the acquisition hardware. Nevertheless, these features also come with the associated complexity of supporting every possible implementation that could be done in the FPGA. If an EPICS Device Driver must be implemented for every FPGA configuration, the system will be very difficult to maintain, introducing considerable stability risks and unnecessary complexity. Regarding the processing requirements, the FPGA alone may not achieve the necessary performance for certain applications, or the resources available may not be enough. That is the reason for including a GPU in the ideal system model, but there was no solution implemented for using direct data transfer between PXI devices and GPUs. Another issue that needs to be considered for the solution is the procedure for the deployment of the solutions implemented.

This unsolved issues motivated the development of the methodology presented in this paper. The methodology and the supporting software provided try to solve these problems to offer a solid framework for implementing data acquisition and processing systems integrated into EPICS.

Chapter 3

3 METHODOLOGY FOR THE DEVELOPMENT OF DATA AND IMAGE ACQUISITION SYSTEMS

3.1 Introduction

In this section, the proposed methodology will be explained in detail. This methodology was designed considering the requirements of data and image acquisition systems for fusion devices and its integration in a distributed control system.

The framework of the proposed methodology is based on two different types of elements. The first ones are the open source and commercial products chosen for the implementation of the data acquisition and processing system. That includes EPICS, the distributed control system chosen in this work, as well as the specific hardware devices and the development tools used with these devices. These elements will be presented here, except EPICS that was introduced in the previous chapter.

The other elements of the methodology are the products developed to adjust the commercial products chosen to the requirements of the model. These products are one of the main contributions of this thesis. They are also the key elements for the development and testing cycle defined for the methodology.

3.2 Commercial Products

3.2.1 Hardware Components

In the Fig. 25 shows the different hardware elements that will implement each of the parts of the system. Independently of the type of sensor (analog, digital, or image), the acquisition hardware will be implemented using devices based on FPGA. More specifically the Reconfigurable Input/Output (RIO) devices family from National Instruments (NI)[45]. These devices are part of the ITER Catalog for Fast Controllers [36], which provides a solid framework where to develop and test the methodology. Moreover, the devices in the ITER Catalog are guaranteed to be available during the lifetime of the experiment, and NI has tested the reliability and availability of the devices. Also, the graphical environment used to design the configuration of these devices, LabVIEW, will ease the development of the system, as will be explained below. The RIO devices will implement all the required acquisition configurations and can also be configured to perform a pre-processing of the data acquired. For further processing of the data, the GPU present in the system will act as processing hardware. Both the acquisition and

Fig. 25 Schema of the hardware components used in the methodology and its connections.

the processing hardware will be connected to an industrial host computer implementing the system controller functionality. The host computer will provide a link between both of them and will also interface with the central system using fast network connections. The system will be synchronized to the master clock that provides the absolute time reference using a timing card in the PXIe chassis. This synchronization will be done using the IEEE 1588-2008 Precise Time Protocol v2 [46] (PTPv2). All this architecture matches with the ITER CODAC architecture and the guidelines for Fast Controllers [47].

3.2.1.1 FPGA-based devices

The data acquisition and preprocessing will be carried out in FPGA-based devices. As it was explained before, the FPGA-based devices are very powerful devices capable of acquiring data at high speed and sustaining a high throughput with the system controller to send the acquired data. When using FPGA devices, it is also possible to perform data process in the acquisition device, in parallel with the data acquisition, and obtaining a performance comparable with hardware specifically designed for the processing. The devices chosen were the FlexRIO and CompactRIO devices from National Instruments. Both are part of the RIO family, together with the R-series, but there are basic differences between them. This differences will be explained in the following subsections, but they can be summarized as the type of mounting and connection, the maximum data bandwidth, and the configuration of the input/output.

Independently of the platform chosen, the main function of this hardware will acquire the data from the sensors. Both platforms have a wide variety of input/output adapter modules enabling them to be used with different sensors at different sampling rates, or for example in the case of analog data acquisition, also different resolution in the analog to digital converters. That also affects to the data treatment for each particular case. In some cases all the data acquired can be sent directly to the host computer or the processing hardware using DMA transfers. In other cases, the data will be processed or decimated, and the results will also be streamed using DMA transfers. In particular applications data acquired or processed will be used only internally for control processes and only some samples will be sent to the host via DMA or the host must acquire them manually by directly reading from FPGA registers. All this configuration will be done by the FPGA bitfile obtained using the LabVIEW tools that are explained below. For the most common applications, like the ones described here, templates and bitfiles were developed for this methodology in such a way that will serve as both platform testing and starting point for the development of user custom applications.

3.2.1.1.1 FlexRIO devices

FlexRIO devices will be used for fast data acquisition and a first processing of the data. These devices are based on the FPGA and PXI/PXIe technology. A typical

system based on FlexRIO is composed of one or more FlexRIO devices in a PXI/PXIe chassis. For the system controller there are two options, it can be an embedded controller plugged directly into the chassis or an external computer connected to the chassis using a PCI/PCIe card and an MXI cable. The inputs and outputs of the different FlexRIO devices will be determined by the adapter module used. There are available different adapter modules for digital/analog input/output with different front ends, number of input/outputs, digitalization precision, and maximum sampling or update rates. There is also available an adapter module development kit for developing custom adapter modules. Each FlexRIO device can be connected to a single adapter module. The interface of the FlexRIO device with the chassis varies from PXI (up to 533MB/s) in the PXI-795xR devices, 4xPXIe v1.0 (up to 800 MB/s) in the PXIe-796xR devices, or 4xPXIe v2.1 (up to 1.7 GB/s) in the PXIe-797xR devices. Regarding the FPGA, devices of PXI-795xR and PXI-796xR have different Xilinx Virtex-5 FPGAs while PXIe-797xR devices use FPGAs from Xilinx Kintex-7 family that have more FPGA slices and DSP slices. Most of the FlexRIO devices come with onboard RAM memory that can be handled directly from the FPGA. Recently (late 2015) National Instruments launched a new family of FlexRIO devices (NI 793xR) combining the resources of a FlexRIO PXIe-797xR device with an embedded controller in a single device and conserving the slot for the input/output adapter module. Table 2 and Fig. 26 show a comparison of the different FlexRIO devices and a schema of possible data acquisition systems based on FlexRIO technology.

Fig. 26 Schema of possible data acquisition systems using FlexRIO devices.

Table 2 Comparison of resources available at the different FlexRIO devices

Model	Bus/Form Factor	FPGA	FPGA Slices	FPGA DSP Slices	FPGA Memory (Block RAM)	Onboard Memory (DRAM)
NI 7935R	Stand-alone	Kintex-7 K410T	63,550	1,540	28,620 kbits	2 GB
NI 7932R	Stand-alone	Kintex-7 K325T	50,950	840	16,020 kbits	2 GB
NI 7931R	Stand-alone	Kintex-7 K325T	50,950	840	16,020 kbits	2 GB
PXIe-7976R	PXI Express	Kintex-7 K410T	63,550	1,540	28,620 kbits	2 GB
PXIe-7975R	PXI Express	Kintex-7 K410T	63,550	1,540	28,620 kbits	2 GB
PXIe-7972R	PXI Express	Kintex-7 K325T	50,950	840	16,020 kbits	2 GB
PXIe-7971R	PXI Express	Kintex-7 K325T	50,950	840	16,020 kbits	0 GB
PXIe-7966R	PXI Express	Virtex-5 SX95T-2	14,720	640	8,784 kbits	512 MB
PXIe-7965R	PXI Express	Virtex-5 SX95T	14,720	640	8,784 kbits	512 MB
PXIe-7962R	PXI Express	Virtex-5 SX50T	8,160	288	4,752 kbits	512 MB
PXIe-7961R	PXI Express	Virtex-5 SX50T	8,160	288	4,752 kbits	0 MB
PXI-7954R	PXI	Virtex-5 LX110	17,280	64	4,608 kbits	128 MB
PXI-7953R	PXI	Virtex-5 LX85	12,960	48	3,456 kbits	128 MB
PXI-7952R	PXI	Virtex-5 LX50	7,200	48	1,728 kbits	128 MB
PXI-7951R	PXI	Virtex-5 LX30	4,800	32	1,152 kbits	0 MB

These characteristics make the FlexRIO devices very flexible, and for that are commonly used for fast prototyping and testing. Also, FlexRIO devices are often used for high-performance data acquisition and processing system, like those used for big physics experiments, thanks to the parallelism capabilities of the FPGA and the fast data transfer using DMA. The flexibility of the platform is also very important in this last case, as it helps in the fast prototyping and development of new solutions or when the requirements of the system change

easily, for example in consequence of changes in the machine operation range or new optimizations discovered during research.

The proposed methodology only considers the use of FlexRIO devices without an embedded controller, where the control of the system is carried out by an external computer, although it could be partially applied to those whose embedded controller run a Linux distribution. The devices used for the methodology test and validation are the PXIe-7966R, part of ITER Catalogue for Fast Controllers; the PXIe-7965R, not part of ITER Catalogue but with characteristics similar to the PXIe-7966R. Regarding the adapter modules, the choice made for the validation of the methodology covers analog/digital input/output as well as image acquisition using CameraLink. The modules used to validate the methodology are the NI 5761R for analog input/output, the NI 6581 for digital input/output, and NI 1483 for CameraLink image acquisition. These modules were chosen because they are part of ITER Catalogue for Fast Controllers, but the methodology is equally valid for any other analog/digital input/output or image acquisition modules available, as it is oriented to the functionality, not to a specific adapter module or FlexRIO device.

3.2.1.1.2 CompactRIO devices

CompactRIO devices are rugged FPGA-devices designed for industrial control and monitoring systems. These devices are tested and validated to work with high reliability even in harsh environments with industry certifications for thermal, electromagnetic compatibility, and shock and vibration. A system based on CompactRIO technology is composed by a chassis integrating an FPGA and multiple slots for input/output modules (typically 4, 8, or 14 depending on the model). CompactRIO systems are intended to run unhandled, and the system controller typically only perform monitoring activities. For the system controller there are different options, as depicted in Fig. 27. Some chassis come with an embedded controller running a real time OS and others can be connected to an external controller using USB, Ethernet or MXI connections. The specifications of different CompactRIO chassis can be seen in Table 3. About the input/output modules, apart from the kits to develop custom modules, there are commercial modules for analog/digital input/output, several motor drives, interfaces for various industrial networks like PROFIBUS or CAN, and other communication

protocols like gateway for Wireless Sensor Network or serial interfaces using RS232, RS485, and RS422. There are much more input/output channels in a CompactRIO system than in FlexRIO devices, and the acquisition rate is slower. Therefore these systems are not designed for high-performance computing rather than robust controllers with high reliability. These devices are used for the control of industrial processes or the implementation safety systems as an alternative to PLCs [48].

For this methodology, we chose the CompactRIO architecture that is more similar to the FlexRIO solution without an embedded controller. That is a CompactRIO chassis without an embedded controller connected through an MXI cable to a host computer acting as the controller. That is also the configuration chosen by ITER for the Fast Interlock System.

Fig. 27 Schema of different systems based on CompactRIO

Table 3 Specification of different CompactRIO chassis

Model	Embedded Controller	FPGA Type	I/O Slots	Ports	Max Cable Length	Cable Throughput
cRIO-9033	Dual-Core Intel Atom, NI Linux RT, 8 GB Non-Volatile, 2 GB DRAM	Kintex-7 160T	4	3 USB 2.0 2 GigE 1SD 1 RS232, 1 RS485, 1 Mini DisplayPort	100 m/Segment	-
cRIO-9034	Quad-Core Intel Atom, NI Linux RT, 16 GB Non-Volatile, 2 GB DRAM	Kintex-7 325T	4	3 USB 2.0 2 GigE 1SD 1 RS232, 1 RS485, 1 Mini DisplayPort	100 m/Segment	-
cRIO-9035	Dual-Core Intel Atom, NI Linux RT, 4 GB Non-Volatile, 1 GB DRAM	Kintex-7 70T	8	3 USB 2.0 2 GigE 1SD 1 RS232, 1 RS485, 1 Mini DisplayPort	100 m/Segment	-
cRIO-9065	Dual-Core ARM Cortex-A9, NI Linux RT, 1 GB Non-Volatile, 512 MB DRAM	Artix-7	4	2 USB 2.0, 2 Ethernet 1 RS232	100 m/Segment	-
cRIO-9066	Dual-Core ARM Cortex-A9, NI Linux RT, 512 MB Non-Volatile, 256 MB DRAM	Artix-7	8	2 USB 2.0, 2 Ethernet 1 RS232	100 m/Segment	-
NI 9155	-	Virtex-5 LX85	8	2 MXIe	7 m	250 MB/s
NI 9159	-	Virtex-5 LX110	14	2 MXIe	7 m	250 MB/s
NI 9147	-	Artix-7	4	1 GigE, 1 USB 2.0	100 m/Segment	25 MB/s
NI 9149	-	Artix-7	8	1 GigE	100 m/Segment	25 MB/s
NI 9144	-	Spartan-3 2M	8	2 Ethernet	100 m/Segment	12.5 MB/s

3.2.1.2 GPUs

The proposed system includes a GPU for high-performance computing. In some cases the data process can be done in the FPGA, exploiting the parallelism capabilities of the platform, or in the CPU using the advantages of a fast processor. However, some processes are more suitable for GPUs, like the ones including massive floating point operations over big collections of data. We can find multiple examples of this kind of data process in machine vision, for instance in the calibration of images acquired from a camera is done by floating point matrices multiplication or in pattern recognition algorithms where the process is parallelized to speed it up.

For using GPUs, it is needed to connect them to a host computer, usually in a PCIe slot. The GPU Driver will allow us to manage the memory transfers with the GPU. The usual procedure is to transfer the data from the host memory to the GPU memory, process the data in the GPU, and then copy the results from the GPU to the host. However, GPUs supporting RDMA transfers can also receive data directly from other devices connected to the PCIe bus, as explained in the section 2.2.3. The use of GPUs with RDMA in the system can effectively reduce latencies in the process of the data, as the data could be transferred directly from the FPGA device to the GPU. This technology is only available in some professional grade GPUs, as is the case of NVIDIA Tesla and Quadro cards and AMD FirePro cards. In order to use this technology, the acquisition device driver has to call the GPU driver to obtain the GPU memory addresses to configure the data transfer. This feature is not commonly implemented in data acquisition devices, and therefore it requires to have the source code of the device driver to modify it and include the calls to the GPU driver.

For the system used in this methodology we have chosen NVIDIA cards, more specifically the NVIDIA Quadro K420 GPU card as it is included in ITER Fast Plant Controller Catalogue. The developments done are independent of a specific card, so any of the NVIDIA cards of Tesla or Quadro family can be used. The software developed is not compatible with AMD cards, as the modifications in the acquisition device driver were done only to use NVIDIA driver but in future versions this will also be available, as the only required changes are the GPU driver calls in the acquisition device driver.

3.2.1.3 Host Computer

In the architecture, the host computer will perform the tasks of the system controller. As system controller, the host computer must coordinate the acquisition and processing devices and configure them according to the parameters given by the central unit. In some cases, the CPU can also work as processing unit, performing partially or totally the data process required. In case that data archiving is required, the host computer will be in charge of collecting the necessary data from the acquisition device or the processing device. The collected data must then be stored in the media connected to the computer or sent to the archiving system using the designed network.

For the proposed methodology any standard computer with a 64-bit and PCIe slots for the GPU and the MXIe/PCIe interface is valid. However, the specifications of the computer may impact on the performance of the system. The methodology also requires the installation of a 64-bit Linux OS. Limitations in the election of the OSs are imposed by the NI drivers for FlexRIO and CompactRIO devices, and for the NVIDIA CUDA framework. In the next chapter is described the complete list of the OSs where the methodology was tested as well as the specifications of the computer where the validation and testing were performed.

3.2.2 Development Tools

Once the hardware components of the system have been described, the next step is to introduce the development tools that the users will need to develop their specific data acquisition and processing systems according to the proposed methodology. The development tools, depicted in Fig. 28, are composed of the tools to develop and test the FPGA bitfiles for the acquisition hardware (NI LabVIEW), the tools to develop GPU applications for NVIDIA cards (CUDA), and the application to create and run GUIs to control and monitor de variables of the system through EPICS (CSS). Several templates of every layer or part of the system are provided, as well as ready-to-use examples. With these templates and examples, the most common and simple use cases are covered. However, the development tools will be needed, as the users may need to modify the given templates and adapt them to their specific requirements.

Fig. 28 Schema of the development tools used in the methodology

3.2.2.1 Graphical Hardware Description Tool: LabVIEW FPGA

As explained before, FPGA devices are configured using a bitfile. The bitfile configures the different logic blocks and the connections among them. The bitfile is obtained from the synthesis of a hardware design written in a HDL, like VHDL or Verilog. The synthesis of the hardware description into a bitfile is done by a tool provided by the manufacturer, which translates the user designed functionality in terms of the resources available in the FPGA. However, the use of these languages is complex and it requires a long experience to master. This reason motivated the use of tools that translate a subset of higher level languages, like C, to HDL for being later synthesized. The use of these tools simplifies the development process for FPGA devices, while HDL languages can still be used for critical parts where the HDL offers more possibilities to define the hardware at the lowest level. For the methodology proposes in this thesis, these tools are a better alternative than HDL languages, as they provide a shorter and simpler development cycle where the users do not need to have expertise in hardware designs using HDL. In the case of NI RIO devices, as FlexRIO or CompactRIO, the specification of the FPGA functionality is done by means of a visual programming environment called LabVIEW.

LabVIEW is a system design platform and development environment based on a visual programming language. The first version of LabVIEW was released in 1986 by National Instruments. The basic elements of LabVIEW are depicted in Fig. 29. In LabVIEW, both user interfaces and programs are made by dragging and dropping elements in different places. The front panel is used to create the GUI. In this panel we can place indicators to show the value of different variables of the program and controls to interact and modify its values. In the block diagram the program is “coded.” In the block diagram, the controls of the front

Front Panel

Front Panel Palette

Block Diagram

Block Diagram Palette

Fig. 29 Example of LabVIEW Front Panel, Block Diagram, and the drag and drop palettes

panel will appear as inputs of the program, and the indicators as outputs. Operations over the data are done using blocks connected through wires. Each block can have input and output nodes to receive the data required and retrieve the result of the process. The data come through the wires. Therefore the wires are a representation of the data flow from one block or variable to another. The basic data types available in LabVIEW are the usual of a programming language, from booleans and different number representations including integer, floating point, and complex numbers of different sizes, to complex types to represent file streams, errors, or compound types like arrays and clusters. Together the block diagram and the front panel conform the Virtual Instrument (VI). A VI can include another one in its block diagram, and thereby allow for modularization and hierarchization.

LabVIEW includes a great number of functions oriented to create programs for data acquisition, instrument control and industrial automation compatible with multiple OSs. LabVIEW also includes multiple ready-to-use examples and templates to control and manage a wide variety of industrial devices and instruments, including but not limited to the hardware manufactured by National Instruments. In the case of the RIO devices (R-Series, FlexRIO, and compactRIO), LabVIEW with its module for FPGA is the only way to generate the bitfiles to configure the device FPGA, and this module is only available for Windows systems at the moment. The development process for RIO devices using LabVIEW FPGA is depicted in Fig. 30. Using LabVIEW FPGA the user can define the behavior of the FPGA using the graphical language. The user design will be translated to VHDL and compiled used Xilinx tools. This process is done by LabVIEW transparently to the user. Programming in LabVIEW can be an easier and faster way to obtain an FPGA bitfile, especially for users not experienced with HDL languages. However, in LabVIEW for FPGA designs is also possible to reuse modules developed in HDL languages including them in the FPGA VI. It is also possible to add Xilinx and third-party IP cores as well as constraint files.

However, apart from being used to define the FPGA behavior and obtain its bitfile, LabVIEW will also be used in the proposed methodology for first-stage testing of the FPGA bitfile. The testing can be done by directly interacting with

Fig. 30 Development schema for FlexRIO and CompactRIO devices using LabVIEW FPGA.

the FPGA VI front panel or developing a host VI that will run on the host computer. In the latter case, the host VI will access the FPGA resources using a special element called FPGA-node. Using this node the user can read and write data from/to the FPGA registers, as well as send or receive data using DMA transfers. The FPGA bitfile can be tested either simulating the FPGA device or directly downloading the bitfile to the target device. In both cases, we can also simulate the FPGA inputs using LabVIEW testing and simulation resources.

3.2.2.2 GPU Development Environment: CUDA

The Compute Unified Device Architecture (CUDA) is the parallel computing platform and programming model developed by NVIDIA. Using CUDA's language, which is a superset of C++, it is possible to develop programs using the massive parallel computing capabilities of NVIDIA GPUs. CUDA Toolkit includes all the necessary tools to develop CUDA code, including the NVIDIA's CUDA Compiler (NVCC), but there are other CUDA compilers developed in collaboration with NVIDIA for languages such as Python or Fortran. Included in the CUDA Toolkit there are several optimized libraries for high-performance computing. Some examples of these libraries are cuFFT, for Fast Fourier Transforms; cuRAND for random number generation; or NPP for image and video processing. CUDA also includes support for other parallel programming frameworks as OpenACC[3] and OpenCL[2].

From the developer point of view, the GPU is seen as a set of multiprocessors capable of running multiple threads executing the same operations over different data. This interpretation of the GPU architecture eases the integration of GPU for general purpose computing, as no intensive knowledge on the GPU architecture is needed, even though been aware of the GPU resource distribution and architecture can severely reduce computation time on GPU. For this purpose, CUDA Application Programming Interface (API) is formed by low-level and high-level functions, so experienced users can develop more optimized code profiting of their advanced knowledge using the low lever API accessing directly to driver functions, while others can still speed up the processing using the high-level API and CUDA libraries.

3.2.2.3 Human Machine Interface Development Tool: Control System Studio

Control System Studio (CSS) is a collection of tools to monitor and operate large control systems based. It is a product of the collaboration among different laboratories and universities and it is used in several big physics experiments, like the Spallation Neutron Source (SNS), the Facility for Rare Isotope Beams (FRIB), the National Synchrotron Light Source II (NSLSII), the German Electron Synchrotron (DESY, Deutsches Elektronen-Synchrotron), and ITER. CSS is based on Eclipse, and it is licensed under the Eclipse Public License v 1.0. [49].

Included in CSS there is a tool for the creation of Operator Panel Interface (OPI) called Best OPI Yet (BOY) (Fig. 31), from where the plant Process Variables (PVs) can be monitored and controlled; the Beast Ever Alarm System Toolkit (BEAST), a distributed alarm system to monitor alarm state, establish alarm conditions, and logging capabilities; and tools to view live or recorded data statistics and history. Currently, some of the CSS tools only supports EPICS control system, but the CSS community is continuously developing new functionality, and it is expected that it will fully support other control systems like TANGO[50] or TINE[51].

Fig. 31 Example of BOY OPI developed with CSS.

3.3 Products developed for the methodology

3.3.1 IRIO Software tools

In the previous sections, the hardware devices and development software have been introduced. These elements are commercial or free elements developed by companies or research centers. However, these elements are unconnected by themselves, and middleware is needed to integrate the hardware elements in EPICS. Another fact to consider when using reprogrammable devices, like FPGA devices, is that if the device configuration changes the interface with the control system may need to change also, and this consumes much time and makes more difficult the maintainability of the system. IRIO Software tools were designed to solve this problem. IRIO eases the integration in EPICS of data acquisition and processing systems based on RIO technology. IRIO also automatically detects what it is implemented in the FPGA and provide access to the implemented resources to EPICS.

IRIO is composed of a set of design rules for LabVIEW FPGA, a collection of templates and examples following the design rules that cover the most common use cases, and C/C++ libraries to handle and control FlexRIO and compactRIO devices configured according to the design rules. IRIO can be used to manage the FlexRIO and compactRIO devices through a custom user application or EPICS. For the latter case, IRIO counts with EPICS interfaces using asynDriver, areaDetector, and NDS. The complete set IRIO Software Tools and its layer organization is depicted in Fig. 32.

Fig. 32 Diagram of IRIO Software Tools for the integration of RIO Devices and GPUs in EPICS.

In [52] it was demonstrated the viability of an intelligent data acquisition system directly integrated EPICS with a proof of concept of how to connect an EPICS device support implemented with asynDriver with the resources implemented in the FPGA. This work is a re-engineering and improvement of the techniques and technologies used for the integration of this kind of systems in EPICS. The result a complete methodology were the user effort can be focused only on the

data process to be done. The main contributions of this work with respect to previous work can be resumed as:

- Establishment of a layer structure, separating the functionality of the core library and the different libraries for interfacing with EPICS using asynDriver (IRIOAsyn) and NDS (IRIONDS). The EPICS interface libraries have dependencies with the core library, but each library is installed as a separated package.
- Addition of a new library for the interface of IRIO with areaDetector (IRIOAD).
- Modularization of the core library: the core library, now called IRIO Library now is modularized as several resource handlers, one for each kind of resource supported by IRIO: analog resources (irioHandlerAnalog), digital resources (irioHandlerDigital), signal generators (irioHandlerSG), DMA to host (irioHandlerDMA), and image acquisition resources (irioHandlerImage). There are three additional modules: the main module (irioDriver), handling the acquisition profiles and its dependencies with the resource handlers; the resource finder (irioResourceFinder), used by the resource handlers to search in the bitfile whether or not is a specific resource implemented in the current bitfile; and the module containing the data types and structures used in the core library (irioDataTypes).
- Added a new module to the core library to handle the errors in a user-friendly way (irioError). Errors handled include IRIO errors, NI-RIO Driver errors, and custom upper layer errors.
- Added a new resource handler to the core library allowing DMA to GPU (irioHandlerDMAGPU).
- Added two new profiles (coreDAQGPU and coreIMAQGPU) using the new DMA to GPU resource.
- Added conditional compilation of the GPU profiles to save from the NVIDIA CUDA Drivers dependencies in case that they are not installed.
- Integrated GPU CUDA program compilation for IRIONDS projects.
- Enable compilation for non-CODAC systems. At the moment it is demonstrated to work on vanilla CentOS, RedHat, and Scientific Linux.

- Implemented a VHDL version of the PTP Synchronized clock for FlexRIO devices.
- Support implemented for I/O Interrupt Scan for analog and digital PVs in irioAsyn.
- Verified and validated the libraries using static and dynamic code analyzers.
- Elaboration of the documentation, including user manuals and test plans for IRIOCore and IRIOAsyn.

The IRIO software tools are licensed under GNU General Public License(GPL) v2 licensed and registered by the Universidad Politécnica de Madrid. IRIOCore and IRIOAsyn are publicly available on Github [53], whereas IRIOAD and IRIONDS are currently available only on demand, but there are plans to include all the modules as public repositories.

3.3.1.1 IRIOCore

IRIOCore is composed of the set of design rules, templates, and examples for LabVIEW FPGA, and the IRIO library. This layer of the IRIO Software tools solves the problem of the automatic recognition of the resources implemented in the FPGA device and offers a data acquisition-oriented API to the upper layers or user custom applications.

The typical development cycle for data acquisition and processing systems based on reconfigurable FPGA devices is depicted in Fig. 33. First of all, the FPGA developer implements the device functionality according to the specifications. The FPGA implementation will use a set of input/output registers and other resources to enable the configuration of the data acquisition, and the data communication with the host controller. In order to integrate the FPGA device in the system, a software engineer must develop a custom driver for the FPGA device, taking into account the input/output registers and other resources used in the FPGA implementation. If this is meant to be controlled by a distributed control system, this custom driver must also link all the configuration and data acquisition commands with the distributed control system. All this is a necessary but time-consuming task. However, if for example some error is detected in the FPGA design or the system requirements change, the new FPGA implementation

Fig. 33 Typical development cycle for systems including reconfigurable FPGA devices

may have a different set of registers and resources, forcing to adapt the software driver to use the new set of registers.

In the case of RIO devices, the manufacturer provides a C driver to interact with the device, called NI-RIO Driver. This driver provides an API for downloading a bitfile, get running the FPGA, and read/write operations to FPGA I/O (terminals) resources. However, to perform the read-write operations the user must know the type and address of the resource. For this, the manufacturer provides another tool called NI FPGA Interface C API Generator. This tool takes a bitfile as input and produces a C header file with a relation between terminal names (labels), types and addresses, as depicted in Fig. 34. The problem when using this approach is that in the case of recompiling the bitfile, even if the names and types of the register are not changed, the address can change, and so will change the bitfile signature used to validate a header file against a specific bitfile when downloading it. Therefore, the old header file will not work with the new bitfile, and we will need to generate a new bitfile and recompile the whole program, even if nothing changes on it apart from the register addresses.

Fig. 34 Overview of the relationship among the IRIO Library calls, the NI-RIO Linux Device Driver API, and the LabVIEW FPGA implementation

In the proposed methodology, IRIOCORE is used to solve the two problems previously mentioned. Part of the solution comes from the IRIO Design Rules for LabVIEW FPGA [54]. The design rules define a set of device profiles for the CompactRIO and FlexRIO devices. Table 4 and Table 5 show the device profiles defined in the IRIO Design Rules Document. Each device profile is oriented to a certain device functionality and reachable performance, considering the platform used. For example, for the compactRIO platform there are profiles for DMA and non-DMA based data acquisition (Table 4), as due to the characteristics of the platform the fast acquisition of data using DMA may not be necessary, and reading periodically from FPGA registers can be more adequate. On the contrary, in the FlexRIO platform the data acquisition is always done by DMA, as they are a faster device with a higher bandwidth. The profiles of FlexRIO are divided using two criteria, if it is data acquisition or image acquisition, and if the data will be set to the host controller or directly to the GPU. Combining the two criteria, we obtain the four possible profiles (Table 5).

Table 4 CompactRIO Device Profiles

DevProfile	Info
0	Data acquisition
1	Point by Point acquisition (PBP)

Table 5 FlexRIO Device Profiles

DevProfile	Info	Data acquired are sent to
0	Data acquisition	CPU Memory
1	Image acquisition	CPU Memory
2	Data acquisition	NVIDIA GPU Memory
3	Image acquisition	NVIDIA GPU Memory

These design rules also establish the convention of the name and type of the registers and resources to be used in the LabVIEW FPGA projects. The purpose of this is to make possible the automatic recognition of the resources implemented in the FPGA, as the name of the resources has been previously established. The design rules also define the functionality of a register with a certain name or the meaning of its values. Additionally, there are defined a collection of general purpose input/output terminals with a specific functionality defined. These terminals can be used to control custom functionalities or resources implemented in the FPGA that are not considered in the design rules. Every profile defines its own set of mandatory resources and optional resources, and the functionality of each one. There are also resources that are mandatory for all the profiles, or for a specific platform. Table 6 shows examples of these rules for the name and type of resources for all the cases. About the DMA data transfers, worth to mention that the design rules propose a model based on channels, channel groups, and data blocks. Every channel is a data source, as it could be an analog input. Each channel can be sampled at a different rate and have different byte-length. Samples with the same sampling rate and length can

be grouped in a channel group. All the samples from a channel group will be sent in data blocks to the corresponding host or GPU using a single DMA resource. As the DMA used are 64-bit wide, the block size must be one where the same number of samples for each channel fit in it.

Table 6 Reproduction of part of a table from the IRIO Design Rules document defining the FPGA resource names, types, and functionalities

Terminal Name	Data type	Type	Detail	Information	Values	Initialize before Run?
Platform	U8	Indicator	This terminal defines the form factor used in the FPGA implementation	Mandatory	0- FlexRIO 1- cRIO 2- R Series	YES
Common Terminals for FlexRIO						
FPGAVIversion	Array U8	Indicator	Contains the VI version, two elements. One for MM major version, and the next one mm minor version. MM.mm	Mandatory	For instance 1.1 FPGAVIversion[0]=1 FPGAVIversion[1]=1	YES
InitDone	Boolean	Indicator	This terminal must be set to true when the FPGA is initialized	Mandatory	True=OK False=NOK	N/A

IRIOCore includes examples of LabVIEW FPGA projects following these design rules. The examples cover all the different profiles and devices and show how these design rules can be followed to implement certain functionalities. The examples are completely functional, and they are ready to be compiled using LabVIEW FPGA and used for a data acquisition system. It is possible as well to modify an example to add some functionalities or modify the existing implementation to adapt it to the user requirements, therefore using the example as a template. There are also simple templates available with only the minimum

resources and functionality that can serve as a start point for custom implementations. The only requirement is to follow the previously mentioned design rules. Both the examples and the templates are well documented, and the LabVIEW design is organized in a way that the common functionality of a data acquisition system are easily identifiable (lecture of the sensor, conversion, decimation, data process, data forwarding to the host/GPU). This makes easier its modification and adaptation to different system requirements.

The other part of IRIOCore is the IRIO library. IRIO Library is based on the NI-RIO Linux Driver, and it provides a high-level API oriented to data acquisition to the resources implemented in the FPGA. IRIO library handles the complete execution cycle of the FPGA: download of the bitfile in the FPGA, run the FPGA, provide access to the resources, and close the FPGA session. In order to do this, the IRIO library must be initialized giving it the following parameters: bitfile path and name, header file path and name, and the serial number of the RIO device to use. The device serial number helps the IRIO library to locate and identify the device, and in case that the device is not found, it will raise an error informing about it. The resources present in the FPGA are automatically recognized by the library if the IRIO Design Rules have been followed in the development of the FPGA bitfile. This recognition is done by parsing the header file provided by the NI FPGA Interface C API generator, instead of including it in the compilation. The header file to parse is indicated to IRIO library by its name and path. This implies that as long as the name and path of the header and bitfile are properly passed to the IRIO library, there is no need to recompile the program. It also detects the device profile and checks that all the mandatory resources are present. In the case of any error with name or type of the resources, or due to missing resources, the library will report it before downloading the bitfile, informing of all the problems detected. In the case of errors when downloading or starting the FPGA, or in the access to the resources, it will show a message with the available information about the error and the manufacturer error code. The meaning of this error code can be consulted in the manufacturer documentation.

In resume, IRIOCore can be used to ease the development and maintainability of data acquisition and processing system thanks to the established convention in the LabVIEW FPGA projects and the library solving the resource recognition and

access using a high level and data acquisition-oriented API. This API can be used from a user custom program or through EPICS using one of the already implemented interfaces that are described below. Currently, this methodology is only valid for NI RIO devices (FlexRIO and compactRIO), but the possibility to port this solution to other platforms, devices and manufacturers is under study.

3.3.1.2 IRIO EPICS Interfaces

The other part of the IRIO Software tools is the IRIO EPICS Interfaces. The IRIO EPICS Interfaces are EPICS Device Drivers using IRIOCore and different EPICS Device Driver frameworks. Currently, there are developed three different interfaces with EPICS, named IRIOAsyn, IRIOAD, and IRIONDS, which use asynDriver, areaDetector, and NDS EPICS Device Driver frameworks respectively. Each interface has its characteristics, so the user must choose the interface that best fits the user needs.

All of the IRIO EPICS Interfaces work under the same principles. They rely on the IRIO Library for the FPGA resource recognition, bitfile download, start/stop running the FPGA, and resource handling. The access to all the resources implemented in the FPGA is done through EPICS PVs, as depicted in Fig. 35. For each interface, FPGA profile, and FPGA resource defined in the design there is a PV or set of PVs that grants the control of the resource. Templates for each resource type are provided to facilitate the configuration of the EPICS PV database, so it is only necessary to instantiate as many of them as resources of that type are present in the FPGA. This configuration must be done manually modifying an EPICS record substitution text file before running the IOC, but the modification of this files does not require recompilation of the IOC. There is no problem if there are more PVs instantiated than resources in the FPGA, it will just return errors when trying to get values or put values to the spare PVs. In the opposite case, if some FPGA resources lack of their representative PVs there will be no errors, it just will not be possible to control that resources from EPICS.

The templates provided for each IRIO EPICS Interface are different, as the specific PVs and PV names associated with a specific resource vary from one interface to another. These variations are caused because both areaDetector and NDS define a set of PVs for the different functionalities of the acquisition system, and this nomenclature was follow in the implementation of the device drivers. In

Fig. 35 Schema of the connection between the EPICS PVs defined in the templates and the resources implanted in the FPGA through the IRIO EPICS Interfaces

the case of `asynDriver` there is no nomenclature defined, and the names of the PVs are directly the name of the resources defined in `IRIOCore`.

The data is acquired from the device using the input PVs. The input PVs can be configured in I/O Interrupt mode or periodic scan mode. In periodic scan mode, the EPICS IOC will make periodical calls to the EPICS Device Driver for getting an update value of the PV. In the I/O Interrupt mode, the EPICS Device Driver is the one that triggers the update of a PV value, usually based on an event. This is the case of the data acquired from DMA in IRIO EPICS Interfaces. There should be one waveform PV per data channel in the FPGA, and this PV will always be configured as I/O Interrupt. Before running the FPGA, one thread is created per channel group and another one per channel. The channel group thread will read the data blocks from the DMA and send to each channel thread its corresponding

data. The channel threads will read the data sent by the channel group thread and publish it in the corresponding waveform PV.

In order to use one of the IRIO EPICS Device Drivers in an EPICS application is necessary to have installed both IRIOCore, the IRIO EPICS Interface, and asynDriver. The use of IRIOAD and IRIONDS also requires to install areaDetector and NDS respectively. After that, it only requires to create a folder in the EPICS application with the bitfiles and header files to be used, create the previously mentioned record substitution files, and include the device and database creation call in the IOC script. The device creation call will include the bitfile and header file paths and names, the serial number of the device, and a name to identify the device in EPICS. The EPICS name of the device must also figure in the record substitution files. The control of the device can be done through the instantiated PVs. At this effect, CSS BOY Panel templates are provided for each interface and device profile. In these templates there is at least one instance for each possible resource. These templates can be modified and adapted to whatever resources are implemented in the FPGA.

Although the previously described behavior is equally valid for all the IRIO EPICS interfaces, there are other characteristics that are specific to the interface chosen. These characteristics are presented in the following subsections.

3.3.1.2.1 IRIOAsyn

IRIOAsyn is the IRIO EPICS Interface base on asynDriver, and it is the interface selected by ITER for the implementation of data acquisition systems not related to diagnostics. IRIOAsyn supports all the devices profiles defined in the IRIO Design Rules for CompactRIO and FlexRIO, except the GPU profiles for FlexRIO.

One of the particularities of IRIOAsyn is the possibility to use I/O Interrupt mode with the analog and digital PVs, not only with the waveform PVs used for the DMA data. This functionality was implemented as a requirement from ITER. The problem that motivated this development is that the time interval for the periodic scan PVs is configured changing one of the record fields. This change can be done in the record substitution files or during the execution. However, the interval values are limited and predefined by EPICS to 0.1 s, 0.5 s, 1 s, 2 s, 5 s, and 10 s. These values can be modified changing the record field definition, but this can

be complex for inexperienced users. In the case that a faster sampling rate is required for analog or digital input PVs, the scan mode can be configured to I/O interrupt. In this mode IRIOAsyn will create two additional threads, one for the analog input PVs and one for the digital input PVs. Each thread will periodically read the value of the analog and digital resources associated with the PVs configured in I/O Interrupt mode and publish its values in EPICS. Each thread has associated an additional PV to configure the sampling rate of the thread. These PVs can take any value in the range of 10 – 1000 samples per second.

Another requirement for its use in ITER was the full integration with ITER Self-Description Data (SDD). SDD is the tool selected by ITER for the deployment of EPICS applications in the Instrumentation and Control Plants. The integration with SDD is done through a set of XML files to describe the hardware device, the EPICS Device Driver and the set of record templates to be used. Once these files are imported in the SDD database, the user can use the SDD GUI to configure and generate the IOC, including the creation and configuration of the record substitution files. The only task left is to copy the bitfiles and header files to be used by the driver and include the calls for the instantiation of the record substitution files on the IOC script.

Similarly to IRIOCore, IRIOAsyn comes with ready to use examples for analog, digital, and image acquisition using FlexRIO platform, and point-by-point and DMA-based acquisition for the compactRIO platform. These examples are grouped in a package called IRIOAsyn-examples and organized as different EPICS IOCs. Each example includes the LabVIEW project, bitfiles for the different FlexRIO targets, the IOC script already configured to configure the corresponding record database and instantiate an IRIOAsyn EPICS Device Driver, and GUI based on BOY to control the device. There is also an IRIOAsyn Device Support User Manual [55] to guide the users through all the features and functionalities of the software, including its installation and configuration.

3.3.1.2.2 IRIOAD

IRIOAD is an EPICS Device Support based on areaDetector. IRIOAD makes use of the areaDetector driver structure and the predefined set of PVs to control an image acquisition system based on FlexRIO and CameraLink. Currently, the use of CompactRIO or analog/digital data acquisition with FlexRIO are not

supported by IRIOAD. This lack of support is because originally areaDetector was developed only for image acquisition, and only the newer versions of areaDetector add support for 1-D data acquisition.

The implementation of IRIOAD is based on the use of NDArrays. The NDArrays are the data structure used by areaDetector to handle the data acquired. The use of these structures allows the creation of a processing chain with areaDetector plugins, as explained in the section 2.3.1.2. Additionally, IRIOAD supports the implementation of the PTP-synchronized clock in the FlexRIO devices for timestamping the images acquired. The details about this feature are detailed later in this chapter.

3.3.1.2.3 IRIONDS

IRIONDS is the IRIO EPICS Device Driver based on Nominal Device Support (NDS) [12]. IRIONDS uses NDS driver structure to control the acquisition device. IRIONDS extends the NDS base classes adding the calls to IRIO library to fully support all the resources implemented in the FPGA according to the IRIO Design Rules. IRIONDS device class manage the IRIO library initialization, bitfile download, FPGA start, and other characteristics affecting the complete acquisition as the adapter module configuration. IRIONDS channel group manages the configuration common to a set of channel sharing a DMA resource, as acquisition start/stop, sampling rate, and decimation factor. A thread created in the channel group will read the data coming from the corresponding DMA resource and send to each channel its data. The channels will receive the data from the channel group and publish it in the designed PVs. There are IRIONDS channel implementations for analog/digital acquisition and image acquisition. There are also IRIONDS channel group implementation for DMA transfers to host or to GPU. There is not special implementation for the case of the point-by-point acquisition, as in this case it just does not use the DMA resources and the data is published through periodic scanned PVs. The record templates provided with NDS for the device, channel group, and channels have also been extended to add support for all the IRIO resources. This implementation also adds support to compile the GPU code as part of the module compilation. This support is done using the EPICS compile rules and adding new rules and configurations for the case of GPU code compilation. The integration of the GPU code compilation as

part of the project increase the maintainability of the IOCs developed using IRIO and its DMA to GPU features for data processing. IRIONDS also includes GUI based on BOY. There are GUIs for analog data acquisition, digital data acquisition, and image acquisition. As NDS base classes, IRIONDS classes, PV templates, and BOY interfaces can also be modified or extended to add support for new functionalities added by the user.

3.3.2 Supplementary Developments

3.3.2.1 NDS DAN plugin

As mentioned before, the archiving of the data acquired is one of the main requirements for the data acquisition systems. In big physics this is a challenging issue, due to the large number of different data sources and the high throughput of each individual system.

One possible approach for archiving is to store the data acquired temporary in an intermediate buffer implemented with volatile memories. The permanent storage of the data is done reading the data from the intermediate buffers in the time between operations. However, with this approach the size of the intermediate buffer increase with the duration of each operation shift, making in inviable for experiments running for long periods, as is the case of the W 7-X or ITER. These experiments require an archiving system capable of handling all the data acquired during the operation, independently of the duration of the operation.

In ITER, the archiving system is called ITERDB [56]. This system collects the data from two of the ITER networks. The first one is the Plant Operation Network (PON). This is the network used by EPICS for the plant control. Therefore, the archiving using the PON networks is based on the collection of the data published through EPICS PVs. However, EPICS is not designed for support the high throughputs of some systems, and some data may be decimated for its publication in the EPICS PVs. This limitation leads to the need of the second network, the Data Archiving Network (DAN). The DAN is a dedicated network based on 10-Gigabit Ethernet links, and it is specialized for archiving purposes. The use of the DAN network is based on the publisher-subscriber model [57]. The publishers are the data sources, i.e. the acquisition and processing systems, and

the subscribers are the different servers in charge of implementing the permanent storage of the data, which is done in HDF5 file format [58]. The DAN archiving is based on the use of the DAN API, a library designed for minimizing the data copies between the different data sources and consumers.

The student had the opportunity to collaborate on the project for the implementation of the DAN API, called “The Implementation of Data Streaming and Data Library.” Under this project, the student developed an NDS-based plugin to act as DAN publisher. The implemented plugin could be connected to any of the data channels of a NDS Device Driver. The DAN plugin uses the DAN API to publish all the data of the NDS channel to the DAN subscribers. For the validation of the plugin, a complete example of use was developed connecting the plugin to an IRIONDS image acquisition channel and tested using EDT CameraLink Simulator.

3.3.2.2 FPGA PTP-Synchronized Clock

The synchronization is an important issue in distributed systems like the ones used for big physics experiments. This kind of experiments requires that all the devices are synchronized with the same absolute time reference to coordinate the actions taken by the different systems. Relating the data acquisition systems, it is also important that the data acquired in them is timestamped according to this absolute time reference. A precise timestamp allows the proper correlation of the data acquired in the different places of the experiment. A solution for the synchronization and timestamping in the data acquisition system is integrated into IRIOAD and IRIONDS.

In the system architecture proposed by the methodology it is included a timing card (NI 6683) that can provide an absolute time synchronized with an external time reference using the PTPv2. The PTPv2 is likely one of the most accurate and flexible mechanisms of all of the different device synchronization mechanisms, and there are already several proposals that apply the PTPv2 to big physics experiments [59]-[61]. The PTP is a standardized protocol (IEEE Std 1588-2008 [46]) that provides time synchronization among standard compliant devices connected to a network. The standard defines the specifications for several networks, including Ethernet, DeviceNET, ControlNET or PROFINET. A clock hierarchy is composed of PTP devices connected to the network and can achieve

a synchronization precision of tens of nanoseconds between the defined grandmaster clock and the slave clocks. However, the timestamp mechanism of the timing card relies on an external computer to package the data together with the timestamp. This process may induce a high CPU overload when multiple channels are distributed among different devices and are sampled at a high sample rate.

In [62] it is proposed the implementation of a hardware clock in a FlexRIO device that provides an absolute time synchronized with the timing card housed in the same PXIe chassis, and this functionality proved to work for image timestamping in [63]. In the original work, the clock was implemented in LabVIEW, but in this new proposal, the implementation is done in VHDL and included in the LabVIEW project as Component-Level Intellectual Property (CLIP). The VHDL module prevents unexpected latencies in the clock module derived from LabVIEW design constraints while facilitating integration its integration in any other FPGA-based device that allows VHDL designs. The implemented clock is synchronized with the external timing card using the periodical events generated every second by the external timing card in two PXI trigger lines. Additionally, a set of registers of the FPGA must be written to provide the absolute time when the next event will occur. With this inputs, the implement clock corrects the offset of the time with regard to the external reference and corrects the drift of the internal oscillator, proving a time reference that is constantly synchronized with the timing card. The procedure to program the generation of events by the timing card and the written of the values in the FPGA registers must be handled by the system controller. This functionality is implemented in IRIOAD and IRIONDS by a thread that uses the timing card driver, *nisync*, and specific calls to the IRIO Library to generate the adequate conditions for the synchronization between the clock implemented in the FlexRIO device and the timing card.

The timestamping of the image is done by embedding the time provided by the hardware clock in the image as depicted in Fig. 36. The timestamp is embedded after the two first pixels of the image, as these pixels frequently contain a frame counter controller by the camera that is very useful to check the integrity of the acquisition. The timestamp used follows the two-field structure defined in the PTP standard. The first field is the secondsField, which is defined as the integer

portion of the timestamp in units of seconds. The secondsField is located at the 48 bits that follow the hardware counter (bits 16 to 63). The second field is the nanosecondsField, which is defined as the fractional portion of the timestamp in units of nanoseconds. The nanosecondsField occupies the 32 bits after the secondsField (bits 64 to 95).

Fig. 36 Representation of the timestamp embedded in an image using the PTP-synchronized hardware clock implemented in the FPGA

With this solution, the images can be timestamped with a precision of tens of nanoseconds. The synchronization error of the clock implemented in the FPGA was measured during 2000 seconds, and results are presented in Fig. 37. The measured error had a normal distribution with a standard deviation of 6.86 ns, a mean of 0.08 ns and a maximum offset of 33 ns.

3.3.2.3 SPAD Disruption Predictor

The student also had the opportunity to collaborate with the project “Análisis de Datos Basados en Aprendizaje Automático y Sistemas Inteligentes de Adquisición de Datos: Modelos Avanzados para Entornos de Fusión” (Data Analysis based on Machine Learning and Intelligent Data Acquisition Systems: Advanced Models for Fusion Environments) for the development of a disruption predictor and its integration in JET. Disruptions are one of the major problems in present tokamaks. This phenomenon is currently unavoidable, and it produces large thermal loads, strong electromagnetic forces, and runaway electrons that can severely damage the machine components. Disruption detrimental effects scale with the plasma stored energy, and this fact has to be carefully considered in the operation of current tokamaks (as JET, or KSTAR) as well as in the design and development of future tokamaks devices (ITER, DEMO). Therefore, disruption predictors are needed to apply the mitigation techniques in time. Several plasma disruption mitigation techniques have been developed and tested in current fusion devices, as massive gas injection, killer pellet injection, or Electron Synchrotron Resonance Heating injection. However, these techniques need to be triggered with enough time (> 10 ms in the JET case) prior to the disruption in order to be effective. This leads to the need of accurate and reliable disruptions predictors. Nevertheless, the physical phenomena leading to plasma

Fig. 37 FPGA synchronized clock offset distribution with respect to NI 6683 clock after 2,000 measurements (one measure per second).

Standard deviation = 6.86 ns; mean = 0.08 ns.

disruptions can be only partially explained using complex and non-linear models. This fact makes the development of such predictors very difficult.

In his collaboration, the student implemented the Single-signal Predictor based on Anomaly Detection (SPAD) for the JET experiment. SPAD is a real-time disruption predictor that learns the normal behavior of signal the beginning of a discharge and triggers a disruption alarm when abnormal behavior is detected. This method presents good detections results, comparable with predictors based on machine learning methods but without the need of training process. SPAD uses one of the signals also used for threshold based disruption prediction, the locked mode (LM) signal. This signal is closely related to the disruptive behavior, and the predictor shows a good relation between anomalies in the signal and the disruptions. The signal is sampled at 1kS/s and every 2 ms the latest 32 samples are processed. The predictor uses the time and frequency information of the LM obtained by means of a wavelet transform. In particular, the approximation coefficients of the Haar Wavelet transform are used. These coefficients are used as feature vectors. The representation of this vector as a point in a multidimensional space shows that in non-disruptive discharges and the non-disruptive phases of a disruptive discharge the feature vectors are distributed in a compact cluster, while in the disruptive phase of a discharge these vectors are far from the original cluster. Due to the obvious covariance present among the feature points of the cluster, the Mahalanobis distance was chosen to measure the distance of a point with respect to the centroid of the cluster. Finally, an outlier factor is calculated using statistical measures of the Mahalanobis distance during the discharge as standard deviation and mean. The disruption alarm will be raised when the outlier factor surpasses a certain threshold. This predictor was implemented using MARTE framework, the framework used for JET's real-time applications, and integrated into JET's system. This predictor together with the Advanced Predictor of DISruptions (APODIS), based on Support Vector Machine classifiers, are the only real-time disruption predictors implemented in JET apart from those based only on threshold cross detection. This more sophisticated detectors show improved disruption detection capabilities with regard to threshold based predictors.

This project is partially funded the Spanish National Research Plan 2013-2015 for fundamental research, and the stay was funded by the grant of the Spanish Ministry of Economy and competitiveness for predoctoral short-term stays in R&D centers in 2014.

3.4 Development and testing cycle

In the following subsections will be described the complete procedure to configure, develop, and control a data acquisition system using the proposed methodology. These steps are depicted in Fig. 38, and they are fully described in the IRIO Documentation, composed by IRIO Design Rules for LabVIEW FPGA, IRIO Library User Manual, and IRIO EPICS Device Support User Manual.

Fig. 38 Development cycle using IRIO Software Tools

3.4.1 Choosing the target

The first step implies to choose the hardware that will be used for the data acquisition and processing system. That is highly dependent on the requirements of the system. The factors to consider when choosing a target are the type of

interface needed, the number of input/output channels, the sampling/update rate of the channels, the data bandwidth between the target and the system controller, and if GPU processing will be required. Currently, the proposed methodology supports FlexRIO and CompactRIO platforms. The specific devices and modules supported are described in Appendix B, in Table 15, Table 16, and Table 17. Table 7 resumes these characteristics for the supported devices. Values shown are the best possible values for each characteristic obtainable with the supported hardware.

Table 7 Resume of the characteristics of the FPGA hardware devices supported by IRIO Software tools

Characteristic	FlexRIO	CompactRIO
Available Interfaces	<ul style="list-style-type: none"> • Transceiver: <ul style="list-style-type: none"> ○ Radio Frequency (RF) ○ Intermediate Frequency (IF) ○ Baseband • Analog I/O (up to 16 bit) • Digital I/O • CameraLink 	<ul style="list-style-type: none"> • Analog I/O • Digital I/O
Max. number of channels	32 Analog Inputs or 54 Digital I/O	32 channels (Analog or digital) per slot, up to 14 slots
Sampling/Update Rate	Up to 3 GS/s (Analog input), 1 Gb/s (Digital I/O)	25 kS/s Analog Input, 100 Mb/s (Digital I/O)
Data Bandwidth	1.7 GB/s	100 MB/s
GPU Processing Available	Using RDMA	RDMA not available

In resume, FlexRIO modules are intended to be used for fast, high throughput acquisition and processing systems where the data is sent to the system controller or the GPU for further processing, while CompactRIO is more adequate for systems requiring more but slower channels and the data is mainly consumed in the platform, as the host controller is used mostly for supervision.

3.4.2 FPGA bitfile development and testing

Once the target has been chosen, it is necessary to develop the bitfile with the desired functionality. The bitfile for FlexRIO and CompactRIO devices can only be generated using LabVIEW FPGA, and this IDE is only available for Windows. It is not necessary to connect the devices to the Windows computer to develop the bitfile. However doing this will allow testing the bitfile right in the development machine, and this will be probably faster than testing it in the Linux system controller.

The LabVIEW FPGA development for the acquisition device must comply with the IRIODesign Rules for LabVIEW FPGA [54]. The development can be started from a blank project, using LabVIEW FPGA wizards to create and configure a project for the selected target. Once the project is created, the FPGA developer must decide which of the device profiles defined in the design rules is more adequate taking into account the requirements of the system. After that, the mandatory resources of the profile must be created and configured according to the design rules, as well as rest of the functionality of the device, including the acquisition channels and the DMA data transfers if necessary. When the design is completed and compiled, it can be tested using front panel of the FPGA main VI. However, some of the features can only be tested developing another VI that runs on the host computer, as for example the DMA transfers. In this case, the host VI will manage the bitfile download, start the FPGA, configure the device, and control the resources.

This task can be complex for new users without experience with the proposed technology. At this effects, the IRIOCORE module includes several LabVIEW FPGA projects covering all the possible profiles and configurations. This projects are examples of FPGA designs for each profile and are ready to be used. If the requirements differ from the implementation, it can be modified easily with the advantages of departing from a correct implementation rather than a blank

project. These projects also come with a host VI to test the developed bitfile. These host VIs can also be adapted in order to test also the modification of the bitfile introduced by the user.

3.4.3 Integration in application

After completing the FPGA bitfile development and test the implementation, it is the turn to develop the application to control and monitor the data acquisition and processing. This methodology relies on the use of IRIO Software Tools, thus it will be necessary to install the tools. That step will not be explained here, as is a simple library installation procedure, but detailed instructions are provided with the tools. The other mandatory requirement is to produce the header file for the produced bitfile using the NI FPGA Interface C API tool.

The methodology is oriented to use EPICS as control system. The developed EPICS Device Drivers allow the user to control the implementation without any additional development. Any FPGA implementation done following the IRIO Design Rules for FPGA can be controlled using IRIOAsyn or IRIONDS, while IRIOAD can be used only for image acquisition designs. The configuration of the EPICS IOC in the three cases is similar. It only requires to include the appropriate EPICS Device Driver initialization call including the device serial number and the names and paths to the bitfile and header file, and the configuration of the record database according to the profile and resources implemented in the FPGA. These EPICS Device Drivers based on IRIO will manage all the procedure to download the bitfile, search the implemented resources, and associate the PV read/write operations to the appropriate calls to IRIO library. The GUIs based on CSS BOY provided with the IRIO modules are prepared to work with the PV names used in the templates also includes in the IRIO modules. In case that these names must be changed to adapt them to a certain name policy the GUIs also need to be modified, but the rest of the code remains unchanged.

However, the IRIO library completely independent from EPICS. This allows IRIO Library to be also used directly from a custom application developed by the user or from another control system framework if the corresponding interface is implemented. The use of IRIO library to manage the FPGA device requires following a sequence of specific steps. These steps are depicted in Fig. 39 and can be described as:

Fig. 39 Schema of an IRIO Library-based application for data acquisition

- Initialization of IRIO driver. This is accomplished with the `irio_initDriver` function. This function performs all the library initialization. The main operations implemented in the function are: download the bitfile to the FPGA, resources identification depending on the profile implemented in the FPGA and data structure initialization for later use by other API calls. This function returns a structure of type `irioDrv_t` that contains the mapping of all resources found in the FPGA.
- Use of getters and setters. The applications using the IRIO library can access to read/write the terminal available in the FPGA in order to know

the initialization value or to set this. All getters/setters use a similar prototype with these parameters in common:

- `irioDrv_t* p_DrvPvt`, this is the data structure for the RIO device.
- `TStatus* status`, this is the data structure containing the status and errors obtained once the function has been executed.
- If the application is using DMA (for image acquisition profiles or DAQ acquisition profile), the user needs to configure the DMA using `irio_setupDMAstoHost`.
- Execution of the hardware implemented in the FPGA. In the initialization the bitfile is downloaded, but it is not executed. This means that the FPGA is waiting for the software to send a run command. This run command is executed with the `irio_setFPGAstart`.
- Once the FPGA is running the application, the user can interact with the FPGA terminals using the setters/getters and can trigger data acquisition retrieving the data using the `irio_getDMATtoHost`. There are many more functions explained in the Library API part of this document.
- For closing the driver and release the resources, `irio_closeDriver` is required to be invoked.

3.5 Conclusions

This chapter presented hardware and software elements used to develop data acquisition and processing systems according to the proposed methodology. It was also presented the development cycle that should be followed to develop these systems.

By one side, the methodology is supported by commercial products, mainly the hardware devices (GPUs and FPGAs) and its drivers. The importance of using commercial products, especially in the case of the FPGA devices is that they are guaranteed to work under the harsh environmental conditions associated with big physics experiments, offering the required availability and reliability. It is important also to assure that the manufacturer will maintain the products used during the lifetime of the experiment. The use of FlexRIO devices results in a very flexible high-performance platform. The development process for this devices is also simplified by the use of a graphical language, which abstracts the user from the complexity of HDL languages used for FPGA designs. Moreover, the

provided templates solve the basic data acquisition, and the user is just required to implement the custom data process specific of the application. Some of the issues not solved by the manufacturer, as the timestamping in the acquisition device or the direct data movement to the GPU, were solved and integrated into the methodology.

The other part of the methodology are the software tools developed for the integration of the hardware in EPICS, and the development cycle defined by the methodology, that is supported by the software tools, templates, and examples developed. The developed tools provide a solid a consistent framework for the implementation of data acquisition and processing systems and the integration in EPICS. IRIO Library and the IRIO EPICS Interfaces allow an almost transparent integration of the implementation done in the FPGA with EPICS, where the user is only required to define the EPICS database and design the GUI. Moreover, even for these cases, record templates and GUI examples are provided to help the users in these tasks.

By using the proposed methodology, the user can be focused in the part that is really specific to his requirements, which is the custom data process. The data process can be distributed among the FPGA, the GPU, and the CPU. The process in the FPGA can be done in LabVIEW, even including VHDL modules. The only constraint imposed by the methodology is to follow a simple set of design rules which are basically name and type conventions. These rules will allow the automatic integration of any design implemented in the FPGA with EPICS. The implemented EPICS Device Driver are valid for all the possible implementations done in the FPGA following the design rules, which contributes to the maintainability of the system. Also, it is not necessary to recompile anything when the FPGA design changes. This feature provides a fast deployment process where the changes can be applied easily. This feature very relevant for some systems used in big physics experiments like the diagnostics, as the requirements of these systems are very likely to change during the experiment lifetime.

Finally, the developed solutions have been exhaustively tested and evaluated against the requirements expected for the PXIe data and processing acquisition systems of ITER. This evaluation will be presented in the next chapter along with some applications of the methodology proposed.

Chapter 4

4 EVALUATION AND RESULTS

4.1 Evaluation

In this section will be presented the evaluation of the methodology and its applications. The methodology is evaluated by the applications developed following the design rules, executed on the proposed hardware, and using the different software elements. Before specifying the aspects of the evaluation, the hardware and software architecture for the evaluation test will be defined in the corresponding subsections. This architecture is shared among all the evaluation test performed unless otherwise specified in the test definition. Three main aspects were evaluated: Functionality, Performance, and Integrity. Each aspect will be analyzed in a different subsection, and the setup used in every test will be defined together with the results obtained.

4.1.1 Hardware architecture

The hardware architecture to test the applications developed according to the methodology is based on the ITER Fast Plant Controller Architecture for PXIe Form Factor.

For the test involving FlexRIO devices the PXIe chassis used was an NI PXIe-1065. This chassis have up to 7 PXIe compatible slots with a maximum theoretical throughput of 1 GB/s per slot and 3 GB/s of the complete system. The different FlexRIO device models used were PXI-7952, PXIe-7961, PXIe-7965, and PXIe-7966. The adapter modules tested were NI 5761 and NI 5734 for analog data acquisition, NI 6581 for digital input/output, and NI 1483 for CameraLink image acquisition. The chassis was connected to the host computer using the PXIe-PCIe 8381 interface supporting up to 3 GB/s of sustained throughput. Additionally, connected to the chassis there was an NI 6682 timing card. The timing card is connected through Ethernet to a Hirschmann MACH1040 Gigabit Ethernet Switch capable of acting as PTP v2 Local Master Clock or Grand Master Clock.

For the test involving compactRIO devices, an NI 9159 14-slot chassis was used. The chassis is connected to the host computer using the PXIe-PCIe 8361 interface, supporting up to 250 MB/s of sustained throughput. In the validation, the following CompactRIO IO Modules are tested: NI 9205 (+10V analog inputs), NI 9264(+10V analog outputs), NI 9401 (digital TTL input/outputs), NI 9477 (60V digital sinking output), NI 9476 (36V sourcing output), NI 9425 (24V sinking input), and NI 9426 (24V sourcing input). To cover the wider amount of

scenarios possible, a connection model interconnecting the inputs and outputs of the CompactRIO IO Modules is proposed. The model is described in the Fig. 40 and

Table 8. This configuration covers: $\pm 10V$ analog I/O with the modules NI 9264 and NI 9205, TTL Digital I/O with the module NI 9401, 24V digital sourcing input and sinking output with the modules NI 9477 and NI 9426, and 24V digital sinking input and sourcing output with the modules NI 9476 and NI 9425. Additionally, power sources for the CompactRIO module are needed.

Two different host computers were used during the test of the system. One is a 1.3 PICMG 4U rackmount industrial computer with two quad-core Intel Xeon

Fig. 40 CompactRIO connection schema for testing

EC5549 at 2.53 GHz, 24 GB DRAM, and a RAID-1 SSD drive system. The other computer was a standard computer with an Intel Core i7-4790 CPU at 3.60GHz, 16 GB RAM, and standard 7.200 rpm HDD.

Regarding the GPUs used with the FlexRIO devices, two different NVIDIA Professional Grade GPUs were tested. The first is a NVIDIA Tesla K20c[64], featuring a Kepler architecture with 2496 CUDA cores distributed in 13 multiprocessors, capable of executing a maximum of 2048 threads per multiprocessor and with a total RAM memory of 5 GB GDDR5. The second is a NVIDIA Quadro M4000[65], featuring a Maxwell architecture with 1664 cores distributed in 13 multiprocessor, capable of executing a maximum of 2048 threads per multiprocessor and with a total RAM memory of 8 GB GDDR5.

Table 8 Description of connections among CompactRIO IO modules for testing

Interconnection of channels from modules of the NI9159 Chassis	
NI9264 AO0	NI9205 AI0
NI9264 AO1	NI9205 AI1
NI9264 AO2	NI9205 AI2
NI9264 COMM	NI9205 COMM
NI9401 DIO0	NI9401 DIO4
NI9477 DO0	NI9426 DI0
NI9477 COMM	Power Supply COMM
NI9426 Vsup	Power Supply Vcc
NI9476 DO0	NI9425 DI0
NI9476 Vsup	Power Supply Vcc
NI9425 COMM	Power Supply COMM
NI9476 COM	NI9425 COM

During the tests, additional hardware was used to generate the inputs of the system or measure the outputs. In the case of the image acquisition, three different image sources were used. Firstly, a PCIe8 DVA CameraLink simulator manufactured by EDT was used. This card is capable of stream images using CameraLink standard at 680 MB/s using CameraLink full mode. However, this card was discontinued by the manufacturer and was later substituted by Gidel's ProcCamSim PCIe CameraLink simulator that supports streaming images using CameraLink Deca/Extended mode, resulting in a maximum throughput of 850MB/s. Apart from the simulators, an EoSens 3CL CameraLink camera manufactured by Mikrotron was used in the test. The camera maximum resolution is 1680x1710 8-bit pixels, achieving a maximum throughput of 710 MB/s. For the case of analog and digital signal generator, a 20 MHz dual-channel Keysight 33512B signal generator [66] was used. The measure of the signals generated by the system was performed using an Agilent MSO7014A

oscilloscope [67], capable of sampling four analog channels and 16 digital channels in a 100MHz bandwidth with at 2 GS/s.

4.1.2 Software architecture

The IRIO Software Tools have been designed to work under different software environments. While most of the work is done under ITER CCS environment, tests were also done on other Linux platforms, like CentOS 6.5 and Scientific Linux 6.5. These OSs and versions were chosen for having the same Linux kernel version than CCS 5.4, the latest CCS version. However, the features available depend on the specific architecture. The reason behind this is the only mandatory dependency of IRIOCore, the NIRIO Driver. There are two different NIRIO Drivers developed by National Instruments. One of them is the driver available for all users, downloadable from NI website, and will be referred from now on as public NIRIO Driver. This driver supports most of RIO hardware manufactured by NI. The specific hardware supported depends on the specific version. Versions of this driver are released every year, corresponding to new LabVIEW versions released, and each version supports the three previous LabVIEW versions. There are two branches of the public NIRIO Driver, one for 64-bit Linux OS and other for 32-bit and 64-bit Windows OS. The specific OSs and versions supported also depend on the specific driver version. Starting from public NIRIO Driver version 15.0, each driver branch was divided in two different drivers, one for FlexRIO devices and another one for compactRIO devices. The latest version of public NIRIO Driver is 17.0. The other NIRIO Driver is an open source driver for 64-bit RHEL 6.x OS and will be referred from now on as ITER NIRIO Driver. This driver was developed exclusively for ITER under GNU LGPL v2.1. This driver has more limited hardware support, and there is no specific version release plan. According to functionality and hardware support, ITER NIRIO Driver will be best compared with public NIRIO Driver versions from 12.0 to 14.0. For this reason, the public NIRIO Driver version used will be 14.0, but IRIO has been tested using versions from 12.0 to 14.0. This driver version NIRIO Driver supports Scientific Linux 6.x, RHEL 6.x and 7.x, and openSUSE 12.x and 13.x. Regarding ITER NIRIO Driver, IRIO supports all ITER NIRIO Driver versions since CCS v5.1, but the version used for the test will be one in the latest CCS version, CSS v5.4.

Regarding the GPU support, this is only available when using ITER NIRIO Driver, as this feature was added by Prof. Julián Nieto et al. ([68], [69]) by modifying the ITER NIRIO Driver. Also, NVIDIA Drivers and CUDA framework are also necessary to support GPU handling. However, this dependency of IRIO is optional and can be disabled if the FPGA to GPU RDMA feature is not going to be used. For the evaluation test, NVIDIA Drivers version 367.48-1 and CUDA 6.5-14 was used.

The latest IRIO dependency is related with EPICS. Nonetheless, this dependency is limited only to the IRIO EPICS Interface libraries, which are IRIOAsyn, IRIONDS, and IRIOAD. The IRIO Core Library is completely independent of EPICS and can be used without any EPICS Device Driver by just linking with the IRIO Core Library from a standard C/C++ program. The EPICS framework is formed by an extensive number of complement with some interdependencies among them, dependencies that are also version dependent. Due to this complexity, it was decided to follow the same EPICS setup available at CCS 5.4 for all the tests. The EPICS versions used for the test are as follows: EPICS Base v3.15.5, asyn v4.31, busy 1.6.1, calc 3.6.1, seq 2.2.4 and areaDetector 2.4. NDS was only used in CCS, and it was used in the version 2.3.8, the one for CCS 5.4.

4.1.3 Functionality

This subsection will cover the three main functionalities of the IRIO software tools, which are: the automatic resource recognition and management, including data acquisition; the integration with EPICS; and the integration with SDD. At this purpose, several bitfiles were developed specially for these tests. In the case of the bitfiles for FlexRIO, all the bitfiles are available for all the FlexRIO devices mentioned in the **Hardware Architecture** section. Also, all FlexRIO test require as input the device model and the serial number of the target device so that the test can be run easily on any of the supported devices.

4.1.3.1 Automatic resource recognition and management

The tests described in this section were designed to validate the functionalities of the IRIO Core Library, that is, automatic resource recognition and management, including data acquisition. This features will be validated for the FlexRIO and CompactRIO platforms, including all the possible device profiles, and in the case

of FlexRIO, also the direct data transfer between FPGA and GPU for the case of image acquisition. These tests are performed using already compiled bitfiles and C programs using IRIO Core Library API. These programs and bitfiles are available on Github, as well as the LabVIEW projects from where the bitfiles were compiled. The bitfiles used have been already verified using LabVIEW environment. The test procedures for bitfile compilation and validation of the bitfiles are documented in the IRIO_Library_Test document, as well as step by step description of the same tests that are described below.

4.1.3.1.1 Resource Recognition

These tests are oriented to probe the correct resource recognition and identification. The LabVIEW projects used for this test are not functional in the sense that they do not perform any data acquisition operation, they are basically a definition of registers with the names according to the IRIO Design Rules. There are six projects for this test: four covering the four possible device profiles (CPUDAQ, GPUDAQ, CPUIMAQ, GPUIMAQ); one testing the error detection, with some mandatory resources missing; and one with the maximum number of each type of resource defined, using CPUDAQ profile. This test is carried out by the FlexRIO_onlyResources test program. When the test is executed, it downloads the test bitfiles mentioned above, indicating in any case what it is the purpose of every project, and how should look the output prompt and error messages from IRIO. The GPU test are given commented by default to prevent test failure in case that IRIO was not installed with GPU support.

4.1.3.1.2 FlexRIO data acquisition mandatory and auxiliary resources

The basic read and write operations of an FPGA resource in a FlexRIO device can be tested executing the FlexRIO_noModule test. The bitfile used in this test only has the mandatory resources for CPUDAQ profile, six auxiliary analog output registers (auxAO0-auxAO5), six auxiliary analog input registers (auxAI0-auxAI5), six auxiliary digital output registers (auxDO0-auxDO5), and six auxiliary input registers (auxDI0-auxDI5). Internally each auxiliary analog or digital output register is connected to the auxiliary analog or digital input register, e.g. the auxAO0 output value is connected to the input of auxAI0, and the auxDO1 output value is connected to the input of auxDI1. The first operation of this test is to download and start the bitfile, checking that no errors are detected

by IRIO. After that, the same operation is repeated over all the auxiliary analog resources: write a 0 in the output register, read the output register and check that the value returned is 0, read the input register and check that has the same value than the output register, and repeat all again but writing 100 in the output register. Then, the same operation is repeated for all the digital auxiliary registers but writing first a 0 and then a 1. After this, the FPGA device temperature is read from the appropriate register, which is mandatory for all device profiles, and is then converted to Celsius degrees and prompted. Detailed info on the execution and results obtained during the test including errors are prompted.

4.1.3.1.3 FlexRIO Image acquisition profile

There are three tests for the validation of image acquisition using the NI1483 CameraLink Adapter Module for FlexRIO. These tests require the connection of the NI 1483 to the FlexRIO target device, and connect a CameraLink camera or CameraLink simulator to the module. In the case of CameraLink simulator, it is also required to initialize the simulator and start the image streaming, as the NI1483 module requires the presence of the pixel clock in the CameraLink interface, and normally simulators do not generate the clock signal until the image streaming is programmed. The first test, `FlexRIO_mod1483-uart`, test the serial line interface implemented in the CameraLink cable. This test uses a fully functional bitfile with CPUIMAQ profile, but it could also work with GPUIMAQ profile as the terminals for the serial interface remain unchanged. In this test, after the bitfile download and start, read and write operations are performed over the CameraLink serial line registers defined in the IRIO Design Rules to check the initial status of the serial line and prompt it to the user. After that, a write operation is performed to ensure that the baud rate of the serial line it is set to 9600 baud/s. The next step is to wait for the user to introduce a message in the terminal. This message is read by the program and using the appropriate IRIO Core Library function sent to the FPGA CameraLink serial interface. The FPGA will automatically send the message to the camera or simulator. Then, the test program will try to read an answer from the FPGA CameraLink serial interface and prompt it to the user. The recommended setup for this test is to connect a CameraLink simulator with a utility to manage the serial line, as it is the case of EDT's simulator. If this is the case, using the serial line utility we should be able to see the message sent from the FPGA and send a response, which will be

prompted by the FlexRIO_mod1483-uart test. If a camera is connected instead of a simulator, then we should check the camera command manual, send a camera command message and check that the answer corresponds to the description in the manual. This test program can also be used to set the camera configuration for the next tests.

To test the image acquisition two different test programs are used, one for image acquisition sending the images with DMA to CPU and the other with DMA to GPU, FlexRIO_mod1483-Image-CPU and FlexRIO_mod1483-Image-GPU respectively. The bitfiles for this tests are also different. However, the only changes are device profile register and the names for the DMA resources, as stated in the design rules. The tests are also very similar, just requiring additional procedures in the GPU test. Both of them requires configuring the image source to send images of 256x256 pixels of 8 bit per pixel in CameraLink Full Mode (8Tap8) under 1000 frames per second. The values of the first two pixels of the image should represent an image frame counter, increasing the value by one in each image. This feature is commonly implemented by cameras and simulators and just need to be configured, as is the case of the EoSens3CL camera and EDT CameraLink simulator. In the test, after the bitfile download and start, the DMAs will be configured and the acquisition will be started. In the test, 1000 images will be acquired. The only process done to the images is to read the frame counter embed in the first two pixels and check that values received are consecutive, probing that no frames are lost. In the case of the GPU test, before downloading the bitfile, the program tries to locate and identify the GPU in the system and initialize it and allocate the memory to receive the images via DMA. In the image acquisition process, as the images are sent to the GPU, it is needed to copy the images from the GPU to the CPU before checking the frame counter. In both cases, the acquisition is finished after acquiring 1000 images, and the resources freed, including the allocated GPU memory for the GPU case. The results of the acquisition are shown. This test proved to work correct image acquisition without frame loss for a low throughput, i.e. maximum of 64MB. However, this does not represent the maximum performance of the system, which will be analyzed in detail in the next section.

4.1.3.1.4 FlexRIO data acquisition profile with analog signals

The next test case is for the validation of analog data acquisition using the NI 5761 adapter module connected to a FlexRIO device with CPUDAQ profile. For this test is not need to connect a signal source to the adapter module. The test program for this part is FlexRIO_mod5761. The FPGA project for this test is configured to perform the acquisition of 4 channels of 16 bit each one in blocks of 4096 samples per channel. The channels 0 and 1 (CH0 and CH1) of the acquisition are the AI0 and AI1 inputs of the adapter module respectively. CH2 and CH3 are the outputs of signal generators 0 and 1 (SG0 and SG1) implemented on the FPGA according to the IRIO Design Rules. In this test only the values from SG0 connected to CH2 will be checked. This way also the implementation of the signal generators is tested and the values acquired can be easily validated, without any possible error coming from misconfiguration of the external signal source. The values of SG0 and SG1 are also connected to the auxiliary analog input terminals auxAI9 and auxAI10. In this test, after the bitfile download and start, the signal generator is set to generate a pure DC signal of digital value 2048. To validate this configuration, firstly the auxAI9 register is read to check that the value read is 2048. Then, the DMAs are configured and the acquisition parameter set to sample the channels at 500kS/s per channel. After this, the acquisition is started, and one data block is read (4096 samples of every channel). Every sample of CH2 in the block is compared to 2048 to check the correct behavior of the acquisition and the signal generator. After acquiring and validating the samples, the acquisition is stopped, and the DMA buffers cleared in preparation for the next acquisition. For this new acquisition, the SG0 is configured to generate a sine wave of 10 kHz updating the value generated at 10 MS/s. The sampling rate is not changed and remains to 500kS/s, meaning that in 50 samples a complete signal period should be represented. Once the configuration is done, the acquisition is started again and one block is acquired. The first 60 samples acquired from CH2 are prompted, so the user can check whether or not a complete period is present in 50 samples. After the prompt, the acquisition is stopped and all the resources freed. With this test, the signal generation and the DMA acquisition of one data block data from signal generators are validated. The acquisition from the adapter module input channels is verified in the EPICS integration tests using this same bitfile but with a different test program.

Continuous data acquisition without data loss is validated in the performance test also using the same bitfile.

4.1.3.1.5 FlexRIO digital I/O management

The last test for the validation of basic data acquisition with FlexRIO is the test of digital input/output handling using the NI 6581 adapter module. In this test only the Digital Data & Control A (DDCA) interface of the adapter module will be used. It is necessary that the DIO port 2 is connected to DIO port 1. The recommendation is to do this with the help of the SMB-2163, as recommended by ITER I&C Cubicle internal configuration. In the FPGA configuration, DIO port2 is connected to digital output registers from DO0 to DO7, and DIO port 1 to digital input registers from DI0 to DI7. This test is run by executing FlexRIO_mod6581. In the program, after downloading and starting the bitfile, all the DO registers are changed, first to 0 and then to 1. In every change, the DO register wrote and the corresponding DI register are read. As DIO port 2 is connected to DIO port 1, the value wrote in DO0 should be the same value read from DI0. Once this has been done for the eight digital output registers, the resources are freed.

4.1.3.1.6 CompactRIO profiles

The validation for compactRIO data acquisition is done in two separated tests. The first one, cRIO_IO covers the point-by-point acquisition, acquisition based on the periodical reading of the registers without using DMA. In this test, after the bitfile download and start, all the mandatory resources are read, and its values prompted to the user. Then the acquisition is started, and tests are done over the analog I/O resources. The analog output resources from AO0 to AO2 are then set to 0 V. These resources correspond respectively to the terminals AO0 to AO2 of the NI 9264 module. To check the correct change of value, the values from AO0 to AO2 are read, as well as the values from AI0 to AI2, corresponding to the same terminals of the module NI 9205. If the connection scheme was followed, these analog outputs and inputs would be connected, and the value should be the same with a maximum error allowed of 0.002 V. Then the DebugMode is enabled. In this mode, the value of the analog outputs AO0 to AO2 is fixed to 1.25 V for AO0, 3.125 V for AO1, and -4.75 V for AO2, and so should be in the corresponding analog inputs. The test of the analog part finalizes assigning the

digital values to the analog outputs of 4000 for AO0, 8000 for AO1, and -8000 for AO2. The conversion of this digital values to volts is prompted to the user, as well as the values obtained from reading AI0, AI1, and AI2. For the digital terminals, the three digital outputs used, from DO0 to DO2, are first set to 0 and then to 1. These resources correspond respectively to the DIO0 from the module NI 9401, the DO0 from the module NI 9477, and the DO0 from the module 9476. According to the connection schema, this terminal should be connected in the same order to the DIO4 of the module NI 9401, assigned to the DI0; the DI0 of the module NI 9426, assigned to the DI1; and the DI0 of the module NI 9425, assigned to the DI2. Therefore, the changes in the DO0 to DO2 should be reflected in the resources DI0 to DI2. After this check, the acquisition is stopped and the allocated resources freed.

The validation of DMA data acquisition in CompactRIO is done in the test `cRIO_DAQDMA`. In this test will be performed the acquisition of two analog input channels of 32-bit at 1kS/s in blocks of 128 samples per channel. The channels acquired, CH0 and CH1, are connected respectively to the output of signal generator SG0 and the analog output AO1. In the test the signal generator will be configured a 100 Hz square signal of 0.4V of amplitude updated at 1kS/S. The value for AO1 will be introduced by the user as an input to the test program.

4.1.3.2 EPICS integration

If the previous test demonstrated that IRIO Library can recognize and handle the different profiles and resources implemented in the FlexRIO and CompactRIO devices, these tests are focused on the integration of IRIO Library with EPICS using `asynDriver`, `areaDetector`, and `NDS`. These tests consist of several preconfigured IOC applications where it is only needed to change the device serial number with the one of the target devices to be used, and they are available as a software package in Github.

The first set of tests is dedicated to the test of `IRIOAsyn`, the integration of IRIO Library using the `asynDriver` framework, and therefore it requires the installation of the EPICS environment as explained before. These tests are provided as a separate software package that can be downloaded from Github. The downloaded package contains IOCs to test all IRIO functionalities using the profiles defined in the IRIO Design Rules for FlexRIO and CompactRIO devices,

except for those involving the use of GPU. The full procedure for downloading and installing the package is detailed in the NI-RIO EPICS Device Driver Test Plan, as well as step by step description of the tests described here with the channel access commands that the user must execute.

4.1.3.2.1 IRIOAsyn reasons and interfaces for mandatory resources

The first test of IRIOAsyn uses the CompactRIO point-by-point profile to test the interface with resources mandatory for all RIO devices, the device initialization, the auxiliary analog and digital resources, and de. This test uses the cRIOPBP IOC. Once the IOC is running, the correct recognition of the hardware is checked using the dbior EPICS command in the EPICS IOC shell. After that, and using Channel Access commands in a second terminal, the status of the PVs corresponding to the RIO mandatory resources is checked before starting downloading the bitfile. Then, the bitfile is downloaded setting the PV FPGASTART to 1. To check the correct behavior of the acquisition, the changes in the PVs for the analog input 0 (AI0) and the device temperature (DEVTEMP) are monitored using camonitor. For the next step, the Debug Mode of the device is enabled by setting the DEBUGMODE PV to 1. When the Debug Mode is enabled, the AI0 PV value should be fixed to 1.25. The test of auxiliary analog and digital resources, and the digital resources is done similarly to what was done in the test of IRIO Library. Just as in that case, the auxiliary analog input resources from 0 to 2 are connected respectively with the auxiliary analog output resources from 0 to 2. Moreover, the same is applied to the auxiliary digital input and output resources, and the digital input and output resources. Therefore, values are written in the PVs connected to the output resources, and it is checked that the same values are read from the PVs connected to the corresponding input resources. After this, the IOC is closed.

4.1.3.2.2 IRIOAsyn reasons and interfaces for analog input and analog output resources

The second test of IRIOAsyn uses the CompactRIO Data Acquisition profile and the cRIODAQ IOC. With this IOC it is tested the configuration of analog output signals with the oscilloscope, the data acquisition using periodic scan on analog input PVs. After running the IOC and downloading the bitfile, the FPGA is started as well as the data acquisition by setting the PVs FPGASTART and

DAQSTARTSTOP to 1. Then, the range of the PVs SGUPDATERATE0, SGFREQ0, SGAMP0, SR0, and AO0 are checked. The check is done by writing in them a value out of range, and checking that the PVs STATUS and STATUS-STR indicate a dynamic configuration error due to values out of range. After that, the PVs are set to a value in the range, and the status PVs are checked to see that no error is shown. The next step is to configure the signal generator 0 to output in sequence a DC signal, a sine wave, a square wave, and a triangle wave. The correct generation of these signals is checked with an oscilloscope connected to the AO0 of the NI 9264 module. The signal generated is also checked setting a camonitor on the AI0 PV. The sampling rate functionality is also tested by changing the values of the AI0.Scan field from 1 second to 0.1 seconds, and checking that this corresponds to the update rate of values shown by the camonitor. To test the acquisition using the waveform PVs, the PV CH0 is read, and it should show 128 values corresponding to the sampling of the AI0 resource. After this last check, the IOC is stopped.

4.1.3.2.3 IRIOAsyn reasons and interfaces for image acquisition and CameraLink configuration

The third test of IRIOAsyn is focused on the test of image acquisition using the FlexRIO IMAQ profile and a CameraLink simulator. The test requires the configuration of the camera simulator and the opening of an OPI panel in CSS. After this, the IOC fRIO1483 must be executed, and the adapter module configuration is done before starting the FPGA using channel access write operations in the PVs CONFIGURATION0, SIGNALMAPPING0, LINESCAN0, FVAL0, DVAL0, LVAL0, and SPAREHIGH0. For the next step, the out of bounds error in STATUS and STATUS-STAR PVs is checked after setting the image size PVs SIZEX0, SIZEY0 to negative values. Then, both PVs are set to 256 and the acquisition is started. At that moment, the OPI panel should show the images acquired, a sequence of numbers from 0 to 9. The last test before closing the IOC is for checking the correct behavior of the serial line by writing UARTTRANSMIT0 a string. When UARTRECEIVE0 is read it must contain the same value, as the simulator is configured in loopback mode.

The next tests are dedicated to evaluate the integration of IRIOAsyn with SDD. At this effect, the necessary SDD XMLs are provided. These XMLs describe the

PVs used for the acquisition using CompactRIO and FlexRIO, including all the supported modules. These XMLs need to be loaded in the SDD database using the `sdd-sync` command. Then, the IOCs are generated and configured using the `sdd-editor` tool. In this tool, the platform and IO Modules to be used are selected, as well as resources implemented in the FPGA. This will produce an EPICS IOC application where it is only needed to add the IRIOAsyn device initialization call in the IOC startup script. SDD also generates an OPI panel to operate all the PVs configured. In this tests, the acquisition with and without DMA in CompactRIO, as well as analog, digital, and image data acquisition in FlexRIO with the modules NI 5761 (analog), NI 6581 (digital), and image (NI 1483). The PV read and write operations in these test are exactly the same that the ones described before in this section (signal generator configuration, camera configuration, read/write in analog or digital resources), but using the OPI graphical interface rather than channel access commands.

4.1.3.2.4 IRIOAsyn driver misconfiguration and error management

A final set of test is dedicated to checking the behavior of IRIOAsyn when the device initialization call is misconfigured, that is, wrong device serial number or wrong bitfile and header file name. In this cases, it is expected that no crash happens when the IOC is started, but the STATUS and STATUS-STR should indicate the error.

4.1.3.2.5 IRIONDS for FlexRIO data and image acquisition

The tests for IRIONDS are also provided as a separate software package containing preconfigured IOCs. The IOCs in this package contains the bitfiles, and configurations for data acquisition with FlexRIO devices and the adapter modules NI 5761 (analog), NI 6581 (digital), and image (NI 1483). For each IOC there is an OPI panel specifically designed for that application. In these cases, for the action of downloading the bitfile in the FPGA is done by initializing the NDS device. The NDS device is initialized writing "START" once in the device MSGS. The device status PV will show "INIT" if the initialization is ok. Writing "START" a second time in the device MSGS PV will start de FPGA. In the case of data acquisition with the NI 1483 adapter module, the module configuration needs to be done before starting the FPGA. The bitfiles provided are for acquisition using CameraLink Full Mode, with 8-bit wide pixels and 8 pixels per

pixel clock (8Tap8). The acquisition is started when the command “ON” is written in the channel group MSGS PV. The status PV of the channel group and the channels will show “ON” status. After this, the data acquired will be shown using the CSS widgets for waves and images. The rest of the controls (analog/digital resources, auxiliary resources, CameraLink serial line) can be operated from the moment the FPGA was started. The acquisition can be stopped by writing the “OFF” command in the channel group MSGS PV, and the FPGA is stopped by writing the “STOP” command in the device MSGS PV.

Regarding the GPU usage in IRIONDS, two test IOCs are provided. Both tests use the image acquisition profile of FlexRIO. The first one, FlexRIO1483GPU, does not include any processing in the GPU. Therefore, it does not require to extend any of the IRIONDS classes. The images that arrive at the GPU are copied to the host memory and publish in the channel PV. The OPI panel for this IOC is exactly the same than the one used for the image acquisition without GPU. The second test IOC, FlexRIO1483GPUProcess, include selectable GPU process. At this effects, the IOC application includes classes extending the IRIONDS classes, just reimplementing the processImageGPU function to call the developed GPU library. This library contains two functions, one for the color inversion of the image, and another one to calculate the Discrete Wavelet Transform (DWT) of the image. The selection of the GPU algorithm is done using a new PV, GPUALG, which support is implemented in the class extending IRIONDS channel group. The OPI panel provided to test this IOC is similar to the previous one, but with two image widgets and a new box to select the algorithm. One of the image widgets is used when the selected algorithm “None” or “INV,” and the second when the selected algorithm is “DWT.” The reason why the DWT image needs another widget is that the pixels of the DWT image are 32-bit wide floating point numbers, not 8-bit wide integers like in the other cases. Therefore it needs another widget to read from a different PV. The second widget is also configured with another color scheme, to include representation for negative numbers. In this example, negative numbers are represented on a scale from orange to red.

4.1.3.2.6 IRIOAD for FlexRIO image acquisition with hardware timestamping

The IRIOAD test only covers image acquisition using FlexRIO and NI 1483, as it is the only acquisition implemented in IRIOAD. The FPGA is configured and

started when the application is run, and the data acquisition is controlled by the areaDetector standard PVs. This example also includes the timestamp of the image using the hardware clock implemented in the FPGA, and it is shown in the Timestamp PV defined by areaDetector. The GUI of this test was implemented using MELD and is the example GUI provided by areaDetector. No changes were required as IRIOAD uses the same set of PVs.

4.1.4 Performance

In this section is analyzed the performance of the IRIO Tools. The first measurement of performance corresponds to the maximum data throughput achieved. NI provides performance test applications and bitfiles for FlexRIO, but these applications do not check the consistency of the data, just the throughput. These tests were run and the maximum performance achieved was 800 MB/s. For the test using IRIO Library, the bitfile developed sends 8-bit wide numbers in increasing order constantly. The speed of the data sent is regulated with the SamplingRate control. In the host side, an application developed using IRIO Library acquires the data in blocks of 1024 samples, checking the consistency of the data. The acquisition is firstly configured with a slow rate, and if the acquisition is made without problems for 100 blocks it is stopped before increasing the sampling rate and starting again. With this application, a maximum throughput of 800 MB/s without data loss is achieved. This value is in the limit of the technology, as the maximum theoretical throughput of the FPGA-Host link using FlexRIO devices from the 796x family is 1 GB/s. Similar values were obtained when DMA to GPUs was used, performing the data consistency check both in the GPU or when the data was copied back to the CPU.

The results are different when using the EPICS interfaces. In all the cases, if the data was just acquired and not published in the EPICS PVs, the throughput is still 800 MB/s. However, when publishing the data, above the 600 MB/s data inconsistency is reported after some seconds. This phenomenon is due to the throughput with EPICS was smaller than the FPGA throughput, and the host or GPU memory buffer used by the DMA gets filled, causing overwrites. Nonetheless, this behavior is expected, as EPICS is not supposed to handle this big data throughput.

The processing performance of the system is more difficult to measure. In the case of the FPGA, it is better to talk about the processing capability than the performance, as the process in the FPGA is limited by the number of resources available. Also, the FPGA is better suited for doing data process that can be done during the acquisition, data processing that does not require accumulate many data before starting the process. Examples of these processes are the timestamping of the data [63], calculation of the histogram [70] (explained in 4.2), small pattern recognition [71], pulse counting [72], or the implementation of the control loop. In Table 12 it is shown a resume of the resources of used to implement this features. All these processes were done in real-time and implemented in the FPGA, however, in some cases the way that the functionality is implemented may impose restrictions in the performance. For example, the histogram requires 256 cycles after the acquisition end for sending the results to the host. Assuming that the FPGA clock is 100 MHz, the maximum frame rate is 390 frames/s.

In the case of the GPU even is more complicated to measure, as the time to process the data depends on the specific process implemented and how it was optimized for the target. With a simple process, as image inversion of 256x256 images with 8-bit wide pixels, the maximum throughput is achieved. However, for very small images, and doing the process for just one image at the time, this performance might be affected by the latency of the small data copies and the calls to the GPU kernels. For more complex data process, as image calibration or CCL, the process time will limit the maximum framerate. In Table 10 and Table 11 are presented some of the process time measured for this cases.

Table 9 Examples of FPGA resource utilization for different data processes

FPGA Resource Utilization					
Model	Xilinx Virtex-5 SX95T FPGA (FlexRIO 7966)				
Resources	Total Slices	Slice Registers	Slice LUTs	DSP48s	Block RAMS
Available	14720	58880	58880	640	244
Image Acquisition (10Tap8)	39.8%	19.8%	17.4%	0%	21.7%
Timestamp	7.6%	5.6%	6.3%	3.9%	0.4%
Image Histogram	12.9%	4.5%	10.3%	0.0%	5.7%
Pattern Recognition	8.3%	6.2%	13.7%	0.6%	0%
Analog Data Acquisition (16bit x 2ch)	33.5%	15.6%	14.3%	1.6%	24.2%
Pulse Counting	49.5%	42.0%	28.4%	18.4%	0%

Table 10 GPU Process Time for Calibration, Thresholding and Find-Max

Process	1024x1024	512x512	256x256
Calibration	0.72 ms	0.19 ms	0.05 ms
Thresholding	0.19 ms	0.06 ms	0.02 ms
Find-Max	0.38 ms	0.22 ms	0.22 ms

Table 11 GPU CCL process time for the YACLAB [73] image databases

Image Database	GPU mean time	Database description
3dpes	0.40 ms	3D people surveillance
fingerprint	0.41 ms	Fingerprint images, from 240x320 to 640x480
hamlet	3.62 ms	Scan of Hamlet book, 2.7 Mpixel images, 1.4k labels per image
medical	2.47 ms	Medical images 1.41 Mpixel images, 484 labels per image
mirflickr	0.73 ms	MIRflickr database 25k 0.17Mpixel images, 495 lables per image
tobaco800	4.42 ms	1290 scanned documents Different equipment/resolution, from 1200x1600 to 2500x3200

4.1.5 Integrity for IRIO Library

The integrity of the solution is guaranteed by static and dynamic code analysis tools and the test coverage percentage. These tools were applied to IRIO Library only, as the dependencies for the rest with EPICS base and modules difficult the analysis a lot.

Regarding the analysis tool, the dynamic analysis was performed using Valgrind [74]. Valgrind aims to detect possible memory and thread management

problems. The first time it was applied, it revealed some possible memory leakages in IRIO Library. These problems are now solved, and IRIO is proved to be memory leakage free.

The static analysis was performed using `cppcheck`[75] and `Polyspace`[76]. `cppcheck` tries to detect bugs that compilers may have missed, like type null pointer dereferences, out of bounds access, or uninitialized variables. `Polyspace` is a more complete tool, which apart from this kind of errors it also aims to detect stylistic issues. One of the advantages of `Polyspace` is that it can be set to check the compliance with some predefined guidelines. One of the guidelines included in `Polyspace` is MISRA C, a set of rules developed by the Motor Industry Software Reliability Association (MISRA) aiming to facilitate the code safety, security, portability, and reliability in the context of embedded systems. MISRA C is a reference for critical developments in embedded systems, and it is adopted not only in the automotive industry but also in very demanding applications like aerospace or defense. MISRA is composed by a set of rules that can be required or advisory. In the new versions, it is also introduced the concept of guidelines, a rule whose interpretation is more open to interpretation. Both tools, `cppcheck` and `Polyspace` checking MISRA C, were applied to IRIO Library. A great volume of the issues found in this analysis were solved for the current IRIO Library version. However, some of them still pending. Part of the pending issues are caused in the calls to the NI RIO Driver, and it will require modifications in the driver to solve them. Other issues are related with the use of dynamic memory, strongly discouraged in embedded systems due to the limited amount of memory usually available in those systems.

With regard to the test coverage, the analysis shows that the test described in the previous sections cover the 76.4 % of IRIO Library code, and more than the half of the branches in the code. A detailed inspection of the report showed that the code and branches not covered by the test were the ones dedicated to error handling and error message prompt. This provides a good confidence on the functionality of the code, but in the next versions it is planned to increase the coverage of the test by including a systematic testing of the errors.

4.2 Application Results

In this section, there will be presented some application examples developed with the methodology. The applications presented are the ITER's reference image acquisition and processing system on PXIe, the plasma density measurement implemented in KSTAR using FlexRIO, a virtual 3D lab with ITER PXIe Fast Controllers.

4.2.1 ITER PXIe Image Acquisition and Processing Reference System

The methodology presented here was used for the implementation of the ITER Image Acquisition and processing System (IAS) on PXIe form factor. This system was designed by the student during his stay in ITER, where its implementation started. This system follows the ITER approach for diagnostic implementation, using NDS approach. The architecture of the system is depicted in Fig. 41. It consists of a CameraLink Camera connected to a FlexRIO 7966 device with the NI 1483 adapter module. The PXIe chassis hosting the FlexRIO device is connected to the host computer using an MXIe. Connected to the same host

Fig. 41 ITER PXIe Image acquisition and processing system architecture

computer there is a NVIDIA GPU that will be used for the image processing. The requirements for this application were to implement some image processing algorithms commonly used in nuclear fusion image diagnostics. The image process consists of the calculation of the image histogram, calculation of the maximum pixel value and its position in the image, the calibration of the image including the non-uniformity correction, the binarization of the image using configurable thresholds, and the Connected Components Labeling (CCL) of the image. It was also required to have a precise timestamp of the images acquired. At this effects, the computation load was divided between the FPGA and the GPU. In the FPGA it is implemented the timestamp of the image using the PTP-synchronized clock described in 3.3.2.2 and the image histogram. The rest will be calculated in the GPU. The logical representation of the acquisition and processing system is represented in Fig. 42.

Fig. 42 Logical representation of the ITER reference image acquisition and processing system

The integration of the system with EPICS was done by extending the IRIONDS classes, as NDS is approach selected by ITER for the implementation of its diagnostics.

The IRIONDS classes were extended to support the synchronization mechanism required by the PTP-synchronized clock and the calls to the GPU processing

libraries. It was also added support for selecting on run-time the image processing done on the GPU. The possible choices are none, image inversion, Discrete Wavelet Transform (DWT), Thresholding, or CCL. This last is the one corresponding to the requirements of this application, including the calibration and maximum search as well as the CCL. As depicted in Fig. 42, the system uses two DMAs. One is used to send the calculated histogram from the FlexRIO to the host. The other one is used to send the acquired image to the GPU. When the image is processed in the GPU, all the results are copied back to the host. In the GPU, the first pixels of the image containing the frame counter and timestamp are not changes. When the host receives the data from both the GPU and the FPGA, it publishes the results using EPICS PVs. Some additional PVs were added to support the selection of the threshold applied to the image, which can be modified at run time. Also, several PVs for publishing the images are used, as the data type and color scheme of the images are different depending on the process done. The OPI developed to control all these functionalities is represented in Fig. 43.

Fig. 43 ITER image acquisition system OPI panel.

The different functional zones are: 1) FPGA Control panel. 2) Camera and image configuration panel (Camera Link). 3) Image process selector. 4) Threshold value control panel. 5) Timestamp indicator read from the image. 6) Raw image histogram. 7) Raw and binarized image display. 8) CCL image display.

The occupation of the FPGA resources is detailed in Table 12, indicating the occupation percentage of each functional block. It is important to notice that there still are available resources in the FPGA that could be used to implement more processing algorithms.

Table 12 FPGA Resource Utilization in ITER IAS Implementation

Xilinx Virtex-5 SX95T FPGA (FlexRIO 7966)					
	Total Slices	Slice Registers	Slice LUTs	DSP48s	Block RAMs
Available resources	14720	58880	58880	640	244
Base & DAQ %	39.8	19.8	17.4	0.0	21.7
Timestamp %	7.6	5.6	6.3	3.9	0.4
Histogram %	12.9	4.5	10.3	0.0	5.7
Total %	60.3	29.9	33.9	3.9	27.9

To measure the performance of the all the blocks built for the GPU, the same algorithms were developed for the CPU, just to make a performance comparison between systems with different image sizes. The results in Table 13 show that the use of GPU improves the performance of the system by at least 400%, reaching in some cases the 500%.

Table 13 Comparison of image processing times on CPU and GPU

Resolution	1024x1024		512x512		256x256	
Process	CPU	GPU	CPU	GPU	CPU	GPU
Calibration	3.77	0.72	1.08	0.19	0.26	0.05
Thresholding	0.83	0.19	0.20	0.06	0.03	0.02
Find-Max	1.20	0.38	0.29	0.22	0.07	0.22

The most demanding algorithm in terms of computational resources is the Connected Components Labeling. The performance of this algorithm is directly dependent on the image processed and the threshold applied in the binarization. Therefore, it is not possible to characterize the performance of the algorithm in a deterministic way. The way to measure the performance was to test the algorithm using the binarized images from a public database [73]. The results of GPU processing are compared with the results in an actual Intel CPU (Core i5 6th gen) in Table 14. The results show that the CPU is better suited than the GPU for the CCL process. However, considering total processing time (i.e. including the calibration, thresholding, and maximum search), the GPU present better performance. Additionally, leveraging the CCL computation load from the CPU will allow its use for other necessary processes, as running all the EPICS threads and the archiving of all the data acquired. Moreover, this algorithm will require

less computational effort when used on real fusion images, as the density of areas detected will be much lower.

Table 14 Comparison of CCL processing times on CPU and GPU using different image databases

Dataset	GPU	Best CPU
3dpes	0.40	0.41
fingerprints	0.41	0.28
hamlet	3.62	3.37
medical	2.47	1.43
mirflickr	0.73	0.29
tobacco800	4.42	4.93

This application demonstrates the possibility of using the proposed methodology for the implementation of applications with the same complexity than the ones need for nuclear fusion diagnostics.

4.2.2 KSTAR Application

KSTAR is the Korean superconducting tokamak that currently holds the record of high-performance plasma pulse time. As part of its role in support of ITER experiment, KSTAR is evaluating the use of CCS tools, including IRIO software. After some testing to get the basic knowledge of the tools, they implemented an analog data acquisition example with zero crossing detection and phase shift calculation with regard to a probe signal using a FlexRIO device with the NI 5734 adapter module. Then, they use the knowledge acquired from this example to implement the density control algorithm using the same platform.

The NI 5734 adapter module is was not supported by IRIO, as it is not part of ITER's catalogs, but the IRIO team added minor changes to support it. The changes were just the recognition of the module and the conversion of the values read using the coefficients given by the manufacturer for that adapter module. The SDD files for the combo of FlexRIO 7966 and NI 5734 were also provided.

The zero crossing detection and phase shift calculation were implemented using two approaches. The first was using a third-party VHDL IP core integrated into the LabVIEW project. The schematic of this IP core is depicted in Fig. 44. The second was using just LabVIEW FPGA code (Fig. 45). In the report, it is said that the first version was more compact and efficient, as it took less compilation time

and minor resource usage, but that the use of LabVIEW FPGA had other advantages, as the ease of development and also the simple interpretation of the process being done in a quick view of the code. In this case, the input and the reference signals were sampled at 120 MHz, and only the results of the process were sent to the host, not the signals acquired.

Fig. 44 Schematic of the third-party VHDL-based IP Core for zero crossing detection and phase shift calculation used in KSTAR

Fig. 45 LabVIEW code for zero crossing detection and phase shift calculation used in KSTAR

For the density control algorithm, they took the VHDL implementation they had for the mTCA platform they were using previously and integrated it in the LabVIEW project. The developed application sends 1280 data to the host every 1ms. The same as above, the data sent to the host is only the processed data, not the acquired data. In the host, this data is used to calculate the plasma density and perform the corresponding control actions. In Fig. 46 a comparison of the previously used mTCA solution and the solution using ITER PXIe architecture is represented. The graphic shows how the new solution gives exactly the same results than the previous. The KSTAR team integrated this solution in EPICS developing their own EPICS device driver, and it was used for the real control of the machine during experimental shots from 16326 to 16348.

Fig. 46 Result comparison of density control in KSTAR implemented using data from ITER's PXIe architecture (left) and KSTAR mTCA architecture (right) for the KSTAR 15869 shot.

4.2.3 IRIO-3DLab

IRIO-3DLab [77] is a platform devised to assist developers in the design and implementation of intelligent and reconfigurable FPGA-based data acquisition systems based on EPICS and FlexRIO technologies using IRIO Tools. It is a multi-user 3D Virtual World developed on OpenSimulator where the users can interact with ITER Fast Controllers on PXIe form factors. The platform allows understanding the procedures such as assembly, connection, start-up, shut-down. Additionally, it enables them to carry out a set of predefined tests to

establish different configurations and test their effects on a real system in operation, as well as drive inputs and measure outputs remotely. This is a novel solution in the field of big physics that can contribute to improving the provision of training required for users involved in the design of complex instrumentation systems such as ITER I&C systems.

In IRIO-3DLab, users are represented by avatars that move around a virtual world and interact with the objects in the world. In this world there is a Laboratory building with several facilities, as represented in Fig. 47. Included in this facilities, there are representations of ITER cubicles with the hardware configuration of PXIe Fast Controllers. These cubicles contain virtual representations of the host computer, the PXIe chassis, an oscilloscope, and a signal generator. There are also desks with cables, FlexRIO devices, and FlexRIO adapter modules.

Fig. 47 Facilities of IRIO-3DLab: (a) Laboratory Building, (b) Exposition Room, (c) Meeting Room, and (d) Technical Room

The virtual Fast Controller needs to be managed just like the real system. The configuration procedures need to be done in an established order. When the user does not operate the system following this order, the system will give a warning with explanations about the correct operation order. The current system version enables actions such as: selecting FlexRIO cards and the adapter modules;

obtaining information about equipment characteristics; connecting and disconnecting cards on the chassis; selecting the pre-processing algorithm and hardware resources to be programmed in the FPGA from determined configurations; and starting up and shutting down the chassis, computer or general power supply to the cubicle.

The actions performed by the users in the virtual world are replicated in a real system controlled remotely. The configuration of the system results in the configuration of an EPICS IOC based on IRIOAsyn to be launched in the real system, and once the IOC is running, the users can monitor the real system running through OPI panels in the virtual world. The use of the virtual oscilloscope and signal generator also have the counterpart reaction in the real world. The data shown in the oscilloscope corresponds to real data acquired from the FPGA devices, while output signals of the virtual signal generator correspond to real signals generated by a signal generator and connected to the inputs of the FlexRIO adapter module.

The full architecture of IRIO-3DLab is depicted in Fig. 48, including hardware and software architecture, and the different elements composing the platform. IRIO-VirtualLab is the part running the virtual world and the virtual representations of the instruments. The actions of the users trigger HTTP

Fig. 48 IRIO-3DLab architecture

communications with IRIO-RemoteLab, a Java application running on the host computer of the real cubicle. This application will configure the EPICS IOC and run it. The application delegates the configuration of corresponding to the virtual oscilloscope and signal generator to a third application, IRIO-Test&Measurement. The communication between IRIO-RemoteLab and IRIO-Test&Measurement is done using Java Sockets. IRIO-Test&Measurement is in charge of acquiring the data from the FlexRIO devices and generate the signals in the FlexRIO inputs. This application currently admits three different devices. The first one is the NI USB-6361, a multifunction input-output device that can represent both the signal generator and the oscilloscope at the same time, as it counts with both input and output channels. This device is operated by calls to the NI DAQmx library, provided by the manufacturer. The other two devices are an Agilent MSO7014A oscilloscope and a Keysight 33500B signal generator. Both of this devices count with TCP servers that can be used for the control of the device.

An example of neutron diagnostics was implemented [72] using this tool, to show the possibilities offered by the platform. This implementation is based on a real application for fusion power estimation through the average neutron flux emitted by the plasma, detected using detectors based on a fission chamber [78]. In the developed application, the FlexRIO device is configured to acquire the pulses that will be supposedly produced by the impact of the emitted neutrons in the fission chamber and to apply signal processing algorithms such as filtering to increase the signal to noise ratio or perform calculations to obtain statistics for qualifying the digitalized pulses. The FlexRIO also contain the logic implementing the three most used algorithms to estimate the neutron flux, which includes Pulse Counting, Campbelling and Current counting. In the virtual world, the signal generator is connected to the FlexRIO adapter module inputs to simulate the pulses of the fission chamber. All the FlexRIO configuration is done following the IRIO Design Rules, and it is integrated into EPICS using IRIOAsyn without requiring any modification. To control the implementation, an OPI was developed, and it is represented in Fig. 49.

Fig. 49 CSS OPI panel available inside the virtual world for the control of the neutron diagnostics

Chapter 5

5 CONCLUSIONS, CONTRIBUTIONS, AND FUTURE WORK

5.1 Conclusions

As the main conclusion, a methodology for the implementation of data acquisition and processing systems based on FPGA and GPU and its integration in EPICS has been designed, developed, and tested. The methodology was completely validated, and it proved to solve the problems presented by the current status of the technologies used.

The main achievements of the use of this methodology are:

1. The development of data acquisition and processing systems following the approach adopted for the ITER Fast Controllers in PXIe Form Factor.
2. The use of intensively tested software tools that comply with ITER's defined Software Integrity Level 2 (SWIL-2) and that are currently in use in two important international big physics experiments (ITER and KSTAR).
3. To increase the maintainability of the system by following the same approach and use the same software for different applications and devices, including analog, digital and image acquisition implemented with high-performance FlexRIO devices or robust and reliable CompactRIO devices.
4. To avoid driver recompilations due to bitfile changes, thanks to the automatic resource recognition of IRIO Library.
5. To reduce the development time taking advantage of already developed examples implementing analog and digital data acquisition and image acquisition.
6. To use the provided examples as a base project to include custom data processing in the FPGA, the GPU, or the CPU, taking the advantages of the best-suited platform for each process required.
7. To reduce the latency introduced by unnecessary data copies by implementing direct data transfers between FPGA and GPU.
8. To reach data throughputs up to 800 MB/s to GPU or CPU memory, reaching the limits of the acquisition devices based on PCIe/PXIe 1.0 x4.
9. To get data with precise timestamping by using a PTP-synchronized hardware clock implemented in the FPGA.

10. The easy integration of the developed solutions with EPICS. The resources implemented in the FPGA are automatically interfaced with EPICS PVs. The user is only required to perform configuration tasks.
11. To benefit the EPICS community with the use of open source solutions which can be improved and extended by anyone and everyone can use these enhancements.

The proposed methodology and the developed products address all the drawbacks and difficulties defined in the Chapter 2 conclusions that were derived from the use of the PXIe technology. Therefore, the PXIe system architecture was effectively enhanced for its use in big physics experiments.

5.2 Contributions

The contributions to the science community resulting from the work carried out in the context of this thesis will be presented here. These contributions will be classified into Main Contributions (MC) and Side Contributions (SC) depending on their relation with the main objective of this thesis.

- **(MC#1) IRIO Methodology:** Definition of a methodology for the implementation of data and image acquisition and processing systems based on RIO devices and GPUs and its integration on EPICS. The methodology solves the basic data acquisition and the integration with the distributed control system. The users just need to implement the application specific processing. The methodology is supported by a set of FPGA Design Rules and a set of software tools, with examples available to help the users to implement their own systems using the methodology. The software supporting the methodology (MC#2-MC#4) is integrated into ITER CCS and available under a free software license. The solution provided was verified using static and dynamic analyses and meets the requirements of ITER SWIL-2.
- **(MC#2) IRIO Core Library:** A library for the handling of RIO devices. The library uses the manufacturer driver and offers a higher level API oriented to data and image acquisition, automatic recognition of the resources implemented in the FPGA, and optional support for direct DMA data transfers between FPGA and NVIDIA GPUs with RDMA support.

- **(MC#3) IRIOAsyn:** An EPICS Device Driver for RIO devices based on asynDriver. The use of IRIOCore Library allows to use the same EPICS Device Driver for different data and image acquisition applications, including user custom data process in the FPGA. Database templates are provided for adapting the PV database to the resources implemented in the FPGA. The device driver automatically links the PVs instantiated with the FPGA resources present. It includes examples for analog, digital, and image acquisition with the corresponding GUI based on BOY, as well as integration with SDD to facilitate the application deployment procedure.
- **(MC#4) IRIONDS:** An EPICS Device Driver following the NDS approach using IRIO Core Library. It supports analog, digital, and image acquisition using FlexRIO Devices. The class structure allows the user to extend the class and selectively modify the base functionality or add support for custom developments while inheriting the rest of the functionalities. It includes rules for compiling and linking with CUDA GPU libraries.
- **(MC#5) IRIOAD:** An EPICS Device Driver for CameraLink image acquisition using FlexRIO devices based on areaDetector and IRIO Core Library. It supports camera configuration through CameraLink serial line and frame timestamping using a hardware PTP Clock (SC#1).
- **(MC#6) ITER PXIe IAS:** An advanced image acquisition and processing application for the EoSens 3CL CameraLink Camera. Currently the reference for ITER image acquisition system in PXIe form factor. This application allows the camera configuration using EPICS PVs, hiding the direct use of the serial line. It includes support for frame timestamping using a hardware PTP clock (SC#1), camera firmware update, and GPU processing configurable on-the-fly among image color inversion, Discrete Wavelet Transform, and Connected Components Labeling.
- **(SC#1) Hardware PTP Clock:** A VHDL clock providing absolute time that can be synchronized with an external timing card connected to the PXIe chassis with a precision of tens of nanoseconds. The clock resolution is 10 ns and the absolute time provided is given in the PTP format, with 48 bits for representing seconds and 32 bits for the nanoseconds. The clock design can be imported into LabVIEW as an IP core and used to timestamp the data acquired.

- **(SC#2) NDS DAN Plugin:** An NDS-based plugin using ITER's DAN API for the archiving of data acquired in a NDS Device Driver. The implementation was successfully validated using an IRIONDS-based (MC#5) application for image acquisition.
- **(SC#3) SPAD Implementation:** Implementation on real-time of the SPAD disruption predictor using MARTe framework. The implemented predictor outperforms the detection results of the previously implemented predictors without training.

5.3 Technology transfer and mock-up implementations

The work presented here was developed with the intention of contributing to the scientific community. At this effect, the IRIO software tools are licensed under GPL version 2 license, thus making it free software that can be freely distributed and modified by the users. The methodology was announced in several conferences, in the EPICS forums, and in the research group web page. The code was made publicly available at Github [53]. To protect the intellectual property, it was registered in the Intellectual Property Registry of the Community of Madrid as “**IRIO: Intelligent Reconfigurable Input Output software tools for the Development of advanced data acquisition solutions for EPICS**” with the code **M-004824/2016**.

The methodology proposed is already in use by several projects, including a virtual 3D lab for the training in ITER PXIe Fast Controllers architecture and handling, the ITER's reference image acquisition and processing system on PXIe, and the plasma density measurement implemented in KSTAR using FlexRIO.

5.4 Academic production and dissemination

5.4.1 Articles in indexed journals

Here are presented the articles published in international journals indexed by the Journal Citation Reports (JCR).

- Title: High performance image acquisition and processing architecture for fast plant system controllers based on FPGA and GPU
Authors: J. Nieto, D. Sanz, P. Guillén, S. Esquembri, G. de Arcas, M. Ruiz, J. Vega, R. Castro

Journal: Fusion Engineering and Design

Date of Publication: April 8th, 2016

DOI: <http://dx.doi.org/10.1016/j.fusengdes.2016.04.004>

Impact Factor (JCR): 1.301; Nuclear Science & Technology Q1(6/32) (2015)

Contribution: First inclusion of the direct DMA transfers between FPGA and GPU in an EPICS application.

- Title: Implementation of an image acquisition and processing system based on FlexRIO, CameraLink and areaDetector

Authors: S. Esquembri, M. Ruiz, E. Barrera, D. Sanz, A. Bustos, R. Castro, J. Vega

Journal: Fusion Engineering and Design

Date of Publication: April 29th, 2016

DOI: <http://dx.doi.org/10.1016/j.fusengdes.2016.04.024>

Impact Factor (JCR): 1.301; Nuclear Science & Technology Q1(6/32) (2015)

Contribution: Development of the IRIOAD interface.

- Title: FPGA-based Solutions for Analog Data Acquisition and Processing Integrated in AreaDetector using FlexRIO Technology

Authors: R. Herrero, A. Carpeño, S. Esquembri, M. Ruiz, E. Barrera

Journal: IEEE Transactions on Nuclear Science

Date of Publication: Minor revision required (May 17th, 2017)

Impact Factor (JCR): 1.198; Nuclear Science & Technology Q2(10/32) (2015)

Contribution: Adaptation of IRIOAD for 1-dimensional analog or digital data.

- Title: Hardware Timestamping for an Image Acquisition System Based on FlexRIO and IEEE 1588 v2 Standard

Authors: S. Esquembri, D. Sanz, E. Barrera, M. Ruiz, A. Bustos, J. Vega, R. Castro

Journal: IEEE Transactions on Nuclear Science

Date of Publication: February 3rd, 2016

DOI: <http://dx.doi.org/10.1109/TNS.2016.2516640>

Impact Factor (JCR): 1.198; Nuclear Science & Technology Q2(10/32) (2015)

Contribution: Implementation and testing of the PTP-synchronized clock implemented in the FPGA using VHDL and its use for image timestamping.

- Title: 3D Virtual World Remote Laboratory to Assist in Designing Advanced User Defined DAQ Systems Based on FlexRIO and EPICS
Authors: A. Carpeño, D. Contreras, S. López, M. Ruiz, D. Sanz, G. de Arcas, S. Esquembri, J. Vega, R. Castro
Journal: Fusion Engineering and Design
Date of Publication: February 24th, 2016
DOI: <http://dx.doi.org/10.1016/j.fusengdes.2016.02.052>
Impact Factor (JCR): 1.301; Nuclear Science & Technology Q1(6/32) (2015)
Contribution: Development of IRIO-3DLab.
- Title: Implementing a Neutron-Diagnostic advanced DAQ system use case on a PXIe platform through a 3D remote laboratory
Authors: A. Carpeño, M. Ruiz, E. Mayoral, E. Bernal, S. Esquembri, E. Barrera, G. de Arcas
Journal: Fusion Engineering and Design
Date of Publication: April 1st, 2017
DOI: <https://doi.org/10.1016/j.fusengdes.2017.03.100>
Impact Factor (JCR): 1.301; Nuclear Science & Technology Q1(6/32) (2015)
Contribution: Application of IRIO-3DLab and the methodology for the implementation of a neutron diagnostic.
- Title: Real-time implementation in JET of the SPAD disruption predictor using MARTe
Authors: S. Esquembri, J. Vega, A. Murari, M. Ruiz, E. Barrera, S. Dormido-Canto, R. Felton, M. Tsalas, D. Valcarcel and JET Contributors
Journal: IEEE Transactions on Nuclear Science
Date of Publication: Minor revisions required (May 17th, 2017)
Impact Factor (JCR): 1.198; Nuclear Science & Technology Q2(10/32) (2015)
Contribution: Implementation of SPAD Disruption Predictor in MARTe.
- Title: Overview of the JET results in support to ITER
Authors: X. Lituatón et al
Journal: Nuclear Fusion
Date of Publication: Accepted for publication (January 30th, 2017)
Impact Factor (JCR): 4.040; Physics, Fluids & Plasmas Q1(2/30) (2015)
Contribution: SPAD considered for its application to ITER disruption system.

5.4.2 International Conferences and Meetings

Here are listed the works presented in international conference and meetings in chronological order:

- Title: Hardware Timestamping for an Image Acquisition System Based on FlexRIO and IEEE 1588 v2 Standard
Authors: S. Esquembri, D. Sanz, E. Barrera, M. Ruiz, A. Bustos, J. Vega, R. Castro
Conference: 19th IEEE Real-time Conference (RT 2014)
City: Nara, Japan
Date: May 26th-30th, 2014
- Title: Integration of advanced data acquisition applications using FPGA-based FlexRIO devices in ITER's CODAC Core System
Authors: A. Bustos, M. Ruiz, D. Sanz, E. Bernal, F. Di Maio, E. Barrera, S. Esquembri, R. Castro, J. Vega
Conference: 28th Symposium on Fusion Technology (SOFT 2014)
City: San Sebastián, Spain
Date: September 29th–October 3rd, 2014
- Title: Methodology for the Development of Intelligent Data and Image Acquisition Systems Using EPICS and FlexRIO Technology
Authors: D. Sanz, M. Ruiz, A. Bustos, S. Esquembri, E. Barrera, R. Castro, J. Vega
Conference: Fall 2014 EPICS Collaboration Meeting
City: Saclay, France
Date: October 20-23, 2014
- Title: Implementation of Image Acquisition and processing system based on FlexRIO, CameraLink and areaDetector
Authors: S. Esquembri, M. Ruiz, E. Barrera, D. Sanz, A. Bustos, R. Castro, J. Vega
Conference: 10th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research (IAEA TM 2015)
City: Ahmedabad, India
Date: April 20th–24th, 2015
- Title: High Performance Image Acquisition and Processing Architecture for Fast Plant System Controllers based on FPGA and GPU

Authors: J. Nieto, D. Sanz, P. Guillen, S. Esquembri, G. de Arcas, M. Ruiz, J. Vega, R. Castro

Conference: 10th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research (IAEA TM 2015)

City: Ahmedabad, India

Date: April 20th–24th, 2015

- Title: 3D Virtual World Remote Laboratory to Assist in Designing ITER Fast Plant System Controllers based on FlexRIO and EPICS

Authors: A. Carpeño, D. Contreras, S. López, M. Ruiz, D. Sanz, G. de Arcas, S. Esquembri, J. Vega, R. Castro

Conference: 10th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research (IAEA TM 2015)

City: Ahmedabad, India

Date: April 20th–24th, 2015

- Title: Development of advanced Data and Image Acquisition Systems using RIO Technology and EPICS: Integration in ITER

Authors: M. Ruiz, S. Esquembri

Conference: NI Big Physics Summit 2016

City: Geneva, Switzerland

Date: February 11th–12th, 2016

- Title: Development of Advanced Data and Image Acquisition Systems Using RIO Technology and EPICS: Integration in ITER

Authors: M. Ruiz, S. Esquembri, J. Nieto, E. Bernal, A. Carpeño, E. Barrera, A. Bustos, D. Sanz, R. Lange

Conference: Spring 2016 EPICS Collaboration Meeting

City: Lund, Sweden

Date: May 22nd–27th, 2016

- Title: Real-time implementation in JET of the SPAD disruption predictor using MARTe

Authors: S. Esquembri, J. Vega, A. Murari, M. Ruiz, E. Barrera, S. Dormido-Canto, R. Felton, M. Tsalas, D. Valcarcel, and JET Contributors

Conference: 20th IEEE Real-time Conference (RT 2016)

City: Padova, Italy

Date: June 5th–10th, 2016

- Title: Software Integrity Analysis Applied to IRIO EPICS Device Support Based On FPGA Real-Time DAQ Systems
Authors: D. Sanz, A. Bustos, J. Autrán, M. Fernández, S. Urueña, M. Ruiz, S. Esquembri
Conference: 20th IEEE Real-time Conference (RT 2016)
City: Padova, Italy
Date: June 5th–10th, 2016
- Title: Image Acquisition and GPU Processing Application using IRIO technology and FlexRIO devices
Authors: J. Nieto, M. Ruiz, S. Esquembri, G. de Arcas, A. de Gracia
Conference: 20th IEEE Real-time Conference (RT 2016)
City: Padova, Italy
Date: June 5th–10th, 2016
- Title: IRIO Technology: developing applications for advanced DAQ systems using FPGAs
Authors: M. Ruiz, S. Esquembri, A. Carpeño, J. Nieto, A. Bustos, E. Bernal, D. Sanz, E. Barrera
Conference: 20th IEEE Real-time Conference (RT 2016)
City: Padova, Italy
Date: June 5th–10th, 2016
- Title: Analog Data Acquisition and Processing FPGA-based Solutions Integrated in Area Detector using FlexRIO technology
Authors: R. Herrero, A. Carpeño, S. Esquembri, M. Ruiz
Conference: 20th IEEE Real-time Conference (RT 2016)
City: Padova, Italy
Date: June 5th–10th, 2016
- Title: Implementing a Neutron-Diagnostic use case in a PXIe platform through a 3D remote laboratory
Authors: A. Carpeño, M. Ruiz, E. Mayoral, S. Esquembri, E. Barrera, G. de Arcas, S. Lopez
Conference: 29th Symposium on Fusion Technology (SOFT 2016)
City: Prague, Czech Republic
Date: September 5th–9th, 2016

- Title: Implementation of an FPGA-GPU based image acquisition and processing system using ITER Fast Controller architecture on PXIe form factor

Authors: S. Esquembri, E. Bernal, M. Ruiz

Conference: NI Big Physics Summit 2017

City: Prague, Czech Republic

Date: March 21st–22nd, 2017

5.4.3 Short Courses for International Organizations

- Title: Real-time data acquisition and processing applications using NIRIO FPGAs-based technology and NVIDIA GPUs

Lecturers: M. Ruiz, J. Nieto, S. Esquembri

Duration: 3 hours

Attendees: IEEE Nuclear Plasma Sciences Society

Place: 20th IEEE Real-time Conference, Padova, Italy

Date: June 5th, 2016

5.5 Future Work

The methodology presented here, including the software tools that support it and the set of examples and templates provided, is meant to aid in the development of high throughput data acquisition and processing systems. From its beginnings, it has evolved cover the necessities reported by users or identified by the developers as possible future requirements. At this matter, there are some of those features that have been already considered for future developments. These improvements try to keep the methodology updated with respect to the technologies arising not only in the context of nuclear fusion environments but also in the high-performance acquisition and processing systems. They also try to increase the number of possible users, covering more applications for the system or different hardware elements.

One of the future developments for the methodology will be the implementation of an example of analog and digital data acquisition and processing system using GPU. While IRIO already supports this feature, the examples showing GPU process are focused on image processing, not in 1-dimensional data. It is intended to develop examples showing this capability, so users interested in process 1-

dimensional data on GPU can use these examples as a starting point for their own applications.

Another GPU-related feature to be developed is the integration of the GPU usage in ADIRIO as it is in NDSIRIO. Currently, no examples of GPU usage with IRIO are developed for areaDetector, and it will need some changes in the ADIRIO classes. This also includes to add the support for GPU processing in the 1-dimensional ADIRIO driver that is currently under development, and that will be merged with the original image-oriented ADIRIO. With this improvement and the one mentioned previously, the use of GPU process for one or two-dimensional data on NDS and areaDetector will be covered.

Currently, the direct DMA transfers between FPGA and GPU are limited to the use of NVIDIA Professional Grade GPUs. This is the case because that are the only GPUs added to the ITER Catalog for Fast Plant Controllers, and is needed that the GPU support Remote DMA (RDMA), feature called GPUDirect by NVIDIA. Nevertheless, some AMD Professional Grade GPUs count with a similar feature, in this case called DirectGMA. If direct DMA transfers to AMD GPUs was supported, GPUs of the two main manufacturers could be used with IRIO, increasing the set of possible users.

Another development that can spread the use of IRIO is the implementation of interfaces with other control systems or frameworks commonly used in big physics experiments, like TANGO, TINE, or MARTe. While the purpose and set of users of this systems are different, they all have in common some characteristics. They are used with acquisition and processing hardware, usually requiring high throughput, fast response, or even real-time behavior of the hardware controlled. Adding interfaces between this frameworks and IRIO will benefit both sides. IRIO could increase the number of users and therefore the number of applications for what it is used. This can help to identify more interesting features to be developed that will make IRIO better for all its users. In the other hand, adapting IRIO to this other frameworks will leverage the development and integration process of RIO devices in wherever these frameworks are used. One of the possible approaches for implementing this could be using the new version of NDS developed by Cosylab, NDS3. This new version is a completely new development which intention is that every device

driver implemented using NDS3 could be used by different control systems. Currently, it has already implemented interfaces with EPICS and TANGO, but the structure of the projects allows to add support for other systems. ITER is also planning to include NDS3 as part of CCS in the near future. In any case, it is needed to investigate these technologies and its applications further to determine what could be more interesting, to develop IRIO device support directly using the framework, using another abstraction layer like NDS3, or maybe both.

To finalize, one interesting line of research could be to include OpenCL as part of the methodology. OpenCL is a framework to develop unique programs that can be executed across heterogeneous platforms including CPUs, GPUs, FPGAs, and DSPs. It depends on the target devices to have an implementation conformant with the standard, but some of the most important manufacturers comply with the standard, as AMD, Intel, ARM, NVIDIA, Qualcomm, Altera, and Xilinx, to name some. While if RIO devices currently do not support OpenCL, even if they are based on a Xilinx FPGA, there are still some uses for OpenCL in a methodology like IRIO. The closest application considering the current state of the methodology is to apply OpenCL for the GPU developments. In this way, the processing libraries will be equally valid for AMD and NVIDIA GPUs. Also, in the case that no GPU is present, the same library could be executed by the CPU. The inclusion of OpenCL in the methodology would also allow increasing the capabilities of the system easily by connecting more processing elements, as long as they are compatible with OpenCL, they can be used to execute the processing libraries. In the case of NDS or areaDetector, these additional processing units could also be managed as plugins, as it was demonstrated in [REF Mariano IAEA17] for areaDetector. Taking this into account, the addition of OpenCL will make the methodology even more flexible, and it will be able to be adapted to use one or other processing device depending on the characteristics of the process to be done, selecting the most appropriate platform.

Chapter 6

6 REFERENCES

References

- [1] NVIDIA. (2017). *CUDA Parallel Computing Platform*. Available: http://www.nvidia.com/object/cuda_home_new.html.
- [2] Kronos Group. (2017). *OpenCL - The open standard for parallel programming of heterogeneous systems*. Available: <https://www.khronos.org/opencv/>.
- [3] OpenACC Organization. (10/2015). *Open ACC Reference Guide v2.5*. Available: http://www.openacc.org/sites/default/files/OpenACC_2.5_ref_guide_update.pdf.
- [4] OpenMP Architecture Review Boards. (11/2015). *OpenMP Application Programming Interfaces version 4.5*. Available: <http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf>.
- [5] V. Martin, V. Moncada and J. M. Traveré, "Challenges of Video Monitoring for Phenomenological Diagnostics in Present and Future Tokamaks," *IEEE Transactions on Plasma Science*, vol. 40, (3), pp. 646-652, 2012.
- [6] X. N. Yue *et al*, "Fast equilibrium reconstruction for tokamak discharge control based on GPU," *Plasma Phys. Controlled Fusion*, vol. 55, (8), pp. 085016, 2013.
- [7] Wind River. (2016). *VxWorks Product Overview*. Available: <https://www.windriver.com/products/product-overviews/2691-VxWorks-Product-Overview.pdf>.
- [8] IEEE Computer Society. (1987). *IEEE Standard for A Versatile Backplane Bus: VMEbus*. Available: <https://standards.ieee.org/findstds/standard/1014-1987.html>.
- [9] Argonne National Laboratory. (2017). *EPICS - Experimental Physics and Industrial Control System*. Available: <http://www.aps.anl.gov/epics/>.
- [10] L. R. Dalesio, M. Kraimer and A. Kozubal, "EPICS architecture," in *Icalepcs*, 1991, pp. 92-15.
- [11] M. L. Rivers, "areaDetector: Software for 2-D Detectors in EPICS," *Sri 2009: The 10th International Conference on Synchrotron Radiation Instrumentation*, vol. 1234, pp. 51-54, 2010.

- [12] V. Isaev *et al*, "EPICS data acquisition device support," in *Proc. Int. Conf. Accelerator and Large Experimental Physics Control Systems*, 2013, pp. 707-709.
- [13] P. Helander *et al*, "Stellarator and tokamak plasmas: a comparison," *Plasma Phys. Controlled Fusion*, vol. 54, (12), pp. 124009, 2012.
- [14] Y. Xu, "A general comparison between tokamak and stellarator plasmas," *Matter and Radiation at Extremes*, vol. 1, (4), pp. 192-200, 7, 2016.
- [15] JET Team, "The JET Project–Design Proposal, Rep," *EURJET-R5, CEC, Brussels*, 1975.
- [16] J. Team, "Fusion energy production from a deuterium-tritium plasma in the JET tokamak," *Nucl Fusion*, vol. 32, (2), pp. 187, 1992.
- [17] M. L. Watkins, "JET divertor research in support of ITER," *Plasma Phys. Controlled Fusion*, vol. 36, (12B), pp. B241, 1994.
- [18] G. F. Matthews *et al*, "JET ITER-like wall - overview and experimental programme," *Phys. Scripta*, vol. 2011, (T145), pp. 014001, 2011.
- [19] F. Romanelli. (11/2012). *A roadmap to the realisation of fusion energy*. Available: <https://www.euro-fusion.org/wpcms/wp-content/uploads/2013/01/JG12.356-web.pdf>.
- [20] X. Litaudon, "JET Program for Closing Gaps to Fusion Energy," *IEEE Transactions on Plasma Science*, vol. 44, (9), pp. 1481-1488, 2016.
- [21] F. Sartori *et al*, "Jet operations and plasma control: A plasma control system that is safe and flexible in a manageable way." in *2009 23rd IEEE/NPSS Symposium on Fusion Engineering*, 2009, pp. 1-6.
- [22] H. van der Beken *et al*, "CODAS: The Jet Control and Data Acquisition System," *Fusion Science and Technology*, vol. 11, (1), pp. 120-137, 1987.
- [23] R. Felton *et al*, "Real-time measurement and control at JET experiment control," *Fusion Eng. Des.*, vol. 74, (1–4), pp. 561-566, 11, 2005.
- [24] D. Alves *et al*, "Real-time motional Stark effect in jet," *Fusion Eng. Des.*, vol. 71, (1–4), pp. 175-181, 6, 2004.
- [25] A. J. N. Batista *et al*, "ATCA control system hardware for the plasma vertical stabilization in the JET tokamak," in *2009 16th IEEE-NPSS Real Time Conference*, 2009, pp. 308-313.

- [26] PCI Industrial Computer Manufacturers Group, "PICMG 3.0 Revision 3.0 Specification: AdvancedTCA Base Specification," 2008.
- [27] G. De Tommasi *et al*, "Real-Time Systems in Tokamak Devices. A Case Study: The JET Tokamak," *IEEE Transactions on Nuclear Science*, vol. 58, (4), pp. 1420-1426, 2011.
- [28] A. C. Neto *et al*, "A Survey of Recent MARTe Based Systems," *IEEE Transactions on Nuclear Science*, vol. 58, (4), pp. 1482-1489, 2011.
- [29] A. Werner *et al*, "Cutting edge concepts for control and data acquisition for wendelstein 7-X," in *2013 IEEE 25th Symposium on Fusion Engineering (SOFE)*, 2013, pp. 1-5.
- [30] C. Hennig *et al*, "Continuous data acquisition with online analysis for the Wendelstein 7-X magnetic diagnostics," *Fusion Eng. Des.*, vol. 83, (2-3), pp. 321-325, 4, 2008.
- [31] J. Schacht *et al*, "Overview and status of the control system of WENDELSTEIN 7-X," *Fusion Engineering and Design*, vol. 82, (5), pp. 988-994, 2007.
- [32] Y. Shimomura *et al*, "ITER overview," *Nucl Fusion*, vol. 39, (9Y), pp. 1295, 1999.
- [33] D. Campbell, "ITER goals and project," *ITER Organization*, 2014.
- [34] J. Journeaux, "Plant Control Design Handbook," *ITER Organization*, (27LH2V), 08/04/2013, 2013.
- [35] F. Di Maio, "CODAC Core System Overview," *ITER Organization*, (97W6QN), 29/03/2017, 2017.
- [36] P. Makijarvi, "ITER Catalog for I&C Products - Fast Controllers," *ITER Organization*, (345X28), 11/02/2016, 2016.
- [37] B. Evrard, "ITER Catalogue for I&C Products - Slow Controllers PLC," *ITER Organization*, (333J63), 04/07/2014, 2014.
- [38] G. S. Lee *et al*, "The KSTAR project: An advanced steady state superconducting tokamak experiment," *Nucl Fusion*, vol. 40, (3Y), pp. 575, 2000.

- [39] K. Keeman *et al*, "Present Status of the KSTAR Superconducting Magnet System Development," *Plasma Science and Technology*, vol. 6, (5), pp. 2445, 2004.
- [40] K. H. Kim *et al*, "The KSTAR integrated control system based on EPICS," *Fusion Engineering and Design*, vol. 81, (15), pp. 1829-1833, 2006. . DOI: <http://dx.doi.org/10.1016/j.fusengdes.2006.04.026>.
- [41] M. Kwon *et al*, "The control system of KSTAR," *Fusion Engineering and Design*, vol. 71, (1), pp. 17-21, 2004. . DOI: <http://dx.doi.org/10.1016/j.fusengdes.2004.04.004>.
- [42] MDSPlus Group. (2015). *MDSPlus Official Website*. Available: www.mdsplus.org/.
- [43] K. Clausen, "ESS—the european spallation source," in *16th Meeting on the International Collaboration on Advanced Neutron Sources*, 2003, .
- [44] M. Åberg *et al*, *ESS Technical Design Report*. European Spallation Source, 2013.
- [45] National Instruments. (16/10/2015). *NI RIO Architecture (White Paper)*. Available: <http://www.ni.com/white-paper/10894/en/>.
- [46] IEEE Instrumentation and Measurement Society. (2008). *IEEE Standard for a precision clock synchronization protocol for networked measurement and control systems*. Available: <https://standards.ieee.org/findstds/standard/1588-2008.html>. DOI: 10.1109/IEEESTD.2009.4839002.
- [47] P. Makijarvi, "ITER Guideline for Fast Controllers, I/O Bus Systems and Communication Methods between Interconnected Systems," *ITER Organization*, (333K4C), 08/02/2013, 2013.
- [48] T. Hakulinen *et al*, "Building an interlock: Comparison of technologies for constructing safety interlock," in *Proceedings of ICALEPCS 2015*, Melbourne, Australia, 2015, .
- [49] CSS Collaboration Group. (2017). *Control System Studio Website*. Available: <http://controlsystemstudio.org/>.
- [50] TANGO Collaboration Group. (2017). *TANGO Main Page*. Available: <http://www.tango-controls.org/>.

- [51] DESY. (2017). *TINE Main Page*. Available: <http://adweb.desy.de/mcs/tine/index.html>.
- [52] D. Sanz Hernando, "Modelización E Integración De Sistemas De Adquisición De Datos Inteligentes En Sistemas De Instrumentación Para Dispositivos De Fusión." , Escuela Técnica Superior de Ingeniería y Sistema de Telecomunicación, Universidad Politécnica de Madrid, 2014.
- [53] IRIO Team (I2A2 Research Group). (11/2016). *IRIO GitHub Site*. Available: <https://github.com/irio-i2a2>.
- [54] IRIO Team (I2A2 Research Group). (11/2016). *IRIO Design Rules for LabVIEW FPGA v1.2.0*. Available: http://www.i2a2.upm.es/wp-content/uploads/2016/11/IRIO_Design_Rules_for_LabVIEW_FPGA_v1.2.0.pdf.
- [55] IRIO Team (I2A2 Research Group). (11/2016). *IRIO EPICS Device Support User Manual v1.2.0*. Available: http://www.i2a2.upm.es/wp-content/uploads/2016/11/IRIO_EPICS_Device_Support_User_Manual_v1.2.0.pdf.
- [56] G. Abela *et al*, "ITERDB—The Data Archiving System for ITER," *Fusion Engineering and Design*, vol. 89, (5), pp. 536-541, 2014. . DOI: <http://dx.doi.org/10.1016/j.fusengdes.2014.02.025>.
- [57] R. Castro *et al*, "Data archiving system implementation in ITER's CODAC Core System," *Fusion Engineering and Design*, vol. 96, pp. 751-755, 2015. . DOI: <http://dx.doi.org/10.1016/j.fusengdes.2015.06.076>.
- [58] HDF Group. (2017). *HDF-5 Support Page*. Available: <https://support.hdfgroup.org/HDF5/>.
- [59] D. Sanz *et al*, "Implementation of intelligent data acquisition systems for fusion experiment using EPICS and FlexRIO technology," in *Real Time Conference (RT), 2012 18th IEEE-NPSS*, 2012, pp. 1-8.
- [60] M. Ruiz *et al*, "ITER Fast Plant System Controller prototype based on PXIe platform," *Fusion Eng. Des.*, vol. 87, (12), pp. 2030-2035, 12, 2012.
- [61] M. Correia *et al*, "Implementation of IEEE-1588 timing and synchronization for ATCA control and data acquisition systems," *Fusion Eng. Des.*, vol. 87, (12), pp. 2178-2181, DEC, 2012.

- [62] D. Sanz *et al*, "IEEE 1588 clock distribution for FlexRIO devices in PXIe platforms," *Fusion Eng. Des.*, vol. 89, (5), pp. 652-657, 5, 2014.
- [63] S. Esquembri *et al*, "Hardware Timestamping for an Image Acquisition System Based on FlexRIO and IEEE 1588 v2 Standard," *Nuclear Science, IEEE Transactions On*, vol. 63, (1), pp. 228-235, 2016.
- [64] NVIDIA. (11/2012). *NVIDIA Tesla K20 Board Specification*. Available: <https://www.nvidia.com/content/PDF/kepler/Tesla-K20-Passive-BD-06455-001-v05.pdf>.
- [65] NVIDIA. (2016). *NVIDIA Quadro M4000 Data Sheet*. Available: http://images.nvidia.com/content/pdf/quadro/data-sheets/75509_DS_NV_Quadro_M4000_US_NV_HR.pdf.
- [66] Keysight Technologies. (11/2015). *33500B Series Waveform Generators Data Sheet*. Available: <http://literature.cdn.keysight.com/litweb/pdf/5991-0692EN.pdf?id=2202606>.
- [67] Agilent Technologies. (11/2012). *InfiniiVision 7000A Series Oscilloscopes Data Sheet*. Available: <http://literature.cdn.keysight.com/litweb/pdf/5989-7736EN.pdf?id=1373609>.
- [68] J. Nieto *et al*, "A high throughput data acquisition and processing model for applications based on GPUs," *Fusion Eng. Des.*, vol. 96-97, pp. 895-898, OCT 2015, 2015.
- [69] J. Nieto Valhondo, "Contribuciones en sistemas de adquisición de datos inteligentes para entornos de fusión por confinamiento magnético," pp. 157, .
- [70] S. Esquembri *et al*, "Methodology for the implementation of real-time image processing systems using FPGAs and GPUs and their integration in EPICS using nominal device support," in *Proceedings of 11th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research*, Greifswald, Germany, 2017, pp. 54.
- [71] E. Barrera *et al*, "Test bed for real-time image acquisition and processing systems based on FlexRIO, CameraLink, and EPICS," *Fusion Eng. Des.*, vol. 89, (5), pp. 633-637, 5, 2014.

- [72] A. Carpeño *et al*, "Implementing a Neutron-Diagnostic advanced DAQ system use case on a PXIe platform through a 3D remote laboratory," *Fusion Eng. Des.*, .
- [73] F. Bolelli. (03/2017). YACCLAB: Yet Another Connected Components Labeling Benchmark. Available: <https://github.com/prittt/YACCLAB>.
- [74] Valgrind Developers Group. (2017). *Valgrind Home Page*. Available: <http://valgrind.org/>.
- [75] D. Marjamaki. (2017). *cppcheck Home Page*. Available: <http://cppcheck.sourceforge.net/>.
- [76] MathWorks. (2017). *Polyspace Static Analysis Home Page*. Available: <https://www.mathworks.com/products/polyspace.html>.
- [77] A. Carpeño *et al*, "3D virtual world remote laboratory to assist in designing advanced user defined DAQ systems based on FlexRIO and EPICS," *Fusion Eng. Des.*, vol. 112, pp. 1059-1062, 11/15, 2016.
- [78] D. Sanz *et al*, "Advanced data acquisition system implementation for the ITER neutron diagnostic use case using EPICS and FlexRIO technology on a PXIe platform," in *Real Time Conference (RT), 2014 19th IEEE-NPSS*, 2014, pp. 1-2.

Appendices

Appendix A: Acronyms

Abbreviation	Definition
ADC	Analog-to-Digital Converter
AMC	Advanced Mezzanine Cards
API	Application Programming Interface
APODIS	Advanced Predictor Of DISruptions
ATCA	Advanced TCA
ATM	Asynchronous Transfer Mode
BLED	Beam Line Element Databases
BOY	Best OPI Yet
CA	Channel Access
CCL	Connected Components Labeling
CCS	CODAC Core System (ITER Operating System)
CLIP	Component-Level Intellectual Property
CODAC	Control, Data Access and Communication (ITER Control System)
CoDaC	Control Data Acquisition and Communication (W 7-X Control System)
CODAS	Control and Data Acquisition Systems (JET Control System)
cPCI	Compact PCI
CPU	Central Processing Unit
CSS	Control System Studio
CUDA	Compute Unified Device Architecture

Abbreviation	Definition
DAC	Digital-to-Analog Converter
DAN	Data Archiving Network
DDB	Dynamic Data Buffer
DEMO	DEMONstration Power Station
DESY	Deutsches Elektronen Synchrotron (German Electron Synchrotron)
DMA	Direct Memory Access
DSP	Digital Signal Processor
DWT	Discrete Wavelet Transform
EPICS	Experimental Physics and Industrial Control System
ESS	European Spallation Source
FPGA	Field Programmable Gate Array
FRIB	Facility for Rare Isotope Beams
GAM	Generic Application Module
GPU	Graphic Processor Unit
GUI	Graphical User Interface
HDL	Hardware Description Language
HMI	Human Machine Interface
IAS	Image Acquisition System
ICS	Integrated Control System (ESS)
I/O	Input/Output
I&C	Instrumentation and Control

Abbreviation	Definition
IOC	Input/Output Controller
JET	Joint European Torus
KSTAR	Korea Superconducting Tokamak Advanced Research
MARTe	Multithreaded Application Real-Time executor
MEDM	Motif Editor and Display Manager
MIMO	Multiple Input Multiple Output
MRG	Message, Real-time and Grid (RHEL real-time extensions)
MTCA	Micro TCA
MSE	Motional Stark Effect
MXI	Multiplatform eXtension for Instrumentation
MXIe	MXI Express
NDArray	N-Dimensional Array
NDS	Nominal Device Support
NI	National Instruments
NRFI	National Fusion Research Institute
NSLSII	National Synchrotron Light Source II
NVCC	NVIDIA's CUDA Compiler
LM	Locked Mode
OPI	Operator Panel Interface
OS	Operating System
PCI	Peripheral Component Interconnection
PCIe	PCI Express

Abbreviation	Definition
PICMG	PCI Industrial Computer Manufacturer Group
PLC	Programmable Logic Controller
PON	Plant Operation Network
PTPv2	Precise Time Protocol Version 2 (IEEE 1588-2008)
PV	Process Variable
PXI	PCI eXtension for Instrumentation
PXIe	PXI Express
PXISA	PXI Systems Alliance
RAM	Random-Access Memory
RDMA	Remote DMA
RHEL	Red Hat Enterprise Linux
RIO	Reconfigurable Input/Output
ROI	Region-Of-Interest
RTAI	Real-Time Application Interface
RTDN	Real-Time Data Network
RTOS	Real-Time OS
SCADA	Supervisory Control And Data Acquisition
SDD	Self-Description Data
SNS	Spallation Neutron Source
SPAD	Single-signal Predictor based on Anomaly Detection
TCA	Telecommunications Computing Architecture
VI	Virtual Instrument

Abbreviation	Definition
VME	VERSA Module Eurocard
VS	Vertical Stabilization
VXI	VME eXtension for Instrumentation
W 7-X	Wendelstein 7-X
xTCA	ATCA and MTCA

Appendix B: Hardware Components Supported by the IRIO Software Tools

Table 15 List of FlexRIO Devices supported by the latest version of IRIO Library (V1.2.0)

Model	Bus	FPGA	FPGA Slices	FPGA DSP Slices	FPGA Memory (Block RAM)	Onboard Memory	Supported by IRIO Library V1.2.0
PXIe-7975R	PXIe	Kintex-7 XC7K410T	63,550	1,540	28,620 kbits	512 MB	✓
PXIe-7966R	PXIe	Virtex-5 SX95T-2	14,720	640	8,784 kbits	512 MB	✓
PXIe-7965R	PXIe	Virtex-5 SX95T	14,720	640	8,784 kbits	512 MB	✓
PXIe-7962R	PXIe	Virtex-5 SX50T	8,160	288	4,752 kbits	512 MB	✓
PXIe-7961R	PXIe	Virtex-5 SX50T	8,160	288	4,752 kbits	0 MB	✓
PXI-7954R	PXI	Virtex-5 LX110	17,280	64	4,608 kbits	128 MB	✓
PXI-7953R	PXI	Virtex-5 LX85	12,960	48	3,456 kbits	128 MB	✓
PXI-7952R	PXI	Virtex-5 LX50	7,200	48	1,728 kbits	128 MB	✓
PXI-7951R	PXI	Virtex-5 LX30	4,800	32	1,152 kbits	0 MB	✓

Table 16 List of FlexRIO Adapter Modules supported by the latest version of IRIO Library (V1.2.0)

Adapter module	Supported by IRIO Library V1.2.0
NI 5791 100 MHz Bandwidth RF Transceiver	Supported on demand
NI 5792 200 MHz Bandwidth RF Receiver	Supported on demand
NI 5793 200 MHz Bandwidth RF Transmitter	Supported on demand
NI 5781 100 MS/s Baseband Transceiver	Supported on demand
NI 5782 250 MS/s IF Transceiver	Supported on demand
NI 5731 12-Bit, 40 MS/s, 2 Channel Digitizer NI 5732 14-Bit, 80 MS/s, 2 Channel Digitizer	Supported on demand
NI 5733 16-Bit, 120 MS/s, 2 Channel Digitizer NI 5734 16-Bit, 120 MS/s, 4 Channel Digitizer	Supported on demand
NI 5751 14-Bit, 50 MS/s, 16 Channel Digitizer	Supported on demand
NI 5752 12-Bit, 50 MS/s, 32 Channel Digitizer	Supported on demand
NI 5761 14-bit, 250 MS/s, 4 Channel Digitizer	✓
NI 5762 16-Bit, 250 MS/s, 2 Channel Digitizer	Supported on demand
NI 5771 8-Bit, 3GS/s, 2 Channel Digitizer	Supported on demand
NI 5772 12-Bit, 1.6GS/s, 2-Channel Digitizer	Supported on demand
NI 5781 14-Bit, 100 MS/s, 2 Channel Baseband Transceiver	✓
AT-1120 14-Bit, 2GS/s, 1-Channel Signal Generator	Supported on demand
AT-1212 14-Bit, 1.2GS/s, 2-Channel Signal Generator	Supported on demand
NI 6581 200 Mbit/s, 54 Channel, Single Ended Digital I/O	✓
NI 6583 300 Mbit/s, 32 SE and 16 LVDS Channel Digital I/O NI 6584 16 Mbit/s, 16 Ch, RS-422/RS-485 Digital I/O	Supported on demand
NI 6585 200 Mbit/s, 32 Channel, LVDS Digital I/O NI 6587 1 Gbit/s, 20 Channel, LVDS Digital I/O	Supported on demand
NI 1483 Full Configuration Camera Link	✓

Table 17 List of CompactRIO chassis and modules supported by the latest version of IRIO Library (V1.2.0)

HW ID	Description
CompactRIO 9159	CompactRIO Chassis with Virtex 5 FPGA
CompactRIO module NI9205	Analog input Module
CompactRIO module NI9264	Analog output Module
CompactRIO module 9401	TTL Digital I/O
CompactRIO module 9477	Digital Outputs 60V sinking
CompactRIO module 9476	Digital Outputs 24V sourcing
CompactRIO module 9425	Digital Input 24V sinking
CompactRIO module 9426	Digital Input 24V sourcing