Embedding Instruments & Modules into an IEEE1451-FPGA-Based Weblab Infrastructure

http://dx.doi.org/10.3991/ijoe.v8i3.2113

Ricardo J. Costa[12], Gustavo R. Alves[1], Mário Zenha-Rela[2]
[1]Instituto Politécnico do Porto, Porto, Portugal
[2]Universidade de Coimbra, Coimbra, Portugal

***Abstract*—Adopting standard-based weblab infrastructures can be an added value for spreading their influence and acceptance in education. This paper suggests a solution based on the IEEE1451.0 Std. and FPGA technology for creating reconfigurable weblab infrastructures using Instruments and Modules (I&Ms) described through standard Hardware Description Language (HDL) files. It describes a methodology for creating and binding I&Ms into an IEEE1451-module embedded in a FPGA-based board able to be remotely controlled/accessed using IEEE1451-HTTP commands. At the end, an example of a step-motor controller module bond to that IEEE1451-module is described.**

***Index Terms*—Weblabs, Remote labs, FPGA, IEEE1451.0 Std., Step-motor controller.**

I. INTRODUCTION

Commonly used in educational literature, sciences and engineering terms have different meanings and intrinsic objectives. While sciences are closely related with research activities focused on theory postulation, engineering focus on practices and practical models' formulation [1]. Despite their differences, both require theories and models validations; otherwise these can turn out to be irrelevant, becoming just unused ideas or concepts. In Sciences & Engineering (S&E) courses this is particularly important because the effectiveness of good teaching & learning processes always requires the adoption of practical activities to prove and validate theories and models, incentivizing students to get critical attitudes to construct their own knowledge. These practical activities include the laboratory work commonly provided by the experimental work required in every S&E courses.

Due to the evolution of technology in the last decades, currently this experimental work can be provided by different laboratory types, divided according to the adopted equipment (virtual or real) and their location (local or remote) [2]. Although each type has advocators and detractors [3], real equipment able to be remotely controlled are becoming a widely used option [4], provided by the so-called weblabs that are used to complement or to replace traditional laboratories, where users have a local access to real equipment. Presently, there are several weblabs implemented in different institutions, but developers still have difficulties for providing remote access to their laboratorial equipment and experiments, probably caused by the different architectures and technologies adopted for creating the infrastructures. Therefore, to overcome this aspect, standardization is a direction already proposed in several publications [5][6][7][8], being the main objective of the Global Online Laboratory Consortium (GOLC) [9], that is creating an interoperability standard that suggests using a set of interface definitions and profiles to control, access and interoperate different weblabs.

Despite the importance of standardization, if some efforts are focused on creating a common and cheap hardware platform able to integrate sharable I&Ms, costs will be reduced and collaboration between institutions will increase. It is precisely in this aspect that this paper provides a contribution for creating standard weblabs based on a common platform supported by FPGA-technology, namely by FPGA-based boards, and based on the IEEE1451.0 Std. that describes an architecture to create and network-interface transducers, that can form the I&Ms typically used by weblabs.

Next section provides an overview about the IEEE1451.0 Std. focusing on transducers' operation modes. Section III presents the implemented infrastructure, while section IV details implementation issues to create and bind I&Ms to an IEEE1451-module embedded in a FPGA-based board. According to suggestions made in this section, section V presents an example of a step-motor controller module able to be remotely controlled using standard IEEE1451-HTTP commands. The paper ends with some conclusions and directions for future work.

II. IEEE1451.0 STD.

A. Overview

Defined in 2007, the IEEE1451.0 Std. [10] aims to network-interface transducers through an architecture based on two modules: the Transducer Interface Module (TIM), that controls Transducer Channels (TCs), and the Network Capable Application Processor (NCAP), that provides network access to the TIM and to those TCs. Each module is connected through an interface defined by another standard of the IEEE1451.x family, some already specified according to the IEEE1451.0 Std. (e.g. the IEEEp1451.6 Std. for the CANopen interface) and others intended to be modified in the future (e.g. IEEE1451.2 Std. which defines point-to-point interfaces). The behaviour and features of TIMs are described within optional or mandatory Transducer Electronic Data Sheets (TEDSs) monitored by a status register and controlled by standard low-level commands. These low-level commands, provided by the TIM, may be accessed by IEEE1451-HTTP commands implemented in the NCAP, enabling the remote control/access of TCs, whose generic information and operation modes are defined by TC-TEDSs.

B. *Operation modes*

The operation mode of a TC is defined in its TC-TEDS according to data sampling and data transmission modes. As represented in figure 1, a sampling mode defines the way a TC acquires/outputs data into/from its Data-Sets (DSs) and the data transmission mode represents the way those same data is transmitted/received to/from the NCAP.

A TC may operate in up to 5 sampling modes:

1- Trigger initiated - available for sensors and actuators, after a trigger signal they start storing (sensor) or outputting (actuator) data until all data are processed;
2- Free running without pre-trigger - available for sensors and actuators. A sensors starts storing data when, in the operating state, a trigger is received. An actuator starts outputting data after entering in the operating state. After receiving a trigger, a sensor restart storing data in the first position of the first (or unique) DS and stops when all DS(s) become full. Actuators stop their operation depending on a defined end-of-data-set operation mode, defined in the TC-TEDS, and on the reception of a trigger;
3- Free running with pre-trigger - only available for sensors, they start acquiring data to internal DSs after entering in a operating state, and stops if a trigger is received or the number of samples reached a defined pre-trigger count value defined in the TC-TEDS;
4- Continuous - similar to the Free running without pre-trigger but, after a trigger, a sensor/actuator starts working continuously according to the available DS(s) and the operation attributes defined in the TC-TEDS fields. They stop after leaving the sampling mode or after receiving a reset;
5- Immediate - Storing (sensor) or outputting (actuator) data is made only after the reception of a Read/Write TC low-level command.

Data available within DSs are transmitted to the NCAP according to 3 transmission modes:

1- Commanded - A TIM shall transmit a DS only in response to a Read TC low-level command;
2- Buffer full - Data are transmitted as soon as a DS is full without waiting for the NCAP to issue a Read TC low-level command;
3- Streaming at a fixed interval - DSs are transmitted at a fixed interval. The TIM shall stops using the current DS regardless of how much data are in it, shall begin storing data in another DS, and shall transmit the DS without waiting for a Read TC low-level command.

Based on the IEEE1451.0 architecture and, in particular, on its operation modes, the next section presents the implemented weblab infrastructure supported by an IEEE1451-module (described through HDL files) embedded in a FPGA-based board. It also details the methodology for creating and binding I&Ms required by every weblab.

III. IMPLEMENTED WEBLAB INFRASTRUCTURE

Following the IEEE1451.0 Std. foundations, the implemented solution adopted a hybrid-architecture [11] specifying the NCAP and the TIM in different devices interfaced by a serial connection, as illustrated in the figure 2 and figure 3.

The NCAP was implemented in a micro computer. It provides users' remote access to the weblab infrastructure through an IEEE1451.0 HTTP interface implemented by a software package. The TIM was implemented in a FPGA-based board to accommodate all the I&Ms required for accessing the Experiment Under Test (EUT). The decision for a solution based on a FPGA-based board was made essentially supported by four main reasons: i) it integrates several digital and analog I/O interfaces to access the EUT; ii) it can use I&Ms described through HDL files, which make them easily shared by different weblab infrastructures; iii) it can run those I&Ms modules in parallel like in a laboratory that uses traditional instrumentation; and iv) it is able to be reconfigured [11], enabling to change the entire weblab infrastructure functionality without replacing the hardware platform required to access an EUT.

According to the IEEE1451.0 Std., the NCAP and the TIM should be connected through specific protocols (e.g. Bluetooth) following another IEEE1451.x Std.. However, most of the protocols are not yet compatible with the IEEE1451.0 Std. and the NCAP-TIM connection requires the use of two additional APIs (transducer services and module communication) that, if adopted, will overload de-

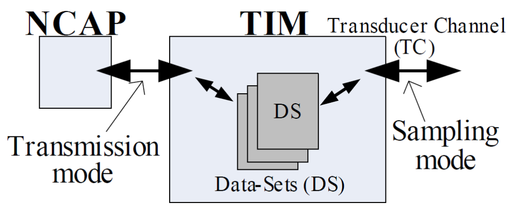
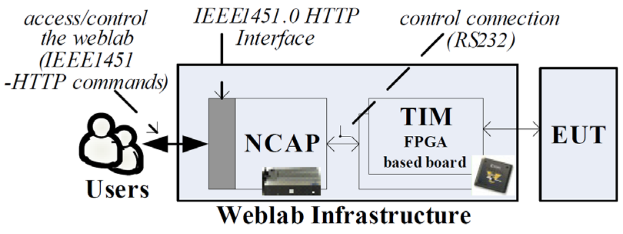


Figure 1. Conceptual diagram of the TIM's operation modes.


access/control the weblab (IEEE1451 -HTTP commands)
IEEE1451.0 HTTP Interface
control connection (RS232)
Users
NCAP
TIM
FPGA based board
EUT
Weblab Infrastructure
Figure 2. Block diagram of the implemented weblab infrastructure.

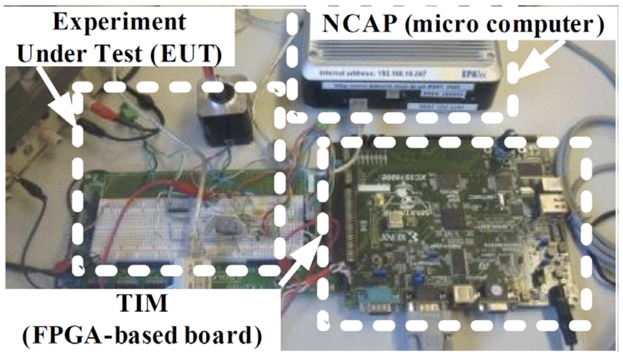


Figure 3. Picture of the implemented weblab infrastructure.

velopments and will add additional computational tasks not bringing any added value for weblab infrastructures that use single NCAP-TIM connections. Therefore, to overcome these issues, current infrastructure adopted a thin implementation, interfacing the NCAP and the TIM through a simple RS-232 connection. It was established a map between low-level commands provided by the TIM, and the IEEE1451-HTTP commands implemented by the NCAP, removing, this way, the transducer services and module communication APIs, as already suggested in [12].

A. *NCAP*

The NCAP integrates a portable software package able to be recompiled for different Linux distributions according to instructions defined in a makefile. The package comprehends a CGI application developed using the C programming language that, integrated in the Apache HTTP web server installed in the Ubuntu operational system [13], allows remote users to use IEEE1451-HTTP commands to control the TIM, and therefore every I&M. The package is organized in a set of directories each with its specific relevance, namely the *1451* directory and the *cgi-bin* directory.

The 1451 directory comprehends a set of symbolic links to access a *server.cgi* file (created after every compilation) to handle IEEE1451-HTTP commands applied according to the standard format *http://.../1451/command*. The *cgi-bin* directory contains all files and directories used by the NCAP package. It comprehends a set of source (*.c) and header (*.h) files integrating all the interfaces defined in the IEEE1451-HTTP API (e.g. IEEE1451 Teds Manager Api.c), whose accesses are made by the *server.c* file used to manage all users' requests. The package also integrates a *utils.c* file that provides some useful functions, and a file named *serial.c* used to control the NCAP serial port. Besides all these files, the NCAP has two other important directories: the first to keep cached-TEDSs which, according to the IEEE1451.0 Std., may represent copies or updates of TEDSs defined within the TIM, and another directory with files and applications required to reconfigure the TIM. This last aspect is not considered by the IEEE1451.0 Std. but, as already suggested in [12], the current NCAP package already implements two additional HTTP commands that enable reconfiguring the TIM, namely the WriteTIM and ReadTIM, both integrated in a new IEEE1451 Reconfiguration API.

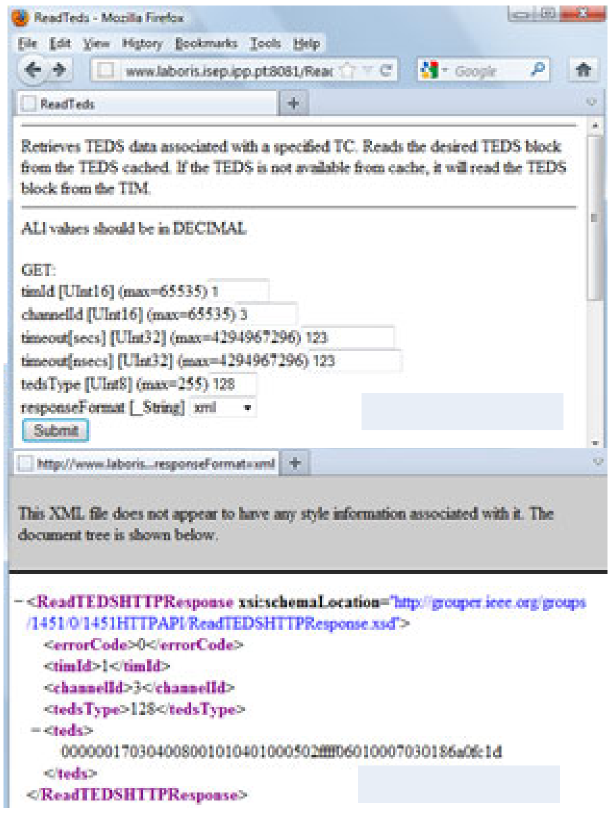
Remote users can access the TIM and the I&Ms through a simple web browser or by a software application able to transmit HTTP commands. For validation purposes, the package also provides a set of HTML pages that enables users to issue IEEE1451-HTTP commands. Figure 4 illustrates the use of the ReadTeds command to read the TEDS number 128 associated with the TC number 3. An XML format response retrieves all TEDS contents and illustrates the correct appliance of the command, as indicated by the error code number 0.

A. *TIM*

The TIM comprehends a generic IEEE1451-module with different I&Ms able to be accessed and controlled according to the IEEE1451.0 Std.. This approach is versatile since it allows binding several I&Ms able to be controlled using standard commands, and it is reusable because both the IEEE1451-module and the I&Ms are described using standard HDL files, which enable them to be embedded into different types of FPGAs.

To simplify and reduce the FPGA resources required to implement the IEEE1451-module, without hampering its operation, current version does not implement most of the optional low-level commands, enables the use of a single sampling mode, and only commanded transmission mode is available, which requires the use of the Read/Write TC low-level commands to transfer data between TIM - NCAP, and therefore to the remote users. Despite these simplifications, current IEEE1451-module is organized in a set of hardware modules so future upgrades can be easily made. As represented in figure 5, and already described in [14], it comprehends 4 modules.

The core is the decoder/controller that manages the behaviour of the whole module decoding and generating commands from/to the UART module that interfaces the NCAP through the RS232 connection. To control the behaviour of each I&M, the decoder/controller also accesses the TEDS controller and the status/states modules to read, write or update their internal memories.


ReadTeds - Mozilla Firefox
File Edit View History Bookmarks Tools Help
www.laboris.isep.ipp.pt:8081/Reac
Google
ReadTeds

Retrieves TEDS data associated with a specified TC. Reads the desired TEDS block from the TEDS cached. If the TEDS is not available from cache, it will read the TEDS block from the TIM.

ALl values should be in DECIMAL

GET:
timId [UInt16] (max=65535) 1
channelId [UInt16] (max=65535) 3
timeout[secs] [UInt32] (max=4294967296) 123
timeout[nsecs] [UInt32] (max=4294967296) 123
tedsType [UInt8] (max=255) 128
responseFormat [_String] xml
Submit

http://www.laboris...responseFormat=xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <ReadTEDSHTTPResponse xsi:schemaLocation="http://grouper.ieee.org/groups/1451/0/1451HTTPAPI/ReadTEDSHTTPResponse.xsd">
    <errorCode>0</errorCode>
    <timId>1</timId>
    <channelId>3</channelId>
    <tedsType>128</tedsType>
  - <teds>
      000000170304008001010401000502ffff06010007030186a0fc1d
    </teds>
  </ReadTEDSHTTPResponse>
```

Figure 4. ReadTeds command issued by a HTML page.

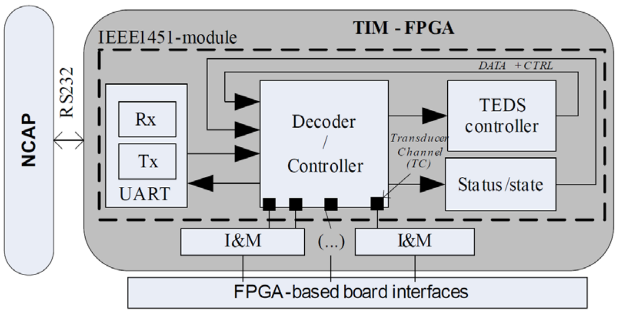


Figure 5. IEEE1451-module implemented within a FPGA.

One of the main challenges of the implemented infrastructure is to define a method to create and bind I&Ms to the decoder/controller module so they can be controlled according to the IEEE1451.0 Std..

II. CREATING AND BINDING I&MS

In a traditional instrument, several commands are required to start a measurement or read/write a specific signal from/into an external device connected to the EUT. For example, to generate a waveform signal using a Function Generator, users should, at least, define three parameters: the waveform type, the amplitude and the frequency. These or others parameters should also be controlled in similar I&Ms bound to the IEEE1451-module. The I&Ms' control is managed by TCs, whose behaviour is defined according to TEDSs, in particular by the contents of the associated TC-TEDS that are able to be changed using low-level commands (e.g. WriteTeds). Depending on the defined sampling mode, different low-level commands may be issued to a particular TC. Per example, if specific data is required to read or write, the Write/Read TC low-level commands should be applied to the correspondent TC, providing an access to their internal DSs, whose contents should have the data of the associated I&M.

Therefore, for using the proposed architecture, users should be able to create and bind the I&Ms with the IEEE1451-module. Current solution considers the use of one or more TCs and suggests an interface between the I&Ms and the IEEE1451-module. This interface is made through a set of bus lines, each associated to a particular TC controlled by TEDSs, according to a handshake protocol managed by tasks included in the decoder/controller module.

A. Required Channels

After analyzing the requirements posed by weblabs', and the IEEE1451.0 Std. specifications, two solutions may be adopted for controlling an I&M: i) using several TCs for individually control each parameter, or ii) using a single TC that may control more than one parameter through a decoding process, as illustrated in figure 6.

In the first solution (figure 6a) users individually control each I&M through several TCs. This means that an I&M requiring the control of several parameters, also requires several TCs, several buses attached to the IEEE1451-module and, at least, one TC-TEDS for each TC. Despite the possibility of a well defined control over the I&M's parameters, this solution tends to be much resource-consuming, which may become impracticable when a single FPGA is used to accommodate the IEEE1451-module and all the I&Ms.

The second solution (figure 6b) tends to be less resource-consuming (less TEDSs and a single bus line attached to the I&M), but it may require an extra-module to decode data received from the TC. Once decoded, the IEEE1451-module can select DSs and/or define the required parameters to control the I&M using the associated TC's bus lines. For example, for an I&M with more than one DS used for defining different I&M's outputs, the use of the Write TC low-level command should specify which DS and I/O will be accessed. Originally, the data fields of this command gather an offset value and data blocks that will be written into DS(s) associated to the output, displaced according to a defined offset. If the I&M has distinct DSs, whose values gather samples for generating different outputs, to fill-in these DSs using a single TC, the Write TC low-level command should specify which DS will be used, for example, by defining a specific code data inside its data blocks. This solution should be considered for all other commands when a single TC is adopted for controlling more than one parameter of an I&M.

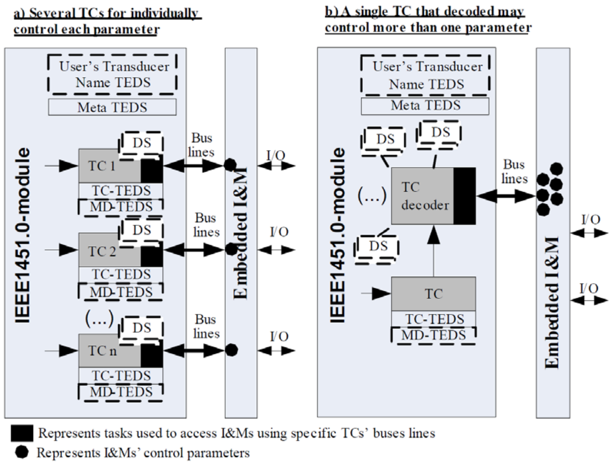

a) Several TCs for individually control each parameter
b) A single TC that decoded may control more than one parameter
User's Transducer Name TEDS
Meta TEDS
IEEE1451.0-module
DS
TC 1
TC-TEDS
MD-TEDS
TC 2
TC n
(...)
Bus lines
I/O
Embedded I&M
TC decoder
TC
Represents tasks used to access I&Ms using specific TCs' buses lines
Represents I&Ms' control parameters

Figure 6. Possibilities for controlling I&Ms through TCs.

Developers may adopt one or both solutions that, despite the difference between them, require TEDSs to characterize the behaviour of the I&M. This is the case of TCs that must be associated with a single TC-TEDS, which, for some situations, does not provide all fields required to characterize its behaviour. When this situation occurs, developers may define extra fields in the TC-TEDS or they may adopt other TEDSs, named Manufacturer Defined TEDSs (MD-TEDS). Both options are in accordance to the IEEE1451.0 Std., and satisfy the requirements posed by every I&M since they allow users to monitor or define the TCs' behaviour through specific fields. However, developers should evaluate its adoption considering the detail level of control required for each I&M and the coherence of the TC-TEDS and other associated TEDS fields with the characteristics of the I/O signals (e.g. associated units, ranges, etc.). In other words, a solution based on a single TC able to control several parameters and I/Os of a specific I&M should only be adopted if TEDS contents describe all relevant features of the associated I/Os.

Whatever the adopted solution is, the following subsections describe implementation issues that should be made both in the IEEE1451-module and in each I&M. It focuses on defining a set of internal tasks for each adopted TC in the decoder/controller of the IEEE1451-module to manage a handshake protocol used to control a set of bus lines.

B. Interface tasks

Each I&M is defined by mandatory and optional tasks, and by modules describing the I&M itself, which mainly integrate processing units and internal buffers. According to the association illustrated in figure 7, these tasks are embedded into the decoder/controller module and accessed when a specific low-level command is applied.

The decoder/controller module automatically decodes the commands and the target TC, and accesses the correspondent task to control the parameters of the I&M according to internal TEDSs' definitions that include the

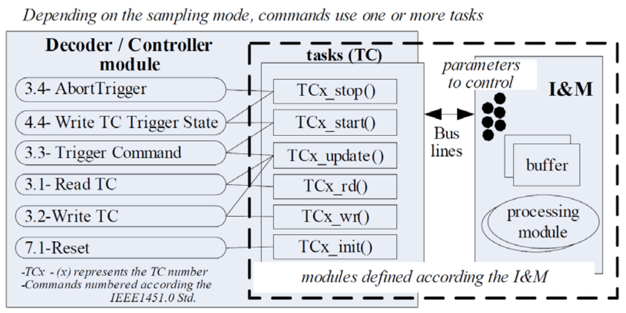

Depending on the sampling mode, commands use one or more tasks
Decoder / Controller module
3.4- AbortTrigger
4.4- Write TC Trigger State
3.3- Trigger Command
3.1- Read TC
3.2-Write TC
7.1-Reset
-TCx - (x) represents the TC number
-Commands numbered according the IEEE1451.0 Std.
tasks (TC)
TCx_stop()
TCx_start()
TCx_update()
TCx_rd()
TCx_wr()
TCx_init()
parameters to control
Bus lines
I&M
buffer
processing module
modules defined according the I&M

Figure 7. Association between low-level commands and tasks required to control each TC bond to the I&M.

adopted sampling mode. Tasks' internal description comprehends directives to access the TEDS controller and the status/state modules to read, write or update internal information according to the operation of a specific I&M, and the UART module to enable the transmission of information to/from the NCAP.

Six tasks (some optional), may be specified for each TC according to the adopted sampling mode defined in the TC-TEDS:

- TCx_stop() and TCx_start() - [optional] - used for I&Ms that adopt sampling modes with triggers. It start/stops the operation of a specific I&M indicating it can start acquiring or sampling data to/from their internal buffer. Accessed by low-level commands 4.4, 3.4 and 3.3.;
- TCx_rd - [optional] - performs a read operation. Buffers are copied into internal DSs. Accessed by the low-level command 3.1;
- TCx_wr - [optional] - only applied to I&Ms acting as actuators, performs a write operation. If the sampling is trigger-dependent, i.e. the *TCx_start* was previously used, this instruction will send data available within buffers directly to the output. If it is in a sampling mode not trigger-dependent, data sent by NCAP will be directly copied into the buffers and outputted. Accessed by the low-level command 3.2;
- TCx_init - [required] - initializes the TC by accessing its TEDS contents and defining its current operation (e.g. sampling times). Accessed by the low-level command 7.1;
- TCx_update() - [required] - updates the TC operation based on TEDS contents. Accessed by low-level commands 3.1, 3.2 and 3.3.

The access to the TEDS controller and status/state modules, and to all the I&Ms are made by controlling a set of bus lines using a handshake protocol.

C. Bus lines and handshake protocol

Since the use of an I&M in the suggested infrastructure requires defining: i) its HDL modules and ii) the tasks described in the previous subsection, it is possible to use any kind of handshake protocol to interface the HDL modules and the IEEE1451-module. A possible one is the Wishbone bus [15] that is an open source hardware computer bus typically used to interface different modules within an FPGA. However, to simplify the current implementation and to reduce the required FPGA's resources, we have decided to use the same solution adopted for interfacing the TEDS controller and the status/state modules with the IEEE1451-module. In current solution, the I&Ms are controlled/accessed through TCs using set of bus signals (some optional others required) described in table I. Table II describes 3 other signals that should be used to interface the I&Ms with the FPGA-based board. A complementary illustration of all these signals is presented in figure 8.

The IEEE1451-module acts as a master, whose slaves are all I&Ms bond through one or more TCs that implement a handshake protocol controlled by the defined tasks. The data handshake can start by a reception of a low-level

TABLE I. SIGNALS USED TO INTERFACE TCS AND I&MS.

Name	Description
Signals for interfacing a particular TC	
clk	[required-out] clk signal used for synchronizing data transmission between a particular TC and the I&M.
out [7:0]	[optional-out] data send from the TC to the I&M (depending on the I&M data within TC's DS(s) may go to the I&M's buffers).
in [7:0]	[optional-in] data received in a TC from the I&M. (depending on the I&M's data buffers may go to the TC's DS(s)).
run	[required-out] starts the handshake.
end_	[required-in] indicates the end of the handshake.
event_	[optional-in] indicates an I&M's event. (only for I&Ms working as event-sensors).
During handshaking an extra synchronization may be required. This way, the following *exec* and *done* signals can control that synchronization through operation-steps. For their appliance, the *run* signal should stay at high.	
exe	[optional-out] executes a step-operation.
done	[optional-in] indicates that a step-operation has finished.
access [n:0]	[optional-input] controls a specific I&M's operation changing the data meaning of *in* and *out* bus signals:
Signal shared by all TCs	
en	[required-out] enables the I&M.
rst	[required-out] initializes the I&M.
error [n:0]	[optional-in] errors codes defined according to the I&M. Tasks should map these codes to the IEEE1451.0 codes.

access	out [7:0]	in [7:0]
= 0	represents data to write into the I&M.	represents data read from the I&M
≠ 0	represents the instruction code for the I&M (depends on the I&M implementation)	represents the reply code instruction issued to the I&M (depends on the I&M implementation)

TABLE II. SIGNALS USED TO INTERFACE I&MS AND THE FPGA-BASED BOARD.

Name	Description
clk_ external	[required-in] clk signal used for defining the operation frequency of an I&M.
data_in [n:0]	[Optional-input] I&M input.
data_out [n:0]	[Optional-output] I&M output.

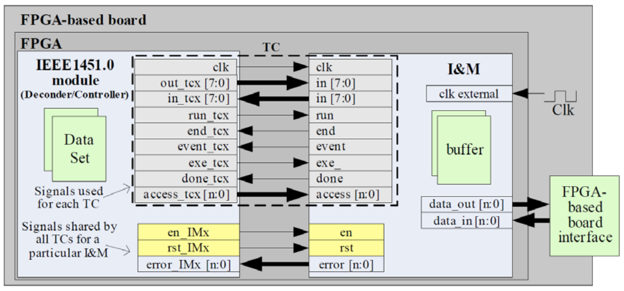


Figure 8. Interface between the IEEE1451-module and the I&Ms.

command or by an I&M's event signal, when the associated TC is defined as an event sensor.

During the handshake protocol, the synchronization between the I&Ms and the tasks is made according to a rate defined by the *clk* signal, a set of signals to run an operation (*run*/*end*), eventually executing step-operations (*exec*/*done*), and the optional *access* bus that may specify instruction codes to define a particular operation on the I&M, which is particularly important for solutions that adopt a single TC to control several I&M's parameters. Figure 9 illustrates these signals, exemplifying the case of a single TC controlling an I&M. In the example, a specific task starts the operation raising the *run* signal that may be caused by a reception of a low-level command or the rise of the *event* signal, when adopted. The operation is then synchronized using the *exec* and *done* signals that allow the IEEE1451-module to read or write data using the *in*, *out* or *access* buses. The operation ends when the *end* signal goes up.

The behaviour of an I&M is defined according to TEDSs' contents available within the TEDS controller module. Depending on the operation mode defined for a particular TC and the applied low-level command, a specific procedure should be used to read TEDS, configure the I&Ms parameters, and to transfer data between DSs and buffers available within each I&M. Figures 10 and 11 illustrate, for each task, the operation sequences when a specific command or event signal is received, illustrating the accessed tasks and the adopted signals used to interface the I&Ms and the IEEE1451-module.

III. STEP-MOTOR CONTROLLER EXAMPLE

A controller for a bipolar step-motor described according to the previous architecture was bond to the IEEE1451-module[1]. Since the control of this step-motor requires six output digital lines, whose sequence defines a specific step, it was adopted a solution based on a single TC, which allowed simplifying the design and reducing the required FPGA resources. As illustrated in figure 12, the step-motor controller comprehends a set of modules (mpp1, mpp2 and clk generator) controlled according to TEDSs and tasks used to establish the interface between those modules and the decoder/controller integrated within IEEE1451-module.

The behaviour and the features of the step-motor controller were defined by a Meta-TEDS and two other TEDSs, namely a TC-TEDS and a MD-TEDS. Within the TC-TEDS, the TC was defined as an actuator using the continuous operation mode associating 6 digital outputs for the step-motor sequences, among other parameters. Additional parameters namely: the direction, number and step modes, and a time divider to control the speed, were defined by a MD-TEDS identified by the number 80 (hex.) represented in table III and described through different fields:

- TEDS identification - field 3 uses 4 octets - identifies the TEDS using 4 octets: i) IEEE1451 standard family number (0 for this standard); ii) TEDS class (80 hex.); iii) Version number (01); and iv) tuple length (01).

[1] The binding was made through a reconfiguration tool not described in this paper.

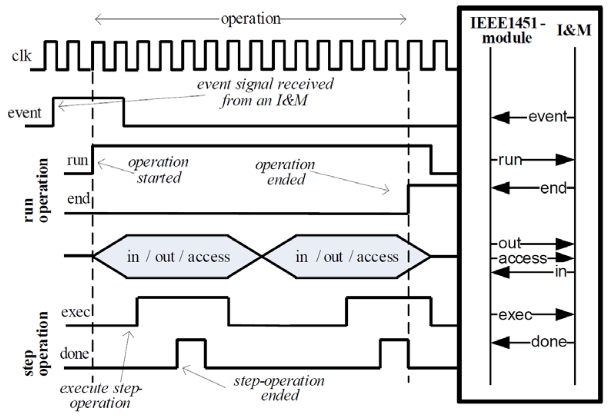


Figure 9. Example of the handshake protocol.

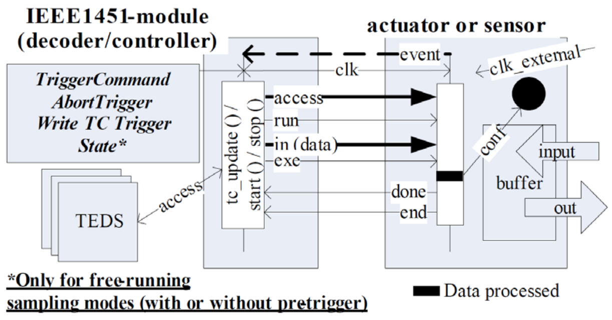


Figure 10. Abort trigger or Write trigger low-level commands during the handshake protocol.

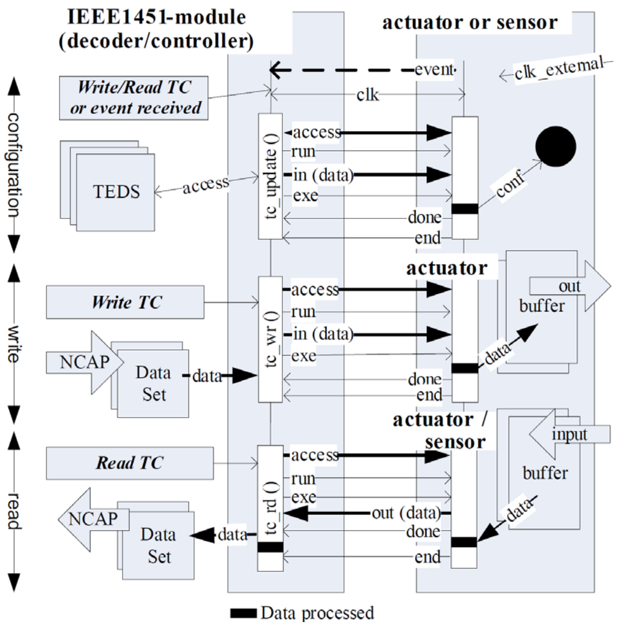


Figure 11. Adoption of the Update, Write and Read tasks during the handshake protocol.

- Direction - field 4 uses 1 octet - indicates the direction of the step-motor (0-left, 1-write);
- Number of steps - field 5 uses 2 octets - indicates the number of steps the motor will do after receiving a trigger signal. If the value is set to its maximum (FF.FF hex) this field becomes irrelevant and the step sequences starts being generated continuously;

- Step mode - field 6 uses 1 octet - defines the type of sequences generated by the controller (0-half step mode; 1-normal drive mode; or 2-wave drive mode);
- Time divider (speed) - field 7 uses 3 octets - defines the internal clk rate of the module, and therefore the speed of the generated sequences used to control a step-motor.

Since this module adopted the continuous operation mode, when a trigger signal is received it starts generating the step-motor sequences through its output lines. It stops after receiving another trigger command signal, namely the optional low-level command named *AbortTrigger* that was implemented in current solution. These commands use the *init*, *update*, *start* and *stop* triggers tasks included into the decoder/controller module. While the *init* and *update* tasks have the main objective of reading the associated TEDS and update the parameters of the step-motor controller, the *start* and *stop* tasks are accessed every time the TIM receives a *Trigger* or the *AbortTrigger* low-level commands. Therefore, to control the step-motor using the IEEE1451-HTTP commands, users should start defining the MD-TEDS contents before applying a trigger command. On each trigger command, the decoder/controller module calls the update task to read the TC-TEDS, and in particular the MD-TEDS contents, in order to define the parameters of the step-motor controller. This way, current operation of the step-motor controller is always available in the MD-TEDS, so remote users may update or query its state using the Write or Read TEDS IEEE1451-HTTP commands.

IV. CONCLUSIONS

Technology evolution and educational requirements in S&E courses led, in the last decades, to a widespread of weblabs. These are supported by infrastructures developed according to different architectures with independent I&Ms that bring network interfaces to enable their remote control/access. Traditionally, some of these I&Ms are expensive, may bring features not required for running specific experiments, and they are implemented by independent and predefined hardware components. This last aspect difficults sharing them by different infrastructures, and does not allow the redefinition of their features according to the requirements of a specific experiment. Although current infrastructures allow sharing remote experiments, contributing for joining efforts in the promotion of better experimental work activities, the collaboration among the involved institutions can be extended to the development phases by adopting: i) a standard solution supported by a common hardware platform and, ii) a methodology for creating and binding I&Ms to the weblab infrastructure. The advantages a solution with these features can bring, led us to create the weblab infrastructure described in this paper.

The weblab infrastructure was developed according to the foundations of the IEEE1451.0 Std. using hardware platforms based on FPGA-based boards. The infrastructure adopts FPGAs to embed I&Ms described through HDL files, taking the advantage to their reconfigurable nature and the ability they have on running different modules in parallel. Additionally, the different layers handled by the IEEE1451.0 Std., that go from hardware to software levels integrating APIs, guaranties a standard solution for controlling/accessing the I&Ms, which can extend the collaboration among institutions during the development phases. It was suggested a simplified architecture supported on the foundations of the IEEE1451.0 Std. using an FPGA-based board to create a flexible solution able to be reconfigured with different I&Ms. A special emphasis was given to the modules embedded into the FPGA, suggesting a methodology for binding a IEEE1451-module with I&Ms, so they can be controlled/accessed using IEEE1451-HTTP commands. To validate the infrastructure and the suggested methodology, we have an example of a step-motor controller module bond with the IEEE1451-module able to be remotely controlled using IEEE1451-HTTP commands.

Current version of the IEEE1451-module still requires specific knowledge on how-to interface the decoder/controller module with the TEDS controller and the state/status modules, which may create some difficulties for developing the suggested tasks to interface the I&Ms. Improvements to this issue will be made in the future, by the definition of a simplified API. Once concluded, it is our intention to invite scientific community to develop others I&Ms compatible with the IEEE1451-module and, therefore, able to be controlled/accessed using IEEE1451-HTTP commands. These I&Ms can then be provided by a

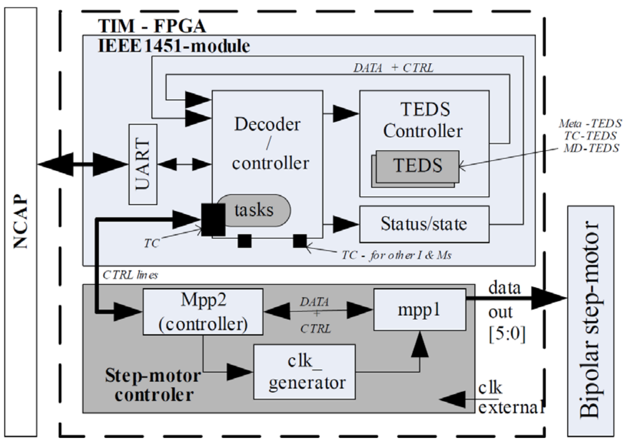


Figure 12. Step-motor controller bond to the IEEE1451-module.

TABLE III. MD-TEDS FOR THE STEP-MOTOR MODULE.

Field num.	Description	Data Type	octets	Value (hex)
-	TEDS length	UInt32	4	00.00.00.17
0-2	Reserved	-	-	-
3	TEDS identification: Family (=0) Class (=80hex) Version (=1) Tuple Length (=1)	UInt8	4	00.80.01.01
4	Direction 0-left or 1-write	UInt8	1	01
5	Number of steps	UInt8	2	FF.FF
6	Step mode defined according to three modes: 0-half step mode; 1-normal drive mode; 2-wave drive mode.	UInt8	1	00
7	Time divider (Speed) Defines the higher level of the clk signal generated in the step-motor controller module. Value=0.5(Freq.clk/TimeDiv.)	UInt8	3	01.86.A0
-	Checksum	UInt16	2	FC.1D

web portal so different institutions may adopt them in their weblab infrastructures created according to the suggestion described in this paper. This way, a worldwide workbench will be available, enabling users to select the required I&M to use in their experiments, without costs.

REFERENCES

[1] [1] Norrie S. Edward, 'The role of laboratory work in engineering education: student and staff perceptions', *International Journal of Electrical Engineering Education*, vol. 39, no. 1, pp. 11–19, Jan. 2002.

[2] [2] Gomes, L. and Bogosyan, S., 'Current Trends in Remote Laboratories', *IEEE Transactions on Industrial Electronics*, vol. 56, no. 12, pp. 4744–4756, Dec. 2009. http://dx.doi.org/10.1109/TIE.2009.2033293

[3] [3] J. Ma and J.V. Nickerson, 'Hands-on, simulated, and remote laboratories: A comparative literature review', *ACM Computing Surveys*, vol. 38, no. 3, p. 7, 2006. http://dx.doi.org/10.1145/1132960.1132961

[4] [4] Javier García Zubía and Gustavo R. Alves, *Using Remote Labs in Education - Two Little Ducks in Remote Experimentation*, 1 vols. University of Deusto Publications, 2011.

[5] [5] A. Agrawal and S. Srivastava, 'WebLab: A Generic Architecture for Remote Laboratories', pp. 301–306, 2007.

[6] [6] Franco Davoli et al., 'Remote Instrumentation and Virtual Laboratories - Service Architecture and Networking', *Springer*, 2010.

[7] [7] V. J. Harward et al., 'The iLab Shared Architecture AWeb Services Infrastructure to Build Communities of Internet Accessible Laboratories', *Proceedings of the IEEE*, vol. 96, no. 6, p. 20, Jun. 2008. http://dx.doi.org/10.1109/JPROC.2008.921607

[8] [8] I. Gustavsson et al., 'The VISIR project – an Open Sources Software Initiative for Distributed Online Laboratories', *Conference on Remote Engineering and Virtual Instrumentation (REV'07)*, p. 6, Porto - Portugal 2007.

[9] [9] GOLC - Global Online Laboratory Consortium, [Online]. Available: http://online-lab.org/. [Accessed: 26-Jan-2012].

[10] [10] IEEE 1451.0 Std., 'Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats', *The Institute of Electrical and Electronics Engineers, Inc.*, p. 335, 2007.

[11] [11] R. J. Costa et al., 'FPGA-based Weblab Infrastructures - Guidelines and a prototype implementation example', *3rd IEEE International Conference on e-Learning in Industrial Electronics (ICELIE'2009), Porto - Portugal*, p. 7, Nov. 2009.

[12] [12] Ricardo J. Costa, Gustavo R. Alves and Mário Zenha-Rela, 'Extending the IEEE1451.0 Std. to serve distributed weblab architectures', in *1st Experiment@ International Conference (exp.at'11), Calouste Gulbenkian Foundation, Lisboa-Portugal*, http://ave.dee.isep.ipp.pt/~rjc/, 2011.

[13] [13] Ubuntu operating system, [Online]. Available: http://www.ubuntu.com/. [Accessed: 24-Jan-2012].

[14] [14] Ricardo Costa, Gustavo Alves and Mário Zenha-Rela, 'Work-in-progress on a thin IEEE1451.0-architecture to implement reconfigurable weblab infrastructures', *International Journal of Online Engineering (iJOE)*, vol. 7, no. 3, 2011.

[15] [15] 'WISHBONE System - on - Chip (SoC) Interconnection Architecture for Portable IP Cores', *http://opencores.org/opencores,wishbone*, p. 128, 2010.

AUTHORS

Ricardo Costa is an Assistant Professor at the Polytechnic of Porto, School of Engineering (IPP/ISEP), and he is a PhD. Student at the Faculty of Sciences and Technology of the University of Coimbra (FCTUC) (e-mail: rjc@isep.ipp.pt / rjc@dei.uc.pt).

Gustavo Alves is a full Professor at the Polytechnic of Porto, School of Engineering (IPP/ISEP), (e-mail: gca@isep.ipp.pt).

Mário-Zenha Rela is an Auxiliary Professor at the Faculty of Sciences and Technology of the University of Coimbra (FCTUC) (e-mail: mzrela@dei.uc.pt).