A Hardware Library for Sensors/Actuators Interfaces in Sensor Networks

J. Portilla, J.L. Buron, T. Riesgo
Departamento de Automática, Ingeniería Electrónica e Informática Industrial, DIE
Universidad Politécnica de Madrid
Madrid, Spain
e-mail: {jportilla, triesgo}@etsii.upm.es

A. de Castro
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
Madrid, Spain
e-mail: angel.decastro@uam.es

***Abstract*— Sensor networks have reached a great relevance during the last years. The idea is to use a high number of nodes measuring different physical parameters in several environments, which implies different research challenges (low power consumption, communication protocols, platform hardware design, etc). There is a tendency to use modular hardware nodes in order to make easier rapid prototyping as well as to be able to redesign faster and reuse part of the hardware modules. One of the main obstacles for rapid prototyping is that sensors present heterogeneous interfaces. In this paper, a VHDL library for sensors/actuators interfaces is proposed in order to have a set of different sensor interfaces that include the most common in the sensors/actuators world, enabling a rapid connection to a new sensor/actuator.**

I. INTRODUCTION

Sensor networks discipline has reached a notable importance recently. Several fields are involved here, like sensor technology, communication protocols, low power consumption techniques, hardware design of the nodes, algorithms, etc [1]. The tendency is promising, and it is expected that the sensor networks market will grow up to $43 billion in 2008 [2].

The hardware node design becomes critical in sensor networks, in order to achieve the targets commented before. In this way, several approaches exist in the state of the art, but there is a tendency to make the node hardware platform modular [3], [4], [5]. With a modular approach, it is easy to redesign the platform to adapt the system to different scenarios and applications. Moreover, modularity allows rapid prototyping. This concept was developed in previous works of this research group, and a modular platform is actually available as a niche for researching and developing [6] (Fig. 4).

This modular platform is divided in four functional layers: communication, processing, power supply and sensing/actuating layer. The processing layer includes a µC and an FPGA, which gives much processing power to the platform, as well as flexibility. Modularity in the hardware node must be combined with flexibility in the processing devices in order to obtain the maximum adaptability. In this context, sensors present heterogeneous interfaces, which make difficult developing applications in a fast way. When adopting a new sensor, most of the work must be started from scratch.

Different works have been done in order to minimize this fact, the most of them with a software point of view [7]. It would be desirable to standardize in some manner sensor interfaces to accelerate development time. Some efforts have been done as the IEEE 1451 standards family, for smart sensors [8].

In this paper, a VHDL library for transducers (sensors or actuators) interfaces is presented, which allows implementation in any custom hardware device (FPGA or ASIC). Different common interfaces for actual sensors have been chosen, like PWM or I2C among others. The purpose is to minimize redesign time when new sensors must be integrated in the system, because the interface is standardized. The designer treats each transducer as a "channel", but all the channels are treated in the same way independently of the actual interface that transducer has.

II. LIBRARY OF INTERFACES FOR SENSORS/ACTUATORS

At the present time, there are a lot of different transducers (sensors/actuators) interfaces. Although there are some interfaces that have reached much diffusion, like I2C, probably there will never be a common unique interface for all the transducers due to commercial interests and special features of every transducer or application.

In order to simplify the connection and use of transducers, it would be desirable to have a library with the most common interfaces for sensors, and to improve and to extend this library with more interfaces for new transducers in the market. This situation would make easier rapid prototyping and redesign.

A VHDL library has been developed in order to achieve this purpose. Different interfaces have been described in VHDL, which makes the solution independent of the final implementation as long as it is based on custom hardware (FPGA or ASIC). The library is composed of different modules which deal directly with every transducer (analog or digital) and present a common interface with the rest of the circuit independently of the specific transducer. The transducer is connected to the corresponding module of the library, which is different for every interface, and finally this

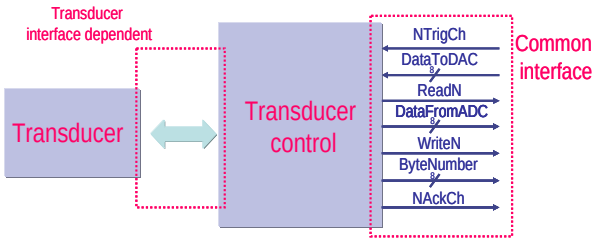

Fig.1. General structure chosen for the control of a transducer.

module is connected through the common interface to the rest of the circuit (Fig. 1). The signals that compose this interface are the following ones:

- Inputs
  - *NTrigCh* (1 bit): Used to request the module to take a measure from a sensor or to write a value in an actuator.
  - *DataToDAC* (8 bits): The data to be written in the actuator is received by this bus (not included in sensor channels).
- Outputs
  - *NAckCh* (1 bit): Acknowledge that shows that a sensor has been read or a value has been written in an actuator.
  - *ReadN* (1 bit): Used to request data to be written in the actuator (not included in sensor channels).
  - *WriteN* (1 bit): Used to alert that a value read from a sensor is put in the bus *DataFromADC* (not included in actuator channels).
  - *ByteNumber* (8 bits): Used to assign a number to each byte sent by *DataFromADC* or received from *DataToDAC*. Each data can include up to 256 bytes.
  - *DataFromADC* (8 bits): The data to be read from the sensors is received by this bus (not included in actuator channels).

Every module has been designed following a philosophy inspired in the IEEE 1451 family of standards, but can also be used without being compatible with them. Each transducer is "seen" as a channel (or set of channels) by the transducer controller. Two kinds of channels are recognized: sensor channel and actuator channel. Some sensors, like the SHT11 from Sensirion, supply different measures (in this case, humidity and temperature). So, for the same sensor two different channels are needed. This will be explained in further detail in the next section.

The heart of every module is a set of nested FSMs (Finite States Machines). At the top, there is a principal state machine, *TrigStates* (Fig. 2), common to every module, which responds to the trigger signal. It controls the lower level FSMs and the acknowledge signal. This FSM follows the 1451 philosophy. At a second level there is another FSM, *DataStates*, that controls the data acquisition process and the communication through the common interface with the rest of the system. This second FSM exists in all the modules of the library, but changes from module to module depending

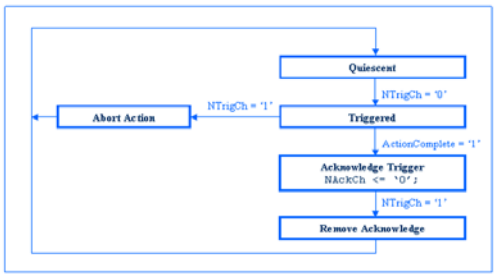

Fig.2. Top FSM, *TrigStates*.

on the specific needs of each interface. Finally, some modules use a third or even fourth FSM if the communication with the transducer is complex. This will be commented in section III.

The different modules developed in the proposed library are used for the following transducer interfaces:

- PWM
- Frequency/Period modulation
- I2C
- Sensirion interface (similar to I2C)
- Interface for analog transducers (module to interact with ADCs and DACs).

There are other notable interfaces to be developed, like SPI and 1-Wire. These will be added in the future.

III. TRANSDUCER INTERFACES

In this section, more detailed information of every transducer interface is given.

A. *PWM*

Some sensors in the market give their measurements using PWM (Pulse Width Modulation). Basically, the sensor generates a signal with fixed period and variable duty cycle depending on the value of the measurement.

Reading this signal demands a lot of resources if done via software. The microprocessor should be reading it continuously during at least one cycle, which is usually in the order of tenths or thousands of µs, or even ms, stopping the rest of tasks during this time. In the other side, an FPGA can process this signal in parallel with the rest of system, without stopping it, and with higher precision thanks to its high processing speed. Therefore, this is a good example of the benefits of implementing transducer interfaces in hardware.

A module for the family of accelerometers ADXL from Analog Devices has been developed. This module may be adapted to any other sensor with PWM interface with a minimal effort.

The acceleration would be normally calculated using the value T_1/T_2 (where T_1 is the time during which the output signal is '1' and T_2 is the period) which implies including a divider in the design, that consumes many resources in hardware terms. In order to avoid the divider, it has been

supposed that T_2 is constant. This supposition is valid once R_{SET} is fixed. A parameter that represents T_2 is included in the module as a constant, so adjusting the module to a different period only needs correcting this constant, but there is no divider circuit.

B. *Period/Frequency*

There are a lot of sensors whose output is codified in period or frequency. The library includes modules designed for these coding strategies, which have been applied to two specific temperature sensors. These sensors are MAX6576 and MAX6577 from Maxim. The former gives the temperature codified in period and the latter in frequency. The modules in the library are easily adaptable to other sensors with similar outputs.

If a period interface is used, the measurement consists on counting the number of clock cycles in each output cycle. In the case of the frequency coding strategy, the same interface could be used, calculating the inverse of the period. However, in order to avoid the divider, which demands a lot of hardware resources, a different measurement method has been used. It is based on counting the number of output cycles in a certain constant time, as this number is proportional to the frequency.

C. *I2C*

I2C (Inter-Integrated Circuit) is a serial bidirectional bus developed by Philips which uses two lines (SCL or clock and SDA or data). It was thought to make easier the communication between peripherals in a motherboard or an embedded system.

This kind of interface is very usual in sensors today, probably the most popular, so it has been included in the library. As an example, a module for the DS1629 temperature sensor from Maxim has been designed. This temperature sensor also includes a real time clock.

The sensor is treated as two sensor channels (temperature and real time clock) and an actuator channel (for programming the clock), using the IEEE 1451 philosophy: including a different channel for each functionality (Fig. 4).

Due to the complexity of this interface compared to the previous ones, the structure of this module uses four levels of nested FSMs. Each level has the next functionality:

- Attending triggers from each channel. There is a separate *TrigStates* FSM for each channel but, as the I2C interface is shared, if all the channels are triggered simultaneously they receive access to the interface depending on their priority level, which is set in the module.
- Managing the requests to the sensor. This second level (*DataStates*) is common in all the modules of the library. The difference is that, in this case, it is not in charge of controlling directly the interface signals due to the complexity of the process. Its task is managing the FSMs at lower levels.

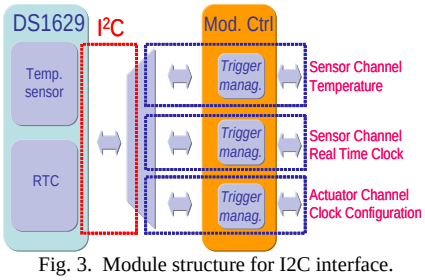

Fig. 3. Module structure for I2C interface.

- Sending and receiving I2C frames. The third level is in charge of managing the frames sent or received from the sensor. Two different FSMs control each kind of frame: *ReadStates* for receiving data and *WriteStates* for sending data.
- Sending and receiving bytes. As all the information in I2C is sent in separate bytes, the task of sending or receiving each individual byte is done in this fourth level. Again, there are two FSMs: *SendStates*, for sending a byte to the sensor, and *ReceiveStates*, for receiving a byte from the sensor.

D. *Sensirion interface (similar to I2C)*

There is a series of sensors (SHT1x/SHT7X) from Sensirion that use an interface similar to I2C. These sensors are very common in sensor networks applications due to their low size and digital output. The Sensirion interface presents mainly the following differences from I2C. Signals are named SCK and DATA, instead of SCL and SDA. SCK is not open-drain, and its default value is '0' instead of '1'. The "start" sequence is also different and there is no "stop" sequence in the Sensirion interface. The communication finishes when there is no acknowledge to a byte.

The chosen sensor, SHT11, is treated as two different channels, because it includes two different measurements: temperature and humidity. In this case, a three level FSMs strategy was used, instead of the four level strategy in the I2C module. However, the idea is almost the same.

E. *Analog transducers*

Many sensors in the market have analog outputs. Because of this, the interfaces library must include a way of dealing with these sensors.

All the previous modules deal with digital signals, so a VHDL implementation can be in charge of them directly. However, for analog transducers an ADC or a DAC will be necessary, depending on whether it is a sensor or an actuator respectively. Therefore, the control of an analog transducer is equivalent to the control of an ADC or a DAC.

In the library, modules for controlling two different ADCs and a DAC have been included. The ADCs have been chosen because they represent the two most usual interfaces in ADCs. Next, every control module is explained in more detail:

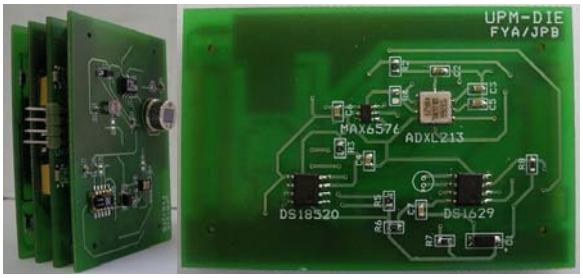

Fig.4. Modular platform for wireless sensor networks and sensor layer with ADXL213 accelerometer, MAX 6576, DS18S20 and DS1629 temperature sensors.

a) AD0808: this ADC has 8 analog multiplexed inputs. This is a problem if every analog input has to be defined as a sensor channel. The reason is that the number of signals in the interface with the rest of the system would be multiplied by 8. The solution was to include an actuator channel in order to tell the interface module which input must be used at each time. In this way, the controller can set the active input from the ADC. Regarding the conversion interface, it is controlled through two signals: "start of conversion" and "end of conversion".

b) HI5805: this ADC converts its input continuously. It only needs an external clock input (0.5-5 MHz) as interface. The analog to digital conversion is done using a pipelined flash structure, which introduces a latency of 3 clock cycles. This is taken into account in the module to synchronize the measurement with high accuracy.

c) DAC8562: this DAC has a 12-bit parallel input. It has a 12-bit latch controlled by the CE signal, and an additional clear signal.

All these three modules include two FSMs, *TrigStates* and *DataStates*, as explained in the previous section.

IV. Verification and Experimental Results

All the modules in the library have been tested using a Xilinx Spartan 3 XC3S200 FPGA. But before starting with the experimental verification, exhaustive simulations have been carried out for each module. In order to do so, a functional model of each sensor has been written in VHDL. This model is connected with its respective module, and the test bench generates the different values that are measured by the model of the sensor. Furthermore, the test bench is also in charge of the signals for every channel, such as the trigger.

Then, two kinds of simulations have been accomplished for every module. The first simulation is a simple one, just to know if the module works correctly with its sensor. This simulation was verified watching the waveforms. The second simulation guarantees that the module works appropriately even in strange conditions. For instance, conditions such as simultaneous triggers of different channels, rapid changes in the sensor values for synchronization verification, errors in the sensor communication, etc, were tested. Very long simulations were necessary for these exhaustive verifications, so visual inspection of waveforms was inappropriate. Therefore, automatic verification mechanisms were included in the test bench, making use of VHDL procedures and functions. All these conditions were reported through text messages. The simulation tool was ModelSim.

After simulations were correct, experimental tests were made in hardware using real sensors. In this way, the modules were synthesized in the FPGA and the different sensors were attached to the system. In order to complete the hardware verification, additional VHDL blocks were developed for managing the channels of each module and showing the results through LEDs and displays.

The last step is to include these interfaces in the modular platform for wireless sensor networks (Fig. 4).

V. Conclusions

A VHDL library for sensor/actuator interfaces has been presented. The library is used on a modular platform for wireless sensor networks. The library makes the sensor almost transparent for the user, who always sees the same signals independently of the sensor being used.

Exhaustive simulations have been carried out in order to make sure that the modules are correct. Furthermore, experimental tests have also been accomplished for every module in the library. Finally, the transducer interface modules have been integrated in the modular platform for wireless sensor networks.

References

[1] Chee-Yee Chong, Srikanta P. Kumar, "Sensor Networks: Evolution, Opportunities and Challenges," Proc. of the IEEE, vol. 91, Nº 8, Aug. 2003, pp. 1247–1256.

[2] Y. Zhang, Y. Gu, V. Vlatkovic, X. Wang, "Progress of Smart Sensors and Smart Sensor Networks," Proc. of the 5th IEEE World Congress on Inteligent Control and Automation, Jun. 2004, pp. 3600–3606.

[3] A.Y. Benbasat, J.A. Paradiso, "A Compact Modular Wireless Sensor Platform," Proc. of the 4th IEEE International Symposium on Information Processing in Sensor Networks, Apr. 2005, pp. 410–415.

[4] L. Nachman, R. Kling, R. Adler, J. Huang, V. Hummel, "The Intel Mote Platform: a Bluetooth-Based Sensor Network for Industrial Monitoring," Proceedings of the 4th IEEE International Symposium on Information Processing in Sensor Networks, Apr. 2005, pp. 437-442.

[5] S. Yamshita, T. Shimura, K. Aiki, K. Ara, Y. Ogata, I. Shimokawa, T. Tanaka, H. Kuriyama, K. Shimada, K. Yano, "A 15x15, 1 µA, Reliable Sensor-Net Module: Enabling Application-Specific Nodes," Proc. of the 5th IEEE/ACM International Conference on Information Processing in Sensor Networks, Apr. 2006, pp. 383–390.

[6] Kept blank for blind revision.

[7] D. Chu, K. Lin, A. Linares, G. Nguyen, J. M. Hellerstein, "Sdlib: a Sensor Network Data and Communications Library for Rapid and Robust Application Development," Proc. of the 5th IEEE/ACM International Conference on Information Processing in Sensor Networks, April 2006, pp. 432–440.

[8] K. B. Lee, R. D. Schneeman, "Distributed measurement and control based on the IEEE 1451 smart transducer interface standards," IEEE Transactions on Instrumentation and Measurement, vol. 49, issue 3, pp. 621–627, Jun. 2000.