Flexible Architecture for Microinstrumentation Systems in Silicon

E. Cretu, J.H. Correia, S.H. Kong, M. Bartek and R.F. Wolffenbuttel
Delft University of Technology, Dept. of Electrical Engineering,
Lab for Electronic Intrumentation/DIMES,
Mekelweg 4, 2628 CD Delft, the Netherlands
phone:+31-15-2781602, fax:+31-15-2785755
cretu@ei.et.tudelft.nl

Abstract— **A flexible architecture for a microinstrumentation system in silicon was developped, and intended for use in a wide range of applications. The MCM device should be considered a genuine (micro)instrument, which supplies high- level data over an external instrumentation bus. Internally, the system should support all vital functions, such as power management, self-calibration and in-system data transfer over a flexible internal data bus. The internal bus is the major subject of this paper. Drivers are designed for a generic sensor module; the application will decide the type of sensors to be used.**

Keywords— **microsystems, internal bus interface**

I. Introduction

The advances in silicon technology allow the introduction of a new system concept, the microsystem, as a merger between microelectronics and micromechanics. The concept of ASIC (Application Specific Integrated Circuits) based on some standard blocks proved to be a good compromise between a reduced time to market and a high performance. It is already widely used in microelectronics. A similar approach could be introduced in the field of microengineering, in particular for the development of microsystems encapsulated in a single package. ASIM (Application Specific Integrated Microsystems) based on a common, standard system level architecture could deliver in a standard way information toward a high-end processor. The communication is based on a high level protocol, independent of the type of the measured quantities. Moreover, the microsystem behaves as an autonomous unit, which manages and tunes internally the functionality of the various sensors. However, new problems appear, such as:

- which fabrication technology to choose for the integration in the same package of different types of sensors and different electronical blocks
- assuming a centralized bus structure, what type of protocol will allow flexible communication between the local master and different types of sensors
- how to increase the ratio between the common part, consisting of application-independent blocks, and the part that depends on the specific application
- the packaging could give rise to major technological difficulties in case of different types of sensors.

In order to obtain a good overall system performance, one aims at a maximum decoupling between the fabrication of the standard unit and of the application specific parts. The system structure will thus consist of a common silicon substrate and surface-mounted devices, e.g. the local microcontroller and the sensors [1](Fig. 1). On the platform several active areas are defined, for local bus drivers and a smart power unit (the common infrastructural functions), together with mounting sites for the application-dependent parts.

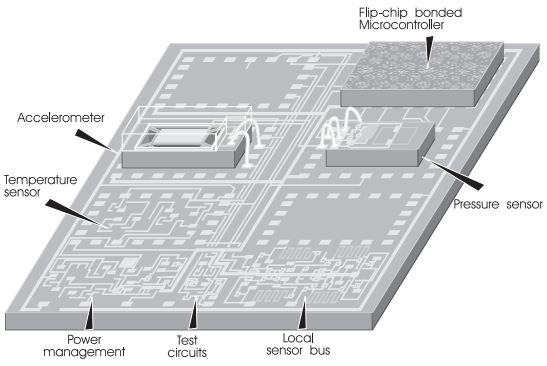

Fig. 1. Microinstrumentation platform

The system structure is based on a single master controller, which communicates with the other devices by sending and receiving messages on a unique, internal bus. The bus protocol must allow both analog and digital signals to be sent/received, in order to increase the range of usable sensors. The compromise flexibility/simplicity was settled by designing an Improved Smart Sensor bus protocol, which adds a series of facilities to the existing IS^2 bus protocol [2].
The generality of the system architecture resulted from the concept of *generic sensor* : a black-box device, for which only the interfacing properties are well defined, and not its internal structure. The goal was to allow a direct connection of different types of sensors to the internal bus, without any modification of the bus driver circuit. This is why the bus driver is designed to support all the facilities provided by the generic sensor block, which offers an extensive set of working modes. Even if a specific sensor will use only a subset, it can be directly connected to the bus, without any modification of the bus driver. ANY specific sensor will be easily integrated in the system if it has at least one operating mode within the set for which the driver was designed. This is in contrast with generic or template objects. There any specific object is guaranteed to have at least the interfacing capabilities of its associated template.

II. The internal bus protocol

The general architecture of the microinstrumentation system consists of a collection of sensor modules connected on a unique bus, controlled by a master microcontroller (Fig. 2).
The protocol used had to cope with the specificity of data acquisition tasks. The requirements imposed were:

- simplicity, for a tight implementation in silicon (for instance,to be a single master on the bus)
- to send/receive both analog and digital data
- on-line sensor calibration
- to allow both a polling mode, in which the master asks a specified device for data, and interrupt mode, in which a device signalizes the availability of data
- to let the master configure an addressed device.

The bus can be seen as consisting of four hierarchical layers, at increasing levels of abstraction:

1. the mechanical and physical layer - the physical structure (number of wires, maximum distance between two consecutive units, etc.)
2. the electrical layer - focused on the electrical characteristics of the modules (input and output impedances, signal attributes, etc.)
3. the logical layer - deals with the mapping from the electrical domain to the '0' and '1' logic levels
4. the protocol layer - associates a meaning with the transmitted signals.

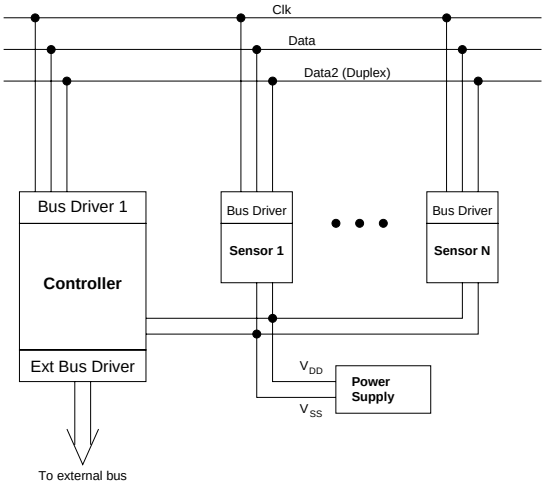

Fig. 2. General Architecture of the microinstrumentation system

At the protocol level one distinguishes, with respect to the semantic of the data, the following classes of functions:

- sending/receiving data
- sending/receiving the state of the subsystem
- sending/receiving control signals.

The designed protocol was built up starting from the basic features of the Integrated Smart Sensor (ISS) bus protocol, previously developed in our laboratory. A series of additions at the protocol layer resulted in a new Improved ISS (**IISS**) bus protocol. This fulfills the imposed requirements and is suitable for a general microinstrumentation architecture [3].The basic ideas are:

- the use of a variable-length frame for sending each message. At the beginning of the frame, the master put a start bit, followed by a synchronously transmitted fixed-length field. This has four bits for address specification and another four for the command coding. After sending the 8 bit-wide field, the master waits for the acknowledgment symbol from the receiver. The remaining of the frame is of variable length, until the master sends an **EOT** (End of Transmission) symbol.
- use of the Manchester encoding scheme for the

transmission of synchronous data at the logical level (inherited from the previous ISS protocol). This allows a compact protocol. There are four available symbols (Fig. 3), so two of them were used for mapping of the logic symbols '0' and '1' and the remaining two were used as metasymbols ('**Free**' and '**EOT**') for separating the messages and distinct fields inside a message frame.

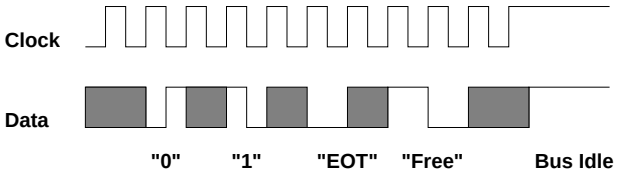

Fig. 3. Manchester code

- at the electrical level, the use of the open-drain technology for the connection on the bus allowed a hardware arbitration and solving of the possible bus conflicts.

The **generic sensor interface** for which the bus driver was developped is presented in Fig. 4. A particular type of sensor could use all or only part of the facilities offered by the interface. The state of a sensor module was divided into two components: a common part, embedded into the bus driver module, independent of the sensor type, and a part supposed to be inside the sensor module, of variable number of bits. The common part at present consists of only two bits, for enabling the **Interrupt Request** (**IRq**) and **Service Request** (**SRq**) modes for the corresponding device. We plan to add a general switch-off bit, in order to disconnect the supply voltage from the unused sensor modules. The IRq and SRq modes differs from the normal **polling mode** in that the slave module informs the master about the availability of data, and does not wait to be asked. An interrupt request has the meaning of an urgent message, so the sensor may signalize even in the middle of another frame. A service request has a lower priority; it may be sent only when the bus is in idle state, that is, outside of any message frame. The interrupt and service request modes allows a flexible and low-power data handling for rare or very important events [4].

In Fig. 5 there is an example of communication on the bus, in which the local controller, after asking the device $a_3\,a_2\,a_1\,a_0$ to send data on its asynchronous output, decides when to stop the data stream.

The functions presented in Table I are encoded in a four bits wide command field $Tx\,c_2\,c_1\,c_0$. On-line cal-

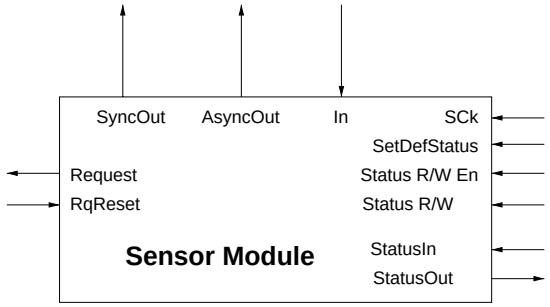

Fig. 4. Generic sensor interface

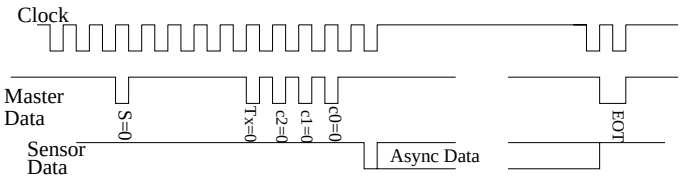

Fig. 5. A message protocol for getting asynchronous data

ibration procedure is included, by the addition of a dedicated second data line. If this facility is not necessary for a specific application, then only two communication wires (clock plus main data line) are needed. The interrupt and service request signals were implemented as short pulses put by the slave devices on the clock line, thus bypassing the default polling mechanism.

TABLE I
THE MAPPING OF THE COMMANDS SET

Tx	c_2	c_1	c_0	Command
0	0	0	0	get async. data
0	0	0	1	get sync. data
0	0	1	x	get status
0	1	0	x	unused
0	1	1	0	start async. duplex
0	1	1	1	start sync. duplex
1	IRqE	SRqE	0	set IRqE,SRqE and default config.
1	IRqE	SRqE	1	set IRqE,SRqE and a new config.

III. Hardware Realisation of the Bus Driver

The block diagram of IISS interface bus is shown in Fig. 6. It was implemented in a 1.6 μm CMOS process [5].

The upper trace is the clock line, while the bottom trace represents the data line voltage level. After 2 clock periods in which the data line remained in

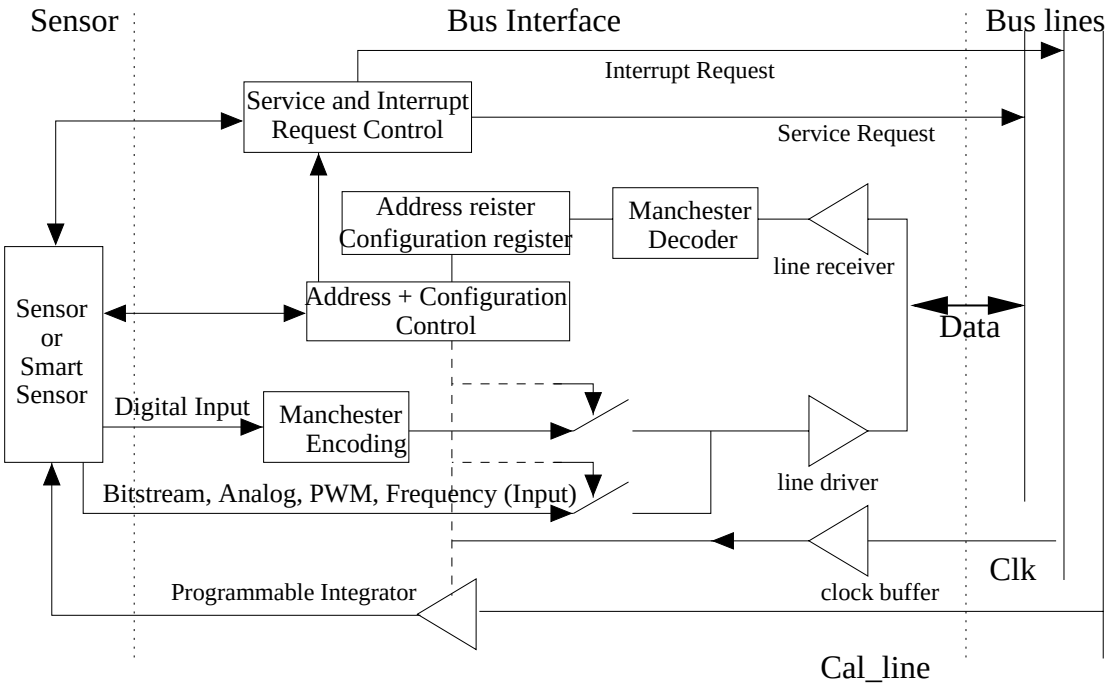

Fig. 6. Block diagram of the bus interface

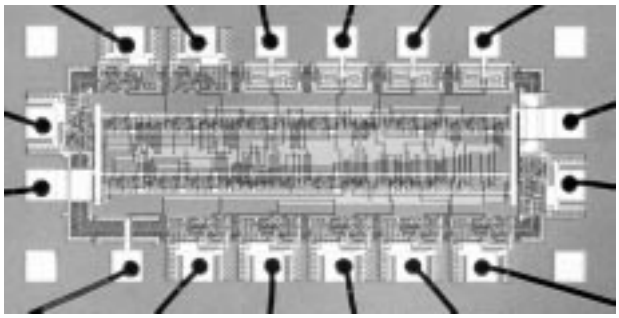

Fig. 7. Photograph of the bus driver chip

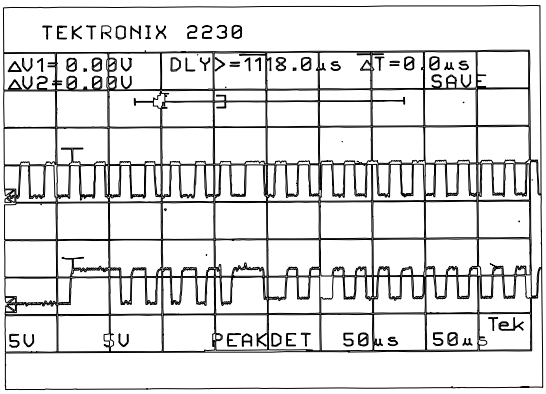

Fig. 8. Oscilloscope record of getting synchronous data frame

"Free" state, the master starts a new message frame. Firstly, it puts a start bit ('0'), followed by the address of the sensor ('0000'). In the present case a simplified variant of the bus driver was used, designated for those sensors which will only deliver data toward the local microcontroller. Their activation is implicitly done by addressing. In response to master's request, the sensor '0000' confirms by putting '1' on the data bus. After that, it simply outputs a data bit stream (in the image, a string of zeros). The transmission is ended when the microcontroller forces an 'EOT' symbol on the data line. The hardware arbitration resulted from the open-drain logic ensures that the 'EOT' symbol is not overwritten by the signals sent by the sensor.

IV. Applications

Presently the architecture is made suitable for applications where extended data processing from multichannel sensors is required. The focus is toward two systems: a miniature spectrometer and a condition monitoring system. The miniature spectrometer consists of an array of tunable Fabry-Perot resonance cavities for the spectral analysis of visible and near-infrared radiation. The sensor similarities suggests that most part of the software needed will be common for all the devices. For the condition monitoring system a set of three accelerometers for measuring the vibration spectrum in three direction is used. Based

on the comparison with pre-recorded pattern features, an early prediction of upcoming failure can be anticipated.

V. Conclusions

The microinstrumentation system architecture presented here is aiming for an optimum compromise between flexibility and simplicity. It can be used regardless of the actual sensors needed for the application. The resulting autonomous instrumentation system exchanges *externally* only high level data with a central processing unit. The *internal* communication between the local microcontroller and sensors is done in the Improved Integrated Smart Sensor bus protocol, suitable for a large range of sensors. The bus drivers are standard modules, independent of the specific sensors. This approach enables the co-integration of sensor and bus driver, but leaves also the option for a two-die solution, with both sensor and driver die bonded to platform. The practical realization in CMOS technology proved that the present approach offers an easily tunable metering system for various applications. The low power consumption required for the internal bus interface makes this solution very attractive for integrated microinstrumentation systems.

VI. Acknowledgments

This work is supported by STW (project DEL 55.3733), TU Delft and JNICT-Portugal (PRAXIS XXI-BD/5181/95).

References

[1] R.F. Wolffenbuttel, "Silicon sensors and circuits: on-chip compatibility", Chapman-Hall, 1996

[2] J.H. Huijsing, F. Riedijk and G.vd. Horn, "Developments in integrated smart sensors", in Proc. Transducers'93, pp. 320-326

[3] J.H. Correia, E. Cretu, M. Bartek and R.F. Wolffenbuttel, " A microinstrumentation system for industrial applications", in Proc. IEEE Int Symp. on Industrial Electronics (ISIE'97), Guimarães, Portugal, July 7-11,1997

[4] A. Mason, N. Yazdi, K. Najafi and K.D. Wise, "A low-power wireless microinstrumentation system for environmental monitoring", in Proc. Transducers'95, pp.107-110

[5] J. H. Correia, E. Cretu, M. Bartek and R.F. Wolffenbuttel, "A low-power low-voltage digital bus interface for MCM-based microsystems", in Proc. 23rd European Solid-State Circuits Conf. (ESSCIRC'97), Southampton, UK, September 16-18, pp. 116-119