VSIB: A Sensor Bus Architecture for Smart-Sensor Network

Md. Sajjad Rahaman, Masud H Chowdhury
Department of ECE, University of Illinois at Chicago, Chicago, IL 60607, USA
Email: {mrahaman, masud}@ece.uic.edu

Irfan Nasir, Lih-Tyng Hwang,
Digital and Physical Realization Center, Motorola Labs, Schaumburg, IL 60196, USA
Email: {Irfan.Nasir, L.Hwang}@motorola.com

Abstract— *There has been a proliferation of real-time sensing application due to the recent technological improvements in small, inexpensive, low-power, and distributed "smart" sensor nodes capable of doing a small amount of data processing and storage. Sensor network coordinates a collection of these sensor nodes to enable a high quality detection and measurement network in great details. Sensors come in great variety with different capabilities from different vendors. Besides, there exists a variety of sensors interfaces i.e. analog (capacitive, inductive, and resistive readout), I^2C, SPI, UART, for communicating sensor signals. Therefore, it is difficult to provide the required level of integration and functional flexibility which are of great importance in smart-sensor systems. Due to these various interfaces, the design of the versatile sensor interface bus is of great importance. This paper presents a Versatile Sensor Interface Bus (VSIB) architecture which offers network-independent communication interfaces for connecting sensors to microprocessors, instrumentation systems, and control/field networks. It is shown that it is easier to deploy commonly available sensors with various communication interfaces to a multi-sensor module.*

1. INTRODUCTION

A sensor network is a collection of nodes that collect data about their environment. In addition to data acquisition nodes, a sensor network may have actuator or output nodes that can be used to manipulate the surrounding environment based on data collected by the sensor nodes. Sensor networks typically span a small physical area, with nodes dispersed throughout. A smart sensor network addresses many of the problems associated with sensor networks. There is an increasing amount of sensors with signal-processing capabilities due to the rapidly improving silicon technology. These are known as smart-sensors. A system consisting of smart sensors generally includes a direct connection of different kinds of physical sensors, and sensor interfaces to a microcontroller, a digital signal processor (DSP) or a computer with a data communication protocol.

In a traditionally designed system the sensor measurement are fed directly into the application. As a result, the application has to deal with a lot of imprecise, ambiguous, and incomplete data streams. In order to provide more robust and more comprehensive view of the environment it is necessary to preprocess the sensor data by fusion method. Sensor data fusion hides data streams of physical sensors behind a sensor fusion unit. A typical sensor data fusion application contains a network of sensors, intelligence to process the sensor data, e.g. by sensor fusion and signal processing algorithm, a control program, that derives the output value based on these sensor data.

Sensors come in great variety with different capabilities from different vendors. Besides, there exists a variety of sensors interfaces for communicating sensor signals. Therefore, it is a problem to provide the required level of integration and functional flexibility which are of great importance in low-power smart-sensor systems. Due to these various interfaces, the design of the versatile sensor interface is of great importance. A versatile sensor interface must this be very generic to support all present and future types of sensors. It should also provide some standard functionality to transmit data in a standard data format [1]. Figure 1 shows a multi-sensor module architecture where several sensors are connected to a multiplexer and controller, data storage and transceiver block handle data communication functionality with a host system.

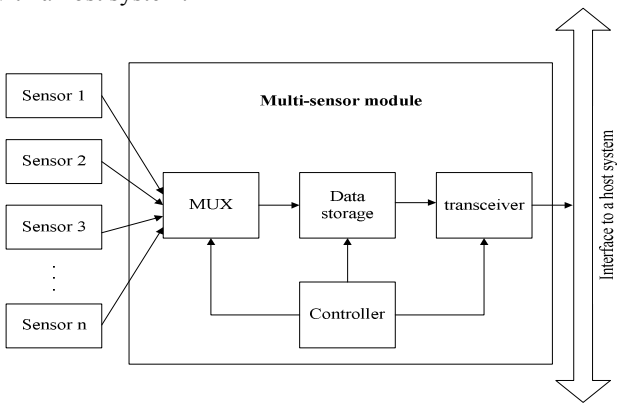

Figure 1: Multi-sensor module

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.1014

This paper proposes a method for implementing a versatile sensor interface which can process sensor data from various sensor interfaces and communicate with a host system with a standard data communication protocol. VSIB can handle sensor with SPI/I^2C, analog, and digital output. It can adapt to sensors with different resolution. It also provides two widely used serial communication protocol SPI and UART to connect to a host system.

The rest of the report is organized as follows. A survey of common sensor communication protocols like I^2C, SPI, and UART are given is section 2. Section 3 gives a brief introduction to IEEE 1451 standards which deal with sensor interface independent sensor communication protocols. Section 3 describes VSIB with its circuit operation and functionalities. Section 4 concludes the report.

2. SENSOR DATA COMMUNICATION PROTOCOLS

2.1 INTER-IC INTERFACE

Inter-IC (I^2C) bus was introduced as a standard for connecting networked IC which may or may not include sensors. I^2C is intended for application in systems which connects microcontrollers and other microcontroller based peripheral devices. It is consist of two wire serial bus to minimize the cost of connecting the various devices in the system. Two wires, serial data, and serial clock carry information between the devices connected to the bus. The serial data wire is bi-directional but data may flow in only one direction at a given time. Each device is identified by a unique address. Device on the bus are defined as masters or slaves. A master initiates a data transfer on the bus and generates the clock. It also generates the control signals which are placed on the data wire. The slave device is a device which is controlled by the master (Figure 2) [2].

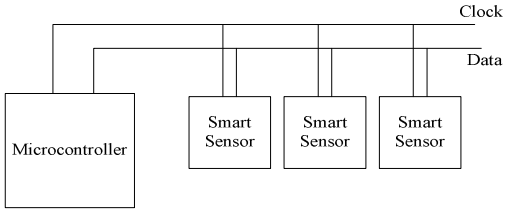

Figure 2 : I^2C bus configuration [2]

The I^2C is a multi-master bus. Therefore, more than one microcontroller can be connected to the bus. An arbitration procedure is implanted to avoid the chaos the might ensue from simultaneous data transfer by multiple micro-controllers.

I^2C bus allows a system design to rapidly progress directly from a functional block diagram to a prototype. The integrated addressing and data-transfer protocol allow systems to be completely software-defined. The topology is simple and the number of connection wire is minimized. However, although the hardware specification allows for simple interfacing, the communication protocol is rigid. The control signals from the masters are encoded on the data wire which increases the required complexity of the interface hardware.

2.2 SERIAL PERIPHERAL INTERFACE

Serial peripheral interface (SPI) is a three wire serial bus for 8-bit data communication application. Two of the three lines transfer data and the third is a serial clock. Similar to I^2C bus, devices on the bus are defined as masters and slaves. A master initiates an information transfer and generates the clock. Each slave device on the bus is controlled by a chip select line, which is a parallel line for each bus node in addition to the three SPI bus signals (Figure 3).

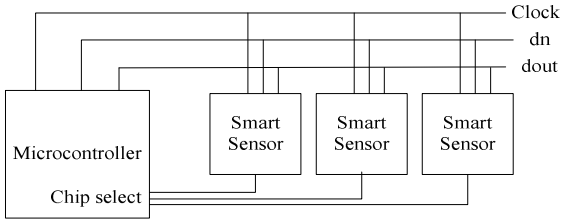

Figure 3 : SPI bus configuration [2]

Two data lines are unidirectional and the clock signal is generated by the master. The *dout* line carries data from the master to the slave while *din* carries data from the slave to the masters. The SPI bus employs a simple shift register data transfer scheme. Data transfer is usually performed in eight or sixteen bit blocks. Data rate could vary from near zero to 1 Mbps.

There are four modes defines for SPI based on the clock polarity and clock phase settings. The clock polarity determines the level of the clock idle state and the clock phase determines the clock edge that places new data on the bus. Any hardware device capable of operating in more than one mode uses a method of selecting the value of these mode control settings. The multi-mode capability combined with the simple shift-register architecture makes the SPI bus very versatile and allows many non-serial devices to be used as SPI slaves.

SPI is an attractive bus for smart sensor microsyetms. Two separated data lines and the simple shift register data transfer scheme make the hardware of an SPI interface much simpler than that of the I^2C bus. However, an individual mode enable line is needed to select the device connected on the bus. Since select lines are generally provided by the master. This limits the number of the sensor nodes in the system, and consequently the size of the sensor network. Beside, unlike I^2C, the SPI bus does not allow new sensors to be easily added to the system without any additional external interfacing.

2.3 UART

A Universal Asynchronous Receiver and Transmitter (UART) is used for communication with serial input/output devices. Typically, the UART is connected between a central processor and a serial device. To the processor, the UART

appears as an 8-bit parallel port, which can be written to or read from. To the serial devices, the UART presents two data wires, one for input and one for output, which serially communicate 8-bit data. The rate of data communication depends on the peripheral device [5].

A UART has a transmitter section and a receiver section. The transmitter converts the bytes into a serial stream of data bits as they are prepared for transmission. The receiver takes the incoming stream of bits and groups them into 8-bit chunks so they can be constructed as bytes.

The UART also monitors input control lines and has the ability to change the state of output lines. UARTs can be used as either Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). They are controlled by a clock usually running at different speeds. A buffer is also used to temporarily hold incoming data. This buffer varies by design and is usually very small, ranging anywhere from 1 byte to 128 bytes [5].

3. IEEE 1451 STANDARD TRANSDUCER INTERFACE

IEEE 1451 family of standard define a set of common communication interfaces for connection transducers to micro-processor-based systems, instruments, and networks in a network independent environment. They provide a set of protocols for wired and wireless distributed applications. The IEEE 1451.0 standards provides a common set of commands, an electronic data sheet format, and communication protocols for the family of IEEE 1451 standards [6]. Figure 4 shows the IEEE 1451 family of standards.

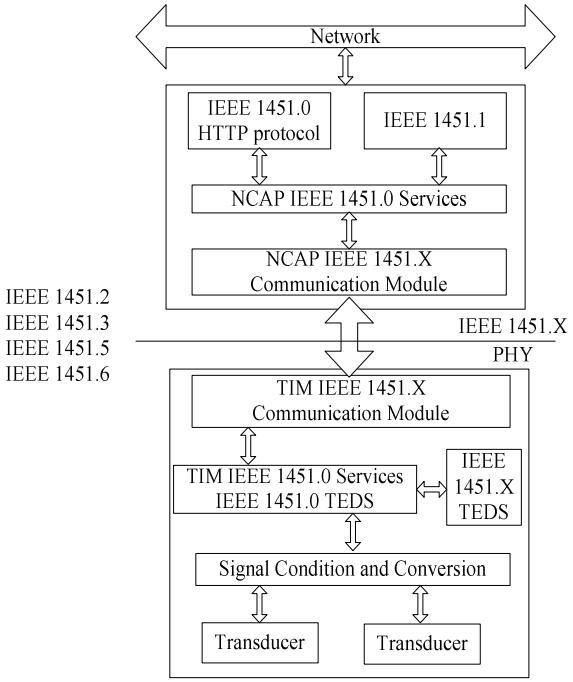

Figure 4 : IEEE 1451 family of standards [6]

A brief comparison between various IEEE 1451 standards is given in Table 1.

Table 1. IEEE 151 sensor bus standard summary

IEEE 1451.0	Encourages compatibility among other IEEE 1451 standard using physical layers
IEEE 1451.1	Defines a common object model for the components of a smart sensor and the network that connects the sensor to the outside world. This model defines a network capable application processor (NCAP).
IEEE 1451.2	It specifies a transducer-to-microprocessor protocol and a transducer electronic data sheet (TEDS) format for digital point-to-point communications.
IEEE 1451.3	It defines a multi-drop configuration for networking distributed transducers with TEDS.
IEEE 1451.4	It defines a mixed-mode interface that allows both digital signals from the TEDS and the analog signal from the sensors to share the same set of wires between the NCAP and the transducer.
IEEE 1451.5	It addresses wireless communications between the NCAP and a transducer with a TEDS.
IEEE 1451.6	This is the information required for the CAN (consolidated auto network)
IEEE 1451.7	It specifies the integration of sensors in RFID infrastructures.

4. VERSATILE SENSOR INTERFACE

The block for a versatile senor interface (VSIB) contains SPI/I^{2}C, digital output, and analog sensors interface and will provide both UART and SPI interfaces to a host system. Figure 6 depicts the system architecture of the proposed VSIB.

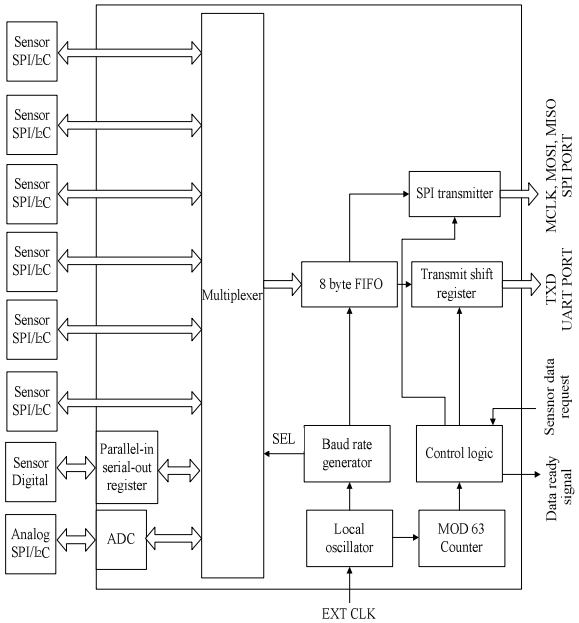

Figure 6: VSIB

As illustrated in Figure, a typical VSIB architecture consists of the following blocks:

i) local oscillator
ii) baud rate generator
iii) multiplexer for sensor data fusion
iv) multiplexer for clock selection
v) 8 byte FIFO
vi) parallel-in, serial-out register
vii) ADC
viii) MOD-63 counter
ix) Control logic
x) transmitter shift register
xi) UART transmitter
xii) SPI transmitter

VSIB provides 8 input sensor interfaces for sensors with various output interfaces: 6 SPI/I^2C, 1 digital, 1 analog. Most of the digital sensors have resolution from 5 bits to 8 bits (Appendix 1). In this case, VSIB allocates 8 bits for each sensor output data. Local oscillator will generate clock signals for controlling various serial and parallel data transmission between sensors and VSIB. There is a provision to drive the VSIB from an external clock generator. Baud rate generator will generate a frequency which is 1/8 times the local oscillator frequency. The factor 8 comes from the fact that there will be 8 sensor nodes. Multiplexer will route data from a selected sensor to FIFO. VSIB will operate in two modes: master and slave. In master mode local clock will drive all sensors sequentially and FIFO will store all the sensor data. VSIB will act as slave when it gets a service request from host system. Therefore, FIFO will have the option to operate at two different frequencies for different modes and clock selection multiplexer will choose between two available clock signals.

Sensor data from an SPI/I^2C interface is transmitted in a serial format. Since each sensor generates 8 bits data, a selected sensor will be allocated a duration equals 8 clock time period. FIFO will store the entire sensor data sequentially during this period. A baud rate generator with a frequency 1/8 of the clock signal will keep the physical connection between serial SPI/I2C output port to FIFO. It will also provide the chip select signal for the sensor with SPI/I2C interface. Baud rate generator is programmable. So, VSIB can accommodate sensors with resolution other than 8 bits with some modification to baud rate generator.

Some sensors come with only digital output. So, an 8 bit parallel-in serial-out register is employed with digital sensor. For, analog sensors, the preconditioned sensor signals in the analog domain will be converted to digital format using an ADC featuring sigma-delta architecture and, then, digital signals will go through an 8-bit parallel-in serial-out register and at last, to FIFO. It can also use commercially available universal transducer interface (UTI) for analog sensors to get SPI output. UTI has also various build in signal processing capabilities to control resolution, range of the sensor data.

Several control logic and signals will be required in a VSIB. MOD 63 counter will set the Data Ready Signal at the end of the accumulation of all the sensor data in the FIFO. So, data ready signal will indicate whether VSIB is ready to transmit the sensor data to a host system with either an UART or a SPI port. Sensor data request is an interrupt signal from a host system. These control signals will be using GPIO ports.

UART transmitter uses the FIFO and a transmit shift register to send data to a host system through an UART or SPI port. VSIB operates as slave in this case and both FIFO and output shift register use an external clock frequency generated by the host system.

5. CONCLUSION

This report presented a versatile sensor interface circuit for a multi-sensor interface module. The VSIB utilizes architecture for connecting sensors with SPI/I^2C, Digital and Analog sensors. All the sensor data is stored in a FIFO and this is driven by a SPI or an UART transmitter to send the sensor data to a host system. Therefore, VSIB acts as a transducer independent module (TIM) as suggested by IEEE 1451 standards without the complexity of microcontroller driven operations. The effort was to keep the module as simple as possible yet to offer great flexibility in choosing various interfaces both to sensors and host systems.

REFERENCES

[1] J. Zhang, J. Zhou and A. Mason, "Highly Adaptive Transducer Interface Circuit for Multi-Parameter Microsystems," *IEEE Trans. Circ. Sys. I,* vol. 54, no. 1, Jan. 2007.

[2] J. Zhou and A. Mason, "Communication Buses and Protocols for Sensor Networks," *SENSORS, (ISSN:1424-8220)*, vol. 2 (7), pp. 244-257, August 2002.

[3] http://en.wikipedia.org/wiki/I%C2%B2C

[4] http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[5] http://xilinx.com

[6] E. Song, and K. Lee "Smart transducer web services based on the IEEE 1451.0 standard," Instrumentation and Measurement Technology Conference (IMTC). Poland, May 2007