DESIGN AND ANALYSIS OF THE ON CHIP INTEGRATED DATA INTERFACES FOR ASIC ADOPTING I²C AND SPI PROTOCOL

J. Trontelj Jr.

Faculty of electrical Engineering, University of Ljubljana, Slovenia

Key words: I²C, SPI, ASIC, Serial synchronous data transmission, eeprom AT24CXX.

Abstract: Application specific integrated circuit design usually requires several input bits for trimming, for changing operating parameters or simply for storing information - such as a product version or a serial number. While reducing the size of the chip, we can not afford to waste silicon for several input pads. Therefore, to satisfy the compatibility issues, we normally have to use, or suitably adopt one of the available serial busses. They vary in protocol types, speed, reliability, complexity and number of wires required. In this article we will briefly introduce some of the most popular serial interfaces. Afterward we will focus on the SPI (Serial Peripheral Interface) and I²C (Inter Integrated Circuit bus) serial protocols. For that purpose we will give some details about two different application specific integrated circuits that we recently designed for serial production. These two circuits have very similar input and output requirements, but they use different interface protocols. Complexity details and space requirements for the mentioned interfaces will be analyzed.

Načrtovanje in analiza podatkovnega vmesnika na integriranih vezjih po naročilu z uporabo I²C in SPI standardov

Kjučne besede: I²C, SPI, Integrirana vezja po naročilu, zaporedna sinhrona komunikacija, eeprom AT24CXX.

Izvleček: Načrtovanje integriranih vezij po naročilu je vse pogosteje povezano z zagotavljanjem možnosti sprotnega vplivanja na delovanje integriranega vezja in tudi možnostjo shranjevanja informacij v integriranem vezju - na primer verzija izdelka ali serijska številka. V ta namen je najbolj ekonomična uporaba zaporednega komunikacijskega vodila, saj zahteva malo vhodnih priključkov na integriranem vezju. Zaradi združljivosti z drugimi napravami, je priporočljivo uporabiti ali prilagoditi katerega izmed znanih serijskih vodil. Seveda pa se slednji med sabo zelo razlikujejo glede načina komunikacije, hitrosti, zanesljivosti, zapletenosti in številu povezav, ki jih potrebujemo za njihovo uporabo. V tem članku bomo na kratko predstavili nekaj najbolj popularnih serijskih protokolov. Bolj natančno pa si bomo ogledali SPI (Serial Peripheral Interface) in I²C (Inter Integrated Circuit bus) načina povezav in to na primeru dveh integriranih vezij po naročilu. Vezji smo nedavno razvili za serijsko proizvodnjo in imata zelo podobne vhodno izhodne komunikacijske zahteve. Za izvedbo smo uporabili dva različna komunikacijske protokola. Primerjali bomo prostorsko zahtevnost in zapletenost izdelave komunikacijskih vmesnikov.

1. Introduction

Modern ASIC (Application Specific Integrated Circuit) design usually requires input capability of several information bits. This is necessary to achieve optimal operation or for storing desired information. Since the silicon chip should be as small as possible, we can not afford to use several pads for parallel communication. As the efficiency of the serial busses increases, the speed advantage of the parallel data transmission gets less important. Consequently, a serial transmission is perfectly suited to decrease the number of space consuming pads on the integrated circuit. The basic principle is that command codes and as well as data values are serially transferred to the integrated circuit. After that they are pumped into a shift register, where they can be available for internal parallel processing.

At the present time we have several serial communication standards available /1/. The most popular are USB (Universal Serial Bus), IEEE1394, SPI (Serial Peripheral Interface), I²C (Inter Integrated Circuit bus), UART (Universal Asynchronous Receiver Transmitter) and CAN (Controller Area Network). Figure one presents an application overview of the mentioned serial protocols. Let us shortly present the most important features:

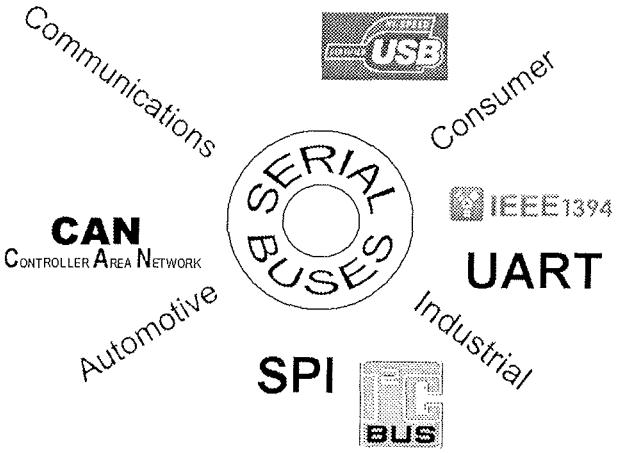

Figure 1: Overview of the most popular serial data busses.

- USB is still emerging standard, developed by Intel and Microsoft for connecting personal computers to peripherals. It uses four wires; two of them are power supply.
- UART is mainly used for asynchronous communications in computer world for speeds up to 1Mbits/s. Parity bit can be added for increased reliability. One wire for each direction of flow.
- CAN is one wire (differential) bus, developed by BOSCH and CIA for industry demanding environments, where high reliability is requested.
- IEEE 1394 is also still emerging standard - trademark of Apple. It uses four or six wires. Mostly used in computer or video applications where constant transfer rates with guaranteed bandwidth is required.
- SPI was developed for connecting peripherals to each other and to microprocessor. It is a four wire full duplex synchronous serial data link. Has no official specification.
- I^2C developed by Philips for communication between integrated circuits on printed circuits boards. Two bus lines are required.

In this article we will take a closer look to the SPI and I^2C serial bus. We recently designed and transferred into the production two different ASICs, with similar input and output requirements, but different interfacing protocols. One was SPI and the other was I^2C.

The SPI-bus is a four-wire (actually three-wire plus slave selects wires plus ground) synchronous serial communications interface, used by many microprocessor peripheral chips. It is standard across many Motorola microprocessors and other peripheral chips. It provides support for a low/medium bandwidth network connection amongst CPUs and other devices supporting the SPI. A synchronous clock shifts serial data in and out of the bus controller in blocks of 8 bits. SPI bus is a master-slave interface. Whenever two devices communicate, one is referred to as the "master" and the other as the "slave" device. The master drives the serial clock. When using SPI, data is simultaneously transmitted and received, making it a full-duplexed protocol. Motorola's names for the four signals are as follows: SCLK for serial clock, which is always driven by the master, MISO is master-in slave-out data, MOSI is master-out slave-in data and CS (chip-select) input is required to enable the slave. Every slave connected to bus needs separate chip-select signal line. There are also the extensions to SPI protocol such as for example QSPI (Queued Serial Peripheral interface) and MicrowirePLUS.

I^2C is a two-wire (plus ground) synchronous full duplexed serial interface standard defined by Philips Semiconductor in the early 1980's. The two active wires, SDA and SCL, are both bidirectional. Where SDA is the Serial Data line and SCL is the Serial Clock line. The interface typically runs at a fairly low speed (100 kHz to 400 kHz) form 1998 specifications also up to 3.2 mega baud. Each device on the bus has a unique address. A device that controls signal transfers on the line and also controls the clock frequency is the master. Device that is controlled by the master is slave. The master can transmit or receive signals to or from the slave, or control signal transfers between the two slaves. Bus supports more than one master on a single bus. To begin the communication, the master places the address of the slave, with which it intends to communicate, on the bus. All devices monitor the bus, to determine if the master device is sending their address. Only the device with the proper address communicates with the master. I^2C has also the acknowledgement mechanism to confirm data reception.

In next paragraph we will shortly describe our approach to the implementation of the SPI and I^2C bus, including some technical background for better understanding the complexity and the differences between the mentioned communication protocols. Later we will show our interface design approach and some area analyses of the implemented methods.

2. Implementation of the SPI interface on the ASIC1

Requests for the ASIC1 interface was to adopt SPI interface for writing, storing and reading eighty bits of information. This information is necessary for trimming DAC (Digital to Analog Converters) and for integrated circuit setup. Interface clock frequency should be up to 1MHz. Eighty bits are organized within ten differently addressed bytes. Since the purpose of previously mentioned bits is to fine trim the operation of the ASIC, these ten bytes of information should be available in parallel. The reason for this is to avoid erratic operation of the chip while trimming. In other words, by using a simple shift register particular fine trimming function would be set up to several undesired positions before the final shift. Therefore we actually need more registers and appropriate logic to perform the desired serial to parallel operations. Figure 2 presents block diagram of the ASIC 1 interface.

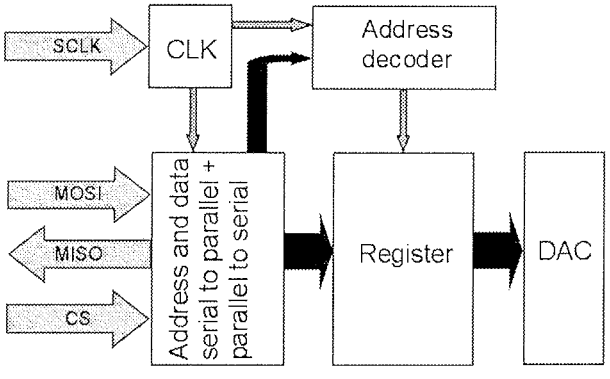

Figure 2: SPI interface block diagram for ASIC 1.

There is no official specification what exactly is SPI interface and what not, so there are no general rules for transitions where data should be latched. In practice there are used four different modes. Figure 3 presents an example of the SPI-mode 0 communication protocol where CPOL (Clock polarity) and CPHA (Clock phase) are both zero. Data on the MOSI and MISO lines are therefore valid on the rising edge of the clock. When chip select for slave device is low, writing instruction is followed by address and data bits.

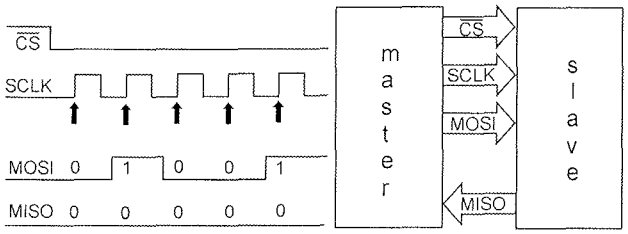

Figure 3: Example of the SPI communication protocol.

From the block diagram on figure 2 we can see, that such SPI interface design can be quite straight forward. There are very few extra blocks necessary. We require one address decoder, one clock multiplier and divider, some electronics for enabling and disabling register operations, eleven eight bit registers and two serial to parallel registers - one for address and one for data.

3. Implementation of the I²C interface on the ASIC 2

Serial communication protocol for the ASIC 2 uses I²C interface for reading fifty-six information bits that are also used for trimming DAC structures and setup the operation of the integrated circuit. This demand is quite close to that for the ASIC 1.

Communication must run between the ASIC 2 and the Atmel eeprom integrated circuit AT24CXX /3, 4/. When one IC that wants to talk to another, the communication procedure goes as follows:

- Wait until there is no activity on the I²C bus. This is when SDA (data) and SCL (clock) wires are both high.
- Reserves the communication bus. All other ICs then wait for the clock and address to appear on the bus.
- Provides a clock signal. It will be used by all the ICs for the synchronization of the transfer. The data on the data wire is valid when clock wire switches from low to high.
- Places on the bus the unique binary address of the IC that it wants to communicate with.
- Puts appropriate bit on the bus, whether it wants to send or receive the information.
- Waits for the other IC to acknowledge the communication.
- After receiving acknowledge, it starts transferring data. After every eight bits of data, it waits for a new acknowledge.
- After all data is transferred, it sends stop bit to the bus.

Figure 4 presents the implemented sequence of protocol requirements for our purpose. Default eeprom address for AT24CXX is "1010" and additional three bits "000" are used for addressing different memory pages or different devices on the bus. If we use for example AT24C01 with 1K of memory, it has address "1010000". Reading starts at byte with address 0x01.

From figure 4 we can see, that we need besides fifty-six information bits also additional thirty-seven bits thanks to the communication protocol. Therefore we actually have to monitor bus clock for ninety-three bits. For that purpose we decided to use two decimal counters, one clock divider and multiplier, input data monitor, action select circuit and seven eight bit shift-parallel registers. Since the data, that is stored in seven eight bit registers also fine adjusts DAC structures, it is necessary to read them in parallel to avoid erratic operation. The same case was in section two while we discussed ASIC 1. Such ASIC 2 structure is generally presented on figure 5. We can easily see that there is extra complexity according to the ASIC 1 structure on figure 3.

At this time we must also mention the Philips's position, that all chips (IC, ASIC, FPGA, etc) that can talk to the I²C bus must be licensed. It does not matter how this interface is implemented /1, 11/. Since we only perform reading

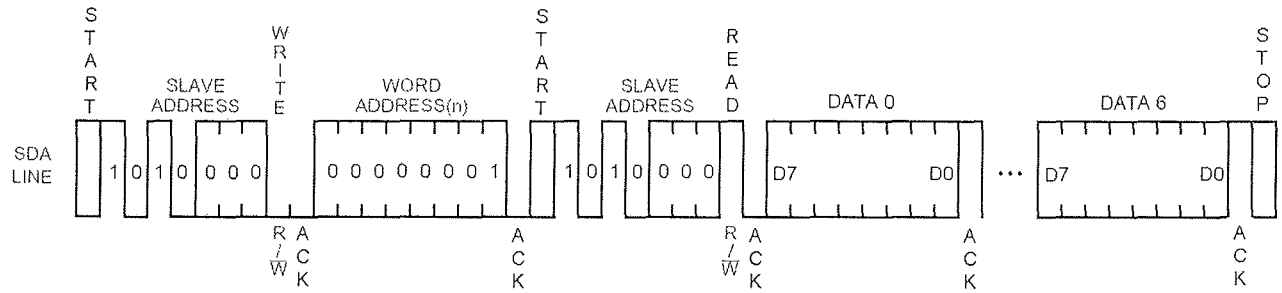

Figure 4: Communication procedure form AT24CXX to ASIC 2.

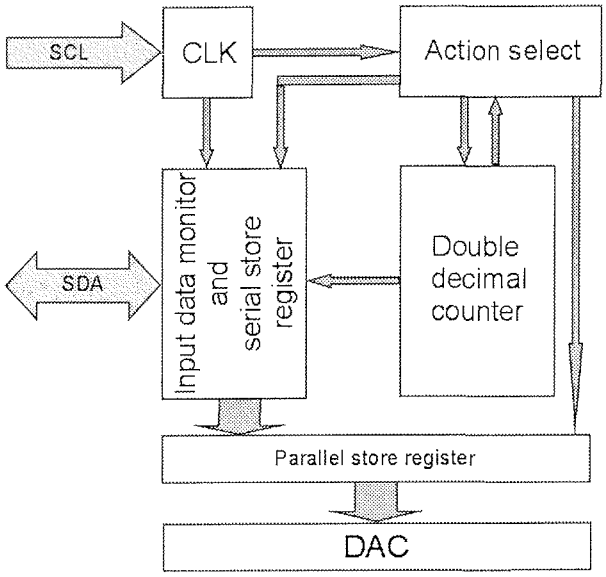

Figure 5: I^2C interface block diagram for ASIC 2.

from the eeprom, such license is already covered by eeprom manufacturer and is not additionally required for our ASIC 2.

4. Comparison of the implemented serial interfaces

Previously described interface approach on the ASIC 1 required for realization 3784 transistors occupying 0.1 mm^2, what represents 1.2 percent of the chip area. ASIC 2 required for the interface realization approximately 4696 transistors occupying 0.18 mm^2, what represents approximately three percents of the chip area.

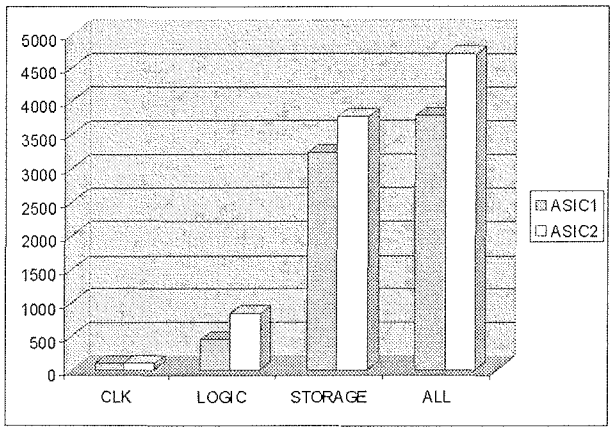

Figure 6: Number of transistors that were necessary for practical ASIC 1 and ASIC 2 interface realization.

Diagram on figure 6 presents the number of transistors that was used for realizing different interface tasks on ASIC 1 and ASIC 2. Realization of the auxiliary clock signals required almost identical number of transistors. It is also clear, that less communication wires required more digital logic gates for input stream processing. The reason for different number of necessary transistors for storage is due to the higher complexity of the I^2C. It requires some additional functions to be included in storage registers. The difference to consider is also the fact that ASIC 1 needs to store eighty bits in ten registers, while ASIC 2 needs to store fifty-six bits into seven registers.

On the other hand, the design of storage registers does not require a lot of designing time, since the same cells are frequently repeated. More demanding is the design and simulation of the interface logic. From figure 6 we can clearly see that choosing SPI interface requires less design effort. Our rough estimation is that necessary time to

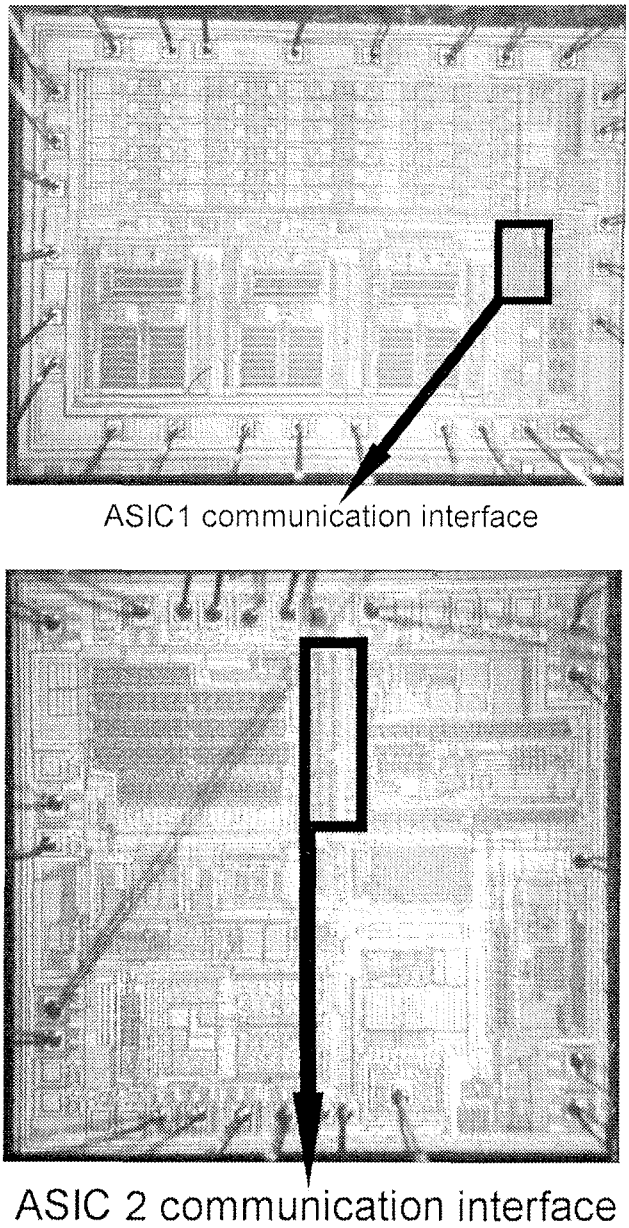

Figure 7: In scale comparison of the ASIC 1 and ASIC 2 with marked areas that are in use by serial interfaces.

incorporate and simulate SPI interface on the ASIC interface takes about half the time necessary for the I^2C interface simulation.

ASIC 1 is realized in 0.6um CMOS technology. It is designed to control 3D strain gauge probe and to process the strain gauge generated signals for 3D computer modeling purposes. ASIC 2 is realized in 0.8um CMOS technology and controls the closed loop current measurement sensor that consists from a full bridge built with MR sensors /12/. Figure 7 presents in scale comparison of the ASIC 1 and ASIC 2 with marked SPI and I^2C data interface areas.

Although both SPI and I^2C provide good support for communication with slow peripheral devices that are accessed intermittently, each of the mentioned communication ways has its own advantages towards each other. I^2C is best if we have just two signal lines to spare or if we need strong line drivers. If the clock speed should be faster than 400 kHz SPI is most likely better and simpler choice. SPI is also better suited for applications that are naturally suited as data streams while I^2C is better for communication with on-board devices that are accessed on an occasional basis.

5. References

/1/ Philips Semiconductors, "*I^2C Manual*", AN10216-01, March 2003

/2/ Fairchild, "*An introduction to Fairchild's SPI Interface EEPROMs*", Fairchild Application Note 832, 1998

/3/ Axelson, J., "*Using serial EEPROMs-Part 1*", Circuit Cellar, Issue 84, July 1997, pp. 74

/4/ Axelson, J., "*Using serial EEPROMs-Part 2*", Circuit Cellar, Issue 85, August 1997, pp. 70

/5/ Hannes, E.., "*SPI Interface and Use in a Daisy-Chain Bus Configuration* ", Infineon Technologies AG, Application Note, V1.2, Feb. 2002

/6/ ST Microelectronics, "*Choice of Serial EEPROMs Requires Understanding of Bus Differences*", Application Note AN-1001, 1998

/7/ Analog devices, "*Creating a Master-Slave SPI Interface between Two ADSP-2191 DSPs*", Engineer to Engineer Note, EE-144, June 2001

/8/ Philips Semiconductors, "*The I^2C Bus Specification - Version 2.1*", document order number 9398 393 40011, January 2000

/9/ Philips Semiconductors, "*PCF 8584 I^2C-Bus Controller Product Specification*", document order number 9397 750 02932, Oct. 1997

/10/ Philips Semiconductors, "*The I^2C bus and how to use it*", April 1995

/11/ Paret, D., "The *I^2C-bus: From Theory to Practice*", Wiley & Sons, Feb. 1997, ISBN 0-471-96268-6

/12/ Trontelj, J., "*Integrated Magnetic Sensors Design Examples*", Inf. MIDEM, Issue 4, 1999, pp. 190-194

Dr. Janez Trontelj Jr.,
jani@kalvarija.fe.uni-lj.si
University of Ljubljana,
Faculty of Electrical Engineering,
Tržaška 25, 1000 Ljubljana, Slovenia
Tel: +386 1 4768471

Prispelo (Arrived): 20.08.2004 *Sprejeto (Accepted): 15.09.2004*