FPGA Implementation of I^2C & SPI Protocols: a Comparative Study

A.K. Oudjida, M.L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui
Microelectronics and Nanotechnology Division
Centre de Développement des Technologies Avancées, CDTA
Algiers, Algeria
Email: a_oudjida@cdta.dz

Abstract— **I^2C and SPI are the most commonly used serial protocols for both inter-chip and intra-chip low/medium bandwidth data-transfers.**

This paper contrasts and compares physical implementation aspects of the two protocols through a number of recent Xilinx's FPGA families, showing up which protocol features are responsible of substantial area overhead. This valuable information helps designers to make careful and tightly tailored architecture decisions.

For a comprehensive comparative study, both protocols are implemented as general purpose IP solutions, incorporating all necessary features required by modern ASIC/SoC applications according to a recent market investigation of an important number of commercial I^2C and SPI devices.

The RTL code is technology independent, inducing around 25% area overhead for I^2C over SPI, and almost the same delays for both designs.

Keywords— Inter Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), Intellectual Property (IP), System-on-Chip (SoC).

I. Introduction

Today, at the low end of the communication protocols we find two worldwide standards: I^2C and SPI [1]. Both protocols are well suited for communications between integrated circuits for low/medium data transfer speed with on-board peripherals. The two protocols coexist in modern digital electronics systems, and they probably will continue to compete in the future, as both I^2C and SPI are actually quite complementary for this kind of communication [2][3][4][5][6].

The I^2C and SPI protocol specifications are meticulously defined in [7] and [8], respectively. Consequently, they will not be discussed here. Instead, a quick overview is provided in table I.

Although the literature on I^2C/SPI protocols is so extensive and the topic is so old (early 1980), to the best of the authors knowledge there is no comprehensive comparison of I^2C/SPI problem. By comprehensive comparison we mean a treatment that start from Philips/Motorola specifications and goes down to the actual ASIC/FPGA implementation, contrasting the two designs and then comparing the obtained results based on included protocol features.

In our attempt to implement universal I^2C/SPI IP cores [9], we first made a market study of an important number of recent commercial I^2C/SPI devices (datasheets) from different vendors [10] to look at the requirements and what features are to be included to satisfy modern ASIC/SoC applications. The key features required for I^2C/SPI IP cores as a result of the market investigation are summarized in table II, and their translation into architectures are depicted by figures 1 and 2, respectively. It's noteworthy to mention that only the slave side of the protocols is dealt with in this paper.

The paper is organized as follows. In this section, we showed the requirement specifications of a recent market investigation for modern I^2C/SPI IPs and their corresponding architectures. Section two contrasts and compares the implementation results. And finally some concluding remarks.

TABLE I. Comparison of I^2C and SPI protocols

	I^2C	**SPI**
Originator	Philips (1982)	Motorola (1979)
Plug & Play	Yes	No
Interface type	Serial (2 wires)	Serial (3+N wires)*
Distance	Short (In-box communication)	
Application	Multi-master register-access	Transfer of data-streams
Protocol Complexity	Low	Lower
Design Cost	Low	Lower
Transfer rate	Limited (100 & 400 KHz and 3.4 MHz)	Free (n x MHz to 10n x MHz)
Power Consumption	Low (2 pull-up resistors)	Lower
Transfer type	Half Duplex	Full Duplex
Time Constraint	Synchronous	
Multi Master+	Yes	No
Multi Slave	Yes	Yes
I/O constraints	Open-drain with pull-up resistors	No constraint
Addressing+	Software (7/10 bits)	Hardware (Chip Select)
Flow Control+	Yes	No
Clock Stretching+	Yes	No

* : N is the number of devices connected to a single master on the bus.
+ : Feature inducing substantial area overhead.

This work was supported by Centre de Développement des Technologies Avancées (CDTA) Algiers, in conjunction initially with Technische Universität (TU) Berlin.

978-1-4244-5091-6/09/$25.00 ©2009 IEEE

TABLE II. KEY FEATURES OF I^{2}C-SLAVE AND SPI-SLAVE IPS

Global Features	Key Features	
	I^{2}C-Slave	**SPI-Slave**
Data transfer	• Data transfer up to 100 Kbps in standard mode, and up to 400 Kbps in fast-mode; • Fixed data width of 8 bits; • Frame data-transfer (unlimited number of bytes); • Bidirectional data transfer; • Software programmable acknowledge bit ; • Repeated START detection; • Interrupt & polling mode transfer.	• Low/medium (n x MHz to 10n x MHz) data transfer rate depending on implementation issues; • Variable transfer rate (SCLK rate) depending on the master SPI baud rate; • User defined SPI word length; • Frame data-transfer (unlimited number of words); • Bidirectional data transfer (full duplex mode); • Interrupt & polling mode transfer.
Addressing	• Software programmable address (to be used in different application fields); • 7-bit & 10 bit addressing format (reduce the risk of conflicting slave addresses).	• Hardware addressable (Slave Select input pin nSS).
Synchronization	• Serial Clock with data acknowledge; • User defined Wait-states insertion period (clock stretching).	• Serial Clock with programmable polarity and phase; • Hardware defined outputs (full & empty) for Master wait-states insertion period (SCLK clock stretching).
Transfer error	• Transfer error recovery (internal status flags, use of a timer).	• Transfer error detection (internal status flags and output signals for overflow and underrun)
Data integrity	• Spike filtering (removing input spikes shorter than a certain number of clock cycles).	
Host Side Interface	• Simple and basic handshaking protocol easily adaptable to any standard SoC bus [11] via a wrapper [12].	
Performance	• User defined multiple-bytes write and read buffer (FIFO) to improve transfer performance (reduce wait state occurrences).	
Low power consumption	• Power conservative state (stop). The I^{2}C and SPI are disconnected from the system and internal activity turned off except for I/O interface.	
Parametrizable Functionalities	• Transmitter only /Receiver only or both; • With/without FIFO; • With/Without Digital Filter; • 7bits/10bits Address modes for I^{2}C only.	
Implementation & test	• Strictly synchronous design with positive edge clocking for straightforward scan-path insertion; • No internal three-state elements.	

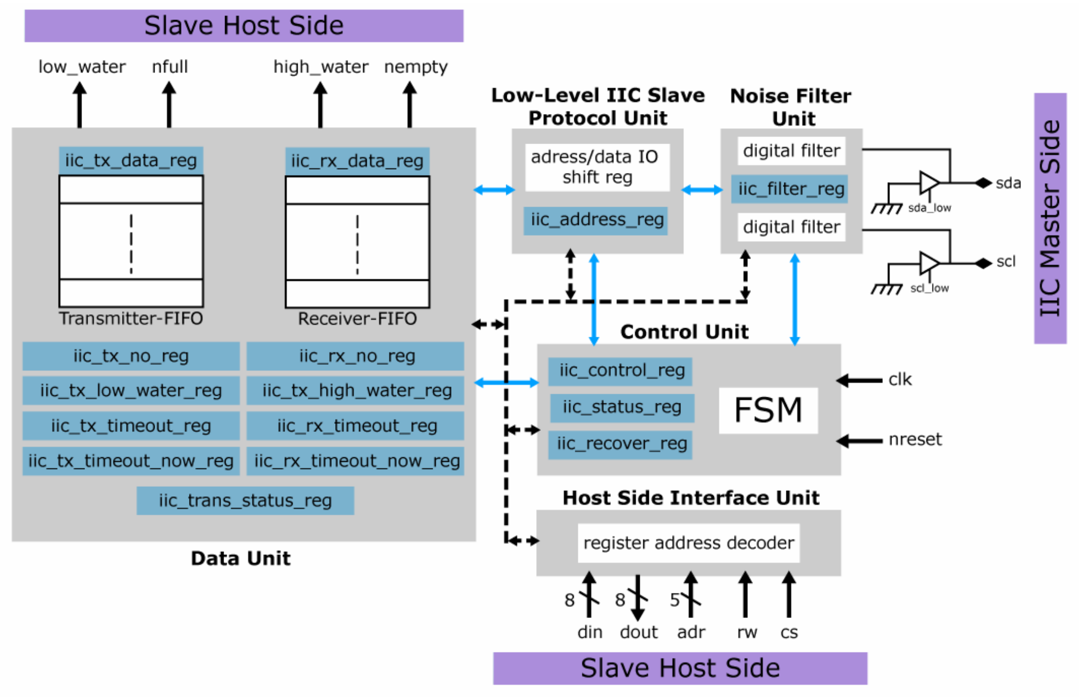
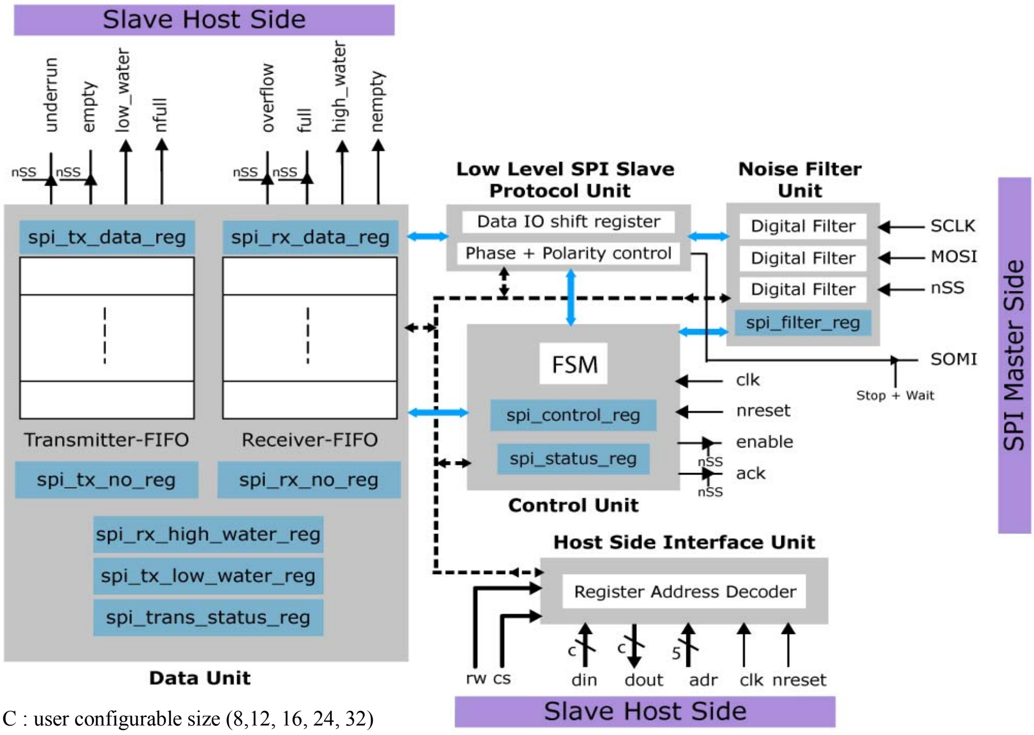


Figure 1. I^{2}C-Slave Transceiver Architecture.



C : user configurable size (8,12, 16, 24, 32)
to cope with various applications

Figure 2. SPI-Slave Transceiver Architecture.

II. Comparison of Implementation Results

For a more precise comparison, early in the design process, some precautions were taken to put both implementations under the same conditions. Starting from the initial specifications (Table II), we first built up the I^{2}C-Slave architecture [10], from which we derived the SPI-Slave one, keeping exactly the same architectural topology with minor modifications except for the Low-Level Protocol and FSM units. Afterwards, the I^{2}C-Slave architecture was translated into a high-quality technology-independent RTL code. The same code was used for the SPI-Slave with necessary modifications. Both RTL codes were written by the same designer to preserve the same coding style.

The whole design code, either for synthesis or functional verification, is implemented in Verilog 2001 (IEEE 1365). The synthesis design code is technology independent and was simulated at both RTL and gate level (post place & route netlist) with timing back annotation using ModelSim SE 6.3f and mapped onto Xilinx's FPGAs using Foundation ISE 10.1 version. Both designs have undergone severe functional software verification procedure according to our own IP development methodology summarized in [9]. As for physical test, both designs were integrated around Microblaze SoC environment using V2MB1000 demonstration board [13] with Xilinx's EDK 9.1i version.

The RTL-Code size of I^{2}C-Slave is about 1.44 times the code size of SPI-Slave (Table III). The 44% extra code size is mainly due to the additional logic required by the I^{2}C-Slave to handle the software addressing (7/10 bits), the control flow, and clock stretching features (Table I).

The mapping of RTL-code including two 4-byte FIFOs and digital filters onto Xilinx's FPGA devices (Table IV), exhibits a slice utilization average around 500 and 360 for I^{2}C-Slave and SPI-Slave respectively, except for Virtex 5 devices where the utilization is around 185 and 140, respectively. This difference is due to the number of look-up-tables (LUTs) per slice, which is: 2 LUTs of 4 inputs each for Spartan 2-3 and Virtex 2-4 devices, and 4 LUTs of 6 inputs each for Virtex 5 devices. Note that some slices are used only for routing. Nevertheless, whatever the FPGA device used; the I^{2}C-Slave induces an average of 25% area overhead over SPI-Slave, which is the counterpart of higher usage flexibility and a more secure transfer.

It is noteworthy to mention that all results, either for slice occupation or delays, are obtained using the default options of the implementation software (Foundation ISE 10.1) with the selection of the fastest speed grade for each FPGA device.

TABLE III. Comparison of RTL-Codes

Architecture Units	I^{2}C-Slave		SPI-Slave	
	Number of Lines	*Size (Ko)*	*Number of Lines*	*Size (Ko)*
Top Module (HSI + FSM)	598	22	434	15
Low-level Protocol	472	19	217	8
FIFO	232	7	232	7
Filter	67	4	67	4
Total	1369*	52	950*	34

* : (1369-950)/950 = 0.44

There is almost no significant difference in terms of delays (Table V). Delays are calculated for two types of paths: Clock-To-Setup and all paths together (Pad-To-Setup, Clock-To-Pad and Pad-To-Pad.) The Clock-To-Setup gives more precise information on the delays than other remaining paths, which depend in fact on I/O Block (IOB) configuration (low/high fanout, CMOS, TTL, LVDS…)

TABLE IV. COMPARISON OF OCCUPIED SLICES

		I^2C-Slave		SPI-Slave	
Xilinx's FPGA Device	***Number of Total Slices***	***Number of Occupied Slices***	***Utilization***	***Number of Occupied Slices***	***Utilization***
xc2s50-6tq144	768^{+}	510	66%	363	47%
xc3s50-5tq144	768^{+}	503	65%	354	46%
xc2v80-6cs144	512^{+}	504	98%	366	71%
xc4vlx15-12sf363	6114^{+}	512	8%	360	5%
xc5vlx30-3ff324	4800^{x}	187	3%	141	2%

+ : each slice includes 2 LUTs of 4 inputs
x : each slice includes 4 LUTs of 6 inputs

The transfer rate for I^2C-Slave is fixed, while for SPI-Slave is unlimited, but for both a timing relationship between the master clock and the synchronous transfer clock must be known to enable the sampling of smallest events depending on the timing constraints of each protocol [7][8]. The master clock (clk) must be at least 5 and 4 times faster than the transfer clock (scl, sclk) for I^2C-Slave and SPI-Slave respectively. In fact only a ratio of 2 is required for SPI-Slave, but the RTL coding style requires 2 additional clock cycles.

TABLE V. COMPARISON OF DELAYS

	I^2C-Slave		SPI-Slave	
Xilinx's FPGA Device	***Clock-To-Setup Paths***	***All Paths***	***Clock-To-Setup Paths***	***All Paths***
xc2s50-6tq144	12.079 ns	19.616 ns	12.234 ns	17.042 ns
xc3s50-5tq144	7.636 ns	14.996 ns	7.835 ns	12.089 ns
xc2v80-6cs144	6.483 ns	12.806 ns	6.604 ns	10.234 ns
xc4vlx15-12sf363	4.863 ns	9.572 ns	5.006 ns	8.880 ns
xc5vlx30-3ff324	3.606 ns	7.998 ns	3.303 ns	6.971 ns

For instance, if we consider the Clock-To-Setup delay of SPI-Slave mapped onto Virtex-5 device (3.303 ns), a transfer rate of 13.212ns can be achieved, corresponding to 75 MBPS.

III. CONCLUDING REMARKS

A practical comparative study of I^2C and SPI protocols has been presented. Our primary concern was the accuracy of the comparison, for which careful measures were set at each development step.

While this comparison is only limited to the slave side of the protocol, logical predictions can easily be stated for the master side: approximately the same delays with a greater area overhead to cope with the multi-master feature of I^2C protocol. For SPI-Master, there will be no significant area overhead compared to SPI-Slave in case a simple counter-based baud-rate is integrated. This will not be the case if a digital-frequency synthesizer is used instead.

The paper has shown up the results of an up-to-date FPGA implementation of the slave side of the two standard protocols I^2C/SPI, which are:

- an utilization ratio of 3% and 2% respectively for I^2C-Slave and SPI-Slave on the smallest Viertex-5 FPGA device;
- a maximum transfer rate of 75 MBPS for SPI-Slave;
- and an area overhead of 25% for I^2C-Slave over SPI-Slave.

As the RTL-code is technology independent, much faster transfer rate can be obtained for SPI-Slave with ASIC implementation using a standard cell library.

REFERENCES

[1] J.M. Irazabel & S. Blozis, Philips Semiconductors, "I^2C-Manual," Application Note, ref. AN10216-0, March 24, 2003.

[2] F. Leens, "An Introduction to I^2C and SPI Protocols," IEEE Instrumentation & Measurement Magazine, pp. 8-13, February 2009.

[3] F. Leens, "Solutions for SPI Protocol Testing and Debugging in Embedded System," Byte Paradigm's White Paper, pp. 1-9, Revision 1.00, June 2008.

[4] L. Bacciarelli et al, "Design, Testing and Prototyping of a Software Programmable I^2C/SPI IP on AMBA Bus," Conference on Ph.D. Research in MicroElectronics and Electronics (PRIME'*2006*), pp. 373-376, ISBN: 1-4244-0157-7, Ortanto, Italy, June 2006.

[5] R. Hanabusa, "Comparing JTAG, SPI and I^2C," Spansion's application note, pp. 1-7, revision 01, April 2007.

[6] P. Myers, "Interfacing Using Serial Protocoles: Using SPI and I^2C". Available: http://intranet.daiict.ac.in/~ranjan/esp2005/paper/i2c_spi_341.pdf

[7] Philips Semiconductors, "The IIC-Bus Specifications," version 2.1, January 2000.

[8] Motorola Inc., "SPI Block Guide V03.06," February 2003.

[9] A.K. Oudjida et al, "Front-End IP-Development: Basic Know-How," Revue Internationale des Technologies Avancées, N° 20, pp. 23-30, December 2008, ISSN 1111-0902, Algeria

[10] A.K. Oudjida et al, "Universal Low/Medium Speed I2C Slave Transceiver: A Detailed FPGA Implementation**,"** Journal of Circuits, Systems and Computers (JCSC), Vol. 17, No. 4, pp. 611-626, August 2008, ISSN: 0218-1266, USA.

[11] R. Usselmann, "OpenCores SoC Bus review," revision 1.0, January 2001.

[12] A.K. Oudjida et al, "Master-Slave Wrapper Communication Protocol: A Case Study," Proceedings of the 1st IEEE International Computer Systems and Information Technology Conference ICSIT'05, pp 461-467, 19-21 July 2006, Algiers, Algeria.

[13] Xilinx Inc., "Virtex-II™ V2MB1000 Development Board User's Guide". Available : http://ww.cs.lth.se/EDA385/HT06/doc/restricted/V2MB_User_Guide_3_0.pdf