FPGA implementation of an ultra-high speed ADC interface

Article · April 2011

DOI: 10.1109/SPL.2011.5782642

CITATIONS
8

READS
8,895

5 authors, including:

Cristian Sisterna
National University of San Juan
12 PUBLICATIONS **108** CITATIONS
SEE PROFILE

Marcelo J Segura
National University of San Juan
14 PUBLICATIONS **172** CITATIONS
SEE PROFILE

Martin Guzzo
National University of San Juan
6 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Gustavo Ensinck
National University of San Juan
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

FPGA IMPLEMENTATION OF AN ULTRA-HIGH SPEED ADC INTERFACE

Cristian Sisterna, Marcelo Segura, Martin Guzzo, Gustavo Ensinck, Carlos Gil

Departamento de Electrónica y Automática
Facultad de Ingeniería, Universidad Nacional de San Juan
{cristian, msegura, m.guzzo, gustavo, cgil} @unsj.edu.ar

ABSTRACT

Nowadays the need for dealing with ultra-high speed Analog to Digital Converter (ADC) is becoming more and more common, from Telecommunications to Precise Instrumentation, every application is increasing the analog to digital interface data rate. The ultra-high sampling rate of the ADCs demands the use of advanced acquisition techniques as well as the latest technology available. The utilization of dedicated Application Specific Integrated Circuit (ASIC) is an expensive solution to deal with the very high throughput from the ADC and its lack of flexibility is a huge drawback. On the other hand, the technology, the architecture and the state-of-the art of the current Field Programmable Gate Arrays (FPGAs) make them especially suitable to act as interface between an ultra-high speed ADC and a data processing unit. Another extra advantage is the reconfigurability of the FPGAs, they can be quickly adapted to different ADCs or different data processing units. The purpose of this paper is to present a practical approach to interface an ultra-high speed 8-bit ADC, MAX104, from Maxim Integrated Circuit, which performs digitalization of the input signal with a sampling rate of 1Gsps and a commercial and popular FPGA, the Virtex2 Pro, from Xilinx Corporation. Once the ADC digital data have been acquired, then they can be processed by either the dedicated FPGA Digital Signal Processing (DSP) blocks, or the FPGA embedded processors or just send the data out to a PC for later processing. Hence, the proposed method of implementation can be used as front-end of a wide range of applications.

1. INTRODUCTION

The analog-to-digital converter (ADC) is a critical component in many of the most demanding applications in the electronics world today. The faster the clock rate of the ADC, the more complex the interface. In recent years the enhancement of ADC architectures and technology [1] and the rapid development of high performance FPGA [2] facilitate the interface between them, eliminating the need of using ASICs. Even though there are some ultra-high speed ADCs in the market, there are not many works done on interfacing these devices with an FPGA at their maximum data rate [3][4][5].

[1] Xilinx University Program (XUP) sponsors part of this work with the donation of a XUP Virtex II-Pro Evaluation Board.

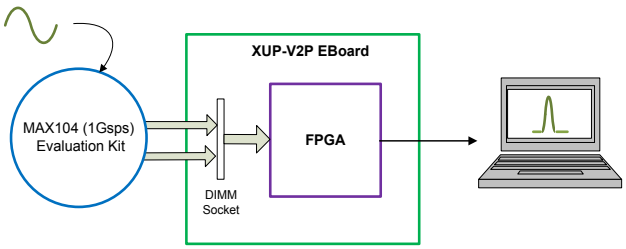

Fig. 1. General scheme of the hardware to test the proposed implementation

This paper proposes an implementation method to successfully interconnect an ultra-high speed ADC and an FPGA. To be able to test the ADC-FPGA interface, a MAX104 Evaluation Kit, from Maxim Integrate Components, and a Xilinx University Program Virtex II-Pro (XUP-V2P) evaluation board[1], from Xilinx Inc., were connected as detailed in Fig. 1.

There are several advantages on using an FPGA as read-out device, mainly because the enormous real time data processing capability that the FPGAs have. The data processing can be done using the embedded Digital Signal Processing (DSP) blocks available in the FPGA, which can work in parallel to process a huge amount of data. Data can also be processed using powerful embedded Soft Core Processors, some of them available for free in the market. Even more, some FPGAs have hard-coded processors, such as the PowerPC in the Virtex family of Xilinx. On the other hand the data can also be processed in a PC through a PC interface implemented in the FPGA. Thus, the data processing power that an FPGA offers is almost unlimited, as it is illustrated in Fig. 2. Consequently, the proposed implementation could become the front end for different kind of applications that would process the ultra-high speed ADC acquired data as front-back.

This paper is structured as follows. Section II describes the details of the high-speed interface between the ADC and the FPGA, emphasizing the mode that the two boards are connected as well as the signal integrity issues faced. Section III describes the proposed implementation. Section IV goes through the actual design consideration, describing in particular detail the different timing considerations and timing equations to be able to successfully interface the two devices. Finally, in Section V the results are shown.

978-1-4244-8848-3/11/$26.00 ©2011 IEEE

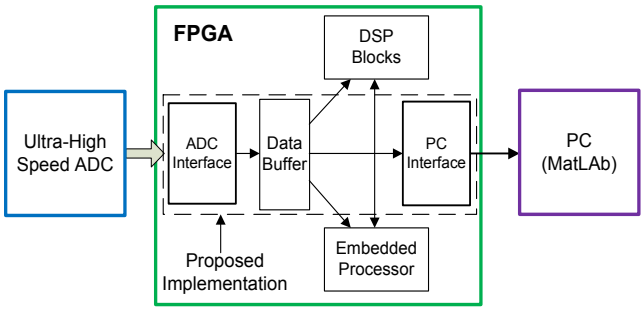

Fig. 2. Different data processing options with the proposed implementation

2. FPGA-ADC HIGH SPEED INTERFACE

2.1. Ultra-High Speed ADC

The MAX104 is an 8-bit, 1Gsps flash analog-to-digital converter with on chip track/hold amplifier and differential PECL-compatible outputs, with an error magnitude of 1LSB [1]. The analog input is designed for either differential or single-ended use with +/- 250mV input voltage range. An integrated 8:16 output demultiplexer simplifies the interface by reducing the output rate to one-half the sampling clocking rate. The demultiplexed outputs are presented in dual 8-bit format with two consecutives samples appearing in the primary and auxiliary output ports on the rising edge of the data ready clock, as it is depicted in Fig. 3, the auxiliary data port contains the previous sample, and the primary output contains the most recent data sample.

The MAX104 Evaluation Kit (MAX104-EKit) contains all circuitry necessary to evaluate the dynamic performance of these ultra-high speed converters, emphasizing special precautions and design feature on the board layout. The sampling frequency for the MAX104-EK is determined by a very stable external sampling clock of 1GHz.

2.2. Xilinx University Program Virtex II-Pro Evaluation Board

The XUP-V2P board has a Virtex XC2VP30 FPGA on it, with a large amount of Configurable Logic Blocks (CLBs), RAM blocks (BRAM), Multi-Giga Bit Transceivers (MGT) and other logic components available to be configured. Among all the available connections on the XUP-V2P board, there is one 184-pin Dual In-Line Memory Module (DIMM) socket that provides access up to 2GB of Double Data Rate (DDR) SDRAM [6]. This socket is available on the XUP-V2P board to interface the FPGA with SDRAM DDR memories. However, to carry out the proposed interface of this work, the usage of the DDR SDRAM socket was altered, as it will be explained in the next points.

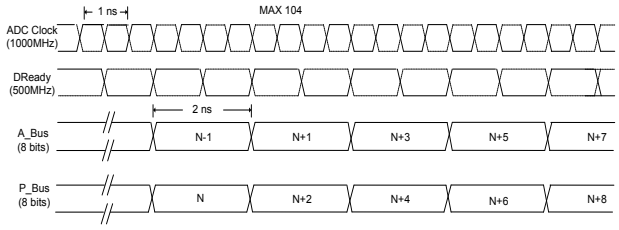

Fig. 3. Demultiplexed data output from the MAX104

2.3. Connecting the ADC with the FPGA

The ADC-FPGA system interface from the clocking point of view it is a source-synchronous clocking system, with the DREADY clock generated by the MAX104 EKit as the source clock for both, A_Bus and P_Bus, data buses, Fig. 3.

Stub Series Terminated Logic for 2.5 volts (SSTL2) is a DDR SDRAM memory bus standard. Thus, it is the I/O standard available at the DIMM socket in the XUP-V2P board. This standard requires a 25 ohms series resistor and a 50 ohm termination resistor pulled to a termination or reference voltage (Vtt). Both, the termination resistor and the termination voltage Vtt, are already implemented and fixed in the board tied to specific FPGA IO pads. Hence, the SSTL2-II I/O logic standard can not be changed for the IO pins dedicated to the DDR SDRAM interface available at the DIMM socket. For the reason previously explained, the I/O standard in the DIMM socket can not be changed. Therefore, a study was done to find out the feasibility of connecting the PECL outputs (from the MAX104 ADC) to the SSTL2-II input (input standard already fixed for the FPGA in the XUP-V2P board). The study concluded that it is feasible to interconnect the two different IO standards as long as two conditions were satisfied. One is to select the right combination of differential IOs of the FPGA available at the DIMM socket to be connected with the right PECL differential signal from the MAX104. Thus, both board schematics were looked at carefully to make the right connections. The other condition to be satisfied is in the MAX104-EKit board, the voltage for Vcco has to be set as lower as possible (using a variable resistor available in the board). Thus, the logic levels of the MAX104 PECL output [7] can be properly recognized by the SSTL2-II FPGA input [6].

It has to be mentioned that the XUP-V2P board does have other multi-purpose connector available to interface the Virtex 2-Pro FPGA with other devices; however, this connector is used for low speed interfaces (<100MHz).

2.4. Boards Interconnection and Signal Integrity

At the planned working frequency signal integrity is a critical issue. The delay caused by the physical length of the trace is one of the problems of signal integrity.

However, this problem is successfully overcome in the XUP-V2P

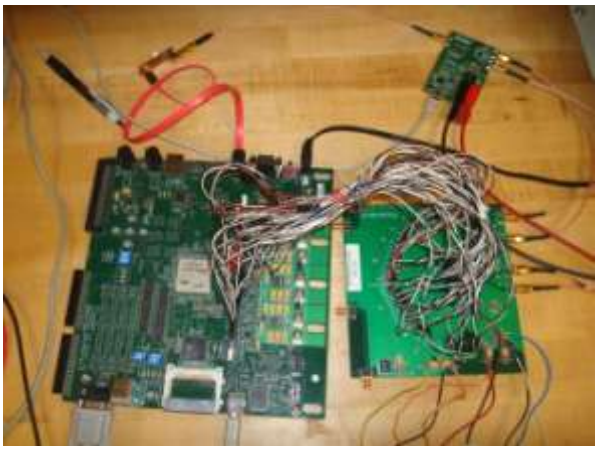

Fig. 4. Connection between the Max104 and the XUP-V2P boards

board by the serpentine traces used between the DIMM socket and the FPGA IO pads for delay match [8], consequently the routing delay among the traces is almost null. Similar approach is utilized on the MAX104-EKit, the traces between the MAX104 outputs pads and its respective connectors [7] have serpentine traces to match the length of the traces.

Other source of signal integrity problem is the quality of the signal. In this case each ADC PECL differential output is wired to the DIMM socket using same length pair of cables. A total of 16 pairs of PECL output signals were connected to the XUP-V2P board via the DIMM socket using same length pair of cables. Due to the good quality of the boards and excellent trace terminations; ringing, overshoot or undershoot values of the data lines were between values that not affected the logic values of the data signals.

Fig. 4 is a picture of the connection between the XUP-V2P board (on the left) and the MAX104-EKit. Note the DIMM connector on the XUP-V2P with the customized PCB board, inserted in the socket, used to connect the two boards.

3. Proposed Interface Architecture

The proposed interface architecture to be implemented in the FPGA is detailed in Fig. 5. Since the FPGA has Dual Data Rate (DDR) flip-flops in the Input/Output Block (IOB), a large amount of CLBs and several internal RAM memory blocks (BRAMs), all the ADC control logic, data read logic and interface with the PC can be done within the FPGA.

The DDR data coming from the ADC are registered at the DDR flip-flops in the IOBs of the FPGAs. These flip-flops are clocked by the 250MHz clock, on both positive and negative edges, coming out from the Digital Clock Manager (DCM), which input is the DREADY ADC clock

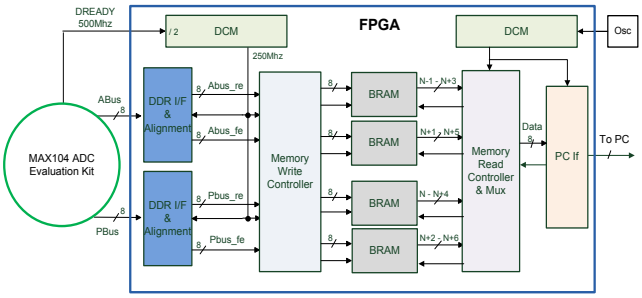

Fig. 5. ADC-FPGA interface architecture

running at 500MHz. Once the acquisition has been done the acquired data is available in the BRAM blocks, implemented as First In First Out (FIFO) memory, ready to be transfer to the PC through a FPGA-PC interface block, also implemented in the FPGA. Next section goes into details of the implementation.

4. FPGA Design Consideration

Among all the considerations that have to be taken into account to be able to meet the timing of the interface, considering that the interface clock period is very narrow, there are three critical items. First of all, the DDR flip-flops within the IOB have to be used to register the ADC data out, avoiding any routing delay to internal flip-flops. The second is that the edge change for data alignment and the other point is that advanced placement and routing techniques have to be used to deal with a data path running at 250MHz in the FPGA.

Fig. 6 illustrates the input logic in the FPGA and the main timing challenges of this interface. The critical timing paths (logic and routing) can be detailed as: the DDR IOB input flip-flops with a capture window of less than 2 ns, the 4 ns data path for the 4 8-bit buses and the 2 ns window of time available to change the synchronization edge of the captured data from negative edge to positive edge.

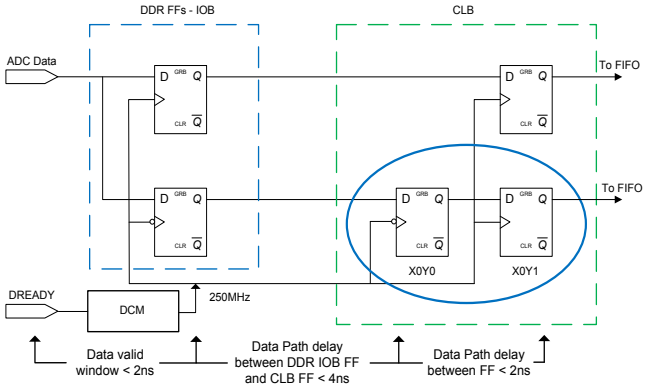

Fig. 6. Detail of the high-speed data path in the FPGA

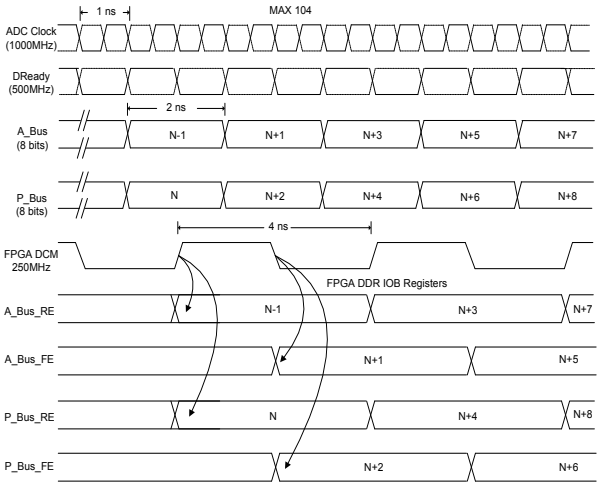

Fig. 7. Time diagram of clocks and data going into and coming out of the DDR IOB

4.1 DDR Interface

The DDR flip-flops in the IOBs were used to capture the data from the ADC. The data capture on these flip-flops is carry out using both, the rising and falling edge of the clock coming out from the Digital Clock Manager (DCM), which input is the DATARDY clock from the ADC, which has 2 ns of period, as it was seen in Fig. 3.

The output from the DCM is a clock signal with 4 ns of period. Both 8-bit buses from the ADC, Auxiliary Data, A_Bus, and Primary Data, P_Bus, are synchronous with DATARDY.

Fig. 7 details the relation between the ADC clock, the two ADC 8-bit buses, A_Bus and P_Bus, the FPGA DCM clock and the resultant 4 8-bit buses coming out of the DDR IOB registers.

Even though the use of DDR registers relaxes the clock requirement to 250MHz, the data from the ADC remain stable at the FPGA IO pad for just 2ns and must be registered within this tiny time frame. Moreover, the 2ns time window is reduced by the following factors:

- The FPGA DCM clock output is deskewed and phase aligned with the incoming clock, DREADY, however it does have jitter in its clock output (ClkJt) used to capture the DDR data, worst case value is 200ps [2].
- Although the clock output from the DCM is routed using the global clock network resources, in which the skew is minimum, some skew actually exists and has to be considered due to the working frequency of this design. The worst case estimate of the clock skew (ClkSk) is approximately 100ps [2].
- Setup time for the DDR IOB register (FFStp), for the XC2VP30 with speed grade -6 is 860ps [2].
- The hold time for the DDR IOB register is -630ps. A negative hold time means that the data is not expected to stay stable after the clock edge. Since its magnitude is less than the setup requirement, it does not affect the 2ns window.
- Skew between the different bits of the data bus (DSk), even though the MAX104 evaluation kit has same trace delay for each bit of the bus, and same length cables were used, an estimation of 150ps worst case skew was used.

Adding all the factors mentioned above and taking them from the DATARDY clock period (DRdyPrd), the available time window for latching the data coming from the ADC in the DDR IOB register can be expressed as in (1) and calculated as in (2)[11]:

$$T_{window} = DRdyPrd - ClkJt - ClkSk - FFStp - DSk \quad (1)$$

$$T_{window} = 2ns - 200ps - 100ps - 860ps - 150ps \quad (2)$$

$$T_{window} = 690ps$$

To find out whether it is possible to meet these challenging timing requirement, several hardware tests were run using two approaches, one using the constant shift feature of the DCM to shift the clock edge forward or backward to be in the middle of the timing window [9], and the other just without shifting the clock. Since the results were the same, the method of no shifting the clock was decided to be used, since it is the simplest. However, shifting the clock coming out of the DCM is an approach to keep in mind when it is necessary to deal with such tight data timing window.

4.2. Data Alignment

At the output of each DDR IOB there are two data, one synchronized with the rising edge of the clock and the other synchronized with the falling edge. Thus, data alignment to a unique edge of the clock had to be performed. Usually, a couple of flip-flops would change the data alignment from the falling edge to the rising edge. However, in this case is critical the delay of the routing between the two flip-flops considering the ultra-short period of time between the edges, 2ns. Consequently, some timing considerations have to be followed to accomplish the edge change with two flip-flops.

The following equation must be true [9][10][11] to be able to meet the timing requirement:

$$DataPath + ClkSk + 2 * |ClkJt| + \%Tlrce < ClkPrd/2 \quad (3)$$

Where,

$$DataPath = FFClktoXQ + RteDly + FFStetupBYinput \quad (4)$$

The timing component $\%Tlrce$ is a percentage of tolerance regarding the period of clock. Usually the percentage is a value between 10 to 15%.

Obtaining the timing parameter values from the FPGA datasheet [2] and analyzing the results from Timing Analyzer and FPGA Editor, it is possible to replace the values in (3) and (4), obtaining:

$$0.38ns + RouteDly + 0.24ns + 0.0ns + 0.120ns + 0.1ns < 2n$$

The only way to get a routing delay between flip-flops of less than 2ns in an FPGA, that has more than 30,000 flip-flops, is to force that the two alignment flip-flops belong to the same CLB. The Integrated Software Environment (ISE), a Xilinx design software suite, takes into consideration different sources to carry out its synthesis and place and route tasks. One of these sources is the constraint file, called User Constraint File (ucf) in Xilinx ISE environment, which specifies timing and placement settings. Thus, using placement constraints per each pair of flip-flops is possible to force the Place and Route tool to use flip-flops within the same CLB, in adjacent SLICEs. The following placement constraints were used for one of the pair flip-flops:

```
INST "reg_inp_fe" RLOC = X0Y0;
INST "reg_out_re" RLOC = X0Y1;
```

After synthesis and place and route the design a routing delay of 0.618 ns was measured in FPGA Editor between the flip-flops.

A point to bring out is the fact that the two flip-flops contained in the same SLICE within a CLB can not be used to carry out the data clock edge change, since they share the same clock edge.

4.3. BRAM Buffer

Once the ADC digital data are aligned, at the outputs of the flip-flop there are 4 8-bit buses synchronized all on the rising edge of the clock. Then, an FSM controls the write procedure to storage the acquired data into its respective FIFO (one FIFO per bus). Likewise, another FSM controls the read procedure to read out the data from the FIFOs and arrange (multiplex) the data to be sent out.

4.4. Interface to the PC

For this experimental work a simple RS-232 interface was written in VHDL and implemented in the FPGA to connect the ADC-FPGA system to a PC, using the serial interface of Simulink (MatLab).

5. FPGA VHDL CODE

All the logic inside the FPGA was described in generic VHDL code, except for the blocks that are FPGA fabricant dependent, such as the DCM, the FIFOs and the IODDRs. These blocks were either generated by Xilinx CoreGen tool or instantiating component primitives.

The Integrated Software Environment (ISE) was used for design synthesis and implementation. ModelSim, from Mentor Graphics, was used for functional and post place and route simulation. In order to achieve the high performance required by the timing constraints, proper VHDL coding techniques were used in the written code [13][14][15].

6. RESULTS

Fig. 8 shows an example of application of the proposed interface. A 2 ns pulse of an Ultra Wide Band transmitter is perfectly acquired in the FPGA and then processed by MatLab in a PC [16].

Table 1 details the FPGA resources utilized for this implementation. As it can be seen there are plenty of resources still available to be utilized for either processing the acquired data inside the FPGA, or embedding this interface in a more complex data processing system.

7. CONCLUSIONS

A practical approach to implement an interface between an ultra-high-speed ADC and an FPGA has been presented. This approach laid the groundwork for implementing inside

the FPGA complex DSP algorithms or embedded processors able to process the very-high throughput of data coming out from the ADC, in applications such as telecommunications, radar/sonar systems, high-speed data acquisition, etc. The high-speed FPGA design techniques presented in this work have been already successfully tested in hardware [16].

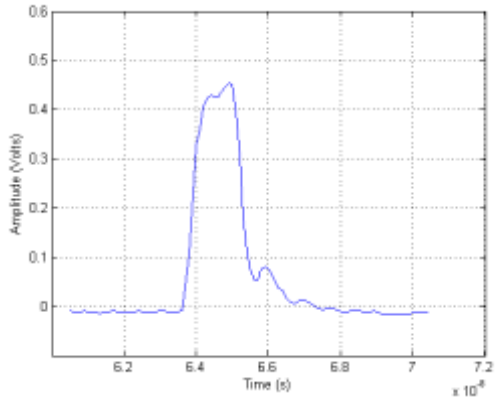

Fig. 8. An example of an ultra-narrow pulse acquired by the proposed interface

As future works, the design and assembly of one board integrating the ultra-high speed ADC and FPGA is projected as well as a higher speed interface to the PC.

Table 1 - FPGA resources utilized

Resource	Available	Used	Percentage Used
SLICEs	13696	194	1%
RAMB16	136	44	32%
DCMs	8	2	25%

8. REFERENCES

[1] MAXIM 104: 5V, 1 Gsps, 8-bit ADC with On-Chip 2.2 GHz Track/Hold Amplifier, MAXIM Integrated Products Inc. [Online].

[2] Xilinx Virtex-II Pro and Virtex-II Pro X Platform: Complete Data Sheet, DS083. Xilinx Inc. March 2007.

[3] C. Wei, L. Xingguang, L. Guiying and C. Dianren, "High-Speed Acquisition and Processing using FPGA." 2009 Third International Symposium on Intelligent Information Technology Application, pp 648-650, April 2009.

[4] T. Lin and Z. Zhengou, "The Implementation of 100MHz Data Acquisition Based on FPGA," in Proc. IEEE, The Third IEEE International Woskshop on System-on-Chip for Real Time Applications. June 2003.

[5] B. Xin, D. Jinsong and F. Wei, "Design e Implementation of an Ultra-High Speed Data Acqusition System for HRRATI," in proc. IEEE, 2009 IEEE Symposium on Industrial Electronics and Applications. October 2009.

[6] Xilinx Inc. Xilinx User Guide 069, UG069, "Xilinx University Program Virtex-II Pro Development System". April 2008.

[7] Maxim Integrated Circuits Inc. Max104/Max106/Max108 Evaluation Kits.

[8] Howard Johnson and Martin Graham, "High-Speed Digital Design, A Handbook of Black Magic," Prentice Hall, 1993.

[9] John F. Wakerly, "Digital Design, Principles and Practices", Fourth Edition, Prentice Hall. July 2005.

[10] Xilinx Inc. Xilinx Application Note, XAPP606, "XGMII using the DDR registers, DCM and SelectI/O-Ultra features." July 2002.

[11] Xilinx Inc. Xilinx Application Note, XAPP259, "System interface timing parameters." April 2003.

[12] Xilinx Inc. Xilinx Application Note XAPP268 "Active Phase Alignment." December 2002.

[13] IEEE Std 1076-2002, "IEEE Standard VHDL Language Reference Manual." July 2002.

[14] Open Cores Organization. "OpenCores Code Guidelines", Rev. 1.1. August 2002.

[15] Pong P. Chu, "FPGA Prototyping by VHDL Examples," Wiley-Interscience, 2008.

[16] M. Segura, H. Hashemi, C. Sisterna, V. Mut, "Experimental Demonstration of Self-Localized Ultra Wideband Indoor Mobile Robot Navigation System." IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN), Zürich, Switzerland. 15-17 September 2010.