Parameterized control layer of FPGA based cavity controller and simulator for TESLA Test Facility

Krzysztof T. Pozniak, Ryszard S. Romaniuk
Institute of Electronic Systems, Nowowiejska 15/19, 00-665 Warsaw, Poland
K. Kierzkowski
Institute of Experimental Physics, Hoza 69, 02-093 Warsaw, Poland

ABSTRACT

The paper describes a functional idea of the parameterized control layer for the FPGA based advanced electronic and photonic systems. The systems under considerations are used (or planned to be used) for the construction of distributed, control, simulation, measurement and data acquisition in the Low Level Radio Frequency (LLRF) part of the TESLA and X-Ray FEL projects in DESY. Practical realization of the control layer was presented. The implementation was done in FPGA Xilinx *VirtexII V3000* chip embedded in *XtremeDSP Development Kit* board by Nallatech. The designed and implemented communications protocol was described. The protocol is based on the standard parallel EPP transmission from the PC.

Keywords: Super conducting cavity control, communication, FPGA, LPT, VHDL, Xilinx.

1 INTRODUCTION

The realization of the FPGA based controller and simulator for the resonant, supeconducting TESLA cavity requires the introduction of a large number of parameters, realization of fast monitoring data acquisition and current, effective system control from the software level. It means, that a reliable and fast communications is required for the FPGA based LLRF system to be able to use fully its potential. A general block diagram of the multilayer structure of the FPGA based system was presented in fig.1.

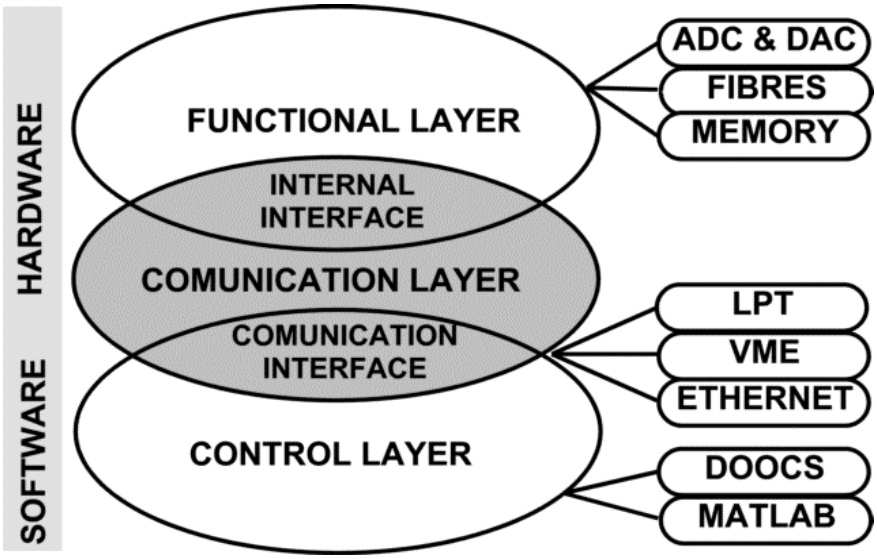

Fig. 1. Multilayer structure of the FPGA based LLFF system.

The Functional Layer embraces fast realization of the DSP algorithms for the TESLA SC cavity controller and simulator implemented in the FPGA chips [1,2,3]. A number of external components is combined with this layer like: AC and CA converters, I/O circuits for clock signals, additional memory blocks, etc.

The Control Layer is build of appropriate software using such programming tools and environments like C++, MATLAB, DOOCS, etc.

The subject of this paper concerns the characteristics and implementation of the *Communication Layer*. Its task is to provide the fast, effective and reliable flow of information inside the FPGA based system. This communication layer should be realized in a form of a standardized interface between the functional and control layers. The interface should be flexible to cooperate with particular communication media like: VME, Ethernet, LPT, USB etc.). It should be adaptable to the frequent changes of the requirements by the control layer, implemented in the FPGA chips. The above problems are considered in two domains:

- *logical*, for which a compatibility should be provided between the space of the communication interface (implemented in FPGA chips and in the control software). A unique procedure to modify the communication space is here a primary requirement. This is realized by the communication layer, called the *Internal Interface* (II), and specified in logical domain in ch.2.
- *hardware*, which provides physical distribution of the data and control signals. Hardware specification of the *Internal Interface* embracing the FPGA chips was described in ch.2. The description of realization and the properties of transmission protocol via the parallel EPP port was described in ch.3.

The work presents practical implementation of the system using the vendor ready PCB set *XtremeDSP Development Kit* by Nallatech (fig.2). The mainboard *BenONE* is equipped with a daughterboard *BenADDA*. The DB is armored in two 14-bit A/C and C/A converters and the

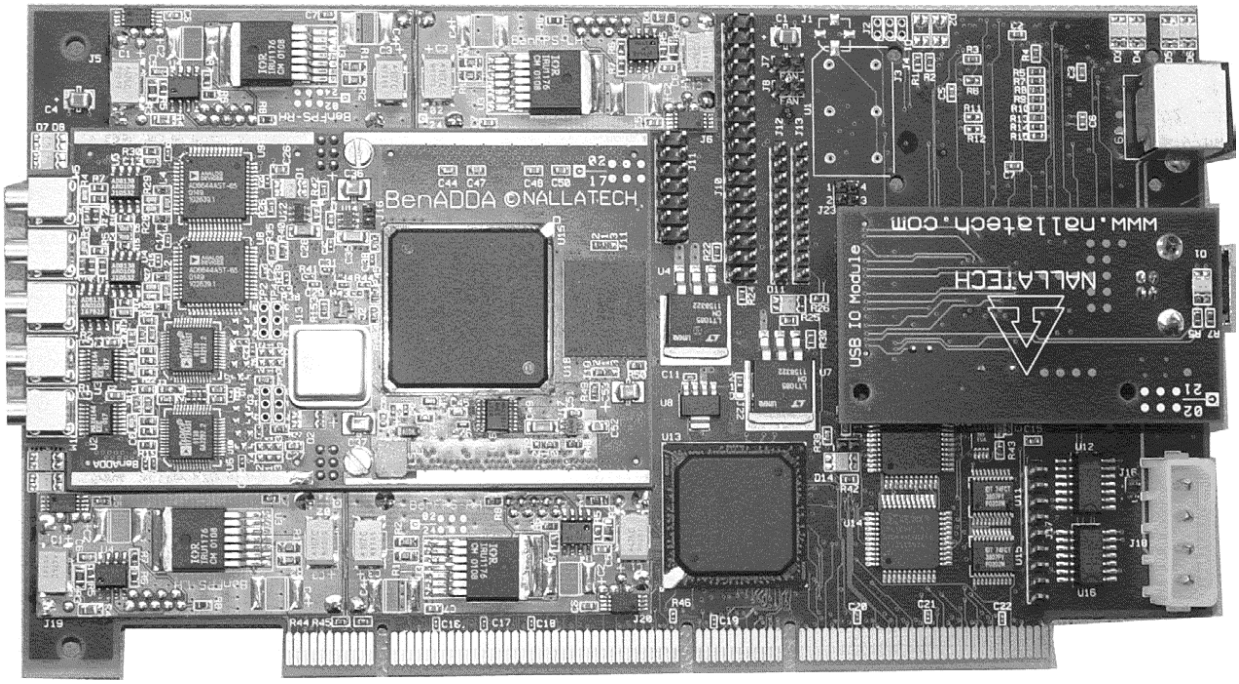

Fig. 2. Laboratory PCB set-up XtremeDSP Development Kit by Nallatech. The MotherBoard *BenONE* a DaughterBoard is placed. The *BenADDA* DB has Virtex-II 3000 [5]

FPGA chip by Xilinx VirtexII (V2000 or V3000) [4]. During the board tests all I/O ports were measured. The USB ver.1.1. chip applied on the board turned out to be unacceptably slow. The PCI connector turned out not to be fully operable. Thus, the both mentioned ports were omitted in the further design. Alternatively, the available EPP port was used practically, and turned out to be quite fast and effective. The EPP I/O port was accepted due to its compatibility with the PC technology, large data transfer, and the simplicity in its implementation in the FPGA chip.

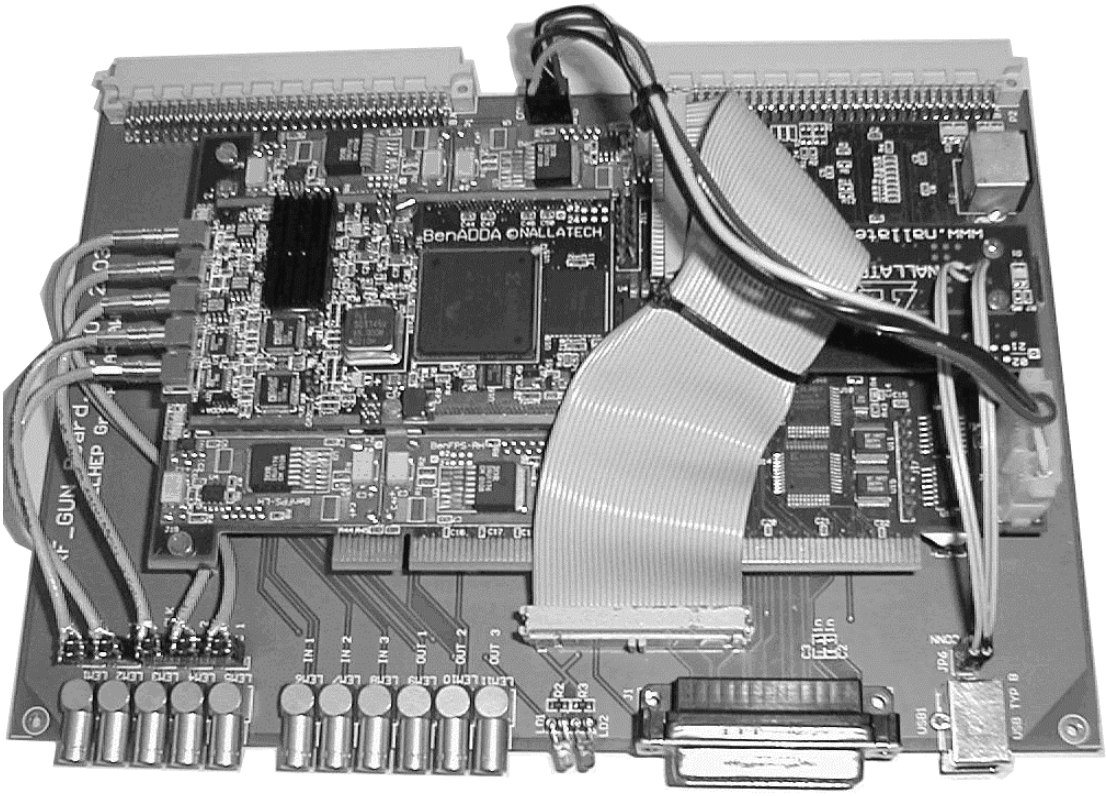

Fig. 3. The set-up of XtremeDSP Development Kit by Nallatech embedded on the standard VME MotherBoard EURO-6HE. The design and construction realized by the authors.

The adaptation of the *XtremeDSP Development Kit* to the EURO 6HE standard was realized with the use of additional carrier MotherBoard (fig.3). The front page of the VME MotherBoard has the EPP interface socket and LEMO connectors for the convenient and safe connections of the analog and digital signals, which are necessary for the LLRF project. The power supply to the *XtremeDSP Development Kit* is provided directly form the VMEbus.

2 PARAMETERIZED COMMUNICATION LAYER FOR FPGA CHIPS

The functional structure of the parameterized communication layer was presented in fig.4. This structure was proposed for the FPGA based TESLA LLRF system electronics. The architecture of the parameterized communication layer is based on the imaging of the hardware functional blocks (called components) to the virtual objects in the software (called objects). A proprietary solution of this hardware to software imaging process was proposed. The *Internal Interface* standard was implemented. Thus, through using the *II,* the access method to the register and memory elements in the FPGA chip gets independent on the practically applied communication bus in the system. The *II* standard provides for design automation of the address space from the hardware side (VHDL) and from the software side (C++).

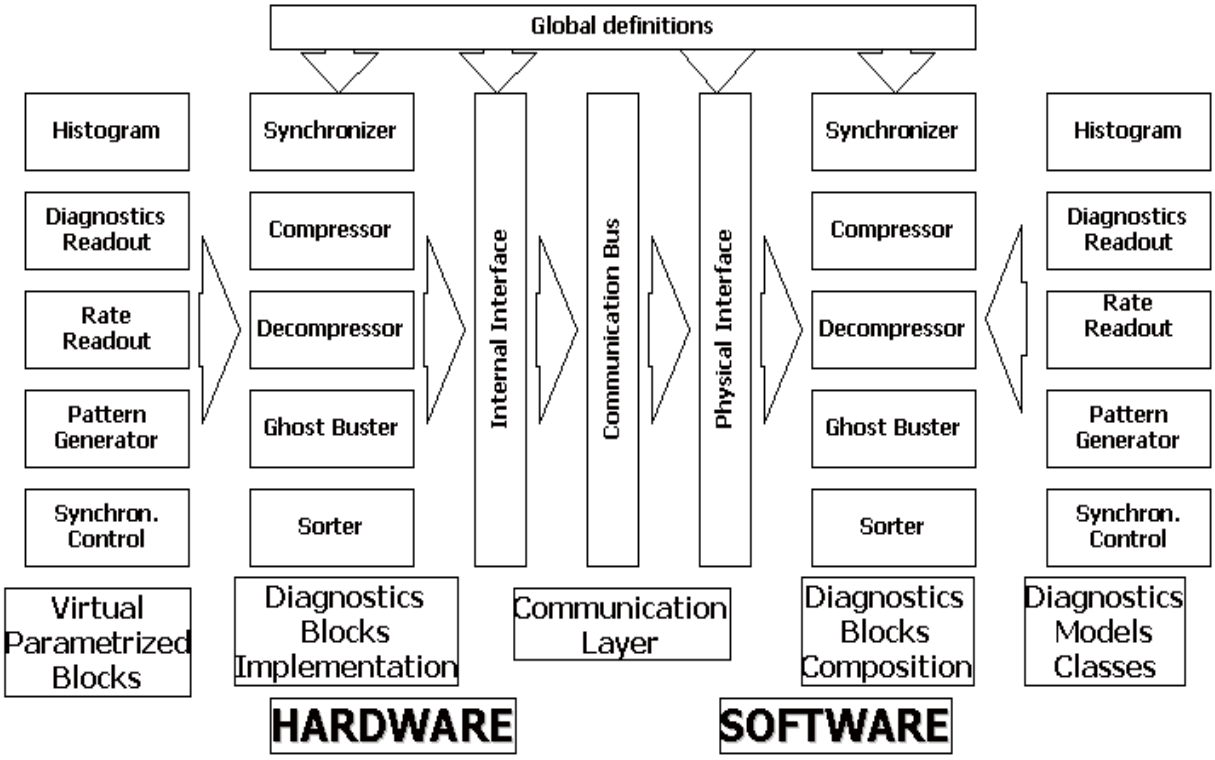

Fig. 4. Functional structure of the internal communications layer.

Addres	D7	D6	D6	D4	D3	D2	D1	D0	remarks
0	A0-7	A0-6	A0-5	A0-4	A0-3	A0-2	A0-1	A0-0	Address space of subarea A
1	A1-7	A1-6	A1-5	A1-4	A1-3	A1-2	A1-1	A1-0	
2	A2-7	A2-6	A2-5	A2-4	A2-3	A2-2	A2-1	A2-0	
3									
4					B0-3	B0-2	B0-1	B0-0	Address space of subarea B
5					B1-3	B1-2	B1-1	B1-0	
6					B2-3	B2-2	B2-1	B2-0	
7									

Fig. 5. Physical imaging, in the FPGA chip, of an exemplary area of the virtual communication space. The area consists of 3 memory cells of 12-bits in the width. The width of the *II* bus is 8-bit. Partition of physical address space embraces two sub-areas, marked as A and B. The A sub-area means the lower 8-bit part of the 12 bit data word. The B sub-area means the 4 higher bits of the same data word. The gray color means meaningless bits of the address space. It was assumed, for the simplicity, that the addressing is initialized from the position of 0. Thus, the relative addresses of 3 and 7 are not used.

The basic foundations to build the virtual communication space are based on: the creation of the parameterized description, assumption of strictly defined syntax (localized in the header file). The header file is processed in parallel to the source codes in VHDL and C/C++. Fig.5 presents an implementation example of the acting algorithm for the memory space exceeding the dimensions of the physical communication *II* bus.

The access processing to the *II* standard, from the VHDL, is realized by the standardized actions. Each line of the definition file is created according to the common rules. The rules are valid for the VHDL and C/C++. The rules concern the bases of grouping, optimization of the address space, concatenating of the group of bits to vectors, etc.

The effect of the process of building the interface is creation of the physical interface implementation, i.e. functional ordering of the addresses with taking into account the requirements of the grouping and data partitioning in case they are oversized relative to the width of the data interface bus.

Table 1. List of parameters of the *II*.

Component	Content	Description and Interpretation	
ItemType	VII_PAGE	Record of a common addressing area.	O
	VII_VECT	Record of common bit vector.	
	VII_BITS	Record of description bit (status bit).	
	VII_WORD	Record of word description (data register).	
	VII_AREA	Record of area description (memory).	
ItemID	natural number	Unique record identifier.	O
ItemWidth	natural number	Data record width (in bits).	F
ItemNumber	natural number	Number of record repeatings (indexing), (for VII_AREA the number of memory cells).	F
ItemParentID	natural number	Binding Identifier ItemID, for VII_BITS is bound with VII_VECT, The rest are bound with VII_PAGE.	P
ItemWrType	VII_WNOACCESS	The component has no write right from II.	F
	VII_WACCESS	The component has right to write from II.	
ItemRdType	VII_RNOACCESS	Component has no read right to II.	F
	VII_REXTERNAL	Component allows for external read to II.	
	VII_RINTERNAL	Component allows for internal read to II.	
ItemName	text	Formal name of component.	S
ItemFun	VII_FUN_UNDEF	Non defined functional type of component..	S
	VII_FUN_HIST	Functional type of component - histogramming.	
	VII_FUN_RATE	Functional type of component – frequency.	
ItemDescr	text	Description of component.	S

Used functions:

1. *O* – obligatory parameter, always compiled, always interpreted,
2. *F* – parameter for physical components (VII_BIT, SVII_WORD, VII_AREA),
3. *P* – linking parameter (VII_VECT, VII_BITS, VII_WORD, VII_AREA),
4. *S* – information parameter for software C++ (ignored in VHDL analysis).

The table 1 contains ordered list of parameters describing particular components of the interface.

The *II* standard is build by the list of its components. The component parameters can be divided to:

- *identifying*, enable unique differentiation of the component (for example, type, name),
- *scaling*, defining the physical dimensions of the components,
- *binding*, allow to realize the grouping operations,
- *access*, defining the access rights to the component,
- *descriptive,* contain information used by the C/C++.

The interface structure is described by two groups of components:

- *physical*, defining the real objects of the interface:
  a. VII_AREA – unified address space, for example an internal memory,
  b. VII_WORD – autonomous bit vector,
  c. VII_BITS – collection of bits requiring grouping operation (for example status flags),
- *grouping*, enabling creation of common space areas (address, data) particular groups of physical components:
  a. VII_VECT – binds to a common vector the components of the type VII_BITS; the constructed vector will be treated as a single component VII_WORD.
  b. VII_PAGE – combines the components of the type VII_AREA, VII_WORD, VII_BITS (ordered previously in VII_VECT) into a common address space, possessing a unified prefix.

The physical space of the interface is defined by two parameters:

- ADDR_SIZE, defines the address space, expressed in the number of address lines; an assumption is that the address lines are indexed from 0 to ADDR_SIZE – 1, what means that the whole address space has the following number 2^{ADDR_SIZE} of address positions;
- DATA_SIZE, defines the data vector expressed in bits; an assumption is that the data lines are indexed from 0 to DATA_SIZE – 1, what means that the data values are include in the area from 0 to 2^{DATA_SIZE}-1.

Declaration of the component is done with the aid of three identical initializes, identically interpreted in the VHDL and C/C++. It is required only that the list of declared components begins with a single use of IIDEC_ITEM_*BEG*, and continue with arbitrary number of IIDEC_ITEM_*CON* and end with a single use of IIDEC_ITEM_*END*. The declaration line has the following structure:

```
IIDEC_ITEM_XXX(_ITEM_TYPE_, _ITEM_ID_, _DATA_WIDTH_,
               _POS_NUMBER_, _PARENT_ID, _WRITE_MODE_,
               _READ_MODE_, _NAME_, _FUNCTION_, _DESCRIPTION_)
```

gdzie: `_ITEM_TYPE_`: type of component – parameter ItemType
`_ITEM_TYPE_`: identifier – parameter ItemID
`_DATA_WIDTH_`: width of data bus – parameter ItemWidth
`_POS_NUMBER_`: number of positions – parameter ItemNumber
`_PARENT_ID_`: group identifier – parameter ItemParentID
`_WRITE_MODE_`: write rights – parameter ItemWrType
`_READ_MODE_`: read rights – parameter ItemParentID
`_NAME_`: component name – parameter ItemName
`_FUNCTION_`: component functions – parameter ItemFun
`_DESCRIPTION_`: component description – parameter ItemDescr

The global parameters of the component allow to use the same values of parameters in both codes the VHDL and in C/C++. To assure parameterized description, it was assumed that the identifiers will be symbolic constants. To create such constants, giving simultaneously respective value, one has to use the initializer:

```
IIDEC_IDEN_VAL (_NAME_IDENTIFIER_, _TYPE_, _VALUE_,
                _COMMENT_)
```

where: _NAME_IDENTIFIER_ : is a symbolic name of parameter,
TYPE : defines type of variable,
VALUE : defines value of parameter,
COMMENT : may be a word, stream of words in „” or be empty – possesses information character.

3 PHYSICAL COMMUNICATION LAYER BASED ON EPP STANDARD PROTOCOL

The realization of the physical communication layer between the FPGA chip positioned on the *XtremeDSP Development Kit* PC computer was realized practically using the EPP communication standard (Enhanced Parallel Port) ver.1.7 [6]. The used configuration allows to obtain maximum transfer of 500kB/s. The hardware interface to the FPGA Xilinx Virtex-II chip, realized in the standard TTL technology was presented in fig.6.

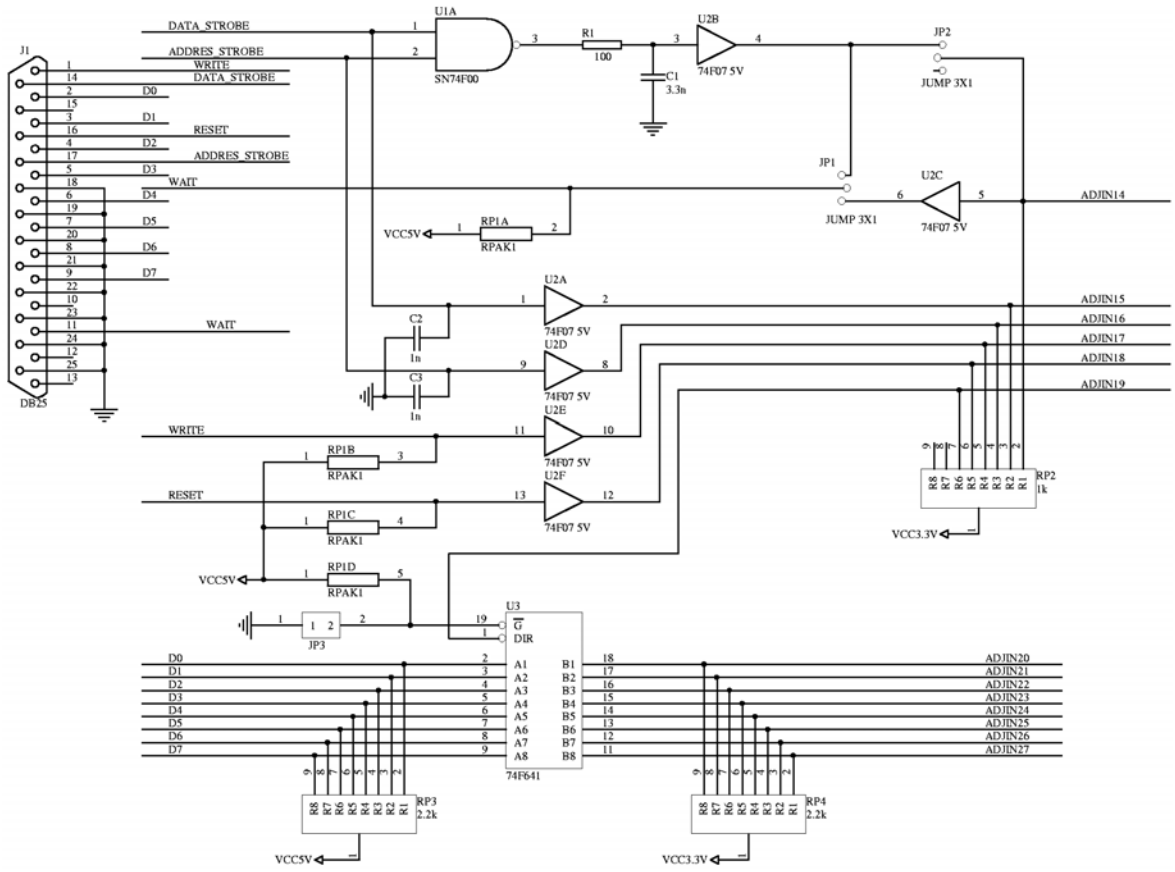

Fig. 6. Diagram of the EPP interface to the FPGA Xilinx Virtex-II chip on *Xtreme DSP Development Kit.*

The U3 (74F641) circuit realizes a bidirectional buffer and voltage converter. The EPP standard uses the 0-5V signals. The Vitrex-II chip accepts 0-3.3V signals. The open collector technology was used in this case in the FPGA chip. A similar solution was applied for control signals using 74F07 gates.

Through the choice of the JP1 jumper, the circuit may work in two modes of WAIT signal operation confirmation:

- *autonomous*, where the access confirmation is generated automatically in the delay circuit realized on the C1,
- *programmable*, where the confirmation signal is provided by FPGA Xilinx.

The data transmission standard is based on the 8-bit words send as data (active signal DATA STROBE) or as addresses (active signal ADDRESS STROBE). Sending of the more complex information packet to the FPGA address space through the II standard is realized as a sequence of 8 bytes. The two first bytes forward the value of 16-bit address, the four following bytes forward include the value of the data word (respectively written and read). The next byte is a control sum. The last byte returns the status word of the realized transmission (confirmation of done operation by the II). An example of the write sequence was presented in fig. 7.

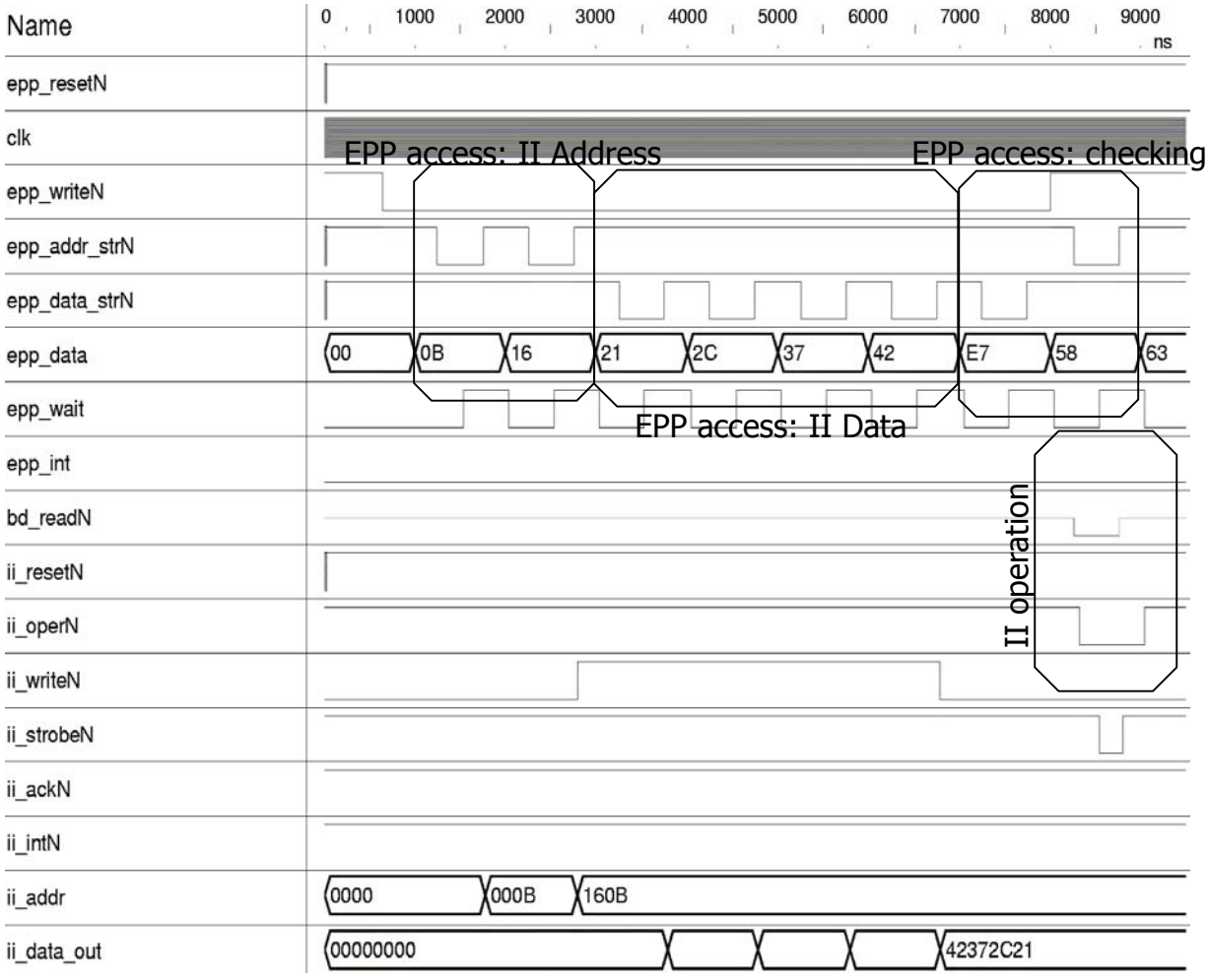

Fig. 7. Exemplary sequence of data transmission (write operation) through the EPP interface to the II standard implemented in the FPGA Xilinx Virtex-II chip.

4 SUMMARY

This work describes concisely the implementation of the communication layer which is used in the FPGA systems. This efficient communication layer was used in the FPGA based TESLA cavity Simulator and controller (SIMCON) system. The efficiency was confirmed by the parameters of the hardware SIMCON. These parameters were described in [7]. The realized system is based on Virtex-II chip residing in the *XtremeDSP Development Kit* PCB by Nallatech. The board is equipped with two 14-bit channel A/C and C/A converters. The described *II* standard was implemented in the FPGA chips and in the control applications in C++. The control application co-operates with the MatLab packet [1] and DOOCS environment [8]. The physical transmission was realized via the EPP and the throughput of 60000 calls/s to the *II* bus was recorded.

REFERENCES

1. T.Czarski, K.T.Pozniak, R.Romaniuk, S.Simrock: "TESLA Cavity Modeling and Digital Implementation with FPGA Technology Solution For Control System Purpose", TESLAReport 2003-28
2. T.Czarski, R.S.Romaniuk, K.T.Pozniak S.Simrock "Cavity Control System Essential Modeling For TESLA Linear Accelerator", TESLA Technical Note, 2003-08:
3. T.Czarski, R.S.Romaniuk, K.T. Pozniak "Cavity Control System, Models Simulations For TESLA Linear Accelerator ", TESLA Technical Note, 2003-09:
4. http://www.xilinx.com/ [Xilinx Homepage]
5. http:// www.nallatech.com/ [Nallatech Homepage]
6. http:// http://www.beyondlogic.org/epp/ [EPP - Enhanced Parallel Port description]
7. K.T/.Poźniak, et al., Functional analyssc of DSP blocks in FPGA chips for application in TESLA LLRF system, TESLA Report 2003-29
8. P.Rutkowski, et. al., FPGA based TESLA cavity SIMCON; DOOCS server design, implementation and application, TESLA Report 2003-32