

# LOGSYS – Development Environment of Embedded Systems

Tamás Raikovich, Béla Fehér, Péter Laczkó

Budapest University of Technology and Economics, Department of Measurement and Information Systems

Magyar Tudósok körútja 2, Budapest, H-1117, Hungary

Phone: (36)-(1)-463-3596, Fax: (36)-(1)-463-4112, rtamas@mit.bme.hu

**Abstract – Thanks to the rapid development of electronics, systems built with programmable logic devices (FPGAs and CPLDs) and microcontrollers are more and more widely used. Their great advantage that arises from their programmable nature as compared to systems using application specific integrated circuits is their flexibility, which cuts back the cost of the development of the prototype to a great extent. Using of programmable devices requires efficient development support. As for simple applications, fast and reliable configuration of the devices is a basic requirement. More complicated applications require efficient support for development, debug and verification. Unfortunately, programmable device manufacturers have such development tools that focus on their own devices and usually offer only a limited communication support with the application itself. The power supply of the target system is also usually left to external instruments. For that reason a new development environment has been created, which integrates the configuration, the communication and the power supply features in a vendor independent manner.**

## I. EMBEDDED SYSTEMS

### A. General Characteristics

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, sometimes with real-time constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. Embedded systems have become very important today as they control many of the common devices we use. Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Physically, embedded systems range from portable devices such as PDAs and MP3 players to large stationary installations like factory controllers. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

Embedded systems are typically implemented with programmable devices: microprocessors, microcontrollers, FPGAs or EEPROM and Flash memories. Contrary to microprocessors that require external peripherals to operate, microcontrollers have built-in peripherals on the chip, reducing the size of the system. There are many different CPU architectures used in embedded systems such as ARM, x86, PIC, AVR, PowerPC, etc. Use of FPGAs enables to create SoPC (System on a Programmable Chip) designs

where all logic is programmed into the FPGA, including the CPU. ASICs are common only in very-high-volume embedded systems because of their very high initial costs.

Power consumption of portable devices is limited because these devices are generally powered from battery. Therefore the low power consumption is very important in this case.

### B. Development Flow

Software for microprocessor systems is usually written in C, C++ or assembly languages. After compiling the source code and linking, the executable file is created. ELF (Executable and Linkable Format) is a common executable file format. With manufacturer tools, the executable can be downloaded to the target or can be converted to different formats such as Intel HEX. Intel HEX files are used to program non-volatile memories.

Designs for FPGA and CPLD devices are written in Verilog or VHDL hardware description language. The development environments create different configuration file formats (BIT, JED, etc.) for these devices.

FPGAs, CPLDs and most microcontrollers support the JTAG configuration interface [1]. This simple synchronous serial interface was originally developed for board-level testing purposes (boundary-scan), but it can also be used to configure programmable devices. The standard SVF file format [2] describes the operations on the JTAG chain. Most manufacturer development environments support this format. Some devices have vendor specific configuration interface such as small Atmel AVR microcontrollers (DebugWire) or Microchip PIC devices (ICD port).

### C. Communication

Embedded systems have their own MMI (man-machine interface) to communicate with the user. Simple embedded devices use buttons, LEDs, and small displays, often with a simple menu system. More complex systems can use a full graphical display with touch sensing. A different option is to provide a web page interface over a network connection.

Various communication interfaces can be found in embedded systems from simple serial interfaces (UART, SPI, I2C, etc.) to high-speed connections (USB, Ethernet, etc.). Some systems also use wireless communication.

For development purposes, typically a UART-like communication interface is used with moderate speed, as a stdin/stdout peripheral.

## II. THE LOGSYS SYSTEM

### A. Design Goals

According to the previous section, many kinds of programmable devices can be found in embedded systems from different manufacturers. Different manufacturers offer different development environments and tools. Therefore the LOGSYS system should be used as a back-end and it should co-operate with the manufacturer development environments by supporting the standard file formats.

As for the configuration interface, the JTAG interface should be chosen because it is widely available in programmable devices. As for communication interfaces, the common serial interfaces (UART, I<sup>2</sup>C, SPI) should be supported. The LOGSYS system should be capable for supplying power to small targets (max. 5 W) and the measurement of the power consumption is also important.

### B. Architecture

A development environment for programmable logic devices consists of two main parts: the development software and the download cable (Fig. 1). The development software serves for design, debug and verification purposes. The download cable connects the target system with the PC.

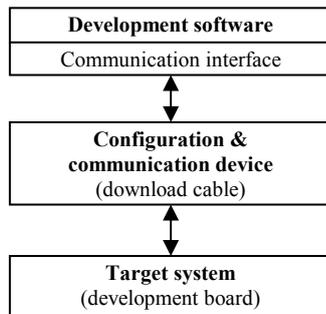


Fig. 1. Architecture of a development environment.

The communication interface determines the usability and the performance of the development environment. An average PC has a number of communication interfaces, like the asynchronous serial port (UART), the parallel port and the USB port. The UART and the parallel port don't provide enough data transfer rate and they will probably disappear from the computers in the future. The USB [3] is a user-friendly way to connect different peripherals to the PC. It is fast (low-speed: 1.5 Mbit/s, full-speed: 12 Mbit/s, high-speed: 480 Mbit/s) and it also provides a short circuit protected 5 V power output. Therefore small devices that consume less than 500 mA current can be powered from the USB port. Because of its numerous advantages, the USB communication interface is used in the LOGSYS system.

Fig. 2 shows the brief architecture of the LOGSYS system. The development cable connects the target system with the PC through the USB port. The dedicated interface between the development cable and the target system is called LOGSYS development port. It provides a configuration interface, a clock and a reset signal, a serial communication interface and a 5 V power output.

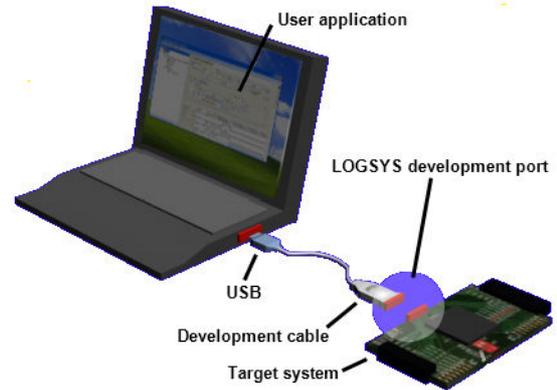


Fig. 2. The LOGSYS development system.

The user application and the device drivers run in a PC environment and they require Windows XP operating system. The user application has a customizable and well-arranged graphical interface for accessing the functions provided by the development cable.

### C. Development Boards

The LOGSYS system can support almost every programmable logic and microcontroller development board. The only requirement is the existence of the JTAG interface. But vendor specific development boards are usually equipped with dedicated connector arrangement, which in general can be handled with flying wire connections.

Therefore a new FPGA board has been created, which is simple enough for the beginners and it is also suitable for implementing more complex applications and it provides direct connection to the LOGSYS development port. Fig. 3 shows the LOGSYS Spartan-3E FPGA card.



Fig. 3. The LOGSYS Spartan-3E FPGA card.

The board contains a Xilinx XC3S250E-TQ144C FPGA device, which enables to implement complex logic and smaller microprocessor systems. This FPGA has 250,000 system gates, twelve 18 Kbit block-RAMs, twelve 18 x 18 bit signed multipliers and four DCM modules.

A 128 K x 8 bit SRAM and a 16 Mbit SPI serial Flash memory are available on the board for storing program and

data. The FPGA can configure itself from the serial FLASH memory.

Other on-board peripherals are one 4-digit 7-segment display, one 5 x 7 dot-matrix display, eight LEDs, eight switches and five push-buttons. A 16 MHz oscillator is available as a clock source.

The LOGSYS development cable can be directly connected to the development port of the board. The board is usually powered by the development cable, but a 5 V power connector is also available for standalone operation.

Two connectors serve for attaching different expansion modules to the board. These modules are called LOGSYS-BLOXES. The pinout of the connectors can be seen in Fig. 4. Each expansion connector provides 13 FPGA I/O lines. 11 of the 13 FPGA I/O lines are bi-directional, the other two are input-only. 5 V and 3.3 V power outputs are available on each connector. However, all FPGA I/O lines are powered from 3.3 V, and they are not 5 V tolerant.

(15)	(13)	(11)	(9)	(7)	(5)	(3)	(1)
Input	I/O	I/O	I/O	I/O	I/O	+3.3V	GND
(16)	(14)	(12)	(10)	(8)	(6)	(4)	(2)
Input	I/O	I/O	I/O	I/O	I/O	I/O	+5V

Fig. 4. The pinout of the expansion connectors.

Fig. 5 shows some expansion modules. A range of expansion modules has been created, including VGA and PS/2 module, 8-channel A/D converter module, audio A/D and D/A converter module, infrared transceiver module, Ethernet module and SD card module.

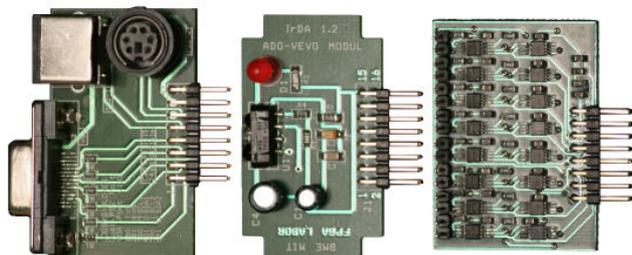


Fig. 5. The LOGSYS-BLOXES modules.

### III. THE LOGSYS DEVELOPMENT CABLE

Fig. 6 shows the LOGSYS development cable. Because of the small size (48 mm x 17 mm), every function is implemented within a single microcontroller. The development cable has a mini-B type USB socket. A standard USB cable can be used to connect the development cable to the PC. The cable is powered from the USB therefore no external power supply is required.



Fig. 6. The LOGSYS development cable.

#### A. LOGSYS Development Port

The pinout of the LOGSYS development port can be seen in Fig. 7. The different interfaces provided by the LOGSYS port are discussed in detail in the following sections.

JTAG TDO	JTAG TCK	CLK	MOSI (ser. out)	V <sub>ref</sub> I/O	5 V
JTAG TDI	JTAG TMS	RST	MISO (ser. in)	GND	V <sub>ref</sub> JTAG

Fig. 7. The pinout of the LOGSYS development port.

Programmable devices may require more than one power supply voltage to operate. For example, the Xilinx Spartan-3E FPGA devices have three types of power supply input [4]: the core requires 1.2 V, the auxiliary supply voltage (JTAG, DCM, differential drivers) is 2.5 V and each I/O bank of the FPGA has dedicated power supply inputs. The I/O voltage level is determined by the used I/O standard, it can be between 1.2 V and 3.3 V. Because different target systems can use different voltage levels for communication, the development cable contains level shifter circuits. The level shifter circuits have two reference voltage inputs: one for the configuration interface (V<sub>ref</sub> JTAG) and one for the control and communication lines (V<sub>ref</sub> I/O). This flexibility enables the development cable to be easily attached to many targets. Reference voltages can be between 1.65 V and 5 V.

#### B. Configuration Interface

The native configuration interface is the JTAG interface for configuring programmable devices. The TCK frequency is 1 MHz. At this clock frequency, the configuration of an XC3S250E FPGA takes about 2100 ms.

The configuration interface also supports the low voltage programming of Microchip PIC18F microcontrollers.

#### C. Communication Interface

The development cable supports a range of synchronous and asynchronous serial communication protocols.

Basically, the popular UART can be used to communicate with the target system. A virtual serial port driver has been created so the UART of the development cable can be accessed from Windows applications.

For simple tests or educational purposes a special communication mode called BitBang I/O is available. In this mode the firmware directly controls the clock, changes the reset and serial data out lines and samples the serial data input line at the rising or falling clock edge.

The development cable also supports the master USRT (synchronous version of the UART), the master SPI and the master I<sup>2</sup>C (TWI, SMBus) communication modes.

TABLE I  
DATA TRANSFER RATES

Mode	Min.	Max.
UART / USRT	4800 bit/s	115200 bit/s
BitBang I/O	1 Hz	1000 Hz
Master SPI	2 kHz	8 MHz
Master I <sup>2</sup> C	1 kHz	400 kHz

#### D. Control Interface

The user can freely control the CLK clock and the RST reset lines when they are not used by the communication interface. In this case, the clock frequency can be set between 1 Hz and 8 MHz and the reset signal is controlled asynchronously.

#### E. Power Supply and Measurement

USB ports have short circuit protected 5 V power output and supply 500 mA current. Because the development cable consumes only a small amount of current, the USB port can be used to power the target system. The development cable has a power switch with adjustable current limits of 450 mA, 700 mA and 950 mA. Setting the current limit greater than 450 mA requires a Y-type USB cable. Systems that consume more than 950 mA current cannot be powered from the development cable, they require external power supply.

The voltage on all power lines (power output, reference voltage inputs) and the output current of the power output are measured by the development cable.

#### F. Connection with the Development Boards

Development boards have various types of configuration ports therefore there is no universal configuration port that is compatible with each target system. The LOGSYS development cable can be connected with the target systems in different ways: directly, using an expansion module or using a flying cable.

LOGSYS development boards are directly compatible with the LOGSYS development cable because all of them have the LOGSYS development port.

Some development boards can be connected with the development cable using a small expansion module. Fig. 8 shows the expansion module made for the Xilinx Spartan-3 Starter Board [5]. In this case, the LOGSYS development cable replaces the power supply, the download cable and the serial cable.

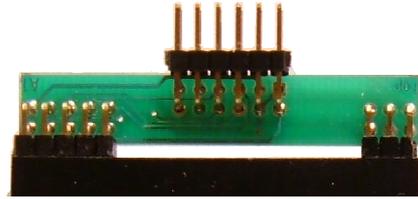


Fig. 8. Expansion module for the Xilinx Spartan-3 Starter Board.

In those cases where the above two methods are inapplicable the user can connect the target system with the LOGSYS development cable using a flying cable.

### IV. THE LOGSYS USER INTERFACE

The main window of the user application can be seen in Fig. 9. The user application has a customizable and well-arranged graphical interface for accessing the functions provided by the development cable. The graphical user interface is implemented using docking windows. Every function has its own graphical interface and they are displayed in separate child windows. Thanks to the usage of docking windows, these interfaces can be arranged in various ways by the user. Services offered by the user application are discussed in details in the following sections.

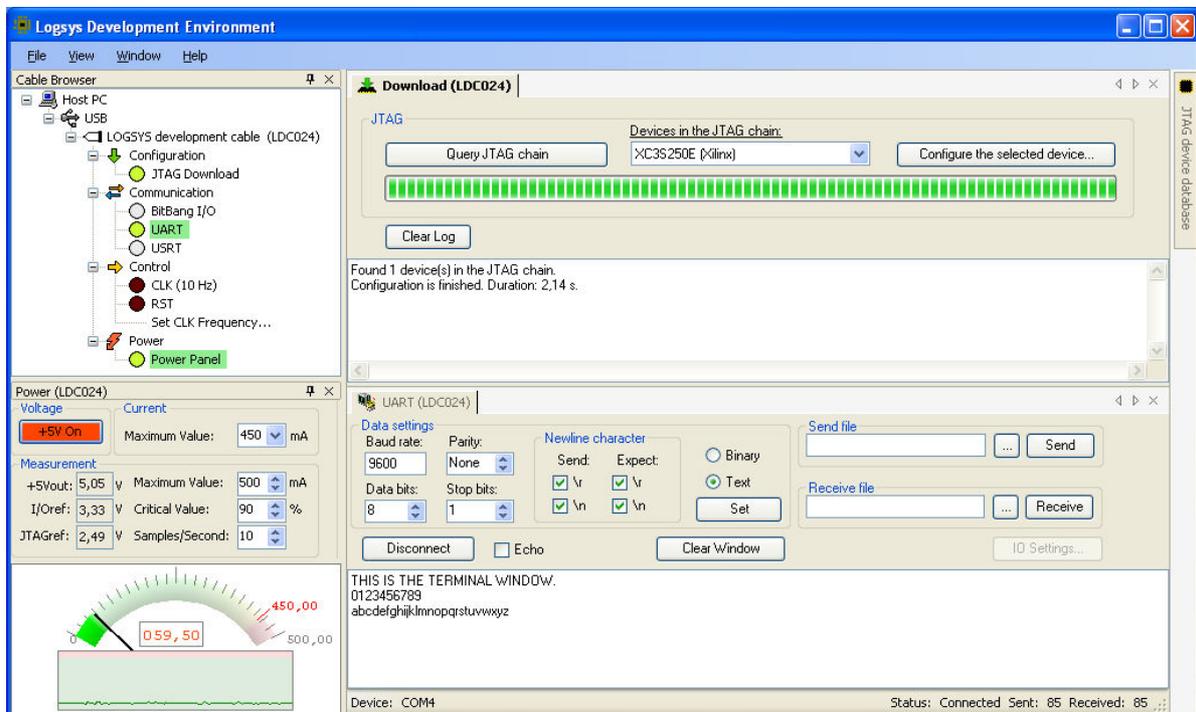


Fig. 9. The main window of the user application.

### A. Cable Browser

The cable browser (Fig. 10) can be found in the upper left part of the main window. It displays the connected development cables and the available functions on each cable. Functions that are currently available to the user are marked with dark green or dark red circle. Functions that are currently used are marked with light green or light red circle. The light grey circle means that the function is currently not available to the user. The user can enable a function by double-clicking on its name. Functions that require the same resources cannot be enabled at the same time.

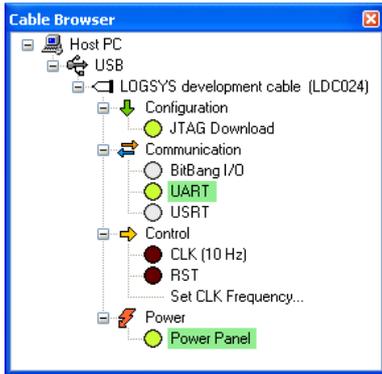


Fig. 10. The cable browser window.

### B. Power Panel

The power panel (Fig. 11) can be found in the lower left part of the main window, if it is enabled in the cable browser. The user can control the 5 V power output and the current limit from the power panel. This interface also serves for displaying the measurement results and the history of the current consumption.

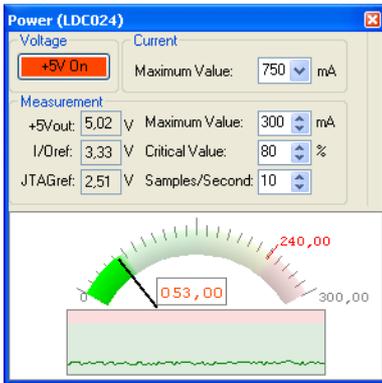


Fig. 11. The power panel.

### C. JTAG Configuration

Fig. 12 shows the JTAG download window. The LOGSYS system uses the industry standard SVF file format to describe the operations on the JTAG chain. Most manufacturer development environments provide a way to create an SVF file that contains the configuration data. In case of Xilinx devices, the BIT and the JEDEC files are

also supported by invoking the manufacturer's iMPACT utility.

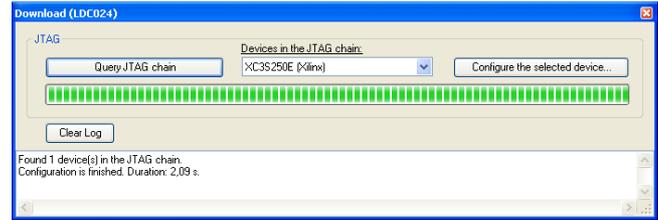


Fig. 12. The JTAG download window.

The LOGSYS system handles JTAG chains with devices from different manufacturers without any difficulties. Let's suppose that the user has a board, which contains two programmable devices from different manufacturers. The first device is an Atmel AVR microcontroller, the second device is a Xilinx CPLD. To configure these devices, the Atmel AVR Studio 4 and the Xilinx iMPACT softwares are needed.

The Atmel software offers a very simple way to handle the composite JTAG chains: the necessary data have to be entered manually (Fig. 14, left). These data are the number of devices before and after the Atmel device, and the number of instruction bits before and after the Atmel device.

The Xilinx software offers a more intelligent way to handle the composite JTAG chains: the necessary data are imported from BSDL (Boundary Scan Description Language) files (Fig. 14, right). These files describe the JTAG devices and they are provided by the manufacturers.

The above ways of handling composite JTAG chains are not convenient at all because those steps have to be repeated every time when the user configures a new target system. In the LOGSYS system, the configuration tool has an internal device database (Fig. 13) to manage the devices from different manufacturers in the JTAG chain. The required data are stored in the database. These data can be entered manually or can be imported from BSDL files.

Thanks to the JTAG device database, composite JTAG chains are handled easily. At the beginning of the configuration process, the devices in the JTAG chain have to be queried first. Then the user can download the configuration file to the selected device.

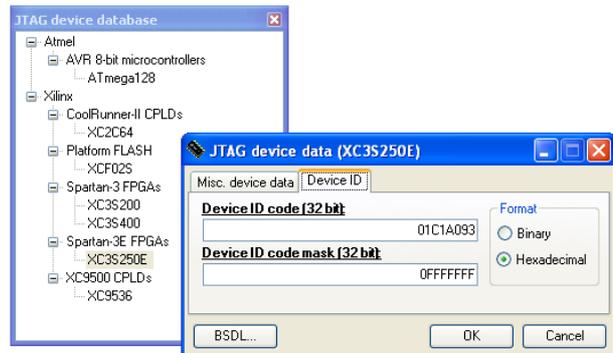


Fig. 13. JTAG device database.

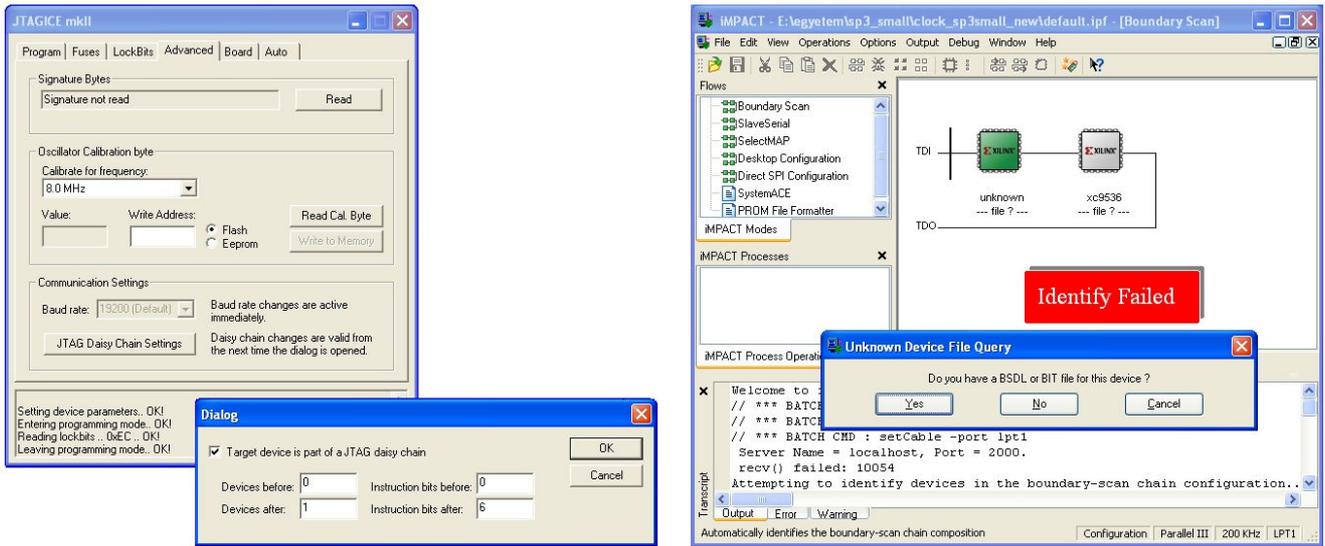


Fig. 14. Handling composite JTAG chains: Atmel AVR Studio 4 (left), Xilinx iMPACT (right).

#### D. BitBang I/O

The BitBang I/O is a special low-speed synchronous communication mode for simple tests and educational purposes. In this mode the software directly controls the clock (CLK), changes the reset (RST) and serial data out (MOSI) lines, and samples the serial data input (MISO) at the rising or falling clock edge. The user interface for the BitBang I/O mode can be seen in Fig. 15.

The clock and reset controls are in the upper part of the window. The user can adjust the clock frequency between 1 Hz and 1000 Hz. The clock can be in free run mode or the user can send a given number of clock pulses. When the clock is stopped the commands are queued, and they going to be executed after the clock is restarted. The reset signal can be set to low or high, also a reset pulse with a given length can be send.

The data I/O controls are in the middle part of the window. Different number systems and file I/O operations are supported. In USRT mode, the data are sent in frames. Every frame consists of a start bit, the data bits (4-16) and a stop bit.

The timing diagram that displays the communication flow can be found in the lower part of the window.

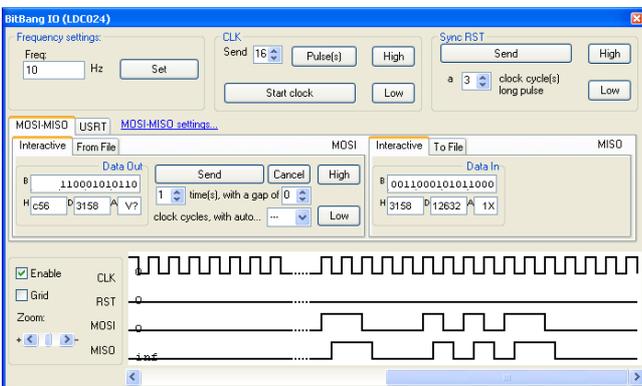


Fig. 15. The BitBang I/O window.

#### E. UART/USRT Terminal

Fig. 16 shows the simple terminal interface available for UART and USRT communication. In the upper left part of the window, the user can adjust various parameters such as the baud rate, the number of data and stop bits, the parity and the newline characters. The terminal interface supports binary and text mode communication, as well as file I/O operations.

In USRT mode, the development cable drives the clock output and the clock frequency is equal with the baud rate.



Fig. 16. The UART/USRT terminal window.

### V. EXTERNAL APPLICATIONS

Currently, there are some functions that are accessible only from external applications. These functions will be integrated into the future releases of the LOGSYS user interface.

#### A. Cable Server

In the Xilinx ISE development environment, the iMPACT application can be used to configure the programmable devices. This application is able to handle download cables connected to remote computers. This feature requires the cable server application, which runs on the remote machine. The iMPACT uses TCP/IP protocol to communicate with the cable server. Thanks to the own cable server application that emulates the standard Parallel-III download cable, the

JTAG interface of LOGSYS development cable can be accessed from the iMPACT utility. In the right side of Fig. 14, the iMPACT uses the LOGSYS development cable through the cable server.

### B. PIC Programmer

The configuration interface of the LOGSYS development cable also supports the low voltage programming of the Microchip PIC18 microcontrollers. With the PIC programmer application, the user can configure the older PIC18Fxxx devices as well as the newer PIC18FxxJxx devices.

### C. Applications for I<sup>2</sup>C and SPI Communication

There are numerous devices equipped with I<sup>2</sup>C or SPI serial communication interfaces. Such devices are microcontrollers, serial Flash and EEPROM memories, I/O expanders (GPIO ports), A/D and D/A converters, real time clocks, programmable clock synthesizers, etc. Because of the many different functions, the user interface for I<sup>2</sup>C and SPI communication is device dependent.

Two applications for general I<sup>2</sup>C (Fig. 17) and SPI communication, an application for programming I<sup>2</sup>C EEPROM memories and an application for programming SPI Flash memories are currently available. The general applications support all SPI and I<sup>2</sup>C devices, but often a specialized application is more useful.

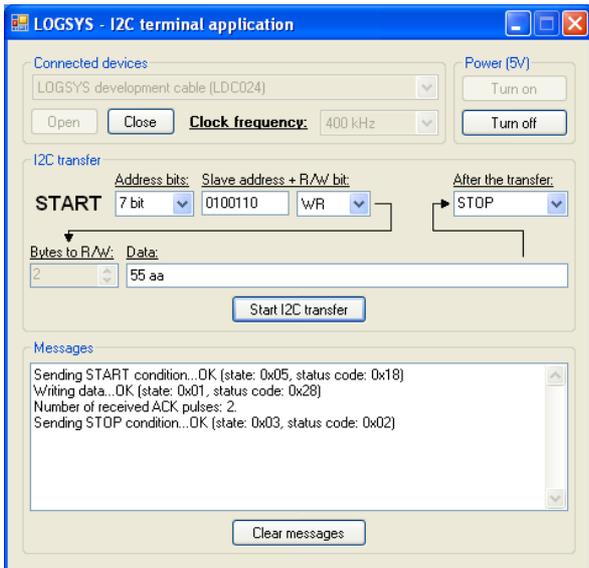


Fig. 17. Application for the I<sup>2</sup>C communication.

Programming the serial Flash memory on the LOGSYS Spartan-3E board requires a special method because there is no direct connection between the memory and the development port. First, a design has to be downloaded to the FPGA, which connects the SPI interface of the Flash memory with the development port. After that, the SPI communication mode of the cable can be used to program the Flash memory. An application is available that performs these tasks.

## VI. EXAMPLE DESIGNS

### A. State Machine

At Digital Design I. class held by our department, students learn about the basics of the digital systems. Designing simple state machines is a common exercise for the students. Fig. 18 shows the state diagram of a Mealy machine, which recognizes the 010 and 1001 bit patterns.

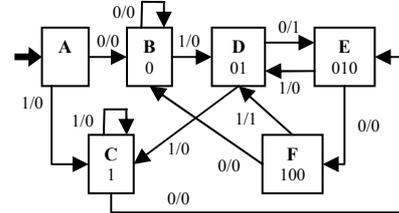


Fig. 18. State diagram of the pattern recognizer.

The LOGSYS system is useful to implement and test these simple designs. Students can use the BitBang I/O communication mode to shift out the input data bit by bit and the timing diagram displays the result. Fig. 19 shows the test result of the above state machine.

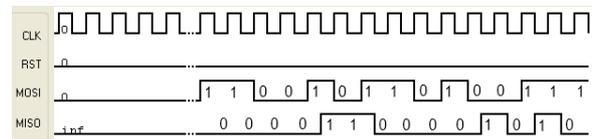


Fig. 19. Using the BitBang I/O to test the state machine.

### B. XSOC

The XSOC [6] is a simple SoPC design. The original version of the XSOC system consists of the XR16 CPU, 32 Kbyte external SRAM, a simple parallel I/O interface and a monochrome VGA controller. The advantage of this design is its small size. The whole system was implemented in a Xilinx XC4005XL FPGA (only 196 CLBs!).

The XR16 is a 16-bit RISC soft processor core with sixteen 16-bit general purpose registers. It is well suitable for educational purposes because its Verilog source code is freely available and it is well documented. As for the software development support, an assembler, the LCC C compiler and a simple simulator are available.

The XR16 CPU has been ported to the LOGSYS Spartan-3E FPGA board and it has been modified to support multiply and barrel-shift operations [7]. Because of other required modifications, the old peripherals became incompatible with the new system. Therefore new peripherals have been created, including UART, external memory controller, GPIO, timer/counter, 7-segment display controller and interrupt controller. A graphical environment is available (Fig. 20) to help the software development. By using this application, the user can:

- edit and compile the C source code
- download the executable code to the target system
- execute the program step by step
- insert breakpoints into the program
- modify the contents of the memory and the registers
- control the on-chip peripherals

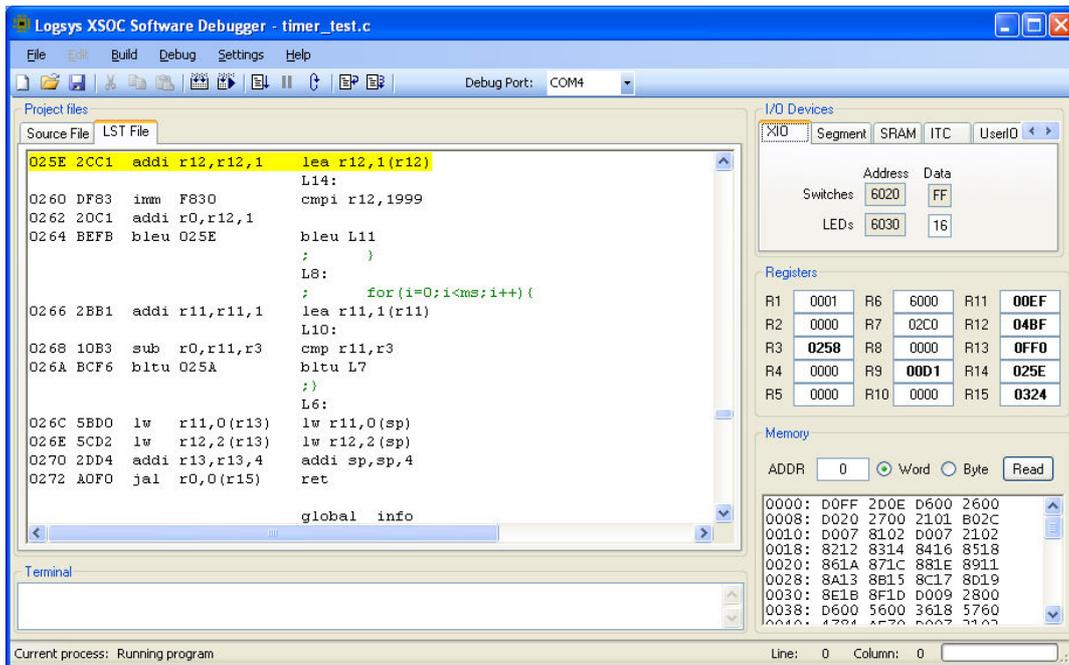


Fig. 20. The XSOC software debugger application.

### C. Debug System for FPGAs

The purpose of this design is to support the high-level FPGA development with a component that provides a test environment for the user logic. Fig. 21 shows the block diagram of the debug system. A Xilinx PicoBlaze processor controls the whole system and it communicates with the PC using the UART interface. The basic idea is that the debug system provides the input data for the user logic and after it has finished the operation, the user can read back the output data. After the input data has been arrived, the PicoBlaze notifies the user logic to start the operation. The user logic has to notify the PicoBlaze when it has finished the data processing. The input data can be stored in a 2 Kbyte block-RAM (MEM), in a 16-byte FIFO or in four 8-bit registers (REG). The same resources are also available at the output side.

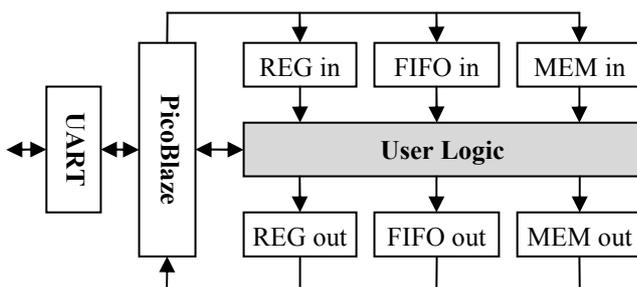


Fig. 21. The block diagram of the debug system.

A software library has been created for MATLAB, which enables the communication with the debug system. Using this feature, users can test simple signal or image processing related hardware. A part of the algorithm can be implemented in external hardware and the MATLAB can visualize the results in a user-friendly way.

### VII. SUMMARY

This paper introduced the LOGSYS system, which is a general purpose configuration and development tool for programmable devices. Manufacturers provide the development environment and development tools for their own devices, but most of them cannot be efficiently used when the target system contains devices from different manufacturers. Other problem is the limited communication support with the application itself. Therefore the LOGSYS system has been created, which integrates the configuration, the communication and the power supply features in a vendor independent manner.

The USB based development cable connects the target system with the PC. The functions offered by the cable can be accessed from a user-friendly graphical interface, which co-operates with the manufacturer development environments by supporting the standard file formats.

The LOGSYS environment is still in development. New functions and capabilities will be added to the LOGSYS environment in the future.

### REFERENCES

- [1.] R. G. Bennets, *Boundary Scan Tutorial*, ASSET InterTech, Inc. [http://www.asset-intertech.com/pdfs/boundaryscan\\_tutorial.pdf](http://www.asset-intertech.com/pdfs/boundaryscan_tutorial.pdf)
- [2.] *Serial Vector Format Specification*, ASSET InterTech, Inc. <http://www.asset-intertech.com/support/svf.pdf>
- [3.] *Universal Serial Bus Specification Revision 2.0* <http://www.usb.org>
- [4.] *DS321: Spartan-3E FPGA Family Data Sheet*, Xilinx, Inc. <http://www.xilinx.com>
- [5.] *UG130: Spartan-3 Starter Kit Board User Guide*, Xilinx, Inc. <http://www.xilinx.com>
- [6.] <http://www.fpgacpu.org>
- [7.] P. Czako, *Mikrorendszer megvalósítása FPGA környezetben (Realization of a Microsystem in FPGA Environment)*, Diploma Thesis, BUTE-DMIS, 2007