Bulletin of the *Transilvania* University of Braşov • Vol. 3 (52) - 2010
Series I: Engineering Sciences

HIGH SPEED INTERFACE TESTCHIP FOR STRUCTURED ASIC TECHNOLOGY

T. TULBURE[1] **R. JIPA**[1]

Abstract: *Nowadays high speed interfaces are commonly used in digital electronics, communications and computers. Standards like PCI Express, SATA, GigE (802.3-2002), XAUI (802.3ae), 10G Base-KX4 (802.3ap), 10G Base-CX4 (802.3ak), HDMI, DVI, DisplayPort require high speed connection link. In this paper we present implementation and test of a 30 bit host-LCD panel interface that can work at an effective rate of up to 945Mbs per channel using Structured ASIC Technology. Structured ASIC technology has the advantage of built in Serializer/Deserializer and low-cost prototype that allows building a testchip before the real product implementation.*

Key words: *High speed serial interface, Structured ASIC, LVDS.*

1. Introduction

High-speed Low Voltage Differential Signalling (LVDS) interfaces become commonly used in digital electronics, communications and computers. Low Voltage Differential Signaling is a technology addressing the needs of today's high performance data transmission applications. LVDS solutions provide designers with a new alternative to solving high speed I/O interface problems, delivering hundreds of megabits per second with power consumptions of few milliwatts for today's and tomorrow's bandwidth hungry data transmission applications.

However testing a high-speed LVDS interface requires very expensive equipment that is not available to all companies. For most of them building a testchip that demonstrate the capability of high-speed LVDS connection link is a preferred solution.

This paper presents such a testchip for validation of a high-speed LVDS connection between a Host and a Flat Panel Display that can work up to 945 Mbs per LVDS channel to support pixel data transmission from NTSC up to SXGA+ resolutions.

In comparison with other similar high speed test interfaces [5] the proposed idea has the advantage of using the same die-package combination that will be used in real product matching exactly the high-speed interface behaviour and package parasitic.

In our case the selected transmitter chip is a five channel 135 MHz 30 bits color transmitter part number is THC63LVD103D described in [9]. Next sections describe the functionality of the testchip that will interact with a LVDS transmitter. The testchip generates pseudo random data that is sent to the serializer transmitter chip. The LVDS transmitter can receive the data from the testchip or from another video stream and

[1] Dept. of Electronics and Computers, *Transilvania* University of Braşov.

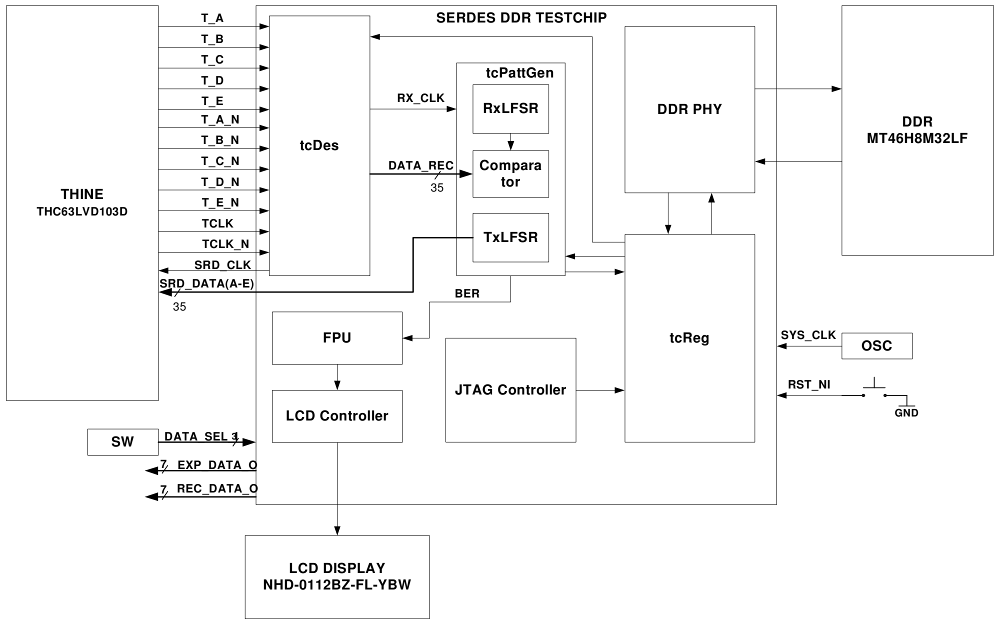

Fig. 1. *LVDS testchip - internal architecture and external connections*

sends the data at 945 MHz frequency back to the LVDS testchip that compares the received data with the expected sequence.

The LVDS Testchip has the following interfaces:

- LVTTL parallel data out to THC63LVD103D at 135 MHz;
- LVDS serial input from THC63LVD103D at 945 MHz;
- mobile DDR interface at 200 MHz;
- JTAG debug interface;
- LCD display interface to NHD-0112BZ-FL-YBW;
- parallel debug port.

The LVDS Testchip has the LCD display interface to allow display of the bit error rate (BER) and can be controlled by a application running on a computer using a JTAG debug interface connection. Other high speed interface tested by the presented circuit is a mobile DDR interface working at 200 MHz, but the current paper focuses mainly on the high-speed LVDS interface.

In section II we will present the architecture of the LVDS testchip. In section III we will describe the test environment. Section IV will depict the implementation. Finally conclusions and further study are presented in Section V.

2. Testchip Architecture

The testchip generates a pseudo random data sequence using a linear feedback shift register (LFSR), that is sent parallel to the transmitter chip which sends the data back serially and the comparison starts. The comparison is using the same LFSR structure. Receive LFSR starts generating data after a latency cycle that is either specified in one of the configuration registers or is determined by sending a training pattern and computing the delay it takes for the pattern data to be received back from the transmitter chip, pseudo-random data following the training pattern.

The deserializer phase alignment interface is driven by configuration registers to solve the phase alignment issue between clock and data.

All the configuration registers can be accessed through a JTAG controller.

The whole configuration structure can be bypassed in order to create a simple loop between the serializer and the testchip.

The top level internal architecture is presented in Figure 1 with the following sub-modules:

- tcDes - serdes block de-serializes high speed data stream;
- tcPattGen - pseudo random pattern generator and comparator;
- tcReg - contains configuration registers and pattern memories;
- JTAG controller - IEEE 1149.1 compliant JTAG controller with OCP (Open Core Protocol) interface;
- FPU - floating point unit for computing bit error rate (BER);
- LCD controller - controller for LCD display;
- DDR physical interface for mobile DDR.

2.1. tcDes - Serdes Block

The transmitter does serialization with factor of 7x and sends high-speed data stream and low speed clock. Figure 2 presents the relationship between clock and data, note that there is a 2 high speed cycle shift between low speed clock and data stream.

The deserializer requires a high-speed clock and a low-speed clock that drive the load-enable-control block, a core clock on which the parallel data is transferred to the core, and a feedback clock to remove the high-speed clock tree insertion delay. The deserializer PLL reference clock originates from the driving serializer, which is received on a dedicated pad.

For both the serializer and deserializer the high-speed clock must be 180° phase shifted (inverted). This can be easily achieved by setting the correct PLL parameter. All other clocks should be non-inverting. The low-speed clock must be the same frequency as the core clock. The high-

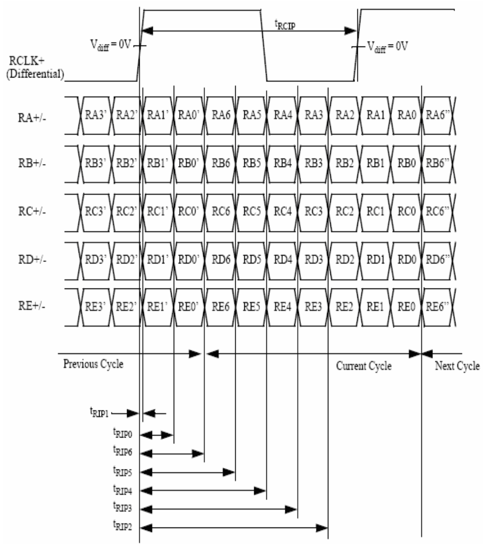

Fig. 2. *LVDS output data position*

speed clock frequency is determined by the data conversion ratio (which is the same as the parallel data width) times the low-speed clock frequency. The following data conversion ratios are supported: 2x, 3x, 4x, 7x, 8x and 10x.

The deserializer takes a serial data stream from a dedicated, high speed differential input buffer and converts it into a parallel data stream for the core.

To adjust for any skew on the data signals, either generated by the PCB or by delays on the die, each channel has a programmable delay, which can be individually adjusted during runtime.

Figure 3 presents the deserializer block for the selected Structured ASIC technology [8].

Load-Enable Control Block (intloaden) generates an internal load enable signal from the high-speed clock and the low-speed clock, which are generated by the PLL.

The load enable signal is driven onto the high-speed clock tree to the serial-to-parallel and parallel-to-serial converter blocks.

The serial-to-parallel converter (int_rx) stores the high-speed data stream from the

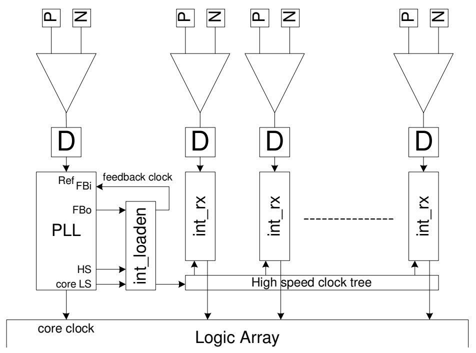

Fig. 3. *Programmable LVDS receiver*

differential input pad, through a programmable delay, into the shift register. On the rising edge of load enable the data is latched into the holding register from where it can be retrieved by the core.

2.2. tcPattGen - Pseudo Random Pattern Generator and Comparator

This module generates parallel streams of data going to the transmitter device. When the enable signal coming from the eDes is asserted the comparison starts. The LFSRs are first initialized with a certain value. The generation of the data continues until the number of patterns is equal with the number programmed coming from the configuration registers bank written through JTAG. Based on stop at error parameter the chip will stop at a specific error number. The number of errors is written in the configuration registers via for software access through JTAG.

The testchip will send a stream of '0' to the transmitter device until the Serdes PLL from the testchip is getting locked. At that moment the tcPattGen will send a training pattern to figure out the latency of the data. A maximum latency register in the tcReg contain the maximum latency expected for the data. If the latency could not be computed then a error bit is output and stored in the status registers. The latency computed can be written in a register for debug purpose.

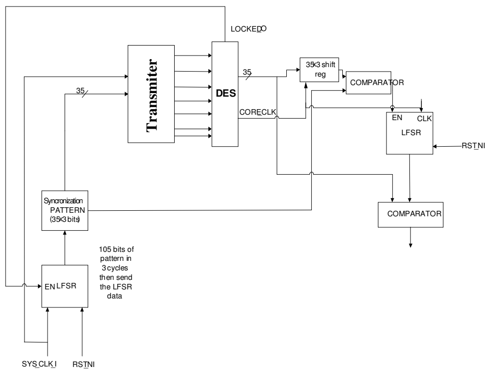

Fig. 4. *Comparison process inside tcPattGen*

Two memory blocks will be written after the training pattern was established every time there is an error. One memory contains the correct data and the other one the wrong data. This way, at the end, both memories can be read back and check the differences. The memories can be read through JTAG. Only when an error appears the memories are written, write enable for the memory is the error signal.

One port will be used for writing data in the memory while the other one for reading the data, on the JTAG_CLK. We can take advantage on the true dual port memory configuration using both ports for independent configuration.

The memory address and data out are connected to the tcReg in order to allow debug via JTAG. The LFSR starts generating data when the latency is computed. It waits for the number of cycles computed in latency register and then it starts comparing the data. The comparison process is presented in Figure 4.

All the data is output also to the primary ports using a custom parallel interface in order to debug them with a logic analyzer.

2.3. JTAG Controller

The JTAG controller implements an IEEE-1149 compliant JTAG tap controller with full boundary scan capability as in [2]. In addition, the JTAG controller maps JTAG user commands into OCP transfers to access the internal devices of an integrated circuit.

This is a reusable IP core that has two standard interfaces: JTAG interface for testing and standardized access purpose and OCP interface that facilitates "plug and play" SoC design as detailed in [6].

JTAG controller has three interfaces:

- JTAG interface - standard JTAG port;
- boundary scan interface - not used in this testchip;
- OCP interface - provide byte wide access to a 64 KB address space implemented in tcReg.

The JTAG solution was preferred because the IP core already has been tested in few implementations on both ASIC and structured ASIC. Also the availability of software libraries for connection of JTAG interface to parallel port or USB port of a host computer make JTAG the preferred solution for the debug interface.

2.4. Floating Point Unit

The FPU is used to compute two bit error rate values. Bit error ratio (BER) is the number of received bits that have been altered due to noise, interference and distortion, divided by the total number of transferred bits during a studied time interval [1], [3].

The total BER is obtained by dividing the total number of wrong bits by the total number of received bits. The other BER is computed relative to a specified period (written through JTAG interface to a configuration register from tcReg). This bit error rate is computed by dividing the number of wrong bits received by the total number of bits received during that specific interval.

The FPU inputs and outputs are represented as floating point number - 64 bit precision [4]. Hence, the dividers and dividends had to be "normalized" (converted from their initial form - fixed point arithmetic, no fractional part, to 64 bit floating point arithmetic, having a sign, mantissa and exponent).

The quotients, however, had to go through the opposite process, "de-normalization" (the values were converted from the 64 bit floating point representation to a fixed point representation, base-10, having a mantissa and a decimal exponent). This was necessary in order to obtain a base - 10 representation that could be further on displayed on the LCD.

The FPU module is a double precision floating point reusable IP core from opencores.org. It features double precision operation for addition, subtraction, multiplication and division.

The IEEE 754 standard defines how double precision floating point number are represented. 64 bits are used to represent a double precision floating point number. The sign bit occupies bit 63. '1' signifies a negative number, and '0' is a positive number. The exponent field is 11 bits long, occupying bits 62-52. The value in this 11-bit field is offset by 1023, so the actual exponent used to calculate the value of the number is 2^(e-1023). The mantissa is 52 bits long and occupies bits 51-0. There is a leading '1' that is not included in the mantissa, but it is part of the value of the number for all double precision floating point numbers with a value in the exponent field greater than 0. A 0 in the exponent field

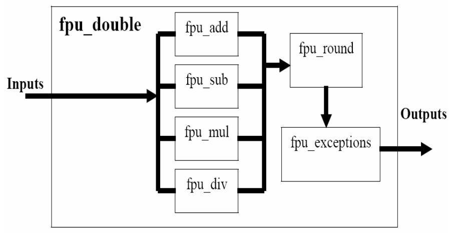

Fig. 5. *Structure of floating point unit*

corresponds to a denormalized number. The actual value of the double precision floating point number is the following:

$$Value = -1\text{^}(sign\ bit) \times 2\text{^}(exponent - \\ - 1023) \times 1.(mantissa),$$

(1.*mantissa*) being a base 2 representation of a number between 1 and 2, with 1 followed by a decimal point and the 52 bits of the mantissa.

The top level, fpu_double, starts a counter one clock cycle after enable goes high. The counter counts up to the number of clock cycles required for the specific operation that is being performed. For addition, it counts to 20, for subtraction 21, for multiplication 24, and for division 71. Once the counter reaches the specified final count, the ready signal goes high, and the output will be valid for the operation being performed. fpu_double contains the instantiations of the other 6 modules, which are 6 separate source files of the 4 operations (add, subtract, multiply, divide) and the rounding module and exceptions module.

2.5. LCD Controller

The LCD controller implements an interface with the NHD-0112BZ-FL-YBW LCD Display. The LCD is used to output the computed BERs. On the first line it displays the total BER and on the second one the BER computed every period. They are represented as floating point numbers, having a mantissa and an exponent.

The controller for the LCD is comprised of a finite state machine, an "lcd write" module and a bcd-to-ASCII conversion module. The FSM is the key part of the controller, generating the control signals that are used in the "lcd_write" module for interfacing the LCD display.

The two BER values that are to be displayed inputs to the controller are represented as 40 bits integer numbers, having an 8 bit exponent. In order to output these values, they were converted from their binary representation to BCD (binary-coded decimal). As the LCD display needs to receive as inputs ASCII codes of the digits/symbols to be displayed, the module "bcd_to_ascii" was implemented. This module contains the logic for the translation from BCD to ASCII.

As the two BER values are fractional numbers, and to be more specific, quite small, the exponential representation was needed in order to maintain the highest precision possible:

E.g.:

The value displayed on the LCD:

1.0253 e – 07.

The value written as a decimal number:

$$1.0253\text{ e} - 07 = 1.0253 \times 10^{-7} = \\ = 0.00000010253.$$

2.6. DDR Physical Interface

The testchip provides a 32 bit mobile DDR interface testing using DFI (DDR PHY Interface) compliant DDR2 physical interface and simplified DDR2 controller.

The DDR2 physical interface is designed to interface a DDR2 memory controller with a DDR2 memory unit and is intended for structured ASIC family only. It consists of DDR2 specific macros which allow translation of information between the controller which works on one edge of a clock signal and the DDR2 memory unit which works on both the edges of the clock thus allowing the controller to perform read and write operations to the memory.

The DDR2 controller part is implemented as a very basic solution where transmit part is under control of data from a 2Xx128 block memory while the received data is stored into a 2kx64 block memory. Both memories can be accessed from debug interface from host computer. Programmable addresses counters are used to control transmit and receive sequences. The implementation provides the possibility to write the test sequence from host computer through JTAG or it can be directly programmed in the initial content of those memories through a bitstream stored into external serial memory. The data received from the external mobile DDR memory is stored into internal block memory. The host computer can read this data and compare it against the expected data for the test sequence.

3. Test Environment

Test environment include a suite of tests that are run using ModelSim simulator. Figure 6 depicts the test environment.

Test environment consists of:

- JTAG master (jtagTasks) which controls and monitors all internal registers and memory. It is configurable through external input file that contains the JTAG sequence in human readable format.

- Transmitter model that receives the parallel stream of data and provides back the clock and high speed serialized data for five channels.

- Error generators that provide the capability to inject errors in high speed stream of data. Each channel has its own error generator with programmable threshold for random error rate.

- Mobile DDR model for the selected DDR part provided by the memory producer.

A test regression composed from multiple tests is automated with a makefile. Test regression contains:

- basic read/write JTAG access to all registers;

- dynamic PLL reconfiguration test;

- basic DDR write/read/check;

- automated detection of transmitter latency;

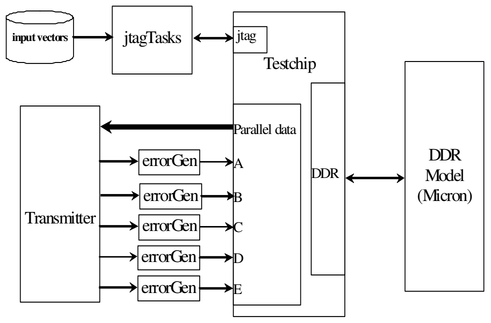

Fig. 6. *Test environment structure*

- phase alignment test;

- error free high speed interface (serdes) test;

- tests with errors injected for each channel and for all channels.

The LCD interface is difficult to test using simulation because of long simulation and non existent simulation model for LCD part. The solution was to implement the LCD controller on FPGA using a Spartan-3E Starter Kit with similar LCD screen. On the two line LCD display we have the bit error rate per period and on second line the total number of errors since latest reset. This LCD interface is helpful for long-term analysis.

4. Implementation

The Testchip was implemented on structured ASIC using eASIC NX1500 device with BG480 package.

The chosen synthesis strategy was the top-bottom one using the Magma® Design Automation environment for structured ASIC 0.09 µm Fujitsu process.

Table 1 presents area results for the testchip.

Testchip area results Table 1

Area results		
Logic	**Memory**	**IO**
15233 eCells = 152330 equivalent logic gates	16 bRAMs = 512 kbits	146 eIO including 98 DDR registers

The final placed and routed netlist was analyzed with a static time analysis tool - Synopsys PrimeTime. The final analysis included parasitic information about the design, silicon die and package. Several iteration from static timing analysis to routing were requires to solve transition and hold violations. Table 2 describes clock frequencies; timing was met for all clocks.

Testchip timing results Table 2

Clock periods [ns]			
CORE_CLK	RX_HS_CLK	DDR_CLK	JTAG_CLK
9.43	1.35	5	100

The implementation took about two weeks with the most difficult part being timing closure for high speed interface. Based on static timing analysis we determine the value for delay elements that ensure clock edge to center of the data eye. Based on this value determined from static timing analysis we coded the rest value for the registers that control the phase alignment for high speed input interface but the values can be later changed by host computer based on experiments.

5. Conclusion

This paper presented the methodology for design and implementation of a testchip for a host-LCD panel high speed interface. The idea can be successfully applied to test other high speed interfaces as well. The simulation proven the correct behavior and implementation provided quantitative performance measurement.

Further studies can be made to design and implement automatic dynamic phase alignment based on training patterns and to improve the reporting of long term error statistics based on feedback from actual silicon testing.

References

1. Hong, D., Ong, C.-K., Cheng, K.-T: *BER Estimation for Serial Links Based on Jitter Spectrum and Clock Recovery Characteristics.* In: Proceedings 2004 International Test Conference (ITC 2004), Charlotte, NC, USA, October 26-28, 2004, p. 1138-1147.
2. Kenneth, P.P.: *The Boundary - Scan Handbook.* 2nd Edition. Kluwer Academic Publishers, 2000.
3. Li, M., Wilstrup, J.: *On the Accuracy of Jitter Separation from the Bit Error Rate Function.* In: Proc. of International Test Conference, October 7-10, 2002, p. 710-716.
4. Lundgren, D.: *Double Precision Floating Point Core Verilog.* Available at: http://www.scribd.com/doc/11091346/Double-Precision-Floating-Point-Arithmetic. Accessed: 22-11-2009.
5. Suzuki, M., Shimizu, R., Naka, N., Nakamura, K.: *High-Speed Interface Testing.* In: Proceedings of the 10th Asian Test Symposium (ATS'01), Kyoto, Japan, November 19-21, 2001, p. 461.
6. Tulbure, T.: *OCP Compliant JTAG Controller - IP Core.* In: Proceedings of the 9th International Conference on Optimization of Electrical and Electronic Equipments **IV**, Braşov, May 20-21, 2004, p. 63-67.
7. *** OCP international partnership. Available at: www.ocpip.org. Accessed: 20-11-2009.
8. *** www.easic.com. Accessed: 20-11-2009.
9. *** www.thine.co.jp. Accessed: 20-11-2009.