Implementation of FPGA based LOW-COST Logic Signal Analyzer (LSA)

Article · November 2010

CITATIONS
2

READS
2,284

2 authors:

Dr S Nagakishore Bhavanam
University College of Engineering & Technology, Acharya Nagarjuna University, Guntur
83 PUBLICATIONS **256** CITATIONS
SEE PROFILE

Vasujadevi Midasala
K L University
38 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Implementation of FPGA based LOW-COST Logic Signal Analyzer (LSA)

S Nagakishore Bhavanam, Vasujadevi Midasala, Madana Gopal.M
Aurora's Technological & Research Institute, Hyderabad, India
Email: satyabhavanam@gmail.com, mekala.madan@gmail.com
Spurthi Engineering College, Hyderabad, India
Email: vasujadevi@gmail.com

Abstract **-- Logic signal analyzers are very essential instrument for digital circuit or board debugging. The existing market solutions offer several features, but the cost of such instruments is very high and most of the time we don't need that much capable instruments. In this paper a low cost logic signal analyzer is implemented around the Spartan-3E FPGA. The FPGA being capable of offering high frequency data paths in them become suitable for realizing high frequency signal capturing logic. The paper work includes development of FPGA based logic signal analyzer using VHDL. The logic signal analyzer will be capable of implementing match conditions, counter based triggering, external clocking and internal clocking features. The blocks such as registers, counters, comparators, state machines will be used in realizing these blocks. The captured data will be stored in memory before transferring the data to PC. The UART core will be developed which, will be used for transferring the data to PC.**

Modelsim Xilnx edition (MXE) tools will be used for simulation. Xilinx FPGA synthesis tools will be used for synthesizing the design for Spartan FPGAs. The developed application will be tested on Spartan 3E development board. By using HyperTerminal we check our design. By using GUI ,we will show the results with respect to bit waveforms.

Index Terms**: Logic Analyzer, UART, Spartan-3E FPGA Board, Modelsim Xilix Edition, Xilinx ISE, Java Based GUI.**

I. INTRODUCTION

A logic analyzer is an electronic instrument which displays signals of a digital circuit which are too fast to be observed and presents it to a user who can then precisely observe with greater ease the operation of the digital system under test. It is typically used for capturing data in systems having too many channels to be examined with an oscilloscope. Software running on the logic analyzer can convert the captured data into timing diagrams, protocol decodes, state machine traces, assembly language, or correlate assembly with source-level software. Presently there are three distinct categories of logic analyzers available on the market:The first is mainframes, which consist of a chassis containing the display, controls, control computer, and multiple slots into which the actual data capturing hardware is installed. The second category is standalone units which integrate everything into a single package, with options installed at the factory. The third category is PC-based logic analyzers. The hardware connects to a computer through a USB or LPT connection and then relays the captured signals to the software on the computer. These instruments are less expensive than either mainframes or standalone units although they lack the sophisticated functionality. These devices are typically much smaller, because they do not need displays or hardware input such as dials.

II. LOGIC ANALYZER OPERATION

A logic analyzer may be triggered on a complicated sequence of digital events, and then capture a large amount of digital data from the system under test (SUT). The best logic analyzers behave like software debuggers by showing the flow of the computer program and decoding protocols to show messages and violations. When logic analyzers first came into use, it was common to attach several hundred "clips" to a digital system. Later, specialized connectors came into use. The evolution of logic analyzer probe has led to a common footprint that multiple vendors support, which provides added freedom to end users. Introduced in April, 2002, connectorless technology (identified by several vendor specific trade names: Compression Probing; Soft Touch; D-Max) has become popular. These probes provide a durable, reliable mechanical and electrical connection between the probe and the circuit board with less than 0.5pF to 0.7 pF loading per signal. Once the probes are connected, the user programs the analyzer with the names of each signal, and can group several signals into groups for easier manipulation. Next, a capture mode is chosen, either timing mode, where the input signals are sampled at regular intervals based on an internal or external clock source, or state mode, where one or more of the signals are defined as "clocks," and data is taken on the rising or falling edges of these clocks, optionally using other signals to qualify these clocks.After the mode is chosen, a trigger condition must be set. A trigger condition can range from simple (such as triggering on a rising or falling edge of a single signal), to the very complex (such as configuring the analyzer to decode the higher levels of the TCP/IP stack and triggering on a certain HTTP packet).At this point, the user sets the analyzer to "run" mode, either triggering once, or repeatedly triggering.Once the data is captured, it can be displayed several ways, from the simple (showing waveforms or state listings) to the complex (showing decoded Ethernet protocol traffic).

Field-programmable gate arrays have become a common measurement point for logic analyzers. Today, the most common method of data capture for logic analyzers is through a probe. A logic analyzer can measure anything electronic if it has the proper probe connected. The probes

try to tap into the electronic signals being passed through a data bus or wire

III. ARCHITECTURAL DESIGN

The block diagram of logic signal analyser is shown in Figure 3-1. The block diagram contains 2 basic modules namely capture module and communication module. Capture module is responsible for signal capture. It contains a capture state machine and samples counter. The counter is 8-bit counter. The communication module contains UART and FIFO. The FIFO (Trace Memory) is used to save the data temporarily before sending that to PC through UART.

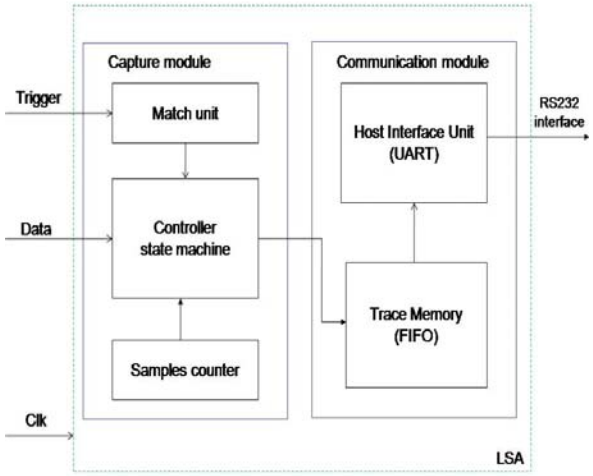

Fig. 3-1. Block diagram of Logic Signal Analyser

The state machine for logic signal analyzer is shown in Figure 3-2.

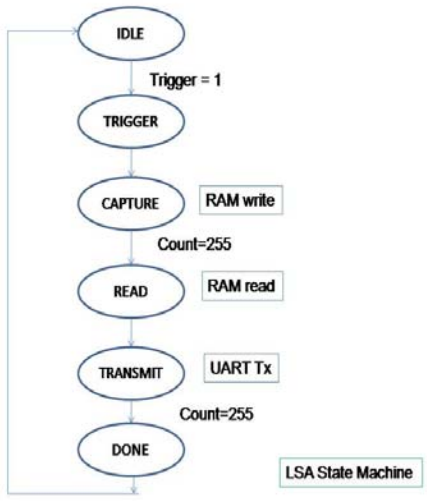

Fig. 3-2. LSA State machine

The state machine contains 6 states namely IDLE, TRIGGER, CAPTURE, READ, TRANSMIT and DONE. The state machine stays in IDLE state by default. When trigger input is '1', the state machine enters TRIGGER state. The state machine enters CAPTURE state after that. Memory write takes place in CAPTURE state only. 256 memory writes are done in this state which corresponds to 256 samples being captured. Once sampling is finished, the state machine enters READ state where RAM read takes place. The read back data is transmitted through UART in TRANSMIT state. Once 256 data transmission finish, the state machine enters DONE state which indicates end of signal capture. The state machine goes back to IDLE state.The block diagram of match unit is shown in Figure 3-3.

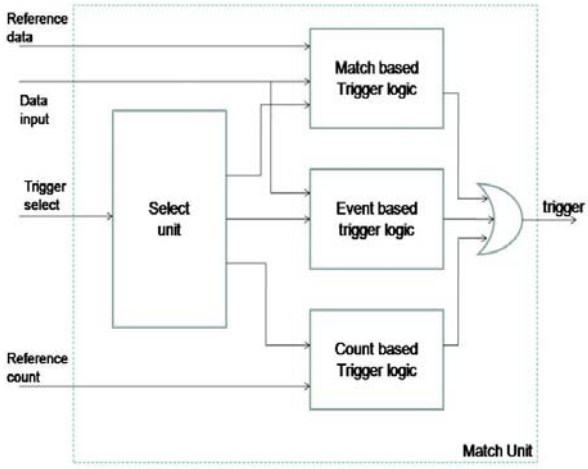

Fig. 3-3. Match unit

As shown above, the match unit contains 4 sub blocks namely select unit, match based trigger logic, event based trigger logic and count based trigger logic.

A. Select unit:

The function of select unit is to generate enable signals for other 3 blocks based on the trigger select. 3-bit trigger select is used since there are 8 conditions in total to be enabled.

B. Match based trigger logic:

This block contains logic to implement the trigger logic based on match condition. It contains 2 data inputs for comparison namely reference data and data input. Reference data is the data with which matching needs to be done and data input is the current sampled data. Various match condition supported are

Din = ref data, Din /= ref data, Din < ref data, Din > ref data, Din <= ref data, Din >= ref data

There are 6 enable signals to this unit to select one of the above match conditions.

C. Event based trigger logic:

The functionality of this block is to detect an event on the data input. It compares two successive samples todetect the event. To avoid unnecessary glitches pick up, both the current data and previous data latched (or synchronized) versions are used.

D. Count based trigger logic:

This block contains a counter internally which will be counting the number of samples. The purpose of this block is to enable trigger after a particular number of samples are finished. In this sense, counter based triggering is delayed triggering based on the input sample count. The reference count input decides the number of samples to be discarded (or delayed). The outputs of these 3 trigger logic modules are logically ORed to generate final trigger signal which would go to the state machine in LSA top.

The block diagram of host interface unit is shown in Figure3-4.

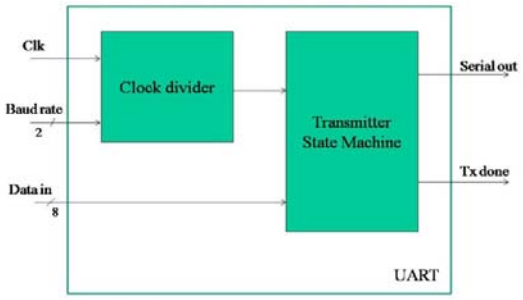

Fig 3-4. Block diagram of host interface unit

The UART contains 2 modules namely clock divider and transmitter state machine. Clock divider generates different clocks corresponding to different baud rates selected. UART transmitter state machine is shown in Figure 3-5.

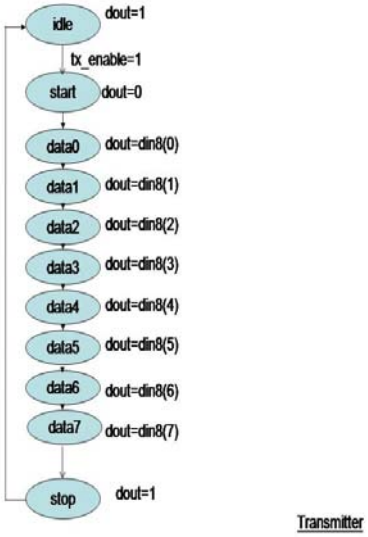

Fig 3-5. UART transmit state machine

The transmitter stays in IDLE state unless transmit enable (tx_enable) is made as '1'. The data transmission starts with tx_enable = 1. As mandated by the protocol, a '0' is transmitted to indicate start of transmission or start bit. This is done in START state. Then data bits 0 to 7 are transmitted in states DATA0 to DATA7. If parity is enabled in configuration register, the data is attached with a parity in PARITY state. Then transmitter enters STOP state and sends a '1'. This indicates the completion of transmission. Then the transmitter enters the IDLE state and waits for next data transmission. An asynchronous FIFO is used as Trace memory in LSA communication module. A FIFO is required since the output data rate of UART is much smaller than the sampling rates supported by capture module.

IV. SIMULATION RESULTS

The simulation results for capture module are shown in Figure 4-1.

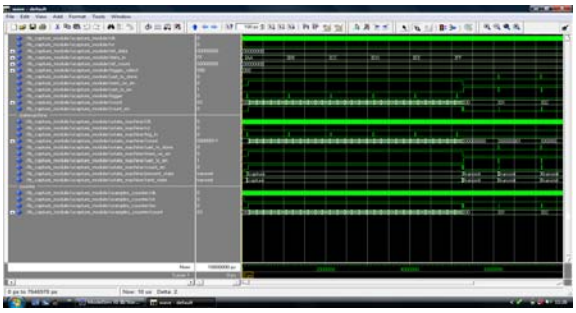

Fig. 4.1 Waveforms for capture module

In the waveform data_in refers to the data being captured by LSA. Trigger_select value selects the type of triggering to be used. Ref_data and ref_count are used for different triggering mechanisms of match unit. The signals after divider are state machine signals.

Figure 4-2 shows the simulation waveforms for even based trigger logic and count based trigger logic at the top level.

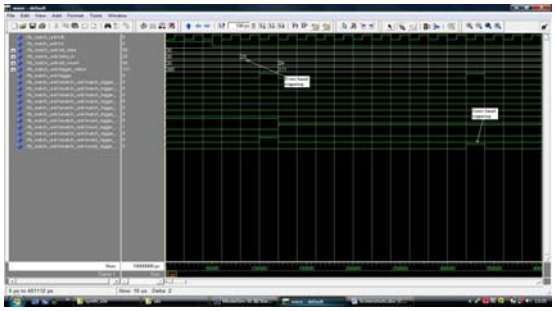

Fig. 4.2 Simulation waveforms for different triggers

In this screenshot, initially even based triggering is enabled (trigger select = 000). When the input data (data_in) changes from X"00" to X"20", an event is triggered. We can see the corresponding output on event_trigger_out first and then on trigger output.

Later, count based triggering is enabled with trigger_select = 111. Here a counter is started in count based trigger logic and when it reaches the reference count (ref_count) value of X"0A" a trigger is generated.

Figure 4-3 shows match based trigger logic waveforms from top level.

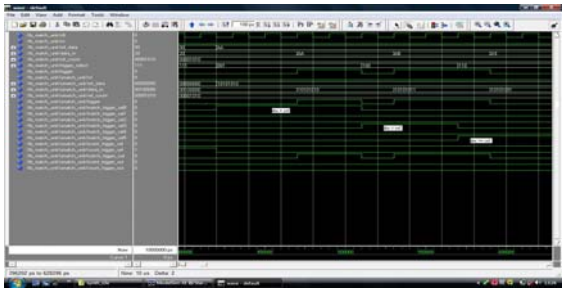

Fig 4.4.simulation results for match based trigger.

Here 3 match conditions are exercised, din=ref, din>ref and din>=ref. As we can see first trigger comes when the data_in value reaches the ref_data of X"AA" since the match_trigger_sel0 checks for equality condition. Then match_trigger_sel3 is enabled which means that din>ref condition is checked. When the data_in value changes to X"AB" which is greater than ref_data of X"AA", trigger is generated. Third condition set is din>=ref. Since it is already satisfied the trigger continues to be generated. When the data_in changes to X"A9", trigger becomes 0.

Figure 4-4 shows the simulaton results for communication module.

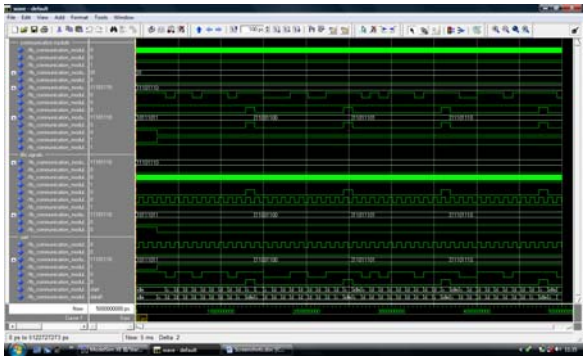

Fig.4.4 Simulation Results for communication module

Figure 4-4 shows 2 set of signals one for top level communication module waveforms and the other is FIFO signals. For the communication module, data comes in as *fifo_wdata* from capture module. This data is saved in FIFO. This data is read back from FIFO as *fifo_data_out* and is given to UART. Final UART serial output can be seen on *sout*. Coming to FIFO signals, we can see the *wclk* and *rclk*, clocks used for write and read. As mentioned earlier, the read clock is much slower than write clock. The *wdata* shows the write data and *rdata* is the readback data.

UART (host interface unit) simulation waveforms are shown in Figure 4-5.

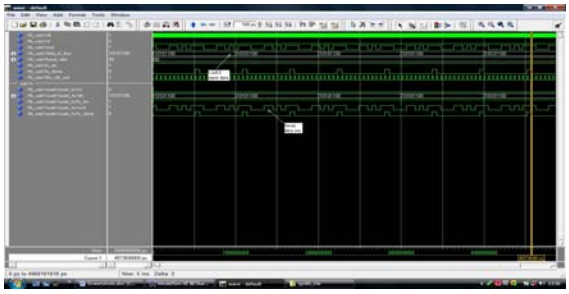

Fig. 4.5 UART simulations

Figure 4-5 shows simulation for UART, where *data_in_bus* refers to the parallel data input to be transmitted.

The baud rate is controlled by *baud_rate* input and *clk* is the system clock which is 50MHs currently. The serial output from UART state machine is given on *sout* and corresponding clock can be seen on *fifo_clk_out* which goes to FIFO.

The UART transmission happens only when tx_en signal is '1'.these are the simulation results for our design.

A. Test setup:

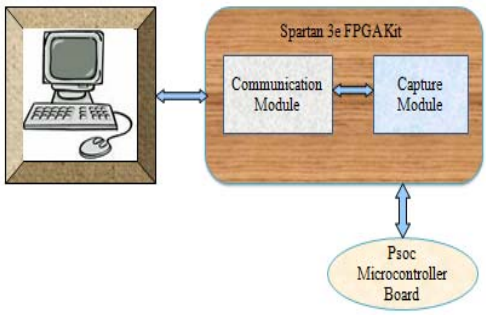

Fig. 4.6 Test Setup

The abow figure can represents the test setup for our design. Here our design (VHDL code) is dumped into Spartan 3e FPGA board after that by using requirements we connect our setup to PC (power supply for Spartan 3e, USB connector, RS-232 cable etc..).

B. HyperTerminal Results:

After test setup we are giving the input data as a hexadecimal to our design the results will be shown by using hyperterminal interms of characters(refer hex to ascii table).the results shown below.

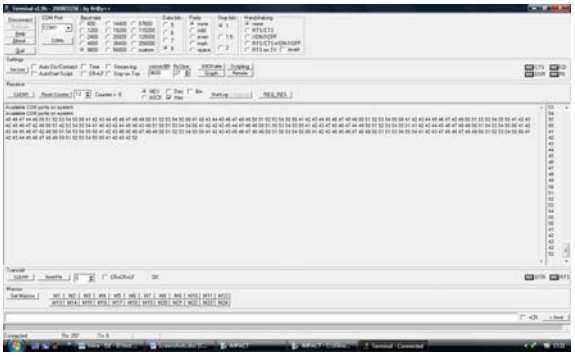

Fig.4.9 hyperTerminal results

C. GUI results:

What we are obeserved the results from the hyperterminal we will see that results with respect to bit waveform by using java based GUI setup file for LSA.The below figure can represents the GUI results for LSA.

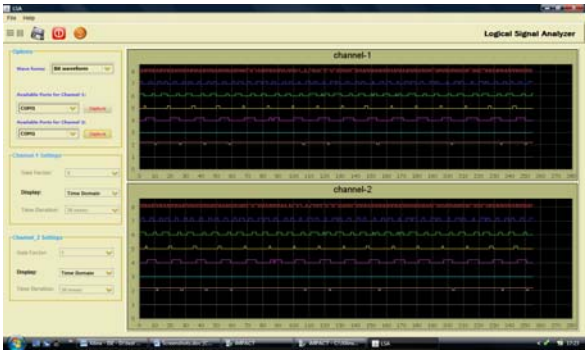

Fig. 4.10 GUI results for LSA

D. Test Board Results:

Here the Hardware test board (Ic used: CY8C29466-24P×I), is directly connected to the Spartan 3e FPGA kit, using wires. By using syp tool we dump the counter code into the IC. The coding used here is “c language”. For every 16 samples the counter will be incremented from “00” to “FF”.we observe the results for this counter operation interms of hexadecimal by using HyperTerminal, by using Java Based GUI tool we observe the results with respect to bit waveform.

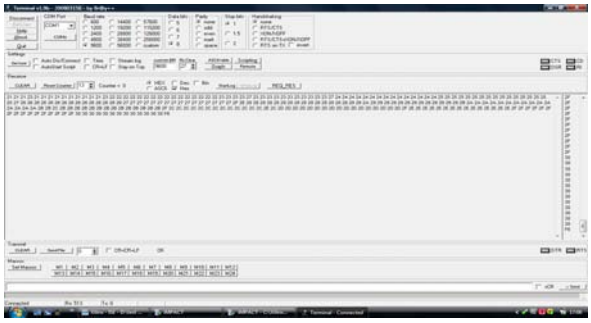

Fig 4.11 Hardware Test Board results for LSA using Hyperterminal

Fig 4.12 Hardware Test Board results for LSA using GUI

V. CONCLUSION

A low cost Logic Signal Analyzer is designed targeted to Xilinx FPGA. The design contains 16-channels and samples at a maximum frequency of 25Msps. The design is functionally verified using ModelSim simulator. Hardware verification is done using a test board with PSoC microcontroller. The design occupies 20% of Spartan 3E FPGA XC3S500E. A Java based GUI is used to plot and verify the results on PC.

VII. REFERENCES

[1] Integrating logic analyzer functionality into VHDL designs,2008 conference.

[2] Altera Corporation, “*SignalTap II Embedded Logic Analyzer Documentation*”.

[3] First Silicon Solutions, “*FPGAView Software*“, www.fs2.com/fpgaview.html

[4] P. S. Graham, “*Logical Hardware Debuggers for FPGA-Based Systems*”, PhD Dissertation, Bringham Young University.

[5] www.agilent technologies.com

[6] www.wikkipedia/freeencyclopedia.com

[7] www.xilinx .com/ise

[8] www.tectronix.com

[9] www.java/netbeans.com

[10] www. Xilinx.com/s3eboards

[11] Apply Error Vector Measurements in Communications Design, ken Voelker, Microwaves & RF, December 1995.

[12] Vector signal analysis of digital baseband and if signals within an FPGA”, autotestcon, 2005. Ieee

[13] Testing of digital circuitry using xilinx chipscope logic analyzer”, cas 2005 proceedings, orest oltu, petru lucian milea.