eASIC®
The configurable logic company™

Low-cost Implementation of Digital Oscilloscope in Nextreme™ Structured ASIC

By Mircea Moldovan, Dan Nicula, and Traian Tulbure,
eASIC Corporation

Whitepaper WP-1 Version 1.0

Rapid Implementation of a Digital Oscilloscope in Nextreme Structured ASIC: eScope-90

Nextreme Structured ASICs provide quicker time-to-market and lower development costs than cell-based ASICs, while providing higher performance, lower power and lower unit cost than FPGAs.

In the deep-submicron design era, it is clear that Standard Cell ASIC has become far too costly for most applications and its lengthening turnaround time is an additional drawback. The high up-front mask charges, together with costly EDA tools and large design teams required for completing a design, inhibit the access to the advantages of Standard Cell: lowest unit cost and highest performance. At the other end of the custom design spectrum, the FPGA solution often lacks the performance or power required, and although the upfront NRE it eliminated, the unit cost is the highest. To satisfy the electronic designer's project needs, Structured ASICs are increasingly becoming the ideal choice for custom silicon applications.

A subset of the Structured ASIC category is the Via Configurable ASIC, in which via layers are used to customize the design rather than metal layers. Such technology is employed in eASIC's Nextreme Structured ASIC where all metal layers are standard/pre-fabricated. Using a unique routing technology, four metal layers are applied for efficient segmented routing, and up to two via layers are customized to implement the design in Nextreme fabric.

The following case study describes the implementation of a digital Oscilloscope on eASIC's Nextreme 90nm Structured ASIC fabric. This design is dubbed eScope-90. It includes a two channel digital sampling oscilloscope and an arbitrary waveform generator in a single USB-powered module. The Nextreme device includes the digital logic (sample buffer memory interpolating digital trigger logic, waveform output buffer memory, data sequencers, and USB IO interface logic) and interfaces to external analog circuitry and USB transceiver logic. A single on-board oscillator drives the on-chip PLL clock generators to create separate clock domains for each digital input and output channel as well as for the USB IO channel.

The eScope-90 design is connected to a PC through USB. A graphical user interface (GUI) on the PC is used to view and process the acquired data. The following discussions detail the eScope implementation.

1. Overview

The eScope-90 design features a PC-based digital sampling oscilloscope. PC-based oscilloscopes offer real cost savings over desktop oscilloscopes. One can use the existing large color display, fast processor, and large disk storage of the PC instead of having to buy a stand-alone oscilloscope. Digital sampling oscilloscopes use the equivalent-time sampling method to capture and display signal samples.

Sampling oscilloscopes can measure signals up to an order of magnitude faster than real-time oscilloscopes. As such, these oscilloscopes are ideal tools for capturing and characterizing computer, datacom and telecom signals. As shown in Figure 1, the eScope system requires only the eScope board and a host computer.

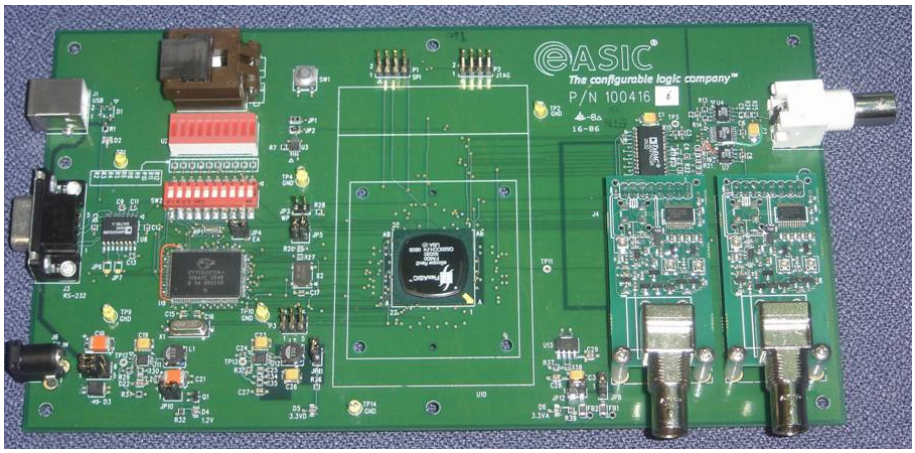

Figure 1: The eScope-90 board

2. The eScope-90 Architecture

The eScope-90 design is implemented using nine sub-modules; it employs the Open Core Protocol (OCP) to interconnect a common shared memory to various data access ports. The sub-modules are as follows:

- clkGen: clock/reset generator for eScope.
- adcInput: synchronizers for the data samples from ADC.
- dacOutput: FIFO synchronizers for the waveform data to DAC.
- trigGen: trigger generator includes an OCP initiator port.
- waveGen: waveform generator logic includes an OCP initiator port.
- hostIf: USB module host interface includes an OCP initiator port.
- sampleMem: block memory shared for storing data samples and wave generator data.
- ocpMerge: 3: combinatorial OCP merge.
- ocp2mem: OCP to memory interface converter includes an OCP target port.

The sub-modules interconnections are as shown in Figure 2.

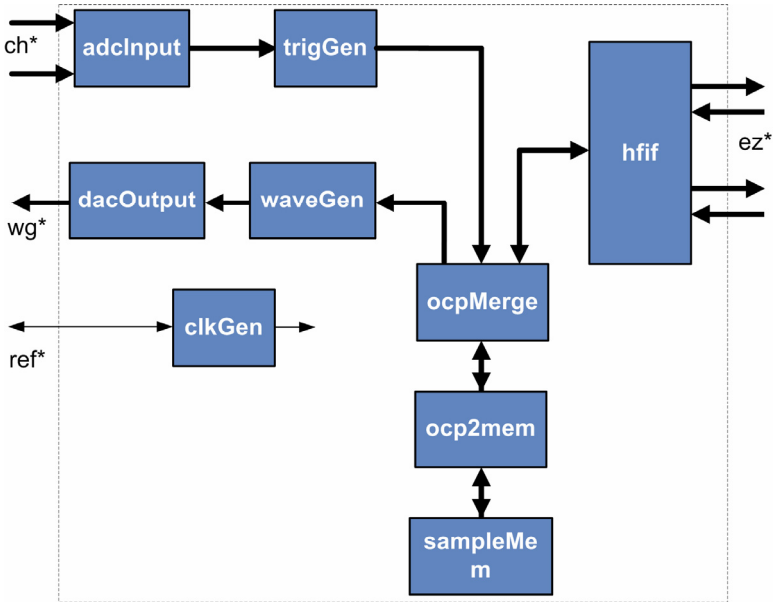

Figure 2: The eScope-90 internal architecture

3. The Acquisition Path

The acquisition path is implemented by three modules: clkGen, adcInput and trigGen. The clock generator (clkGen) creates the clock and reset signals for each clock domain on the chip. eScope-90 performs acquisition on the scClk4x signal and waveform generation on wgClk, both running at 2 0 MHz. Other on-chip clocks include ezClk (24 MHz) and hfClk (48 MHz).

The adcInput module receives 2-bit data samples from the ADCs are packs them into 96-bit width words. These words are sent to the trigGen module. In addition, the time base is implemented in this module.

2-bit data from the ADCs are generally transferred to memory as parallel data streams. When eScope-90 is configured for double sampling mode, the sample clocks between channels are offset by 80 degrees and the data from both channels is interleaved and sent to memory. Samples are propagated forward based on triggering and the 6-bit hfFreqDiv_i parameter. This parameter specifies how many data samples are preserved from the ADC data streams. The time base is implemented based on the following formula:

```
No_sample * T_sampled * (hfFreqDiv_i + 1) = No_divisions * Time/division
```

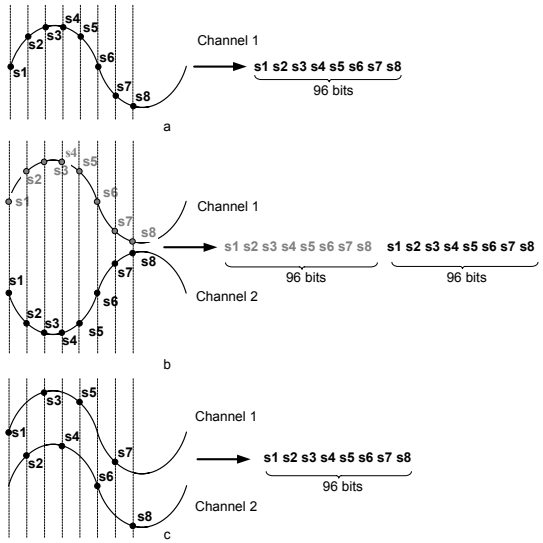

Figure 3: Packing data into 96-bits words: (a) single channel mode, (b) double channel mode, and (c) double sampling mode

The trigGen module implements all trigger logic controlled on acquisition parameters. 96-bit widths words are written in memory using the OCP interconnect channels. The trigger logic operates according to sequencing as presented in Figure 4. A -bit signal hfStartAcq_i created by hfIf for one clock period, initializes the trigger logic. A 1-bit signal hfEndWindow_o returned to hfIf

indicates the end of an acquisition. During an acquisition period, the aqPend signal is active. The trigPend signal becomes active during the acquisition, before the post trigger period, and after hfTriggerPos_i * 6 memory addresses * 8 samples are written to memory. Trigger events are positioned in the memory area dedicated to the specific channel, depending on the 8-bit parameter hfTriggerPos_i from hfIf. The search for trigger events takes place only when the trigPend signal is active. This happens when a trigger event occurs as indicated by the doTrigger signal becoming active.

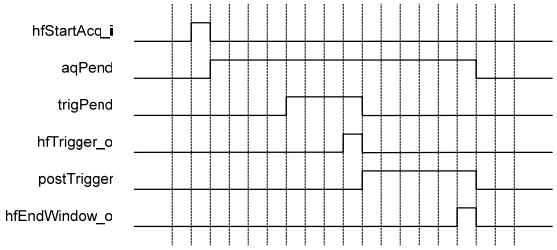

Figure 4: Trigger control logic

The address of the first data to be read by the CPU (hfOffsetAddr_o) and the offset of trigger event into the 8 sample from a 96-bits width word (hfOffsetTrigger_o) are computed and stored in the hfif module. After each trigger event occurs, a post-trigger period follows, indicated by the postTrigger signal being asserted. This signal becomes active when doTrigger occurs and inactive when hfEndWindow_o occurs. All of these signals are used as control signals for different operations that take place inside the trigger logic module.

The eScope's acquisition part can work with four different trigger types as specified through 2-bit hfTriggerType_i input from the hfIf module. These four types are as follows (and as illustrated in Figure 5):

- Trigger on positive edge;
- Trigger on negative edge;
- Trigger on positive pulse;
- Trigger on negative pulse.

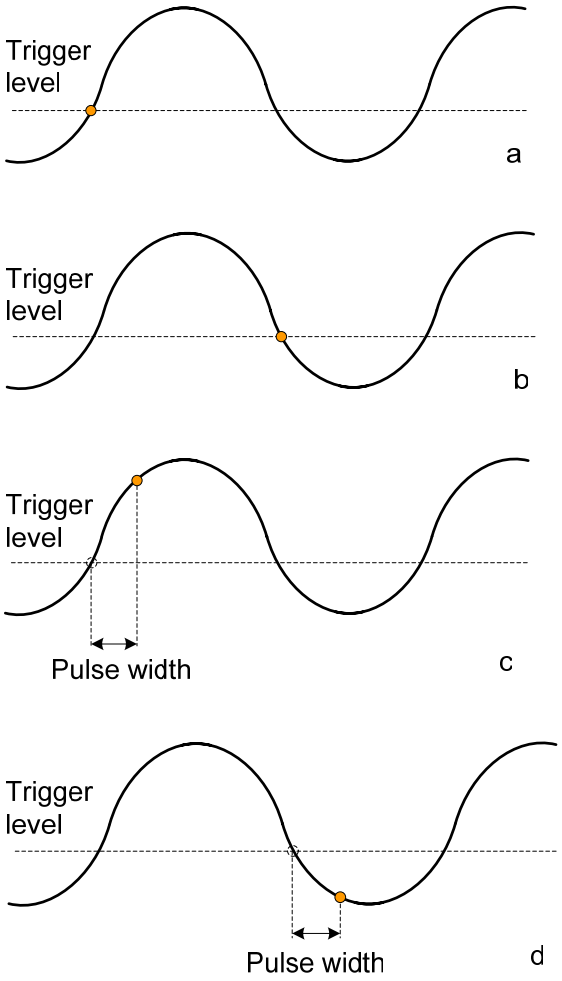

Figure 5: Trigger methods: (a) positive edge, (b) negative edge, (c) positive pulse, and (d) negative pulse

Valid samples are packed into 96-bits words and are written into memory using the OCP protocol. Each channel has a separate address counter. When a write request occurs, the proper counter is selected based on the acquisition mode. The entire memory is managed as illustrated in Figure 6.

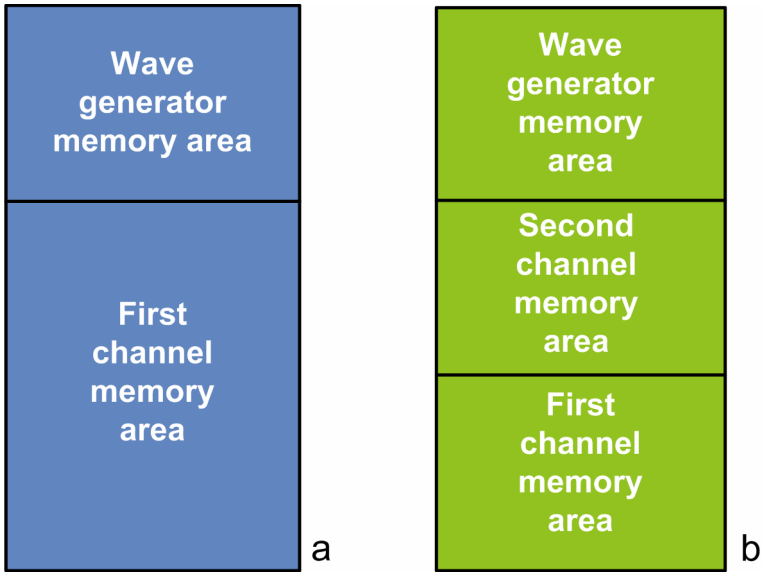

Figure 6: Memory management: (a) single and interleaved sampling mode, and (b) double channel acquisition mode

4. The Wave Generator Path

The main on-chip memory is shared between acquisition and wave generator functions. User defined samples can be uploaded into the wave generator memory. Once this is done, the wave generator outputs this signal at a frequency selected by the user.

The wave Generator output path is formed by two modules: waveGen and dacOutput. The role of the waveGen module is to generate OCP read commands to memory and transfer the received 96-bits data words to the dacOutput module. The transfers are done using OCP protocol synchronized to the scClk4x_i clock.

The dacOutput module receives data from the waveGen module and sends it to the DAC outputs. The dacOutput module synchronizes the data from the scClk4x_i clock to the wgClk_i using asynchronous FIFO memories. These FIFOs are 96-bit wide by 32-locations deep. The FIFO memory is created using the dual-port RAM memory that is available throughout the eASIC Structured ASIC logic fabric.

Each 96-bit data word is sent out to the DAC on a 2-bit interface over (8 x hfSampleWidth_i) consecutive wgClk_i clock periods. Each sample is held on DAC input a for hfSampleWidth_i periods of wgClk_i. This mechanism implements a simple frequency divider for the waveform generator output. The frequency divider parameter (hfSampleWidth_i) is written in the eScope-90 control registers. Memory requests from the waveGen module are made by an embedded FSM sequencer that issues memory requests when space is available in the FIFO memories.

5. The Functional Scenario

All accesses to eScope-90 are made using a GUI in conjunction with the EZ-USB driver procedures, which are summarized as follows:

- The write procedure configures hfIf registers by specifying register address on address bus and also by specifying control signals.
- The read procedure gets data from hfIf registers by specifying register address on address bus and by specifying control signals.
- The burst write procedure stores samples in wave generator memory area by specifying samples on data bus and by specifying control signals. hfIf module manages the address.
- The burst read procedure gets samples from eScope acquisition memory area by specifying control signals. hfIf module manages the address.

Using these four procedures, the application environment is programmed to support acquisitions and wave generator operations. The acquisition path functional scenario is as shown in Figure 7.

By comparison, the wave generator path functional scenario is as follows:

- Setup registers configuration using write procedure.
- Burst write wave generator samples into memory.
- Start wave generator using write procedure.
- Run

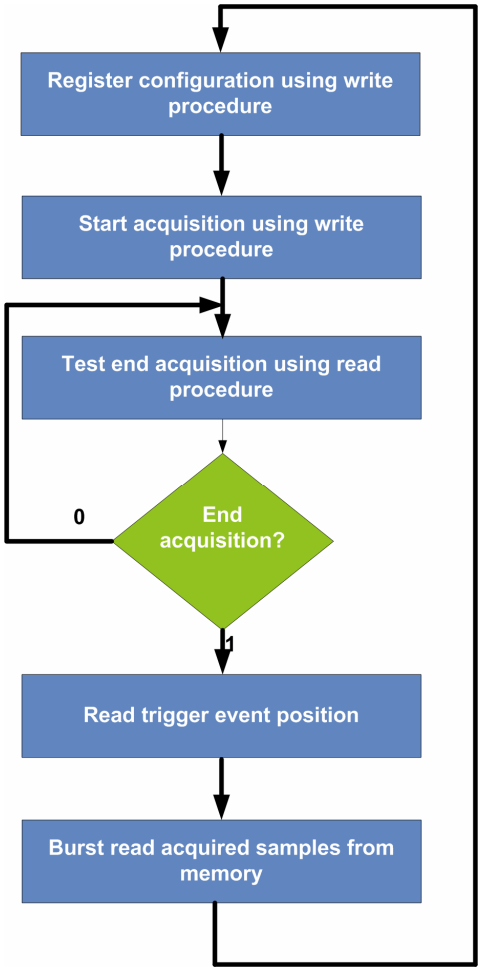

Figure 7: The acquisition path functional scenario

6. The Simulation Environment

The simulation environment is as illustrated in Figure 8. The simulation bench contains an eScope-90 instance, which includes logic (instantiated in eScopeLogic), a clock signals generator (in eScopeClk), pad instances (in eScopePads); ADC and DAC instances, an EZ-USB interface called ez_usb_dummy and two additional modules that generate reference clock (clk_gen) and reference reset (reset_gen) signals. An ezusb.h file is included which describes the functionality of EZ-USB interface. This file programs the design with different functional scenarios.

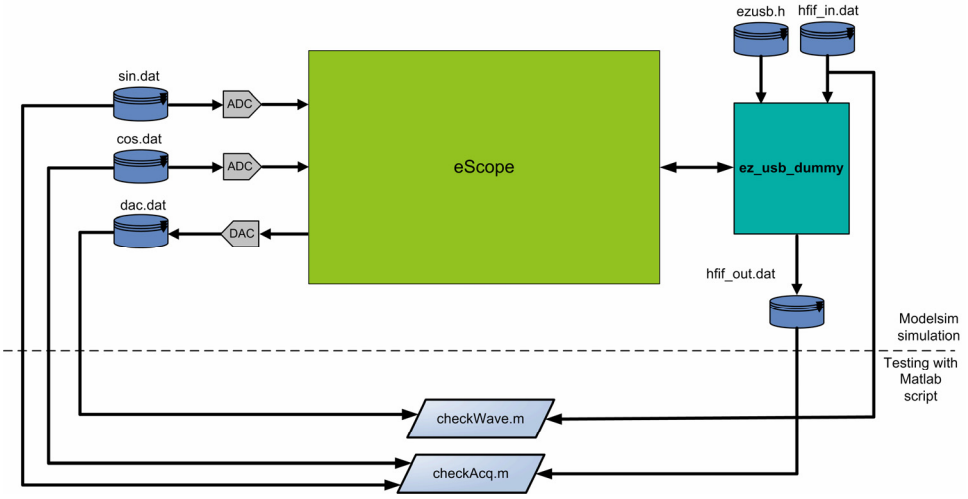

Figure 8: Simulation environment

The ADC modules used for simulation contains a memory for each channel from which samples are read. These memories are loaded from initialization files. Each sample is multiplied by a gain parameter to amplify the signal sent to the eScope-90 inputs. The DAC module receives data from wave generator into a file for analysis.

The ez_usb_dummy module implements four tasks (“procedures”) for accessing eScope-90: write, read, burst write and burst read as previously described. Because eScope operation is highly configurable, it is not possible to simulate all operating conditions. Operating conditions were created by initializing each register with one of four possible values: low limit value, high limit value, and two intermediate values. A 5-bit number is created using a seeded random function generator. Groups of bits from this number are associated with registers their initialization value for a particular simulation.

Sine, cosine, and complex waveforms were used as test signals during simulation, and the simulations were partitioned into 3 types:

- Tests which determine if a trigger event is correct using Verilog simulator assertions.
- Tests which check if the acquired data matches the original “generated” data from the ADCs. The sampled signal is transferred over the USB interface to a virtual host and checked against the original data signal using a MATLAB script.
- Tests which check that the wave generator outputs match the original data uploaded in wave generator memory, also using a MATLAB script.

7. Implementing the eScope-90 on Nextreme

The design flow used for this implementation is illustrated in Figure 9 and is based on MAGMA BlastCreate SA suite. This tool flow leverages existing ASIC design methodology and infrastructure as much as possible.

The MAGMA design flow creates a fully placed design that is routed with eASIC's design tools suite called eTools™.

eWizard is a graphical tool that provides designers the ability to do I/O pin-pad assignment and clock planning Logic synthesis, packing and placement is performed with the Magma BlastCreate SA using eASIC's synthesis library containing all of the two, three, four, and five input functions that can be implemented with combinations of LUT configurations in the eCell, as well as the inverting buffer models and a number of DesignWare components that have been optimally implemented in the eCell logic.

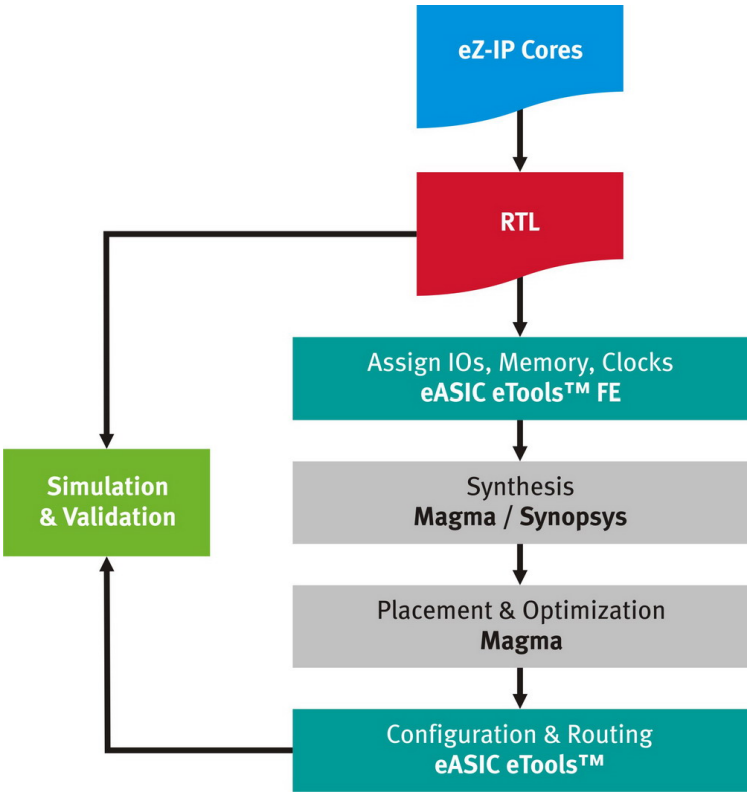

Figure 9: The Nextreme design flow

The placed netlist produced by MAGMA BlastCreate SA Design Compiler is passed through the eASIC-developed tools: Design Configurator tool (called eDC) that performs design mapping to the selected Nextreme family member.

Design is then routed using the eASIC-developed via router (called eVrouter).

The results from the implementation of eScope-90 on eASIC's Structured ASIC fabric using the MAGMA BlastCreate SA as described above are reflected in Figure 10. The design was implemented into NX750 device of the Nextreme family, both in VL (via-programmable logic) and SL (SRAM programmable logic) versions. The package was 208BT.

Clocks	Synthesis (ns)	Mapping (ns)	Place & Route (ns)
Wave generator	3.66	4.12	4.04
System	3.74	4.6	5.04
Host interface	-	-	3.5
Area		4087 eCells	

Table 1: eScope-90 implementation timing and area results

The back-end design flow uses eASIC's eDC and eVrouter tools to implement the final place and route. Parasitic extraction was performed to annotate the final netlist for back-end (signoff) static timing analysis using Primetime. Backannotated simulation and static timing was performed using parasitic annotation data produced by the eTools™ router. The eTools™ suite supports multiple simulation and STA solutions from leading EDA vendors.

In parallel ATPG vectors are created for production testing using Synopsys TetraMax.

eASIC eTools™ generate all of the necessary data required for final implementation. This includes the via configuration and the bitstream files (the later is for Nextreme SL only).

8. eScopeSW - Graphical User Interface

eScopeSW is a GUI (graphical user interface) application that controls the eScope-90 hardware. It was developed as a LabVIEW "VI" using the LABVIEW development environment and C libraries. The USB interface is the physical medium used for data transfer between the eScope-90 hardware and the PC running the eScope-90 software. The communication between GUI and eScope-90 device uses a 2-layer structure as follows: EZ-USB driver (written only for the Win32 platform).

C high-level access functions that access the low-level driver's functions.

The EZ-USB driver is responsible for data transfer between the software and the hardware modules. It provides low-level functions that are used in the C libraries.

There are two kinds of C function:

- Low-level functions (read/write samples/register values, connection status) that use the features implemented in the EZ-USB driver.
- High level functions that the driver. These functions are called by the LabVIEW application when an operation accessing the hardware module is done.

A screen shot of the eScopeSW is shown in Figure 10.

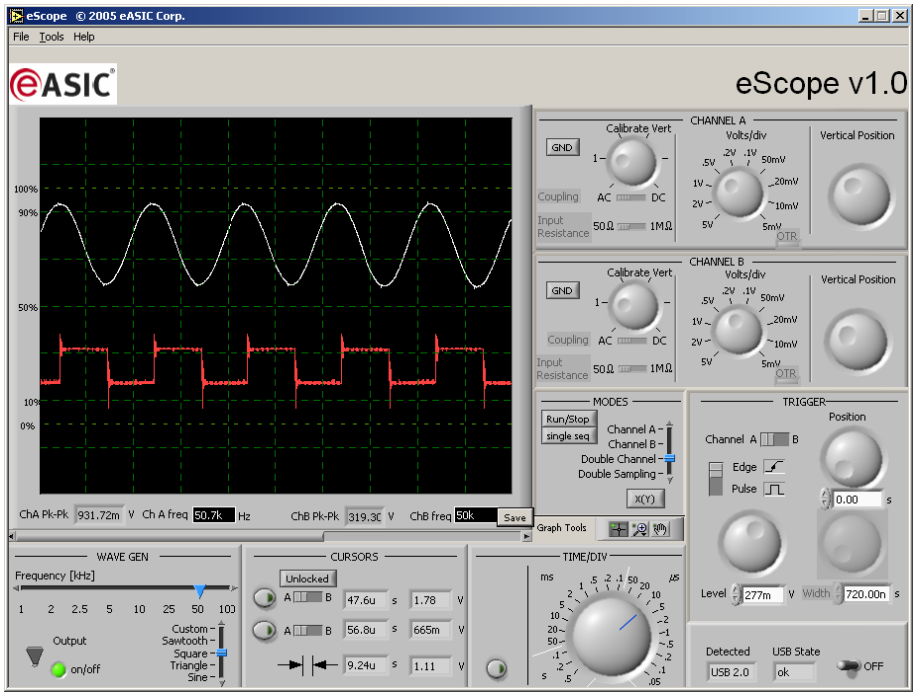

Figure 10: The eScope-90 GUI application - eScopeSW

Graphical objects include controls and indicators. The controls are divided into Boolean controls (push-buttons), toggle buttons, horizontal switches and numerical controls: horizontal and vertical sliders with multiple values. The indicators are divided into numerical, text indicators, leds, and graphical waveform indicators. The controls and the indicators from the front panel are grouped by function as might be found on a traditional desktop oscilloscope.

9. Summary

eScope-90 was developed as a reference design to be used in tutorials for the Nextreme design flow and to demonstrate the technology advantages of eASIC's Structured ASIC fabric for high-speed logic design. In addition, eScope-90 demonstrates that high-speed, low-cost products can be created using innovative Structured ASIC technology from eASIC Corp.

10. References

1. eScope-90 design specification, eASIC Corp.
2. “OCP-based memory access arbitration for a digital sampling oscilloscope”, by Traian Tulbure and Dan Nicula, 2005, Programmable Logic DesignLine.
3. The www.ocpip.org website.
4. The PC Oscilloscope section on the www.oscilloscopeguide.com website.
5. “Oscilloscopes,” Fifth Edition by Ian Hickman – Reed Educational and Professional Publishing Ltd 200. ISBN 0 7506 47574
6. Tektronix TDS5000B Series Digital Phosphor Oscilloscopes Quick Start Manual 07 - 355-02 by Tektronix Inc.

Dr. D. NICULA is a professor at Transilvania University of Braşov, Romania, Department of Electronics and Computers. He graduated the Polytechnic Institute of Bucharest, Romania, in 1990 with an MSc degree in Electronics. In 1997, he was awarded the PhD in Computer Sciences by Transilvania University of Braşov. His research interests are mainly in the area of VLSI design and HDL RTL level modeling. Teaching duties include HDLs, microprocessors and digital electronics. He is author of Romanian books regarding VHDL and Verilog. He is the General Manager of the Romanian eASIC Corporation subsidiary. Dan can be reached at dan@easic.ro.

Traian Tulbure is a lecturer and PhD candidate at Transilvania University of Braşov. He graduated from Transilvania University in Braşov, Romania, Department of Electronics and Computers, in 2000 with MSc degree in Electronics and Computers. His graduation thesis was entitled: “EDA tools for eASIC technology.” His research interests are in the area of VLSI design, HDL modeling, C/C++ programming, and synthesis. Traian can be reached at traian@easic.ro.

Mircea Moldovan is an application engineer at eASIC Corporation. He graduated from Transilvania University of Braşov, Romania, Department of Electronics and Computers, in 2005 with BSc degree in Electronics and Computers. His graduation thesis was entitled: “eScope - Digital Sampling Oscilloscope on a Chip.” He also got an MSc degree in digital communication networks from Transilvania University of Braşov in 2007. His research interests are VLSI design, HDL modeling, C/C++ programming and synthesis. Mircea can be reached at mircea@easic.ro.

11. Revision History

Date	Version	Summary of Changes
14/06/2007	1.0	Initial release

eASIC Corporation

2001 Walsh Ave

Santa Clara, CA, 95050

Tel: +1 408 855 9200

Fax: +1 408 855 9201

Email: info@eASIC.com