How to implement a digital oscilloscope in Structured ASIC fabric

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

Page 1 of 3

Programmable Logic DesignLine
(07/12/2006 7:07 PM EDT)

As the development costs for Standard-Cell design in deep-submicron technology approach the multi-million dollar level, it is inevitable that some designers will shift to an alternative solution that can reduce development costs, even if there is some penalty in overall cost or performance.

Structured-ASICs have emerged as this alternative to standard-cell design. Bridging the gap in performance and cost between Standard Cell ASICs and high-density FPGAs, Structured ASICs maintain the best aspects of both technologies. Designers can achieve quicker time-to-market and lower development costs than standard ASICs while also achieving higher performance and lower unit costs than FPGAs.

A subset of the Structured ASIC category is the Programmable ASIC, which is a Via-customizable, rather than metal-layer customized. In Programmable ASIC arrays, all metal layers are standard/pre-fabricated, out of which four layers are used for efficient segmented routing, and only a single via-layer is customized to implement a design.

The following case study describes the implementation of a digital Oscilloscope on the eASIC Programmable ASIC fabric. This design is dubbed eScope. It includes a two-channel digital sampling oscilloscope and an arbitrary waveform generator in a single USB-powered module.

eScope was implemented on a 130nm Programmable ASIC device. The chip 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 80MHz 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 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.

Overview

The eScope is 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 Fig 1, the eScope system requires only the eScope board and a host computer.

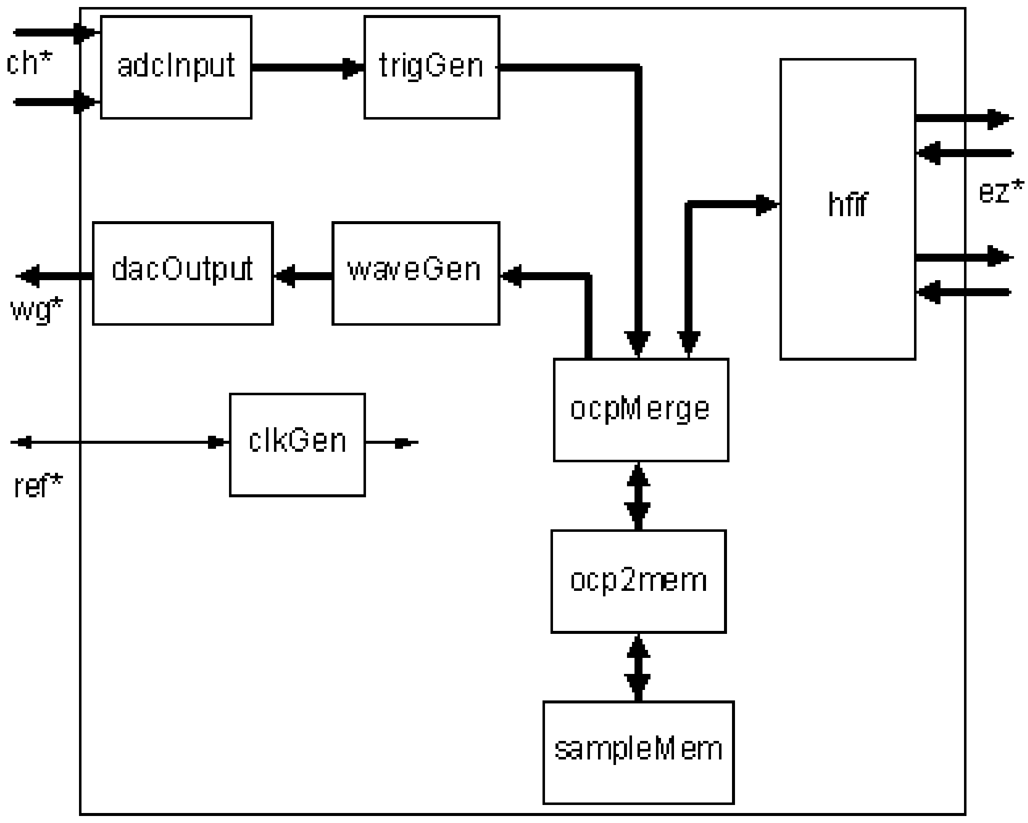
1. The eScope system.

The eScope architecture

The eScope 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:*** *3:1 combinatorial OCP merge.*
- ***ocp2mem:*** *OCP to memory interface converter, includes an OCP target port.*

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



2. The eScope"s internal architecture.

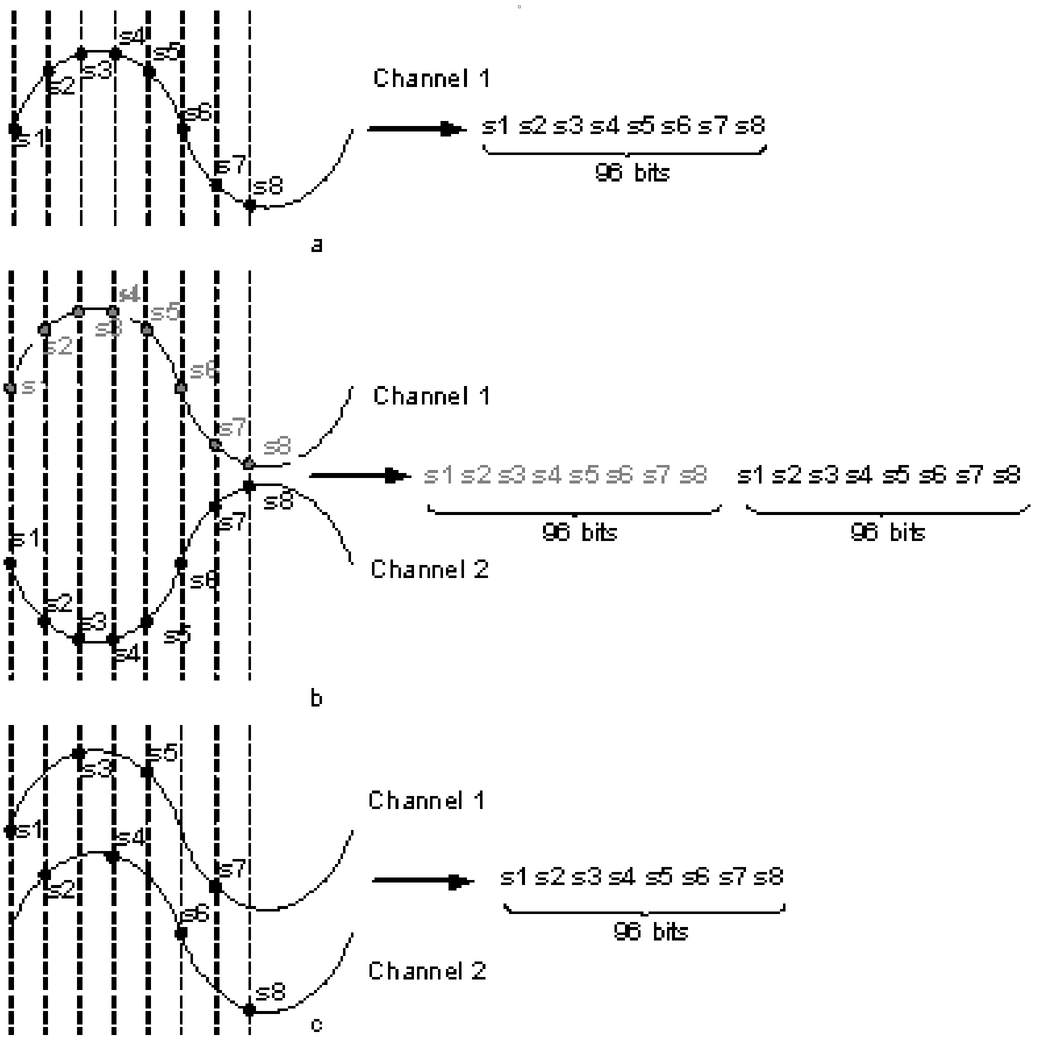
The acquisition path

*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 performs acquisition on the scClk4x signal and waveform generation on wgClk, both running at 210 MHz. Other on-chip clocks include ezClk (24 MHz) and hfClk (48 MHz).*

*The **adcInput** module receives 12-bit data samples from the ADCs are packs them into 96-bit width words. These words are sent to the **trigGen module**. Also, the time base is implemented in this module.*

12-bit data from the ADCs (ch1Data_i and ch2Data_i) are generally transferred to memory as parallel data streams. When eScope is configured for double sampling mode, the sample clocks between channels are offset by 180 degrees and the data from both channels is interleaved and sent to memory. Samples are propagated forward based on triggering and the 16-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:

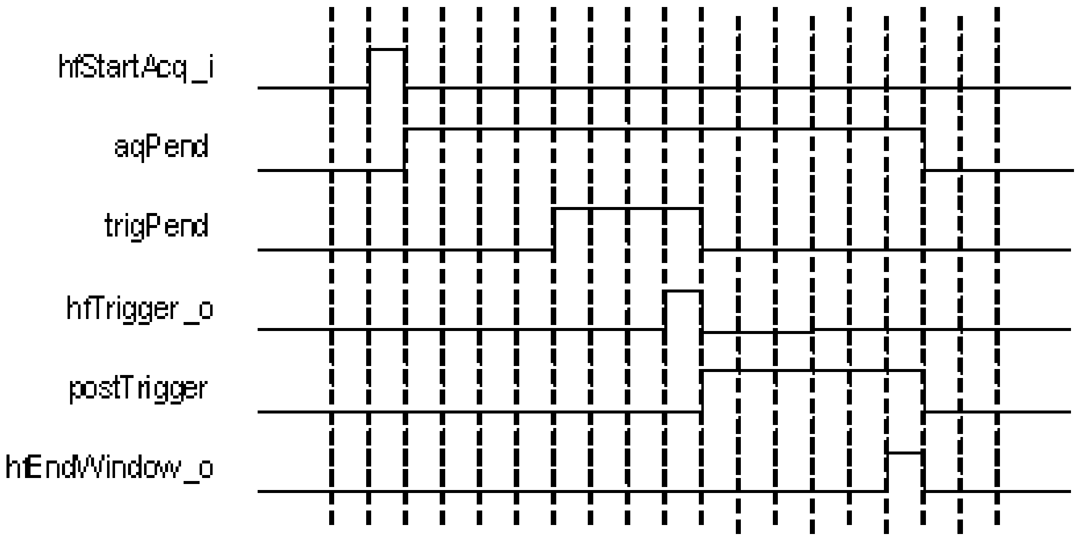
No_sample * T_sampled * (hfFreqDiv_i + 1) = No_divisions * Time/division



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 width words are written in memory using the OCP interconnect channels.*

*The trigger logic operates according to sequencing as presented in Fig 4. A 1-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 * 16 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.*



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.