Logic Analyzer Triggering Techniques to Capture Elusive Problems

Efficient Solutions to Elusive Problems

For digital designers who need to verify and debug their product designs, logic analyzers provide breakthrough triggering features that capture real-time digital system operation. The logic analyzer triggering functions provide an effective and efficient way to verify and debug digital systems designs. This note explains how to use logic analyzer triggering to meet your design schedule and improve your design quality by capturing elusive problems and verifying your design implementation.
Logic Analyzer Triggering Techniques

Introduction

A logic analyzer verifies and debugs digital hardware, real-time embedded software, and the interaction between hardware and software. There are two approaches to debugging a digital system. The first approach is to use the logic analyzer to acquire large portions of data and then search through that data to find faults or verify correct operation. This technique is time consuming and error prone. It also depends on being extremely lucky in capturing the errors when they occur.

A more productive debugging technique is to use the logic analyzer to monitor the digital system for faults and then capture them when they occur. To do this, the logic analyzer is set up to trigger on and capture the fault. The logic analyzer will monitor the system as long as you desire and you will capture the faults that occur with confidence. In other words, you capture only the important data you want, thus avoiding being buried in useless data.

You can set up Tektronix TLA logic analyzer triggers by using either EasyTrigger or PowerTrigger. EasyTrigger provides an easy-to-use selection menu of commonly used trigger setups, with a description for each setup and a graphical waveform diagram of the triggering. PowerTrigger gives you full access to the logic analyzer triggering capability, which includes trigger states and resources. Triggering applications below will show examples of using both EasyTrigger and PowerTrigger setups.

The following examples apply to a wide range of digital circuits used in applications from consumer electronics to aerospace applications. These examples start with simple “trigger immediately”, progressing to powerful “glitch and setup/hold violation triggering”, and end with “triggering on digital systems with microprocessors”.

Capturing a Signal Immediately

A simple way to trigger a logic analyzer is to trigger it immediately. See Figure 1. The logic analyzer triggers as soon as the Run button is pushed and the logic analyzer does not wait for any specific pattern or event to start its acquisition.

We will use the logic analyzer to verify a flip-flop circuit for correct operation. Setting the logic analyzer to trigger immediately captures the flip-flop clock, D input and Q output waveforms in Figure 2. The circuit seems to be operating as expected because the Q output latches the D input on the rising edge of the clock.
The top waveform in Figure 2 is the LA 1: Sample ticks that represent the logic analyzer’s deep timing sampling rate of 2 ns (500 MHz). In addition to deep timing all Tektronix logic analyzers have high-resolution MagniVu™ 125 ps (8 GHz) timing that is running simultaneously with deep timing using the same probes. It is like having two logic analyzers in one: a deep timing logic analyzer and a high-resolution timing logic analyzer.

We display the MagniVu high-resolution timing waveforms time aligned with the deep timing waveforms without reacquiring the data as shown in Figure 3. Notice the additional details that are on clock waveform with the 125 ps high-resolution timing. The clock signal has a narrow pulse problem at the beginning of every clock pulse.

Next, we use the logic analyzer’s iView™ measurement capability to debug the clock signal by integrating a TDS oscilloscope with the logic analyzer to take an analog measurement of the clock signal and display it on the logic analyzer display. In Figure 4, analyzing the time-correlated analog waveform with the digital waveform shows that a non-monatomic rising edge is causing multiple logic transitions on the leading edge of the clock signal.

The lack of decoupling capacitors on the power supply lines of the clock driver was the root cause of the poor clock edge. The fixed circuit waveforms are shown in Figure 5.
Capturing Glitches

In system design, glitches are annoying pulses that can be very difficult to detect and capture. Their effects on the digital system are often unpredictable. Glitches are caused by any number of conditions such as crosstalk, race conditions, termination errors, driver errors or timing violations.

Glitches are usually very narrow pulses and require high-speed timing to measure them. They can be intermittent and extremely hard to find because they may not be occurring when you are capturing the signals. Triggering on glitches is a quick way to debug them.

On analyzing the waveforms in Figure 5, one might assume that the flip-flop is working correctly. This assumption would be based on just that instant of time that the logic analyzer captured the data.

iView™ Display

Today almost every design is a high-speed design with fast clock edges and data rates. For these designs, you need to see the analog characteristics of high-speed digital signals in relation to complex digital events in the circuit. The iView display is your window into the digital and analog world. The iView capability seamlessly integrates and time-correlates data from the logic analyzer and oscilloscope, which automatically transfers the analog waveforms to the logic analyzer display. View time-correlated analog and digital signals side-by-side and pinpoint the source of elusive glitches and other problems in moments.

We can configure the logic analyzer to monitor the system for signal faults, as long you desire. We can count the number of glitches over a user selectable time span. Finally, we can capture the glitches for analysis with the logic analyzer’s 125 ps high-resolution timing and iView oscilloscope waveforms.
The logic analyzer is configured to count the number of glitches and uses a timer to show how long the logic analyzer was running. In Figure 6, the logic analyzer was stopped after 60.819 seconds but it could have run counting glitches over a two-day weekend or longer. In 60 seconds the flip-flop had five glitches as shown by the counter value of five in Figure 6.

The logic analyzer's PowerTrigger was used to count the number of glitches. In Figure 7, it shows that one timer, one counter, one glitch detector, and two trigger states were used. The first trigger state resets a counter and starts a timer. The second state increments the counter for each sample that contains a glitch. The logic analyzer stays in state two until the user stops the logic analyzer using the Stop button.

Next, we configured the logic analyzer to trigger and capture the glitch on the flip-flop Q output signal. As shown in Figure 8, the logic analyzer will stop when it triggers on a Q output glitch; otherwise, it keeps looking for a glitch until one is found or the user stops the logic analyzer with the Stop button.

Figure 9 shows the waveforms acquired by triggering on a glitch on the flip-flop Q output signal. The logic analyzer shows the location of the glitch with a red glitch bar on the Q output deep timing waveform. The center MagniVu™ waveforms measure the glitch with 125 ps resolution and bottom iView™ waveform shows the analog characteristics of the glitch.
Using the MagniVu waveforms, we measure the output glitch to be 1.5 ns wide. For a normal operating flip-flop we expected the output to stay high because the D input was high when the rising clock edge occurred. Upon reviewing flip-flop electrical characteristics in Figure 10 we see that the flip-flop 3.0 ns setup time was violated because the D input went high 1.5 ns before the rising clock edge. When the setup/hold violation occurs, the output of the flip-flop goes into a potentially metastable state and the outputs are unpredictable. This can result in the flip-flop Q output producing a glitch, the output could briefly oscillate, or the output may not change at all. These types of failures are usually intermittent and can be difficult to find when using instruments without glitch triggering and a glitch display. In this case the circuit design flaw is that the D input signal is asynchronous with the clock signal and it occasionally changes in the setup/hold window of the flip-flop clock. A circuit redesign is needed to fix this error.

Logic analyzer glitch triggering can be done on circuits with wide buses with hundreds of signals. Every signal at every sample point is checked for glitches and red glitch bars will indicate glitch locations on the deep timing waveforms.

Capturing Setup/Hold Violations
A setup/hold violation does not always produce a glitch in the flip-flop Q output. A way to verify the flip-flop setup/hold operation is for the logic analyzer to trigger on a setup/hold violation. Figures 11 and 12 show the logic analyzer configured to trigger on a D input 3.0 ns setup and a 1.0 ns hold violation with respect to the rising clock.
edge. We can configure the D input setup/hold window from 16 ns before the clock edge to 8 ns after the clock edge with 125 ps resolution. The 3.0 ns setup and the 1.0 ns hold values were chosen based on the flip-flop electrical characteristics in Figure 10.

Figure 13 shows the logic analyzer triggering on a D input setup violation of 1.875 ns before the clock edge. Notice that the Q output did not glitch or change to a low even though the D input was low before the clock edge.

All synchronous digital circuits have a setup/hold requirement. Using the logic analyzer setup/hold violation triggering you can verify that your design is meeting the device’s setup/hold specifications. For example, you can verify a 128-bit address bus and a 32-bit data bus operation for setup/hold violations using the logic analyzer setup/hold triggering.

Verifying Flip-Flop Operation

The flip-flop operation can be verified by using the logic analyzer’s PowerTrigger capabilities. The flip-flop operates by looking at its D input at the rising edge of the clock. The Q output changes to the D input value after the propagation delay (typical 4 ns.) of the flip-flop. Since the propagation delay is shorter than the pulse width of the clock, we can use the logic analyzer to see if the Q output at the falling edge of the clock is the same value as the D input at the rising edge of the clock. Figure 14 shows the logic analyzer’s PowerTrigger setup to check every clock cycle and trigger if the Q output at the falling edge of the clock does not equal the D input at the rising edge of the clock.

Figure 15 shows the logic analyzer triggering on the falling edge of the clock when the flip-flop Q output was low. The logic analyzer first looked at the rising edge of the clock and the value of the D input, which was logic high. Next, the logic analyzer looked for the falling edge of the clock and the value of the Q output, which was logic low. The logic analyzer triggered because the Q output did not match the D input.
Triggering on Microprocessor Startup Operation

Debugging the startup operation of a microprocessor is very difficult using a software debugger because the software debugger requires the microprocessor to be correctly running with the I/O drivers and its hardware communicating with the software debugger on the external host or an operating debugger interface on the target system.

The logic analyzer non-intrusively monitors and captures the microprocessor control bus, address bus, data bus and I/O signals. The logic analyzer does not affect or depend on correctly operating microprocessor. As result, the logic analyzer is an excellent tool in verifying and debugging the microprocessor kernel and the I/O ports that are needed to run the software debugger.

Verifying the startup code and hardware operation of a microprocessor-based system is straight forward using a logic analyzer. You trigger the logic analyzer on the reset vectors, the reset address or the control signal(s) that defines the reset operation. Once the logic analyzer triggers, you display and verify software execution in the logic analyzer listing window (see Figure 16) and verify the hardware operation in the logic analyzer waveform window (see Figure 17).
Capturing Stack Over and Under Flow

You can check for stack over flow and under flow by triggering the logic analyzer on the addresses above and below the memory stack. Make sure you allocate unused memory above and below the stack so that the logic analyzer does not trigger on valid software operations on these memory locations. Analyzing the real-time code execution before the logic analyzer trigger will determine if the stack memory size is too small or if there is an imbalance in the storing and retrieving data from the stack.

Checking Interrupt Latency

Interrupts are used by the microprocessor to respond to external events in a timely manner. Errors can occur if the interrupt latency is too long. Interrupt latency is the time it takes a microprocessor to respond and to process the interrupt. Interrupt latency is divided into the time from the initial request to the start of the interrupt software routine and the length of time to complete the interrupt software routine handling the interrupt.

You can measure interrupt latency by using two logic analyzer trigger timers. As shown in Figure 18, the first timer, in states 2 and 3, measures the length of time from when the hardware requests interrupt service to the start of the interrupt software routine. The second timer, in states 3 and 4, measures the length of time it takes for the software interrupt routine to complete. The logic analyzer uses two timers, two comparators, one edge detector and four trigger states to implement this PowerTrigger setup.

In Figure 19, the first timer measured 3.124 µs from the time the hardware requested interrupt service to the start of the interrupt service routine. The second timer measured 5.688 µs for the service routine to complete serving the interrupt.
Examples of interrupt errors are: interrupts not being serviced because higher priority interrupts or non-maskable interrupts are inhibiting lower priority interrupts from being serviced, incorrect interrupt priorities assigned to the interrupting tasks, interrupt service routine with non-reentrant interrupt code that responds a second interrupt before it completes servicing the first interrupt, interrupts occurring too fast, and interrupt software routines taking too long to complete their operation.

Summary
The logic analyzer triggering on and capturing digital system faults speeds up the debugging process and helps in verifying system operation. A logic analyzer with a broad set of trigger functions and flexible trigger configurations will help you capture elusive digital and embedded software faults that threaten your development schedule without burying you in useless, time-consuming data.
Providing turn-key solutions to meet your individual testing needs.

Designers face a wide variety of challenging test solutions as they bring new products to market. To keep pace with the complex and shorter design cycles, engineers (designers) need to optimize their testing. In order for you to be successful in these challenging circumstances, Tektronix offers a wide suite of test tools ranging from a broad variety of specialized oscilloscopes to Signal Sources and Real-time Spectrum Analyzers.

Contact Tektronix:

ASEAN / Australasia / Pakistan (60) 6356 3900
Austria +43 2236 8982 262
Belgium +32 (2) 715 89 70
Brazil & South America +55 (11) 3741-8300
Canada 1 (900) 681-5625
Central Europe & Greece +43 2236 8582 301
Denmark +45 44 850 700
Finland +358 (9) 4783 400
France & North Africa +33 (3) 1 69 86 80 34
France +49 (221) 94 77 400
Hong Kong (852) 2585-6688
India (91) 80-22275577
Italy +39 02 25086 1
Japan 81 (3) 674-3010
Mexico, Central America & Caribbean 52 (55) 5666-333
The Netherlands +31 (2) 23 569 555
Norway +47 22 07 07 00
People's Republic of China 86 (10) 6235 1230
Poland +48 (2) 22 521 53 40
Republic of Korea 82 (2) 529-5299
Russia, CIS & The Baltics +358 (9) 4783 400
South Africa +27 11 254 8300
Spain (+34) 901 988 054
Sweden +46 8 477 6503/4
Taiwan 886 (2) 2722-9622
United Kingdom & Eire +44 (0) 1344 392400
USA 1 (800) 426-2200
USA (Export Sales) 1 (503) 627-1916
For other areas contact Tektronix, Inc. at: 1 (503) 627-7111
Last Update August 13, 2004

For Further Information
Tektronix maintains a comprehensive, constantly expanding collection of application notes, technical briefs and other resources to help engineers working on the cutting edge of technology. Please visit www.tektronix.com

Copyright © 2004, Tektronix, Inc. All rights reserved. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specification and price changes privileges reserved. TEKTRONIX and TEK are registered trademarks of Tektronix, Inc. All other trade names referenced are the service marks, trademarks or registered trademarks of their respective companies.

Tektronix
Enabling Innovation