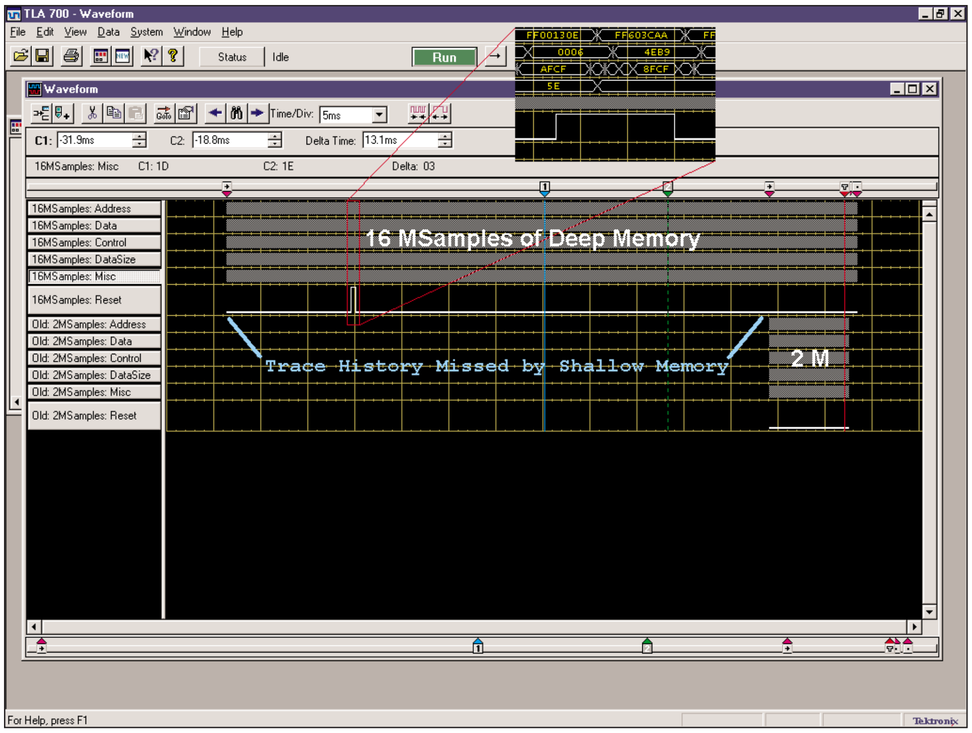
Tektronix

Application Note

Using Deep Memory To Find The Cause Of Elusive Problems



The TLA700 with the TLA7P4 can capture 16 Msamples of data, enabling the user to analyze the data around a glitch that would not be captured by shallow memory.

Introduction

The frequency of an anomaly in a hardware or embedded software system can vary from once every bus cycle to once every million bus cycles or more. The ability to capture anomalies in a digital stream of data is enhanced significantly when a logic analyzer with "deep memory" is used. This application note will show how deep memory can be used effectively to debug hardware and embedded software errors. Problems such as memory leaks, stack overruns, and hardware glitches will be used as examples to demonstrate the usefulness of deep memory. Additionally, this application note will discuss how to select a logic analyzer with deep memory based on features such as memory acceleration that affect the speed of operation, increase productivity, and prevent the "swallow and wallow" phenomenon.[1]

Why Do I Need Deep Memory?

Figure 1 shows a simplified view of how a logic analyzer captures data. The logic analyzer is connected to a data source and is set to continuously collect data into a circular buffer. When the buffer fills, new data overwrites the oldest data in the buffer. If you stop the data capture at any point, you have a "window" of data that you can scan to look back and get a picture of what happened up to that point. How far you can look back in time is determined by how much memory you have in the circular buffer – i.e., how deep is your memory. Earlier logic analyzers functioned well with 512 Ksamples of memory per channel. However, the speed and complexity of today's designs demands troubleshooting and analysis tools with ever increasing power. To answer that need, the Tektronix TLA7P2/4 and TL7N1/2/3/4 deep memory modules for the TLA700 series logic analyzers provide up to 16 Msamples of memory per channel.

Typical Problems That Can Be Solved With Deep Memory

Real-time software problems. Real-time software problems are difficult to debug because they only occur when the system is running "at speed." In these instances, debug monitors fail to provide visibility because they do not have real-time trace capability. Emulators can often help, but sometimes lack the triggering capability or the acquisition memory depth to find the problem. Logic analyzers with deep memory allow users to perform "real-time trace" on large amounts of historical data to identify the problem. Real-time trace records the activity of the program without stopping execution. Debug monitors and emulators only display the current status of the program (at the time it is stopped), not how you got

1 "Swallow and wallow" is the term that some people use to describe acquiring vast amounts of data (swallow) and then "wallowing" around in it aimlessly trying to find what you need. As we describe here, sophisticated data handling techniques can eliminate the confusion and help you quickly find the exact data you need.

there. In real-time systems, you often cannot stop the program while data is coming in without losing a significant amount of information. Thus, real-time trace is critical for debugging these types of routines. The deeper the trace memory the better.

Crash problems. Embedded systems differ from computer applications in that they generally do not have protection from a stray program crashing the entire system. Computer operating systems have many schemes for isolating the system from a misbehaving application – embedded systems often do not. Thus, when your embedded software system crashes, it frequently takes the whole system down, losing any information that may help determine the cause. Logic analyzers can provide the history to quickly determine the cause of the crash. Here again, the deeper the memory the more data you have to analyze to find the problem.

Deeper memory also means that the source of the crash can be further away, or "decoupled," from the actual crash. As embedded application software complexity increases, the decoupling of cause (problem) and effect (crash) can greatly increase.

Memory leaks. A memory leak is an error in a program's dynamic memory allocation logic that causes it to fail to free up memory that is no longer used, leading to eventual collapse due to memory exhaustion. These leaks often caused immediate crashes in older designs with small fixed-size address spaces. With the increasing amount of memory available in systems today, it may take a longer amount of time for the crash to occur and this makes it more difficult to isolate the fault. However, deep memory provides a trace buffer large enough to accommodate the needs of modern designs.

Coupling deep analyzer trace memory with conditional storage with context capability can greatly extend the time window acquired by the analyzer. For example, acquiring just the memory allocation/deallocation routines along with the context just prior to and after the routine can limit the analyzer acquisition to just the areas of interest and greatly extend the capture window. Contextual storage provides a means of having the analyzer automatically capture windows of activity around an event(s) of interest.

Hardware glitch and timing. A glitch or timing error can occur when the inputs of a circuit change, causing the outputs to change to some random value for a brief time before they settle down to the correct value. If another circuit inspects the output at the wrong time and reads the random value, the results can be wrong and very hard to debug. If the logic analyzer has deep memory and glitch storage, the user has the ability to trigger on the symptom and still acquire the glitch that occurred much earlier, causing the eventual failure.

Stack overruns. Stack overruns occur when a program attempts to push more information onto the stack than it can hold. The maximum size of a stack is set first by the size of numbers the relevant register can hold; second by the initial value of the stack pointer. If a logic analyzer does not have deep memory, it is difficult to trace historical stack pointer cycles to capture overrun data.

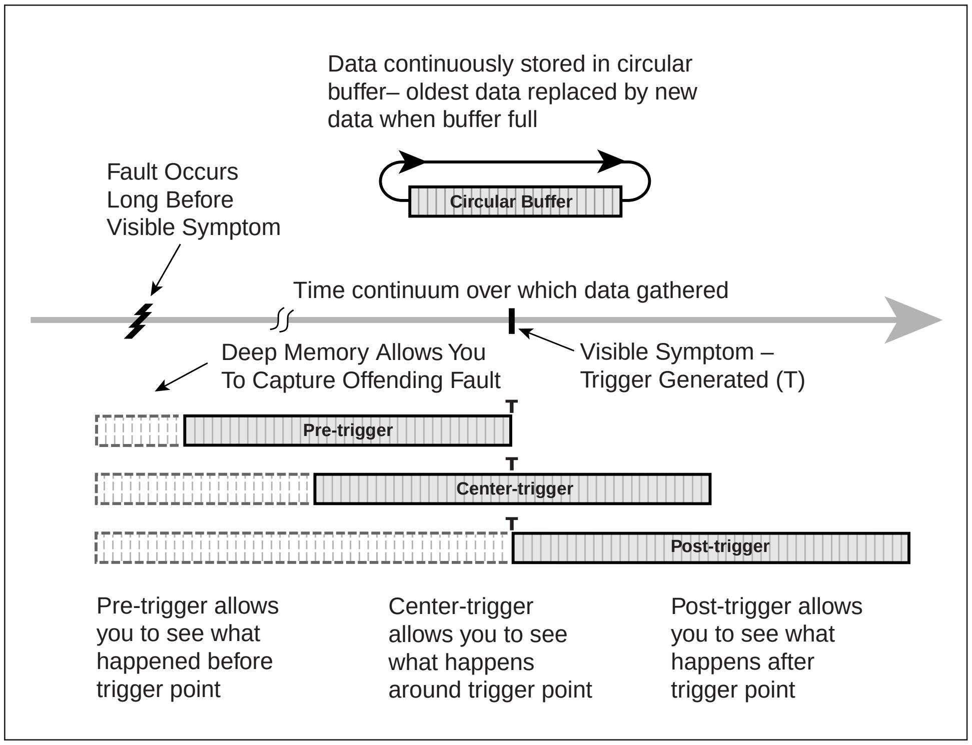

Data continuously stored in circular buffer– oldest data replaced by new data when buffer full
Circular Buffer
Fault Occurs Long Before Visible Symptom
Time continuum over which data gathered
Deep Memory Allows You To Capture Offending Fault
Visible Symptom – Trigger Generated (T)
T
Pre-trigger
T
Center-trigger
T
Post-trigger
Pre-trigger allows you to see what happened before trigger point
Center-trigger allows you to see what happens around trigger point
Post-trigger allows you to see what happens after trigger point

Figure 1. How a logic analyzer captures data.

What Should I Look For In A Logic Analyzer?

You have a lot of choices and every manufacturer will tell you their's is the best. However, there are three primary selection parameters that should be considered when evaluating a logic analyzer with deep memory to avoid the problem of "swallow and wallow." Here's a summary of the features you should look for in choosing a logic analyzer.

Usable memory. When choosing a logic analyzer, a key consideration should be how the logic analyzer manages the large amount of data acquired. Hardware acceleration is an approach that greatly improves the manageability of this large amount of data. Rather than have the mainframe controller's CPU process this data, a more efficient method is to use special hardware capabilities that assist the controller for key operations such as:

- **Waveform display (Zoom and Scroll).** Hardware acceleration quickly provides data so that the waveform display can be drawn in seconds rather than minutes. For example, in a Waveform Window with data from a TLA7N4 (4 Msamples), the time/division setting can be changed and the display updated in 3 to 5 seconds; for a TLA7P4 (16 Msamples), it's 7 seconds.
- **Search.** Hardware acceleration enables the logic analyzer to quickly search the acquired data to find an anomaly. For example, in a Listing Window with data from a TLA7N4 (4 Msamples), searching through the entire acquisition memory for a hexadecimal value in a 32-bit group totaled 2 seconds; for a TLA7P4 (16 Msamples), it's 7 seconds.

Timestamp. Timestamp is a tool that significantly increases the useability of deep memory. Logic analyzers with this capability store a separate timestamp with each data sample.

- **Elapsed Time Between Samples.** One use of timestamp information is to indicate the elapsed time between samples or the total time from the beginning of the acquisition or trigger.

  When choosing a logic analyzer with deep memory, it's important to understand the autonomy of timestamp memory from acquisition memory. When timestamp memory is separate from the acquisition memory, it's easier for the logic analyzer to maintain time-correlation between samples and show time between samples which is useful with data qualifications.
- **Data Correlation.** A second use of the timestamp information is to time-correlate data between different acquisition modules. If a common reference point such as the start of an acquisition or a system trigger can be established between the modules, the data between the modules can be accurately correlated. We all know how important data correlation is when looking at mixed analog and digital signals. But viewing logic analysis data acquired from multiple modules connected to different bus structures (a microprocessor and a peripheral bus such as PCI or RAMBus™ for example) running at different rates can actually present an even greater challenge. Every timestamp counter is, of course, driven by a clock source. Every clock source drifts relative to its designed center frequency. As logic analyzer memory depths increase, the time covered by the acquisition window starts to get long enough that you can see significant timestamp errors between acquisition modules.

  For example, suppose there are two logic analyzer modules that have 1 Msamples of memory depth. Each of these modules uses an independent clock source for its timestamp counter. Let's assume that these cards use a 100 MHz oscillator to run the timestamp counter and that the oscillator has 100 ppm accuracy. If one logic analyzer's clock source is even slightly fast and the other is only marginally slower, by the time we look at the 1 millionth sample we can see correlation errors of up

PACQMEM (Packed ACQuisition MEMory) for the TLA 700 Series

Tektronix has a large library of support software for the TLA700 Series. This software is complementary and available from your local Tektronix Account Manager. One such package that is available is called PACQMEM or Packed ACQusition MEMory. Oftentimes, an application (e.g., video, radar, disk drives, serial communications, etc.) would benefit from trading channels for memory depth. Tektronix offers a special support package for the TLA700 Series called PACQMEM. PACQMEM comes in a wide variety of channel and memory configurations. Depending on the width of the TLA700 Series logic analyzer module, channels can be traded for memory at up to a 16-to-1 ratio. This means that a single TLA7P4 136 channel logic analyzer module with 16 Msamples could provide 1/16 or 8 channels at 16X the memory depth for a total of 256 Msamples. A merged set of three 136 channel logic analyzer modules could provide 24 channels at 256 Msamples.

For further details on PACQMEM, please contact your local Tektronix Account Manager for further details or contact the Tektronix Customer and Sales Support Center (Inside US: 1-800-835-9433, ext. 2400, Outside US: 503-627-2400 or send e-mail to tm_app_supp@tek.com).

to ±100 samples. This is a problem common with older logic analyzer architectures that often require rather elaborate workarounds in an attempt to compensate for this problem.

The TLA 700 Series is based on a modern architecture where all of the logic analyzer modules are automatically phase-locked to the same clock source in the TLA700 Series mainframe. This means that every module's timestamp counter will remain in perfect alignment, and regardless of the memory depth of the module, no module-to-module timestamp drift can occur. This will be true no matter how deep Tektronix makes memory on the logic analyzer modules – 1 M, 4 M, 16 M sample or beyond, it doesn't matter.

Transitional storage. Deep memory applications can often be classified into two categories: externally clocked (i.e., synchronous) or internally clocked (i.e., asynchronous).

- **Synchronous/Externally Clocked.** Oftentimes, the raw clock on a target system is used to acquire data; however, large amounts of redundant data are often stored. With transitional storage, the TLA700 Series can be configured to acquire data only when a specific channel group has a data change.

  For example, if acquiring data from a target system where only one in four samples contains data of interest, a logic analyzer module with 1 M depth and transitional storage can effectively store the same amount of data as a logic analyzer module with 4 M depth that doesn't have transitional storage.

- **Asynchronously/internally clocked.** Sampling data asynchronously with a logic analyzer isn't significantly different from doing so with an oscilloscope. In both cases, the data should be oversampled to ensure faithful data reproduction.

  With a logic analyzer, one should strive to oversample by at least 5X the fastest data rate in the target system. In the resulting acquired data, however, it's often possible that four out of every five samples shows the same or unchanging data. Using a logic analyzer with transitional storage, however, only the data that changed is stored. Each stored sample is timestamped to ensure that the data is accurately displayed, thereby preserving the time relationship.

  For example, assuming a 5X oversample rate, a logic analyzer module with 1 M depth and transitional storage can effectively store the same amount of data as a logic analyzer module with 5 M depth that doesn't have transitional storage.

Whether acquiring synchronously or asynchronously, all TLA700 Series logic analyzer modules have transitional storage and separate timestamp memory, i.e., they don't trade memory depth for timestamp.

Conclusion

Deep memory allows you to analyze and troubleshoot even the most challenging problems found in today's designs. The Tektronix TLA700 series logic analyzers have met your needs in the past. With the addition of the new deep memory modules and expanded mainframes they meet the demands of your current designs. And with their modular design, they'll be there to meet your needs in the future as well.

Tektronix

5/99 TD/XBS 52W–13153–1