Embedded Instrumentation Using XC9500 CPLDs

XAPP076 January, 1997 (Version 1.0) Application Note

Summary

This application note shows how to build embedded test instruments into XC9500 CPLDs.

Xilinx Family

XC9500

Introduction

Systems that use embedded remote diagnostic techniques can identify and isolate potential problems even before field personnel are assigned. This reduces equipment down time and gives field personnel the additional information that saves maintenance effort and money. Now, using the advanced features of the XC9500 CPLD family, designers can easily build advanced remote diagnostic capability into any system and achieve these remarkable benefits.

The XC9500 family has JTAG boundary scan capability built in, to quickly and easily perform a functional device or system test. However, in some cases it is beneficial to design-in other, more precise, diagnostic capabilities as well. This application note describes how to apply unused device resources, or even automatically and remotely reprogram XC9500 devices in the field, to create embedded signature analyzers, logic analyzers, and programmable test points to precisely pinpoint system failures.

Creating a Signature Analyzer

Various nodes (outputs or signal points) in a system will exhibit a repeatable, measurable pattern (signature) when stimulated appropriately. Therefore, failures can be diagnosed by comparing a measured signature with its expected value. Signature analysis identifies a set of critical “test points” in a system and then captures an encoded binary pattern using a polynomial encoding scheme similar to that of cyclic Linear Feedback Shift Registers (LFSRs). The circuitry consists of a shift register with XOR functions embedded at key positions which generate distinctive signature patterns. Figure 1 shows a typical schematic for a signature analyzer.

XC9500 CPLD Signature Analysis Support

Xilinx high performance CPLDs are frequently used in memory and system controllers. These controllers are centrally located within a system, and multiprocessing architectures may have several CPLDs on different cards, making them a natural place to embed diagnostic circuitry.

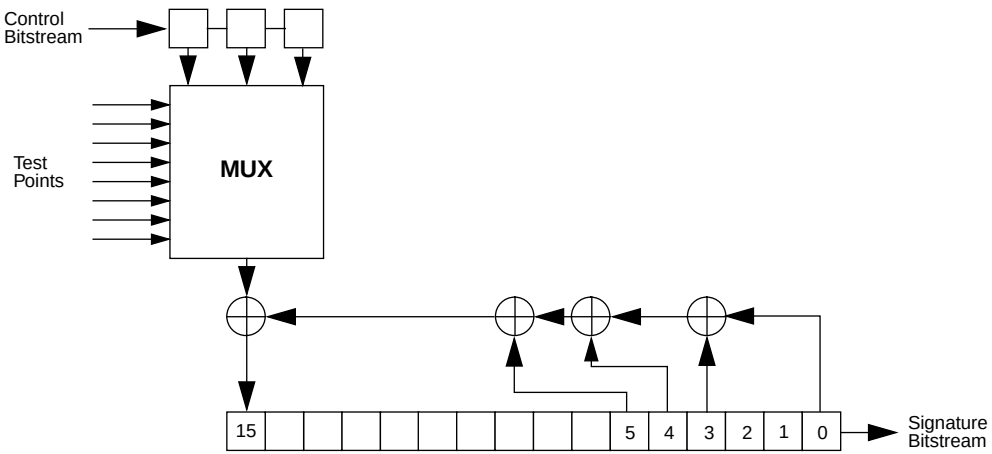

Figure 1: Signature Analysis Using a Cyclic Linear Feedback Shift Register

A typical embedded diagnostic strategy is shown below:

- Verify microprocessor operation at initialization by capturing signatures while transitioning out of reset.
- Verify more signatures as the processor steps through it's bootstrap ROM.
- Capture signatures as the processor tests the memory and subsequent I/O operations.

Figure 2 shows how such a configuration could be built.

Dedicated XC9500 CPLDs can easily implement any or all of the system shown in Figure 2. However, it is also possible to build the signature analyzer by reprogramming unused parts of the system.

For example, the DRAM controller CPLD may not be involved in the system diagnostics while the processor is booting from ROM. In this case, the DRAM controller logic can be erased from the CPLD and the signature analyzer can be constructed from it's logic resources. The DRAM controller is reprogrammed after the diagnostic routine is finished. Then, while the DRAM is being tested, unused data communication's or I/O circuitry can be reprogrammed to act as the signature analyzer. In this way, different system pieces can be reused or adapted to suit the diagnostic needs.

The ability of XC9500 devices to retain pinouts after making design changes, and their ability to perform a minimum of 10,000 program/erase cycles, makes embedded diagnostics and reprogrammability possible.

Signature Analysis Test Strategy Overview

The signature analyzer uses a multiplexed shift register which must be initialized with a pattern other than all zeros (which would make it fail). Therefore, the ability to initialize the register to a known pattern is necessary.

The test points shown in Figure 1 are selected with a three bit serial register and a multiplexer. The signatures must be unique, and each test point must have an identical starting point.

Before signature testing occurs, capture known good signatures and store them for future comparison using the procedure outlined below:

1. Aim the multiplexer at a chosen test point.
2. Initialize the Linear Feedback Shift Register (LFSR).
3. Enable the LFSR.
4. Enable the device under test (DUT).
5. Count operation cycles of DUT until complete.
6. Disable the LFSR and DUT.
7. Read back signature.
8. Compare this signature to the known good signature.

Managing the sequence of captured patterns is important. Once a board has passed all signature tests, the scope of the diagnostic can be increased. For example, analyzing data communication circuits with selected loopback tests. Additional tests may also be performed using the signature method as shown in Figure 2.

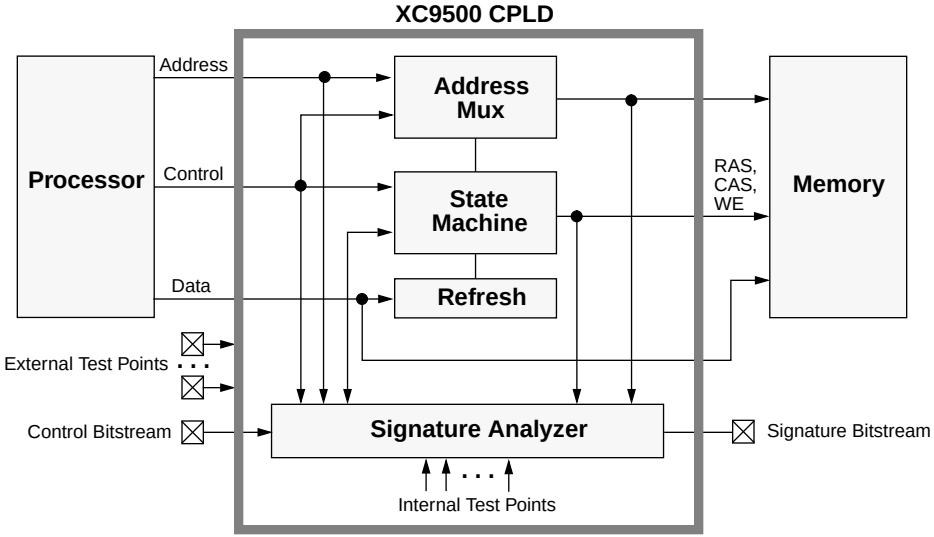

Figure 2: Embedded Signature Analyzer

Practical Considerations

Use the following guidelines when designing signature analysis experiments:

- Synchronous behavior is the most repeatable and the most desirable to use for signature analysis.
- Selection of test points should be done so their global influence is a factor for selection. Any processor status/ control lines which enable data transfers, generate interrupts or make bus requests are candidates.
- Most signatures are unique. However, there is a remote possibility of obtaining aliases. Check for aliases during the initial capture of base-line behavior.
- Table 1 supplies a set of feedback equations for building LFSRs which are the basic components for signature analyzers. Data entry can be exclusive ORed with the feedback input variable (n in this case).
- Avoid initializing the signature register with all 0s.
- Supplement signature analysis with other diagnostics such as memory tests, disk CRC diagnostics, and JTAG boundary scan.

Table 1: LSFR Feedback Equations

n	XOR Feedback Equation
2	X2 = X1 ⊕ X0
3	X3 = X1 ⊕ X0
4	X4 = X1 ⊕ X0
5	X5 = X2 ⊕ X0
6	X6 = X1 ⊕ X0
7	X7 = X3 ⊕ X0
8	X8 = X4 ⊕ X3 ⊕ X2 ⊕ X0
12	X12 = X6 ⊕ X4 ⊕ X1 ⊕ X0
16	X16 = X5 ⊕ X4 ⊕ X3 ⊕ X0
20	X20 = X3 ⊕ X0
24	X24 = X7 ⊕ X2 ⊕ X1 ⊕ X0
28	X28 = X3 ⊕ X0
32	X30 = X22 ⊕ X2 ⊕ X1 ⊕ X0

Note: n equals the number of shift register taps

Creating a Logic Analyzer

An embedded logic analyzer can capture and store a complete picture of system functionality at any number of simultaneous probe points. Figure 3 shows a typical implementation.

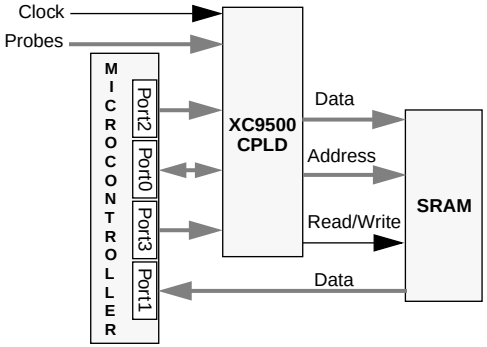

Figure 3: Three Chip Logic Analyzer

The microcontroller enables the logic analyzer and performs housekeeping activities as well as functional testing of the SRAM and CPLD connections. The CPLD drives the data to the SRAM, manages the SRAM addresses, generates memory strobes, and implements breakpoint logic to permit pre-triggering on patterns in the probe data. The SRAM stores the collected data in consecutive locations.

Figure 4 expands the detail for each block of the CPLD. 16-bit data is captured and passed through an output multiplexer to the SRAM data input port. A 16-bit mask register and a 16-Bit enable register combine with the 16-bit probe data to generate an equality compare for triggering the analyzer. The mask register provides the trigger value while the enable register allows for “don’t care” bits.

The control unit automatically delivers memory read and write strobes when the analyzer is enabled by the microcontroller. The captured probe data is stored in the SRAM automatically. When the probe data pattern matches the mask register, a counter inside the control unit is enabled and decrements toward zero on each successive memory transfer. At zero, the control unit disables the storage process.

The counter shown in Figure 4, drives the SRAM address lines and increments on each transfer. The bus (shown in the lower half of Figure 4) permits the microcontroller to load the address counter and pass data through the data multiplexer. The data is read back through the microcontroller bus (not shown).

The microcontroller can enable, disable, and read back SRAM data which is forwarded to a personal computer or workstation for handling the housekeeping and display. The design can also be modified to perform as a simple digital oscilloscope by adding an analog to digital converter where the probes enter the CPLD.

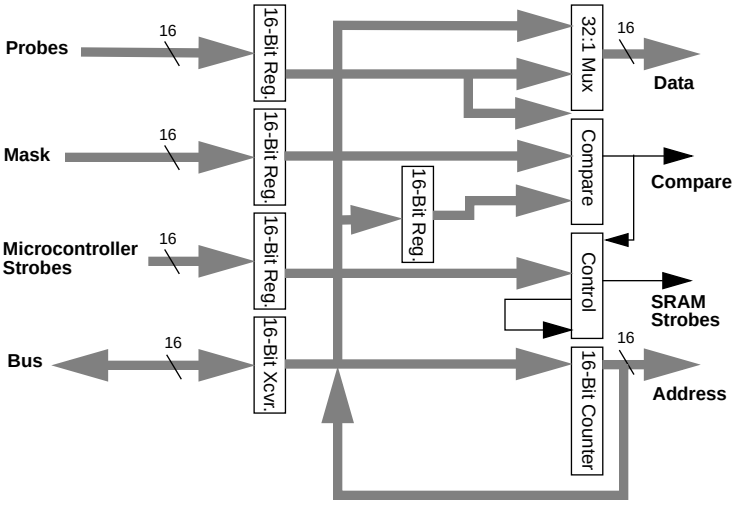

Figure 4: XC9500 Logic Analyzer Implementation

Creating Checkpoint Probes

Because of the small physical size of the pins used in high pin count packages, it is often difficult to attach oscilloscope or logic analyzer probes to a CPLD. In addition, its a good idea to not use physical probes due to the possibility of pin damage. By incrementally connecting a series of internal CPLD test points to dedicated, permanent probe points on the PC board, designers and technicians can access any number of internal nodes by using only one, or several, dedicated probe test points.

Figure 5 shows a CPLD with two dedicated “observation” output pins used for probing internal nodes. These pins can be connected to permanent test points that are soldered to the PC board allowing easy and reliable access for oscilloscopes or other test equipment.

XC9500 designs can be re-compiled to connect any internal node to the dedicated observation pins as needed.

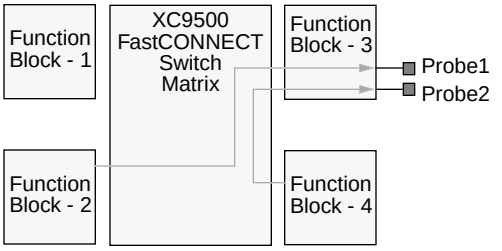

Figure 5: Creating Programmable Checkpoint Probes

Conclusion

XC9500 CPLDs provide the advanced technology for building effective yet inexpensive embedded test circuitry in a wide range of microprocessor systems. This capability is not limited to microprocessor diagnostics, and may be easily expanded to include a wider range of digital systems. With reliable pin-locking and a guaranteed minimum of 10,000 program/erase cycles, many new possibilities exist for creating systems with unsurpassed reliability and maintainability

References

1. Logic Design Principles (with emphasis on Testable Semicustom Circuits), E.J. McCluskey, Prentice-Hall, 1986
2. Design of Testable Logic Circuits, R.G. Bennets, Addison-Wesley, 1984
3. Digital Design (Principles and Practices), J. Wakerly, Prentice-Hall, 1994