EMBEDDED TEST FOR A NEW MEMORY-CARD ARCHITECTURE

David Resnick, Cray Inc., Chippewa Falls, WI

Requirements for a new Cray Inc. computer system mean that the system's memory cards have high-speed SerDes interfaces and multiple memory controllers in a semicustom IC close to the memory chips. As a result, the functionality of a card is much greater than is current practice, and the cards can not be connected to existing memory testers. An embedded test system is designed for the card that can test all aspects of the card including the SerDes paths, a large L3 cache, the memory controllers, and the memory parts mounted on the cards, with testing driven from a JTAG port. The test capability is based in a microcoded controller, with data packets being generated and checked rather than directly controlling the logic being exercised.

Some of the test implementation discussed here is patent pending.

Introduction

New computer system implementations and the still rapidly increasing performance requirements for those systems result in needs for memory bandwidths that are greater than the memory bandwidth improvements that have been 'standard' as new memory ICs are introduced. This is seen in the greater and greater memory latencies in current systems (measured in the number of processor clocks), in the number of memory channels that connect a processor to memory, and in other ways like forcing larger and larger caches into new processor implementations.

A few years ago there was the promise that Direct Rambus™ memory parts would provide much higher bandwidths; but the technology has not succeeded in the marketplace.

The memory manufacturers and some of the system houses, through a JEDEC memory standards group, have defined a new generation of standard memory parts called *DDR2* for Double Data-Rate, 2nd version. DDR2 memory parts are now becoming available and are expected to become the new DRAM standard memory. These parts will have higher performance (533 and 667 MHz data rates) than current DDR parts (400 MHz) and have some significant new features including reduction in power dissipation. However, the new parts do not provide enough improvement for some users, and in particular for systems that implement shared memory, to allow reasonably simple implementation of memory systems with sufficient bandwidths. Moderately complex or functionally demanding implementations of DDR2 will have issues in getting high performance given the control protocol and the electrical requirements of the memory interface including impedance control, timing skews, noise, etc.

As a result there is a need to explore new memory system architectures and implementations. Multiple issues must be addressed:

- Provide ways to get memory bandwidths in the range of 100s of gigabytes per second on a module while keeping pin counts, net counts, the number of module layers and similar issues within reasonable limits.
- Allow for implementations that support memory part counts in the multiple hundreds on a module while not raising electrical issues from long lines, impedance, noise, and line-loading problems in addition to all the other concerns and engineering issues like making systems work with small timing budgets.
- Provide for the possibility of memory parts of different densities, speed grades, and packaging while keeping the base module the same for all variations.

The memory-bandwidth requirements of each processor, along with pin count limits of IC packaging, and with system requirements that multiple processors share a common memory means that each memory card must have multiple access ports and that each port interface be a multi-gigahertz parallel SerDes (Serializer-Deserializer) implementation. This indicates that an interface IC must be part of the card design to provide the SerDes logic and to provide a crossbar function so that each processor port can get to one of a multiple set of memory controllers and associated memory sets. In addition, the interface must implement a communication protocol for the SerDes function. The protocol logic is responsible for things like managing buffer space, checking that data transfers are error free, enabling automatic recovery in the case of a communication or data error, etc.

0-7803-8580-2/04 $20.00 Copyright 2004 IEEE

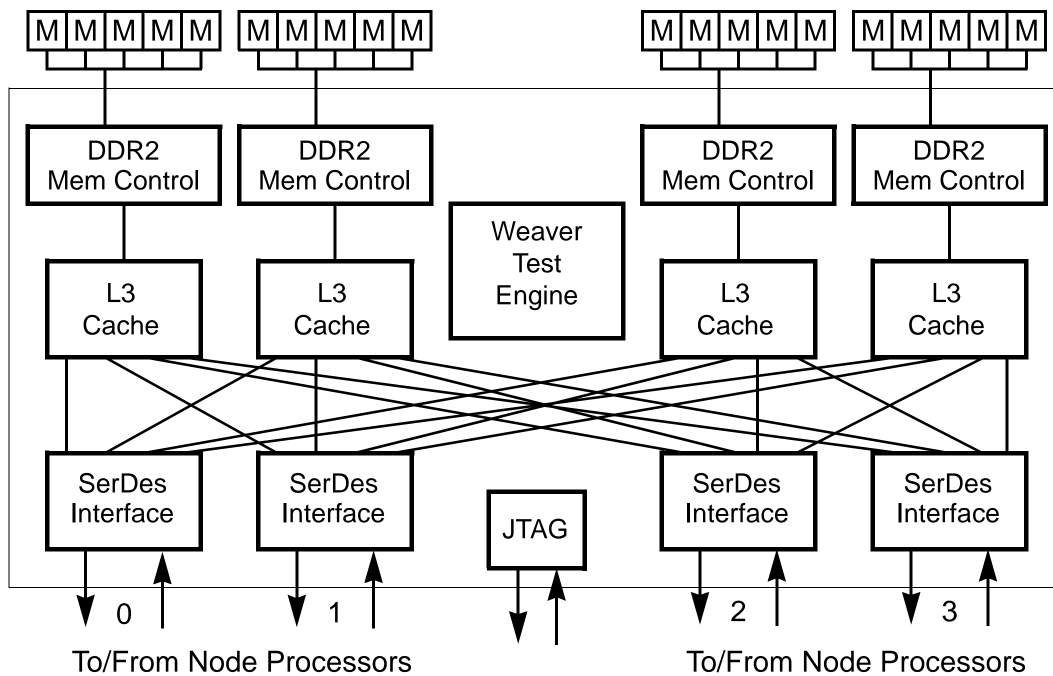


Figure 1 Memory Card Block Diagram

Without the interface IC on the memory card the electrical interface requirements would be difficult to meet. The need for several hundred memory parts on each system module indicate that it would be arduous to keep short lines, control line impedance, lay things out to prevent noise problems, etc., such that the physical and electrical requirements can be met so that a reliable and cost effective system can be built. By putting memory controllers in an IC mounted on the memory card, connections to the memory chips are very short, on the order of a couple of inches, removing most electrical issues at a stroke. This is like current registered DIMMs, but includes the data interface as well.

The remainder of this paper shows the card architecture and discusses the test functions designed into the IC (called Weaver) on each memory card: a microcode test engine, test features of the SerDes logic and its control and management logic, and the memory controller.

Basic Memory Card Architecture

The block diagram of the memory card, emphasizing the Weaver IC, is shown in Figure 1.

All memory parts are 8 bits wide so that each controller is driving a 40-bit-wide data interface. The path is normally 32 data bits, 7 SECDED (Single-Error Correction, Double-Error Detection) bits and an active spare bit. The controllers, and indeed all data paths can support 32/64-bit data widths or 40/80-bit widths for testing. The spare bit can substitute for any of the other 39 data-bit paths and can be inserted into the data path while the system is in operation, in real time.

Each of the processor ports has multiple multi-gigahertz SerDes serial paths at the IO pins of the Weaver. They are grouped so that if one of the signal paths in a processor port fails the SerDes logic first attempts to retransmit the data and can automatically recover the processor link with the particular failing net removed in the presence of permanent failures. This is done with out losing any data.

Each SerDes port is full-duplex so that each port can be moving read and write data at the same time. Data packets are distributed across all SerDes paths in each port, with logic that breaks apart packets as they are transmitted and logic in the receivers that reconstructs packets in order to increase total bandwidth and to minimize path latency.

The JTAG interface is given access to all aspects of the chip by being capable of reading and writing internal parameter, control, and status registers as

well as the contents of the data caches, tag memories, and data buffers. These registers are not only available to the maintenance logic, but are also accessible to the Test Engine for control and status and are memory mapped so that they are available to OS software.

The L3 cache is divided into four blocks on the backside of a fully duplex 4-in/4-out crossbar switch; each cache block is fully independent. Two memory address bits select a path to an individual cache block/memory controller from each port. Each cache supports multiple ways, and has mechanisms so that one processor can not evict data held privately by another processor, but to allow for full data sharing where that is the intention.

Data requests that miss in each cache block are sent to the associated memory controller. Memory requests can also bypass the cache, going directly to the memory controllers, in order to reduce cache pollution. The memory controllers support some special functions like AMOs (atomic memory operations) and also has some special features in support of error recovery. The controller can retry operations when multi-bit errors are detected and also has a spare bit path that can be inserted for any other bit-path (for the spare bit mentioned above) while system operation continues.

Basic Test Implementation

Given that the memory card can not be connected to a normal tester because of the extremely high speed (greater than four gigahertz data rates at the IC's IO pins) of the SerDes interface, the following is being done to enable testing of the IC and the memory chips on the card:

- Add a fairly small microcode memory driving packet-generation logic so that arbitrary test sequences can be generated with the ability to generate tests for all functions of the Weaver IC.
- Add result-checking logic, a copy for each of the four major slices of the Weaver. The checking logic is done so that the microcode sequencer does not have to explicitly test any return result. A data miscompare can halt test execution, if desired, or can simply count errors in a production environment, for example, to see if a successful test was completed. This speeds testing and reduces the size and complexity of the microcode and the test generation logic.
- Rather than have each microcode command be able to generate completely independent test operands, provide a set of test-data source buffers and test-data result buffers. A microcode command can specify a particular source buffer for write functions or a result buffer that is compared with returned data for read functions. This works well because most all tests are highly repetitious with respect to data patterns and saves a good fraction of what would otherwise be a large microcode memory. The test-packet generator simply points to a test data entry—or its complement, which further reduces the size of the test data buffer by half—under control of a small microcode field and the test generator then includes the data as needed by referencing the data-source/test-data buffers.
- Give the test logic the ability to test logic within each of the four logic slices at the same time to reduce total test times and to enable path conflicts, for example, to be fully exercised.
- Use the capability of logic that would be part of normal operations. Cray systems normally have registers that specify timing parameters, routing selections, different operation modes, etc. These memory-mapped registers are normally set at system startup time, but provide additional resources that can ease the total burden on test engineering, given that they can be controlled by the Test Engine.
- And, of course, make sure the other logic is testable and can be driven by the Test Generator, and that the Result Checkers can observe and test any results.

See Figure 2 for a block diagram of the test generation functions and Figure 3 for a block diagram of the test result checkers.

'Fill Data from JTAG' in the figure are the contents of microcode memory needed to run a test and the contents of the Test Data Buffers which become the test data items supplied during a test sequence.

The test logic is able to generate any request sequence that can be generated by processors through the memory-card ports, at full rate. Similarly the test-result checkers have the ability to verify any test results at full bandwidth, thus giving the effect of four copies of a test all running simultaneously. The result sequences can be different for the result checkers as things like memory timing can cause read data timing and ordering to be different, even if the request sequences are identical for all copies of a test.

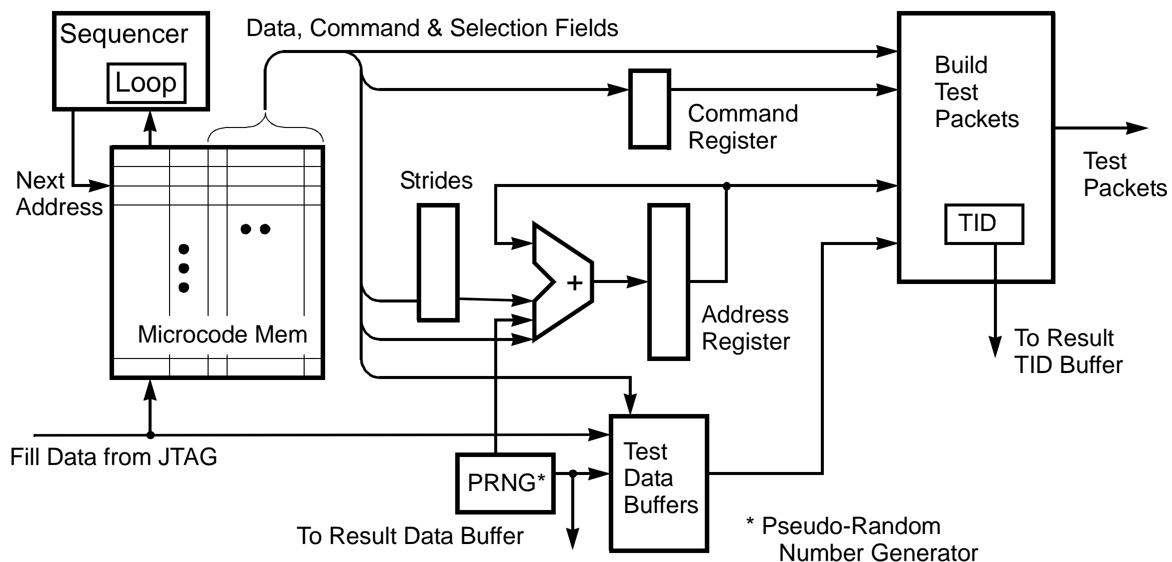
The test logic—including the microcode memory, the test generation logic surrounding it, and the four result checkers—takes about 3.5% of the die area of the Weaver IC.

Functions and Capabilities of the Microcode Test Generator

The basic idea of the microcode implementation is to be able to generate any command and data sequence that a processor can, in addition to being able to exercise and control all test modes and functions.

Processors communicate with memory in the form of data packets. Each data packet has a set of command, address, data descriptors and similar fields and may contain data. Packets can vary in size from consisting only of a header function (e.g.: a function completion signal) or can have from four data bytes to that of a whole cache line, as needed. Rather than having the microcode-driven test logic directly control hardware, it generates data packets. In addition to being able to generate any packet sequence that a processor can, the Test Engine can generate test function packets to exercise specific logic functions or to test memory.

A processor only moves data in 4-byte data words and multiples of that. The Test Engine can supply packets with 4-byte/32-bit or 5-byte/40-bit data items or multiples of those sizes. This is done so that memory, for one example, can have the SECDED bits tested in the same way and can use the same data patterns as the other data bits, using 40-bit data, and can test the SECDED logic, using 32-bit data, when that function is to be exercised. Special test functions are also used to test the internal cache memory and functions, the coherency logic, and other logic and memory blocks including the SerDes functions and the interface protocol.



Note: Connection dots indicate that all data of a path are taken; rounded connections indicate that only a part of the data in a path are taken.

Figure 2 Weaver Test Engine Test Generation Functions

The command field for a packet comes from the microcode memory and is sent to a packet generator. At the same time as the command is coming from the microcode memory, address logic is used by the packet generator, along with other register and microcode data, to generate a specific test packet. If the command is a Write-Memory operation (a 'put' in our parlance), the microcode also supplies a pointer to a Test Data Buffer, and a description of the data shape so that the specified data packet can be generated. If the command is a Read-Memory operation (a 'get'), the packet generated passes a TID (Transaction ID) to the result checkers, so that when data is returned, they can look up and compare the expected data with the data actually read.

The data-source buffer in the Test Generation logic and the test-result buffers in the Test-Result logic are normally loaded before a test sequence starts, through the JTAG port by a system maintenance or test control processor, as part of the sequence that also loads the test code into the microcode memory.

Immediate values from the microcode also can be loaded into the buffers under test control. A pseudo-random number generator can generate test data or test addresses under control of the microcode.

The test generator has five levels of looping control and has several address 'stride' registers. The stride registers hold constants that can be added or subtracted from the current memory reference address so that immediate values are not constantly needed from the microcode memory. This is another way that the test function is kept small while retaining needed functionality, as most tests (March tests are a good example) can be implemented as multilevel loops.

The multiple levels of looping mean that most test sequences can be written and held in a small microcode memory. A full march-pattern test, with multiple modes (the ability to test all memory banks in parallel, test one bank at a time, etc.), with the ability to test multiple memory densities, and with the ability to stop on error or to simply count errors, takes about 100 microcode addresses. The test includes code to look for column errors by running several different data patterns. The microcode memory is 80 bits wide and 512 words deep. Most any test algorithm—march tests, data sensitivity tests, cache associativity testing, etc.—can be executed at the maximum rate supported by the Weaver chip's functional logic.

In addition to having address increment, decrement, and stride capability, the generator also can generate pseudorandom addresses and data. The data can be used either in 32-bit mode, where the memory controller generates and checks data checkbytes, or in 40-bit mode, where all the data bits are under full test control. The pseudorandom addresses have a masking capability so that only a specific range of addresses are generated, if needed. This enables, for example, pseudorandom addressing in a specific bank in the memory chips.

The same test capabilities that are available for memory testing are also available for testing of the internal L3 cache, the associated cache-tag RAMs, and the directory logic. A test can generate a cache request, for example, for an item that is supposedly held by a remote processor, observe that the eviction/return request is generated by the directory logic asking the remote processor for the data item, supplying the needed item (by generating another data packet to the requesting directory), and finally observing a return packet in reply to the original request. Thus the full cache protocol can be tested as well as testing the cache memory blocks.

The result checker receives any returned data items; both test items and internal status and

parameter registers can be compared and observed. See Figure 3.

Not shown in Figure 3 is a data-masking capability, so that specific portions of an item can be ignored in a compare operation. A parameter register can hold several data fields, for example, and the masking capability enables a specific field, or even a specific bit, to be observed and tested as needed.

Test results can also be sent to a small data memory. This memory is essentially a test-point buffer, that enables test sequences to be captured and observed externally. Associated with this test-point logic is a set of selectable data triggers and operation modes, so that a small logic analyzer is actually implemented.

One feature that is not obvious in Figure 3 is that the order of the returned test operands is disconnected from the order that the Test Generator generated them. Each returned data item has TransactionID returned along with the test data. This means that the result checker simply looks up the expected result using the TID, as each data item is returned. This capability is needed because the memory controllers have the capability to shuffle the order of data requests (within limits) to maximize the memory bandwidth. If a memory controller's next request is for a memory bank that is busy, the controller looks further into its request queues to find a request for a non-busy bank. Required memory ordering (reads and writes to the same data items) is controlled by not allowing the controller to reorder references for the same bank. The reordering feature can also be turned off, if necessary, to allow for fully deterministic exercising of the memory parts.

The memory controllers support AMOs (Atomic Memory Operations), having a small cache specifically to support AMO functions. Requests to that cache can cause data to be written to memory in addition to data being returned in response to the request. The data that was written to memory can then be requested from memory and tested. Cache evictions can also result from operations on the L3 cache, generating two results to be observed for each test operand

Test Functions of the SerDes and Interface Logic

There are several functions of the SerDes that are useful for test and troubleshooting in addition to multiple functions that are included specifically for test.

When a SerDes transmission block is initialized, either at power-up or by a master-clear command, each serial port of the block sends out a continuous synchronization sequence. The attached receiving block, when initialized, waits for the syncs and then uses them to generate its received clock. The transmitter is then commanded to send out a pseudo-random data sequence, and is then ready for normal use of the data channel. The receiver checks the data sequence and, if good, sets a flag, and is then also ready for use. If the data check is bad, the receiver goes into a state expecting another attempt, starting with a sync. Status indicating failure is posted and the received data is available for observation, if that is needed.

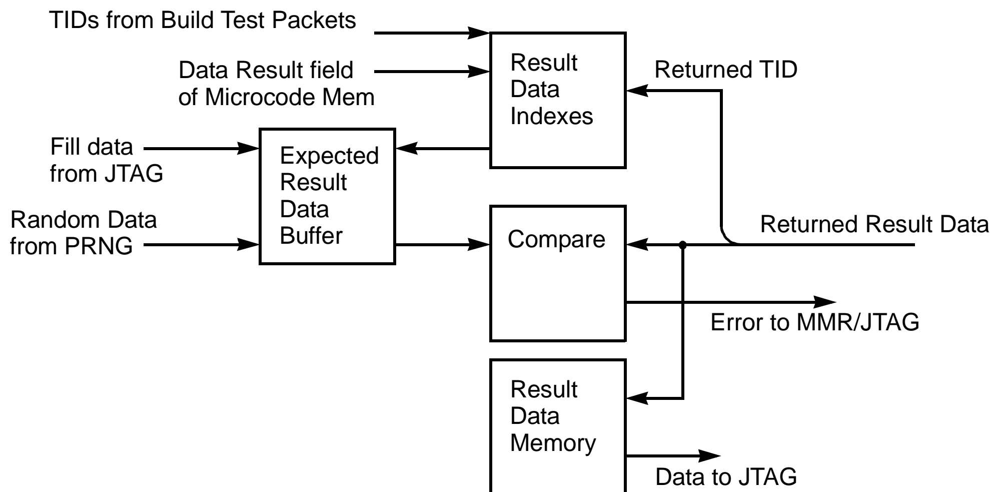


Figure 3 Weaver Test Engine Test Result Functions

The SerDes logic, in a specific test mode, can loop back data from a port's transmitters to the same port's receivers. This capability can be exercised and tested by the Test Engine, and can also be accessed and controlled from the JTAG port as the SerDes function is basic to the operation of the IC. The SerDes logic also has control registers that enable clock and data margins to be set, exercised and tested. Again, these functions can be driven directly from JTAG, by loading and observing data buffers at the start and end of test sequences, or by the microcode Test Engine.

There are also several test functions added to enable testing of the communication protocol. These functions enable easy testing of the logic that generates and checks packet ECC, and the retry and data buffering logic that retransmit data, possibly several times, before a port interface is declared inoperative. The wait time for each port before it starts retransmitting, the number of times to retry, as well as other control values, are some of the parameters that can be controlled for normal operation and can be manipulated by the Test Engine to verify correct operation, to margin a part, or to enable troubleshooting as needed.

Further extending the test capability

The Test Engine, even though it is very capable, is limited by the size of the microcode memory and by the number of different data test words that can be active at one time. While this should be more than adequate for production testing and for most characterization and troubleshooting needs, there are some cases where more capability might be needed. To handle those cases, the test logic is extended slightly so that one memory card can be used to drive test sequences into another card.

A memory-dump function is included in the test design so that the memory can be streamed from each memory block to each respective output port. This means that, if needed, four fully independent and complex streams can be sent to a unit-under-test.

Memory can be loaded through the JTAG interface establishing the test sequence in a 'gold' unit. The data put in memory are data test packets so that the memory card being tested sees a normal data stream, just as if it were in full operation.

Implementing this 'SideCar' test capability requires a more complex test setup than that of a single standalone card, in that card-to-card wiring is needed as well as additional clocks and power. A variation of the SideCar idea is to connect one of the SerDes ports on a card being tested to another port on the same card. This eliminates the need for a second card but requires some short high-speed wiring. In this mode, test operands and data packets are placed into one memory block and the resultant test streams are directed to another block. As the test capability is symmetrical, each port can test another port, for example, wiring port 0 to port 1 and port 2 to port 3.

When this last test mode is used, i.e., connecting ports on a single card, the test capability is not the same as for two connected memory boards because the clocking of the ports on a single card do not have the same flexibility as if two cards are connected together, allowing two different frequencies or multiple voltage margins, for example.

Conclusions

As interfaces, even to memory cards, operate increasingly faster, and as the electrical requirements of memory systems become more complex, putting a control IC directly on a memory card becomes more attractive. The Test Engine discussed here is a fairly small embedded test resource with great generality and good capability. Implementing this test function is not free, however, by requiring additional software—a microcode development system—to be designed and implemented. This effort is reasonable however, given the benefits of the resultant system. Memory bandwidths become largely separate from some of the limitations of current system designs, and other benefits happen almost as side effects. The Test Engine will be used to initialize memory as a system is booted, will be able to capture extended error status, will greatly improve failure isolation and will provide other, similar benefits.

The direct test costs in using this test capability will be much less than that for a full-up high-speed tester, particularly a tester whose capability matches that of the memory cards. As was stated before, there is no existing tester that could be used because of the special SerDes interface.

Test times are the same as for a dedicated memory tester, as the Test Engine can generate tests at full rates, running the same test to each memory controller and checking results from all four result checkers in parallel. While most of the tests are made to be used for production testing, the capability can and will be used in the field. Because of

the generality of the tester, the test capability can be used as new problems arise. The generality has also been very useful because the logic design and its verification proceeded without needing knowledge of specific testing requirements as would be the case for designing memory-BIST test capability into the chip. If a new or modified test requirement is needed, 'simply' load the test into the Test Engine and run it.