TIERED BUILT-IN SELF-TEST (BIST) ARCHITECTURE FOR TESTING DISTRIBUTED MEMORY MODULES

[0001] This application claims the benefit of U.S. Provisional Application Serial No. 60/456,452, entitled "An Apparatus and Method for a Memory Built-in Self-Test Engine," filed March 20, 2003.

TECHNICAL FIELD

[0002] The disclosure relates to electronic devices and, in particular, built-in self-test architectures for use in electronic devices.

BACKGROUND

[0003] Built-in self-test (BIST) units are now commonly incorporated into memory chips and other integrated circuits to test their functionality and reliability. For example, a BIST unit incorporated into a particular memory module operates by writing and reading various data patterns to and from the associated memory module to detect any possible memory faults. By comparing the data written and the data subsequently returned from the memory module, the BIST unit is able to determine whether any memory cell of the memory module is faulty.

[0004] The integrated BIST unit typically generates a variety of predetermined test patterns and asserts or deasserts an output signal based on the results of the memory test. A variety of algorithms may be used for detecting memory faults. For example, test patterns of all zeros, all ones, or a "checkerboard" pattern having alternating zeros and ones may be written throughout the memory cells. Moreover, the data may be written to the cells in any order, such as consecutively in an increasing or decreasing addressing scheme.

[0005] Thus, BIST units are commonly included in many types of integrated circuits that use or otherwise incorporate memory modules and operate according to some predetermined algorithm to verify the functionality of the internal chip circuitry. However, electronic devices typically comprise more than the internal circuitry of a single chip. Normally they are constructed from many integrated circuit chips and many supporting components mounted on a circuit board.

[0006] As the complexity of a typical computing device increases, the number of memory chips and other integrated circuits increases. For example, conventional computing devices typically include a plurality of the memory modules, which are often of different types. The memory modules within a single computing device may include various combinations of random access memory (RAM), read-only memory (ROM), Flash memory, dynamic random access memory (DRAM), and the like. These various types of memory modules often require different testing procedures, and have different bit densities, access speeds, addressing requirements, access protocols, and other particularities. As a result, a typical computing device may have a respective BIST unit for each memory module, and each BIST unit may be particularized to test the associated memory module.

SUMMARY

[0007] In general, the disclosure is directed to a distributed, hierarchical built-in self-test (BIST) architecture for testing the operation of one or more memory modules. As described, the architecture includes three tiers of abstraction: a centralized BIST controller, a set of sequencers, and a set of memory interfaces coupled to memory modules.

[0008] The BIST controller provides centralized, high-level control over the testing of the memory modules. The BIST controller communicates centrally stored and maintained test algorithms to the sequencers for application to the memory modules. The BIST controller communicates the algorithms as a set of generalized commands that conform to a command protocol described herein. Moreover, the command protocol allows algorithms to be generically defined without regard to any timing requirements, physical organization or particular interface characteristics of the memory modules. As a result, a variety of test algorithms may easily be defined and centrally-maintained for distribution throughout an electronic device as needed. Consequently, common test patterns need not be redundantly stored within memory modules.

[0009] The sequencers provide a second level of abstraction. The sequencers are distributed within device blocks that include one or more memory modules. In this manner, each sequencer is associated with one or more memory modules. The sequencers receive the high-level commands from BIST controller. In response to the commands, the sequencers issue a sequence of one or more memory operations to their

respective memory interfaces to carry out the commands. For example, a sequencer may issue commands to sequentially access a range of addresses in response to a single command from the BIST controller. The sequencers report the results of the tests to the BIST controller.

[0010] The sequencers control the application of the operations in accordance with the timing characteristics of their respective memory modules. For example, each of the sequencers controls the application speed of the sequence of operations in accordance with the access speed of the respective memory module. A single sequencer may control the application of the test algorithms to a plurality of memory modules that operate on a common clock domain. Consequently, logic for controlling application timing and sequencing of the test pattern domain is incorporated within the sequencers, and need not distributed within the individual memory modules or maintained by the BIST controller.

[0011] The third tier, the memory interfaces, handles specific interface requirements for each of the memory modules. Each of the memory interfaces may be designed in accordance with the particular signal interface requirements and physical characteristics of the respective one of memory modules. Each memory interface receives memory operations from a controlling sequencer, and translates the memory operations, including associated address and data signals, as needed based on the physical characteristics of the respective memory module. For example, a memory interface may translate addresses supplied by the controlling sequencer based on the rows and columns of the memory module to fill a memory in a row-wise or column-wise fashion. As another example, the memory interface may translate the data to create specific bit patterns, such as a checkerboard pattern or as "striped" rows or columns in which adjacent rows or columns have opposing patterns.

[0012] In one embodiment, a system comprises a built-in self-test (BIST) controller, a sequencer, and a memory interface coupled to a memory module. The BIST controller stores a set of commands defining an algorithm for testing the memory module. The sequencer receives the commands and issues one or more memory operations in accordance with the commands. The memory interface applies the memory operations to the memory module in accordance with physical characteristics of the memory module.

[0013] In another embodiment, a system comprises a plurality of memory modules, a built-in self-test (BIST) controller, and a plurality of sequencers. The BIST controller stores an algorithm for testing the memory modules. The plurality of sequencers are respectively coupled to different subsets of the memory modules, wherein each subset of the memory module is selected to include the memory modules having common clock domains. Each sequencer controls the application of the test algorithm to the respective subset of memory modules in accordance with the common clock domain of that subset of memory modules.

[0014] In another embodiment, a device comprises a first-level built-in, self-test (BIST) means for issuing commands that define a BIST algorithm for a plurality of distributed memory modules having different timing requirements and physical characteristics, and a second-level BIST means for processing the commands to generate sequences of memory operations in accordance with the timing requirements of the memory modules. The device further comprises a third-level BIST means for generating translated address and data signals from the memory operations based on the physical characteristics of the memory modules to apply the BIST algorithm to the distributed memory modules.

[0015] In another embodiment, a method comprises issuing commands from a centralized BIST controller to a sequencer, wherein the commands define a memory test algorithm to be applied to a set of distributed memory modules without regard to physical characteristics or timing requirements of the memory modules, and processing the commands with the sequencer to generate one or more sequences of memory operations in accordance with the timing requirements of the memory modules. The method further comprises applying the memory operations to the distributed memory modules to test the memory modules.

[0016] The techniques described herein may achieve one or more advantages. For example, the techniques may allow a variety of test algorithms to easily be defined and maintained centrally in the form of generalized commands. The generalized commands can be distributed to sequencers located throughout an electronic device for interpretation and application to memory modules. As a result, common test algorithms need not be redundantly stored within the memory modules.

[0017] In addition, the techniques may provide for the simultaneous application of algorithms to different memory modules, which may reduce overall test time and more thoroughly test for inter-memory effects. Moreover, the distributed, hierarchical nature

of the architecture may allow the techniques to be readily applied to existing chip designs.

[0018] The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0019] FIG. 1 is a block diagram illustrating an example electronic device having a distributed, hierarchical built-in self-test (BIST) architecture.

[0020] FIG. 2 is a block diagram illustrating an example embodiment of a BIST controller.

[0021] FIG. 3 is a timing diagram that further illustrates communication between the BIST controller and a set of sequencers for a single command of a generic BIST algorithm.

[0022] FIG. 4 is a block diagram illustrating an example embodiment of a device block.

[0023] FIG. 5 is a block diagram illustrating an example embodiment of a sequencer.

[0024] FIG. 6 is a block diagram illustrating an example embodiment of a memory interface.

[0025] FIG. 7 is a block diagram that illustrates an example embodiment of a data generation unit.

[0026] FIG. 8 is a block diagram illustrating an example data structure of a command issued by the BIST controller.

[0027] FIGS. 9A-9E illustrate exemplary data structures in accordance with the command protocol described herein.

[0028] FIG. 10 is a flowchart illustrating example operation of the distributed, three-tier self-test architecture.

DETAILED DESCRIPTION

[0029] FIG. 1 is a block diagram illustrating an example electronic device 2 having a distributed, hierarchical built-in self-test (BIST) architecture. In particular, electronic device 2 includes a built-in self-test (BIST) controller 4 that provides centralized, high-level control over testing of device blocks 6A through 6N (collectively "device blocks

6"). Each of device blocks 6 includes a sequencer 8, and a set of one or more memory interfaces 10 and one or more respective memory modules 12.

[0030] In general, BIST controller 4 provides and communicates test algorithms to sequencers 8 for application to device blocks 6. BIST controller 4 communicates each of the algorithms to sequencers 8 as a set of commands that conform to a generic and flexible command protocol. Each command specifies an operational code and a set of parameters that define one or more memory operations without regard to the physical characteristics or timing requirements of memory modules 12. In this manner the command protocol allows a variety of test algorithms to easily be defined and distributed throughout electronic device 2. Consequently, BIST controller 4 provides centralized control over the maintenance and distribution of the algorithms. As a result, common test algorithms need not be redundantly stored within device blocks 6.

[0031] Sequencers 8 interpret and execute the test algorithms provided by BIST controller 4. In particular, sequencers 8 receive high-level commands from BIST controller 4 that define a complete BIST algorithm. A single command, for example, may define a particular bit pattern to be written over a range of one or more addresses. In response to the commands, each of sequencers 8 issues one or more sequences of memory operations to their respective memory interfaces 10 to perform the commands. Moreover, sequencers 8 control the application of the operations in accordance with the timing characteristics of their respective memory modules 12. For example, each of sequencers 8 controls the application speed of the sequence of operations in accordance with the access speed of the respective memory module 12. A single sequencer 8, e.g., sequencer 8A, may control the application of the test algorithms to one or more memory modules, e.g., memory modules 12A, that operate on a common clock domain. Memory modules 12 may be grouped and assigned to respective sequencers 8 based on any of a variety of criteria, such as clock domains for each of the memory modules and any pre-existing hierarchy or grouping of the memory modules. Consequently, logic for controlling application timing and sequencing of the test pattern for memory modules 12 operating on a common clock domain may be incorporated within a common sequencer 8, and need not distributed within the individual memory modules.

[0032] Memory interfaces 10 handle specific interface requirements for each of memory modules 12. For example, each of memory interfaces 10 may be designed in accordance with the particular signal interface requirements and physical characteristics

of the respective one of memory modules 12. As a result, each of memory interfaces 10 may be viewed as providing an interface "wrapper" around the particular interface signals, e.g., address, data, and control signals, for each respective memory module 12. In this manner, the BIST architecture of electronic device 2 comprises a three-tier, distributed arrangement including BIST controller 4, sequencers 8, and memory interfaces 10.

[0033] Memory modules 12 may be any type of memory, such as random access memory (RAM), read-only memory (ROM), Flash memory, dynamic random access memory (DRAM), SDRAM, RDRAM, DDR-RAM, combinations thereof, and the like, and the techniques described herein are not limited in this regard. Moreover, electronic device 2 may be any device that incorporates memory modules, such as an embedded computing system, a computer, server, personal digital assistant (PDA), mobile computing device, mobile communication device, digital recording device, network appliance, mobile positioning device, and the like.

[0034] FIG. 2 is a block diagram illustrating an example embodiment of BIST controller 4. In this illustrated embodiment, BIST controller 4 includes an algorithm memory 20 that stores a set of N test algorithms. As described below, each algorithm is defined in accordance with a set of binary commands. In one embodiment, for example, a complete algorithm may be specified by a set of 32-bit commands, where the commands define all necessary parameters for performing one or more sequences of memory operations over address ranges of device blocks 6 (FIG. 1) to test the functionality of memory modules 12.

[0035] User interface 22 invokes algorithm controller 26 in response to external input, such as a control signal from an external testing apparatus. Alternatively, algorithm controller may be automatically invoked upon power-up of electronic device 2. Once invoked, algorithm controller 26 provides an algorithm select signal (ALG_SELECT) to multiplexer 24 to select one of the algorithms stored within algorithm memory 20. Once selected, a stream of binary commands that comprises the selected algorithm is applied to device blocks 6 as command data (CMD_DATA).

[0036] Algorithm controller 26 controls the delivery of the algorithms to device blocks 6 based on acknowledge signals (SEQ_ACK) received from each sequencer 8 of the device blocks. In particular, algorithm controller 26 sequentially delivers each command of the selected algorithm to sequencers 8, and proceeds from one command to

the next upon receiving an acknowledge signal from each of sequencers 8. In this manner, algorithm controller 26 ensures that each sequencer 8 has completed application of a current command to memory modules 12 via memory interfaces 10 before proceeding to the next command. Algorithm controller 26 may be programmatically or statically configured to establish the number of device blocks 6 and, in particular, sequencers 8 that are present within electronic device 2. In addition, algorithm controller 26 may be programmatically configured to apply a given algorithm to one, all, or any combination of memory modules 12 using any combination of device blocks 6.

[0037] In addition to the algorithms stored in algorithm memory 20, user interface 22 may programmably receive algorithms via external input. User interface 22 delivers the received algorithms to multiplexer 24 in a form similar to that of the stored algorithms, i.e., a sequence of binary commands in which each command defines a test within the overall algorithm. In this manner, BIST controller 4 provides a centralized, first tier of the three-tier, distributed self-test architecture.

[0038] FIG. 3 is a timing diagram that further illustrates communication between BIST controller 4 and sequencers 8 for a single command of a generic BIST algorithm. As illustrated, at a time T1, BIST controller 4 asserts the command request (CMD_REQ) interconnects, and communicates a current command of the algorithm on the CMD_DATA interconnects.

[0039] Upon receiving and applying the command, sequencers 8 assert a corresponding SEQ_ACK signal. For example, in the illustrated example, each of sequencers 8 assert respective signals SEQ_ACK[1], SEQ_ACK[2], SEQ_ACK[N] and SEQ_ACK[0] at times T2, T3, T4 and T5, respectively. In response, BIST controller 4 de-asserts the command request (CMD_REQ) signal at a time T6, causing sequencers 8 to de-assert their respective SEQ_ACK signal. BIST controller 4 asserts SEQS_DONE signal when all SEQ_ACK signals have been de-asserted, allowing BIST controller 4 to initiate another command at a time T7.

[0040] FIG. 4 is a block diagram illustrating an example embodiment of a device block, e.g., device block 6A, in more detail. As illustrated, device block 6A includes sequencer 8A, a set of memory interfaces 10A-10C (collectively "memory interfaces 10"), and a set of memory modules 12A-12C (collectively "memory modules 12"). As illustrated in FIG. 4, each of memory interfaces 10 corresponds to a respective memory module 12.

Sequencer 8A may be programmatically and/or statically configured to establish the number of memory interfaces 10 to be controlled, as well as the characteristics, e.g., maximum address, for the largest of memory modules 12.

[0041] FIG. 5 is a block diagram illustrating an example embodiment of a sequencer, e.g., sequencer 8A, in more detail. Sequencer 8A receives high-level commands from BIST controller 4 that collectively define a BIST algorithm. As illustrated in detail below, BIST controller 4 issues the commands in a generic, flexible format, and a single command may define a particular bit pattern to be written over a range of one or more addresses.

[0042] In general, sequencer 8A receives the generic BIST commands, and controls the application of each command as a sequence of one or more memory operations applied to a set of respective memory modules. In the illustrated embodiment, sequencer 8A includes a command parser (CMD PARSER) 30 that receives command data (CMD_DATA) from BIST controller 4. Command parser 30 processes the received command to identify the specified operation, e.g., by identifying an operational code (op-code) specified by the command.

[0043] Based on the specified operation, command parser 30 may extract one or more parameters from the command, and select a corresponding one of command controllers (CMD CONTROLLER) 34A-34N. In other words, each one of command controllers 34 corresponds to a different command that may be specified by CMD_DATA. Command parser 30 invokes the selected command controller 34, and passes the parameters extracted from the received command. Although illustrated separately, command controllers 34 may be combined and/or integrated into a single functional block having logic to perform each of the supported commands.

[0044] In response to each command, the invoked one of command controllers 34 issues a sequence of one or more operations to each memory interface 10. In particular, the invoked one of command controllers 34 sequentially drives the appropriate command control signals (CMD_CTRL_SIGNALS) to carry out each operation of the sequence. The command control signals may include signals to provide a memory address and data to the receiving memory interfaces 10 and to direct the receiving memory interfaces to invert bits, perform read or write operations, invert rows, and the like, as further described below.

[0045] Moreover, command controller 34 controls the application of the operations in accordance with the timing characteristics of their respective memory modules 12. Consequently, logic for controlling application timing and sequencing of the operations for memory modules 12 operating on a common clock domain may be incorporated within a common sequencer 8, and need not distributed within the individual memory modules.

[0046] Sequencer 8A receives data from the tested memory modules 12, e.g., via signals MEM[0]_DOUT throughout MEM[N]_DOUT, and selectively communicates the data back to BIST controller 4 or an external device via multiplexer 37 and data selection signal 39. In this manner, sequencers 8 allow for the analysis of the data to identify any failures.

[0047] Thus, sequencers 8 allow BIST controller 4 to centrally manage the storage and issuance of algorithms using generic and flexible command format. Sequencers 8 receive the generic BIST commands in accordance with the command protocol, and control the application of the commands by generating and issuing sequences of one or more memory operations for application to a set of respective memory modules 12, thereby providing a second tier of the distributed, hierarchical self-test architecture.

[0048] FIG. 6 is a block diagram illustrating an example embodiment of a memory interface 41 that provides the final layer of abstraction of the distributed BIST architecture by handling specific interface requirements for a respective memory module 12.

[0049] In the illustrated embodiment, memory interface 41 includes a layer of multiplexers 45, 46 that isolates the memory module 12 from normal functionality when electronic device 2 is operating in BIST mode. In particular, under normal operating conditions, BIST enable (BIST_EN) is de-asserted, causing multiplexers 45, 46 to select the address / control signals (ADDR/CTRL) and data signals (DATA), e.g., as provided by a programmable processor. When electronic device 2 is operating in BIST mode, however, the BIST enable signal causes multiplexers 45, 46 to select the BIST address / control signals (BIST_ADDR/CTRL) and the test data provided by a respective higher-level sequencer. In this manner, the sequencer controls multiplexers 45, 46 of memory interface 41 to selectively isolate the respective memory module, thereby allowing BIST algorithms to be applied to that memory module.

[0050] Memory interface 41 further includes a data generation unit 44 that receives BIST_DATA signals and default data (DEFAULT_DIN) as provided by the sequencer 8, and generates transformed BIST data signals 49 (BIST_DATA_T) based on control signals (BIST_DATA_GEN_CTRL) provided by the sequencer and the specific physical characteristics of the corresponding memory module 12. More specifically, as described in further detail below, data generation unit 44 generates the exact data (RAM_DIN) applied to the memory inputs during each operation of the algorithm. Data generation unit 44 generates the transformed BIST data based on the physical configuration, e.g., physical row and column configuration, of the respective memory module 12, as well as the bit pattern specified by sequencer 8A, e.g., checkerboard, inverted, and the like.

[0051] For example, sequencer 8 may request a checkerboard bit pattern that requires columns of data within the memory module to alternate between ones and zeros. Different memory modules 12, however, may be arranged with different row and column configurations. As a result, the memory cell for a given address may be located in a different row and column for memory modules 12 having different row-column configurations. For example, a 256 bit memory module 12 may be organized as 128 rows and 2 columns, 32 rows by 8 columns, 16 rows by 16 columns, and the like. As a result, writing a particular pattern throughout this matrix, such as a checkerboard pattern, requires knowledge of the specific row-column configuration of the memory module. Data generation unit 44 processes the data provided by the sequencer 8, and transforms that data, e.g., inverts the data, as needed to ensure the desired bit pattern is correctly written, as described in further detail below.

[0052] Similarly, address generation unit 42 generates the address applied to the memory module 12 based on the addressing requirements specified by sequencer 8A and the physical configuration of the rows and columns of the memory module. For example, in some BIST tests, sequencer 8A directs memory interface 41 to write BIST data in a row (or column) first fashion, i.e., requiring each row (or column) to be completely written before proceeding to the next row (or column). As these requirements are specific to the row and column configuration, address generation unit 42 transforms the address provided by sequencer 8A to an appropriate to traverse an entire row (or column) before proceeding.

[0053] As a result, the sequencer is able to issue operations by sequentially traversing an address space defined by the respective command, limited by the maximum-sized memory module associated with that sequencer. Each memory interface, e.g., memory interface 41, transforms the received address as needed to apply the operation to an appropriate memory cell. For example, for a column first BIST algorithm, the sequencer 8 issues the operations to the addresses in sequence, while address generation unit 42 computationally transforms the received address to access memory cells that fall within along the columns of the memory module 12. In this example, address generation unit 42 transforms the addresses to access the memory module 12 in a column-wise fashion from a lowest-significant column to a highest-significant column based on whether the sequencer 8 is incrementing or decrementing through the address space.

[0054] Comparator 48 that determines whether the data (RAM_DOUT) read from the memory module 12 equals the data last written to that address of the memory module as expected. When the comparison fails, i.e., when the data read from the memory module 12 does not equal the previously written data, comparator 48 asserts the BIST_FAIL signal to indicate that a memory error has been detected.

[0055] In this manner, memory interface 41 processes the sequential memory operations issued by the higher-level sequencer, and transforms the data and addresses provided by the sequencer as needed based on the particular physical characteristics of the memory module, thereby providing a third tier of the distributed, hierarchical self-test architecture. As a result, the sequencer is able to issue operations for complex BIST algorithms without requiring detailed knowledge regarding the physical characteristics and capacities of each memory module 12.

[0056] FIG. 7 is a block diagram that illustrates an example embodiment of data generation unit 44 (FIG. 6). In the illustrated embodiment, data generation unit 44 receives a BIST_DATA signal, a default data signal (DEFAULT_DIN) provided by the sequencer 8. In addition, data generation unit 44 receives a number of control signals in the form of BIST_INVERT_BITS, BIST_INVERT_ROWS, LSB_ROW_ADDR, and BIST_WRITE. Based on the received data and these control signals, data generation unit 44 generates transformed BIST data signals 49 (BIST_DATA_T) based on control signals (BIST_DATA_GEN_CTRL) provided by the sequencer 8 and the specific physical characteristics of the corresponding memory module 12.

[0057] In particular, the sequencer 8 asserts and deasserts the invert bits signal (BIST_INVERT_BITS), the invert rows signal (BIST_INVERT_ROWS), and the invert columns signal (not shown) to specify data patterns, such as solid, checkerboard, horizontal and vertical striped data patterns. If neither the invert bits nor the invert rows signals are asserted, then the BIST_DATA signal passes through XOR gates 52, 54 without modification. As a result, data generation unit 44 generates the transformed data signal 49 (BIST_DATA_T) to fill the memory module 12 with the data dictated from the sequencer 8 without modification.

[0058] If the BIST_INVERT_ROW signal is set then data generation unit 44 inverts the values every time a row is crossed, as indicated by the least significant row address bit (LSB_ROW_ADDR). As a result, AND gate 50 causes XOR gate 52 to invert the values dictated by the sequencer, and thereby invert values written to neighboring rows. Similar functionality may be provided to invert values as columns are crossed. When the BIST_INVERT_BITS signal is asserted, data generation unit 44 automatically inverts the values dictated by the sequencer. This may be useful in inverting values between corresponding cells of the different matrices.

[0059] The DEFAULT_ DIN field establishes a default data value applied to the memory modules during read operations for memory modules 12. Multiplexer 56 selects between the DEFAULT_DIN data and the data generated by XOR gate 54 based on whether a write or a read operation is being performed, as indicated by the BIST_WRITE signal.

[0060] FIG. 8 is a block diagram illustrating an example data structure of a command issued by BIST controller 4. In the illustrated embodiment, command 60 includes a sequencer identifier (ID) 62, and a payload 64. Sequencer ID 62 identifies a sequencer, e.g., sequencer 8A, to which command 60 is being issued, as described in further detail below.

[0061] BIST controller 4 broadcast commands to all sequencers 8, or may issue a command in unicast fashion to a particular one of the sequencers. In particular, BIST controller 4 sets sequencer ID 62 to a unique identifier for one of sequencers 8 to send a unicast command to that sequencer. For a broadcast command, BIST controller 4 sets sequencer ID 62 to a broadcast identifier, such as 0x0.

[0062] Payload 64 of command 60 carries binary data that defines the command itself. In particular, payload 64 includes an operational code (OP CODE) 66 and a set of

parameters 68. In general, OP CODE 66 specifies a particular function to be performed by the receiving sequencers 8. The following table enumerates an exemplary set of operational codes:

TABLE 1

CMD	OP CODE	MESSAGE TYPE	DESCRIPTION
RESET	000	Broadcast or Unicast	Resets the sequencer(s) and selectively enable BIST mode.
EXECUTE	001	Broadcast or Unicast	Apply a defined sequence of one or more memory operations as specified by the parameters the command.
TRIGGER STORED TEST	010	Broadcast or Unicast	Initiate a specified test algorithm stored within a specific sequencer.
TEST MEM	011	Unicast	Used to select a particular memory module within a device block for participating in a test or for failure analysis.
SET ADDRESS	100	Unicast	Sets a specific starting address as well as a maximum address limit for a test algorithm as applied to the memory modules.
SINGLE WORD ADDRESS	101	Broadcast or Unicast	Executes one or more operations on the address specified by the current BIST address of the sequencer. May increment or decrement after access.

[0063] In one embodiment, OP CODE 66 and parameters 68 comprise three bits and twenty-nine bits, respectively, to form a 32-bit command. The format and meaning of parameters 68 is dependent upon the type of command, as specified by OP CODE 66.

[0064] FIG. 9A illustrates an example data structure of parameters 68 for the RESET command. As illustrated, the only relevant bit of parameters 68 is bit 28, which can be asserted or de-asserted to selectively enable and disable BIST mode. When asserted, the receiving sequencers 8 enter a mode for applying test algorithms to test the respective device blocks 6. When de-asserted, the receiving sequencers reset and terminate the current algorithm.

[0065] FIG. 9B illustrates an example data structure of parameters 68 for the EXECUTE command. As illustrated, for the EXECUTE command, parameters 68 directs the receiving sequencers 8 to apply a sequence of memory operations over a range of addresses in accordance with the particular timing characteristics of their respective memory modules 12.

[0066] Once received, by default, sequencers 8 repetitively execute the specified memory operation over the address range of the largest memory module 12 within the respective device block 6. However, if SINGLE ROW (SR) bit is enabled, sequencers 8 execute the defined sequence of memory operations for all columns of the memory module 12 that has the largest column-bit option while maintaining the row address constant.

[0067] When applying the sequence of memory operations, sequencers 8 either increment or decrement through the address range supported by the largest of the memory modules 12 within their respective device blocks 6 based on the state of the ADD INC/DEC bit of parameters 68. If ADD INC/DEC is asserted, for example, sequencers 8 apply the defined memory operations to each address starting from zero and proceeding to the maximum addresses. If ADD INC/DEC is deasserted, however, sequencers 8 apply the defined memory operations to each address starting from the maximum address and decrementing to zero. The DEF DIN field establishes a default data value applied to the memory modules 12 during read operations for the memory modules.

[0068] The rippling row (RR) field directs sequencer 8 to apply the defined memory operations in a column-wise fashion, i.e., by applying the operations to an entire column before proceeding to the next column. In other words, each of sequencers 8 hold the column address constant while applying the memory operations and "rippling" the row address.

[0069] The invert bits (IB) field, invert rows (IR), and invert columns (IC) fields can be used to specify data patterns for testing memory modules 12, such as solid, checkerboard, horizontal and vertical stripes data patterns. More specifically, if BIST controller 4 does not set either of the IR and IC fields, then the receiving sequencers 8 direct the memory interfaces 10 to fill memory modules 12 with the values dictated from the sequencer. If the IR field is set, then the values dictated by the sequencers 8 will have inverted values written to neighboring rows. Similarly, if the IC field is set, then the values dictated by the sequencers 8 will have inverted values written to neighboring columns. As a result, if both the IR field and the IC field are set, values are inverted between each column and between each row to create a checkerboard pattern within the memory modules. Finally, as described above, a given memory module 12 may be configured as more than one matrix. If the invert bits (IB) field is set, then the

memory interfaces 10 automatically invert the values dictated by sequencer 8 between corresponding cells of the different matrices.

[0070] Operations fields (OP1 – OP8) can be used to define a set of operations to be applied to each memory address. For example, each operation field, such as OP1, may comprise two bits. The first bit may be asserted or deasserted to indicate whether the operation is a read or a write. The second bit may be set based on the data to be written, i.e., a zero or a one. The number of operations (NUM OPS) field instructs sequencers 8 as to how many operations have been defined for application to each memory address. In this manner, a single command may be used to generically define a step within an overall BIST algorithm, and each step may define one or more operations to be applied to each address of the memory modules 12 with device blocks 6 of the receiving sequencers 8.

[0071] FIG. 9C illustrates an example data structure of parameters 68 for the TEST MEM command. For this command, parameters 68 include an FA / BIST bit to interpret the TEST MEM command as a failure analysis command and as a BIST command. When set to failure analysis command, the value specified by the MEM ID field is used by the receiving sequencers 8 to select the data output of a specific one of memory modules 12 for failure analysis. When set to a BIST command, the value specified by the MEM ID field is used by the receiving sequencers 8 to select the data output of a specific one of memory interface 10 for participation within a particular test. In this manner, the algorithm can be selectively applied to individual memory modules 12 within a device block 6. The MEM BUS SLICE field is used to indicate which portion of the multiplexed data bus from the memory modules 12 is used for the failure analysis.

[0072] FIG. 9D is an example data structure of parameters 68 for the SET ADDRESS command. For this command, parameters 68 includes and address field (ADDRESS) that sets a specific memory address for application of a BIST step. This may be useful in conjunction with the SINGLE WORD ACCESS command. Parameters 68 also include a limit (LIMIT) field for specifying a maximum address limit for the test algorithm. In one embodiment, the LIMIT field comprises a 2-bit data field for setting the limit to: (1) the maximum address of the largest of the memory modules 12 of the device block 6, (2) the maximum address divided by two, (3) the maximum address divided by four, and (4) the maximum address divided by eight.

[0073] FIG. 9E is an example data structure of parameters 68 for the SINGLE WORD ACCESS command. For this command, parameters 68 includes an enable address change (ENADC) bit that controls whether the receiving sequencers 8 should change their respective current BIST addresses after applying the step. If enabled, the address increment / decrement (ADD INC/DEC) bit controls whether the current BIST address should be incremented or decremented. The invert bits (IB) field, invert row (IR), and invert column (IC) fields can be used to specify data patterns for testing memory modules 12, such as solid, checkerboard, horizontal and vertical stripes data patterns, as described above in reference to the EXECUTE command. The data field (DATA) is used to supply a default value for input data for read operations of the tested memory modules 12.

[0074] Table 2 illustrates an example checkerboard BIST algorithm stored and issued by BIST controller 4 in accordance with the described command protocol. As illustrated, a relatively complex checkerboard memory test algorithm can be described in as few as four commands using the command protocol.

TABLE 2

Op-code	Parameters	Industry Notation	Description
010	01000011100000000000000000	^(W01)	Sequence from 0 to MAX ADDR/write alternating '0', '1'
010	01000011110000000000000000	^(R01)	Sequence from 0 to MAX ADDR/read alternating '0', '1'
010	01000011101000000000000000	^(W10)	Sequence from 0 to MAX ADDR/write alternating '1', '0'
010	01000011111000000000000000	^(R10)	Sequence from 0 to MAX ADDR /read alternating '1', '0'

[0075] Table 3 illustrates an example "Blanket March" BIST algorithm stored and issued by BIST controller 4 in accordance with the described command protocol. As illustrated, this memory test algorithm can be described in as few as six commands using the command protocol. Each of the commands directs the receiving sequencers 8 to issues sequences of memory operations that traverse the entire memory space in a defined direction. Moreover, several of the commands direct the sequencers to apply multiple memory operations to each address within the available memory space. In this manner, complex BIST algorithms can be easily distributed throughout and applied by the constituent components of the hierarchical self-test architecture.

Table 3

Op-code	Parameters	Industry Notation	Description
010	01010000000000000000000000	Din=0 ^(W0)	Sequence from 0 to MAX ADDR/write '0'
010	01110000011100111010000000	Din=0^(R0,W1,R1,W1)	Sequence from 0 to MAX ADDR,rippling-row/read '0', write '1', read '1', write '1'
010	01111000011110010000000000	Din=0^(R1,W0,R0,W0)	Sequence from 0 to MAX ADDR,rippling-row/read '1', write '0', read '0', write '0'
010	00110000011100111010000000	Din=0v(R0,W1,R1,W1)	Sequence from MAX ADDR to 0,rippling-row / read '0', write '1', read '1', write '1'
010	00111000011110010000000000	Din=0v(R1,W0,R0,W0)	Sequence from MAX ADDR to 0,rippling-row / read '1', write '0', read '0', write '0'
010	01010000000100000000000000	din=0^(R0)	Sequence from 0 to MAX ADDR/read '0'

[0076] FIG. 10 is a flowchart illustrating example operation of the electronic device 2 and, in particular, the distributed, three-tier self-test architecture of BIST controller 4, sequencers 8, and memory interfaces 10.

[0077] Initially, BIST controller 4 selects one of the algorithms stored within an internal algorithm memory, e.g., algorithm memory 20 (70). Upon selecting the algorithm, BIST controller 4 issues the first command defined by the algorithm to one or more of sequencers 8 (72).

[0078] Each receiving sequencer 8 parses the command to identify the defined op-code and corresponding parameters (78). For memory access commands, each receiving sequencer 8 initializes a starting address defined by the command (80). Next, sequencer 8 issues a memory operation, i.e., generates the appropriate address, data and control signals (82).

[0079] In turn, each receiving memory interface 10 translates the data and address signals based on the physical characteristics of each respective memory module 12 (92,

94), and applies the translated signals to the memory modules (96). In addition, for read memory accesses (97), memory interfaces 10 automatically compares the data read from the respective memory modules 12 with the expected data (98). Based on the comparison, memory interfaces 10 update respective BIST failure signals to report the status of the tested memory module 12 (100).

[0080] Once the memory operation has been issued by the memory interfaces 10, sequencers 8 determine if additional operations are to be applied to the current memory address within the sequence (84). If so, sequencers 8 issue the commands to memory interfaces 10 in similar fashion (82). If additional operations are not required, sequencers 8 update the address (86) and determine whether the entire address range defined by the command has been sequenced or whether additional addresses remain (88). When the entire address range has been sequenced and the memory operations have been applied to the addresses within the range, sequencers 8 issue acknowledge signals to BIST controller 4 (90).

[0081] Upon receiving acknowledgements from each of sequencers 8 targeted with the command (74), BIST controller 4 determines whether the last command for the selected algorithm has been issued. If not, BIST controller 4 repeats the process and issues the additional commands (72). Once all of the commands have been issued and applied by sequencers 8 and memory interfaces 10 (76), BIST controller 4 terminates the current BIST test.

[0082] Various embodiments have been described. These and other embodiments are within the scope of the following claims.

CLAIMS

1. A system comprising:

a built-in self-test (BIST) controller that stores a set of commands defining an algorithm for testing a memory module;

a sequencer that receives the commands and issues one or more memory operations in accordance with the commands; and

a memory interface that applies the memory operations to the memory module in accordance with physical characteristics of the memory module.

2. The system of claim 1, wherein the BIST controller comprises:

an algorithm memory that stores the set of commands as one of set of selectable memory test algorithms having associated commands; and

an algorithm controller to retrieve the commands from the algorithm memory and issue the commands associated with the selected memory test algorithm to the sequencer.

3. The system of claim 2, wherein the BIST controller includes an interface to receive one or more additional memory test algorithms, wherein the algorithm controller delivers the additional memory test algorithm to the sequencer for application to the memory interface.

4. The system of claim 2, wherein the memory interface comprises one of a plurality of memory interfaces associated with respective memory modules, and the sequencer comprises one of a plurality of sequencers associated with respective subsets of the memory interfaces, and the algorithm controller issues each of the commands to the sequencers in parallel for application to the respective subsets of the memory interfaces.

5. The system of claim 4, further comprising a set of command data interconnects to communicate the commands from the BIST controller to the plurality of sequencers and a set of acknowledgement interconnects to communicate acknowledge signals from

the plurality of sequencers to the BIST controller to indicate the completion of the commands.

6. The system of claim 1, wherein the sequencer controls an application speed of the memory operations to the memory interface in accordance with timing requirements of the memory module.

7. The system of claim 1, wherein the sequencer comprises:

a plurality of command controllers that implement the commands in accordance with a command protocol; and

a command parser to parse each command to identify an operational code and a set of parameters based on the command protocol, wherein the command parser selectively invokes the command controllers based on the operational codes of the commands received from the BIST controller.

8. The system of claim 7, wherein when invoked the command controllers issue the memory operations to the memory interface by sequencing through address ranges defined by the respective commands

9. The system of claim 7, wherein the command controllers issue the memory operations by asserting signals to apply addresses and data to the memory interface based on the commands received from the BIST controller.

10. The system of claim 9, wherein the command controllers issue the memory operations by further asserting control signals to direct the memory interface to automatically store inverted data between least one of neighboring rows, neighboring columns, and neighboring row-column matrices based on the physical characteristics of the memory module.

11. The system of claim 9, wherein based on the physical characteristics of the memory module the memory interface translates the addresses specified by the sequencer for the memory operations.

12. The system of claim 11, wherein the memory module includes memory cells arranged in rows and columns, and the memory interface translates the addresses to fill the memory module in a row-wise or column-wise fashion as specified by the commands from the BIST controller.

13. The system of claim 9, wherein the commands specify a bit pattern to be written to the memory module, and the memory interface translates the data specified by the sequencer based on the specified bit pattern and the physical characteristics of the memory module.

14. The system of claim 1, wherein the memory interface comprises a data generation unit that receives data signals from the sequencer and generates transformed data signals based on the data signals and the physical characteristics of the memory module.

15. The system of claim 14, wherein, in response to a control signal received from the sequencer, the data generation unit automatically transforms the data to store inverted data within at least one of neighboring rows, neighboring columns, and neighboring row-column matrices of the memory module.

16. The system of claim 1, wherein the memory interface comprises an address generation unit that receives address signals from the sequencer and generates transformed address signals applied based an arrangement of rows and columns of the memory module.

17. The system of claim 1, wherein the memory interface comprises a comparator to compare data read from the memory module to data previously written to the memory module and set a state of a failure signal based on the comparison.

18. The system of claim 1, wherein the physical characteristics include at least one of a number of rows, a number of columns, and a number of row-column matrices of the memory module.

19. The system of claim 1, wherein the commands conform to a generalized command protocol that substantially define the test algorithm without regard to physical characteristics and timing requirements of the memory module.

20. The system of claim 19, wherein the command protocol defines a command syntax having a set of supported commands, and each command includes an operand and a set of parameters.

21. The system of claim 20, wherein at least one of the commands includes fields to specify an address range, one or more memory operations to apply over the address range, and a bit pattern for application to the memory module of the address range.

22. The system of claim 1, wherein the BIST controller, memory interface and sequencer are integrated within an electronic device.

23. A system comprising:

a plurality of memory modules;

a built-in self-test (BIST) controller that stores an algorithm for testing the memory modules; and

a plurality of sequencers that are respectively coupled to different subsets of the memory modules, wherein each subset of the memory module is selected to include the memory modules having common clock domains, and each sequencer controls the application of the test algorithm to the respective subset of memory modules in accordance with the common clock domain of that subset of memory modules.

24. The system of claim 23, further comprising a plurality of memory interfaces that are respectively coupled the memory modules, wherein each of the memory interfaces receive address and data signals generated by the sequencer based on the algorithm and translates the address and data signals in accordance with an arrangement of rows and columns of the respective memory module.

25. A device comprising:

first-level built-in, self-test (BIST) means for issuing commands that define a BIST algorithm for a plurality of distributed memory modules having different timing requirements and physical characteristics;

second-level BIST means for processing the commands to generate sequences of memory operations in accordance with the timing requirements of the memory modules; and

third-level BIST means for generating translated address and data signals from the memory operations based on the physical characteristics of the memory modules to apply the BIST algorithm to the distributed memory modules.

26. The device of claim 25, wherein the first-level BIST means comprises:

algorithm storage means for storing the commands as one of set of selectable memory test algorithms having associated commands; and

algorithm control means for retrieving the commands from the algorithm memory and issue the commands associated with the selected memory test algorithm to the sequencer.

27. A method comprising:

issuing commands from a centralized BIST controller to a sequencer, wherein the commands define a memory test algorithm to be applied to a set of distributed memory modules without regard to physical characteristics or timing requirements of the memory modules;

processing the commands with the sequencer to generate one or more sequences of memory operations in accordance with the timing requirements of the memory modules; and

applying the memory operations to the distributed memory modules to test the memory modules.

28. The method of claim 27, further comprising translating address and data signals associated with the memory operations with memory interfaces coupled to each of the memory modules to generate translated address and data signals based on the physical characteristics of each of the memory modules, and wherein applying the memory

operations comprises applying the translated address and data signals to test the memory modules.

29. The method of claim 27, further comprising selecting the memory test algorithm from one of a plurality of memory test algorithms stored within an algorithm memory.

30. The method of claim 28, wherein translating address and data signals comprises translating address and data signals with memory interfaces based on at least one of a number of rows of the respective memory module, a number of columns of the respective memory module, and a number of row-column matrices of the respective memory module.

31. The method of claim 27, wherein issuing commands comprising issuing commands in accordance with a command protocol that defines a set of supported commands having operands and a set of parameters that define the memory operations to be generated by the sequencer.