Embedded DRAM design and architecture for the IBM 0.11-μm ASIC offering

by J. E. Barth, Jr.
J. H. Dreibelbis
E. A. Nelson
D. L. Anand
G. Pomichter
P. Jakobsen
M. R. Nelms
J. Leach
G. M. Belansek

This paper presents an overview of the macro design, architecture, and built-in self-test (BIST) implementation as part of the IBM third-generation embedded dynamic random-access memory (DRAM) for the IBM Blue Logic® 0.11-μm application-specific integrated circuit (ASIC) design system (CU-11). Issues associated with embedding DRAM in an ASIC design are identified and addressed, including fundamental DRAM core function, user interface, test, and diagnosis. Macro operation and organization are detailed and contrasted with traditional DRAM designs. The use of BIST, a key enabler for embedded DRAM, is discussed while highlighting innovations required by the embedded DRAM.

Introduction

As application-specific integrated circuit (ASIC) technologies expand into new markets, the need for denser embedded memory grows. To accommodate this increased demand, embedded DRAM macros have been offered in state-of-the-art ASIC library portfolios [1, 2]. This paper describes an embedded DRAM macro that extends the on-chip capacity to more than 40 MB, allowing historically off-chip memory to be integrated on chip and enabling System-on-a-Chip (SoC) designs. With memory on the chip, applications can take advantage of the high bandwidth naturally offered by a wide-I/O DRAM and achieve data rates greater than those previously limited by pin count and off-chip pin rates. Applications for this memory include network processors, digital signal processors, and cache chips for microprocessors. The integration of embedded DRAM into ASIC designs has intensified the focus on how best to architect, design, and test a high-performance, high-density macro as complex as dynamic RAM in an ASIC logic environment. The ASIC environment itself presents many difficult elements that have historically challenged DRAMs—specifically wide voltage and temperature operating ranges and uncertainties in surrounding noise conditions. These challenges dictate a robust architecture that is noise-tolerant and can operate at high voltage for performance and at low voltage for reduced power. With the advent of embedded DRAM offerings in a logic-based ASIC technology [3], the performance of embedded DRAM macros has improved significantly over that of DRAM-based technologies. Users are increasingly replacing SRAM implementations with embedded DRAM,

0018-8646/02/$5.00 © 2002 IBM

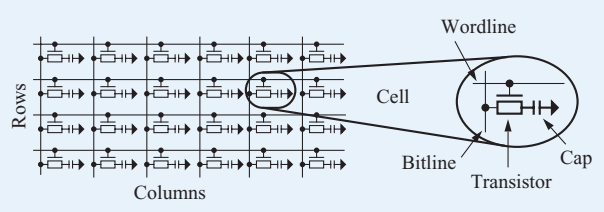

Figure 1

DRAM cell array.

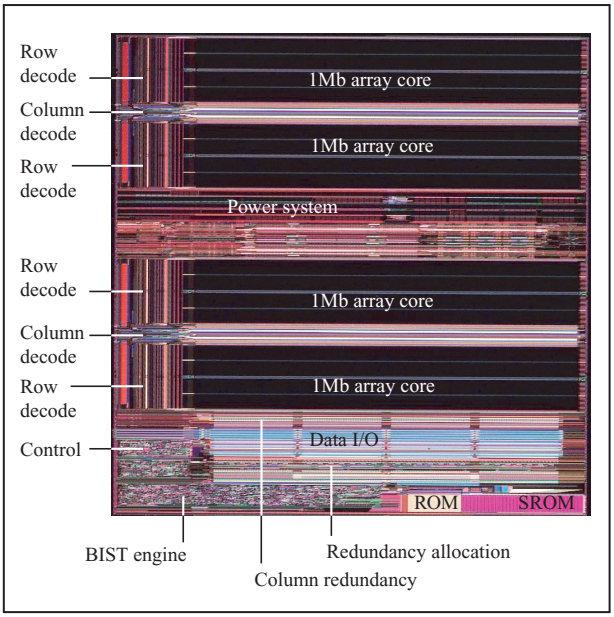

Figure 2

Floorplan of 4Mb macro.

placing additional pressure on macro performance and random cycle time. This pressure extends into testing, where use of traditional direct memory access (DMA) is costly in silicon area and wiring complexity, and introduces uncertainty in performance-critical tests. A more attractive solution to this test problem is the use of a built-in self-test (BIST) system that is adapted to provide all of the necessary elements required for high fault coverage on DRAM, including the calculation of a two-dimensional redundancy solution, pattern programming flexibility, at-speed testing, and test-mode application for margin testing [4, 5]. This paper presents an overview of the macro design, architecture, and BIST implementation as part of the IBM third-generation embedded DRAM for the IBM Blue Logic* 0.11-μm ASIC design system (CU-11), offering a 4× density advantage over SRAM.

Fundamental DRAM operation

DRAM memory arrays are composed of wordlines (or rows) and bitlines (columns); see **Figure 1**. At the cross-point of every row and column is a storage cell consisting of a transistor and capacitor [6]. The data state of the cell is stored as charge on the capacitor, with the transistor acting as a switch controlling access to the capacitor. With the switch on (wordline activated), charge can be read from or written to the storage cell. The rest of the DRAM support circuits are dedicated to controlling the wordlines and bitlines to read and write the memory array.

Overview of embedded DRAM

The CU-11 embedded DRAM macro has been developed around the idea of user simplicity while including a high degree of flexibility, function, and performance. For application flexibility, the embedded DRAM is growable in 1Mb increments to provide macro sizes from a 1Mb minimum to a 16Mb maximum and offers a 256-I/O width and a 292-I/O width for applications requiring parity. The wide I/O was chosen to provide maximum bandwidth; for applications that do not require the full width, bit-write control was included to facilitate masking. Multiple embedded DRAM macros can be instantiated on an ASIC die, enabling customers to make a performance/die-area tradeoff specific to their application. **Figure 2** shows a high-level floorplan of the embedded DRAM. This architecture lends itself well to providing two modes of macro operation: single-bank and multi-bank interleave modes. The single-bank operation provides a simple SRAM replacement function, while the multi-bank mode extends the macro performance by allowing concurrent operations to independent banks.

Single-bank operation

Single-bank operation was intended to resemble an embedded SRAM, supporting simple broadside addressing with read/write control. To improve bandwidth, the user can optionally use page mode, which was carried over from conventional DRAM. The addressing is broken down as follows:

- A0–A2 decodes one of eight page (or column) addresses.
- A3–A11 decodes one of 512 row addresses within a 1Mb block.
- A12–A15 decodes which 1Mb block is to be accessed.

The number of high-order addresses (A12–A15) is determined by the macro size. The diagram in **Figure 3** shows an example of timing for a macro in single-bank-mode operation. The horizontal lines indicate the logical data state of the corresponding signals as they change with time. The macro select signal (MSN) controls the active

(t_{actp}) and restore timing (t_{res}) of the macro and latches the row (A3–A11), column (A0–A2), and block addresses (A12–A15) on every falling edge of MSN (indicated by the vertical lines). For a read cycle, write enable (WEN) is held high; data-out (DO) is latched and transmitted off the macro within time t_{acc}. For a write cycle, WEN is held low; data-in (DI) and bit write (BW) are received and latched. Page mode, controlled by the page signal (PGN), allows access to the additional bits along a wordline not selected at the MSN falling edge. For successive page cycles, falling PGN latches only WEN and a new column address (A0–A2); the row and bank addresses latched during the MSN falling edge are reused. In page mode, DO is latched and transmitted off the macro within time t_{accp}.

Multi-bank operation

For the multi-bank-mode configuration, each 1Mb block of the macro acts as an independent bank that shares a common address and data bus with all other 1Mb blocks within the macro. The number of banks within a macro is determined by the macro size. **Figure 4** shows a 4Mb macro with four banks. A bank select (BS) pin is associated with each bank (1Mb block) and controls activation and pre-charge of that bank. The bank address (BA) is decoded by control logic and arbitrates which bank has control of the data path.

In multi-bank configuration, the macro does not employ broadside addressing. Rather, the embedded DRAM macro operates similarly to a synchronous DRAM (SDRAM), in which addressing is performed in a row-address strobe/column-address strobe (RAS/CAS) manner and the macro select input (MSN) is treated like a master input clock, latching the state of all other input pins with each falling MSN edge. **Figure 5** shows three cycles: Cycle 0 activates Bank 0, Cycle 1 activates Bank 1 and reads or writes Bank 0, and finally Cycle 2 activates Bank 2, reads or writes Bank 1, and pre-charges Bank 0. The MSN input can be cycled at a maximum rate of 250 MHz (4 ns assuming a nominal 50/50 clock duty cycle). All bank select (BS) inputs must be defined at every MSN falling edge to indicate whether each bank is to remain open or closed or whether a bank is to become active/open (from the pre-charge state) or become pre-charged/closed (from the active state). This protocol supports simultaneous activate, read/write, and pre-charge to three different banks.

Any combination of banks within a macro can be active simultaneously as long as each 1Mb bank is opened in a sequential fashion. To avoid power-supply design limits, multiple banks cannot be activated or pre-charged on a single MSN clock cycle. Maximizing the number of banks in a macro improves the probability of avoiding an open (or busy) bank and maintaining the pseudorandom peak bandwidth of 8 GB/s.

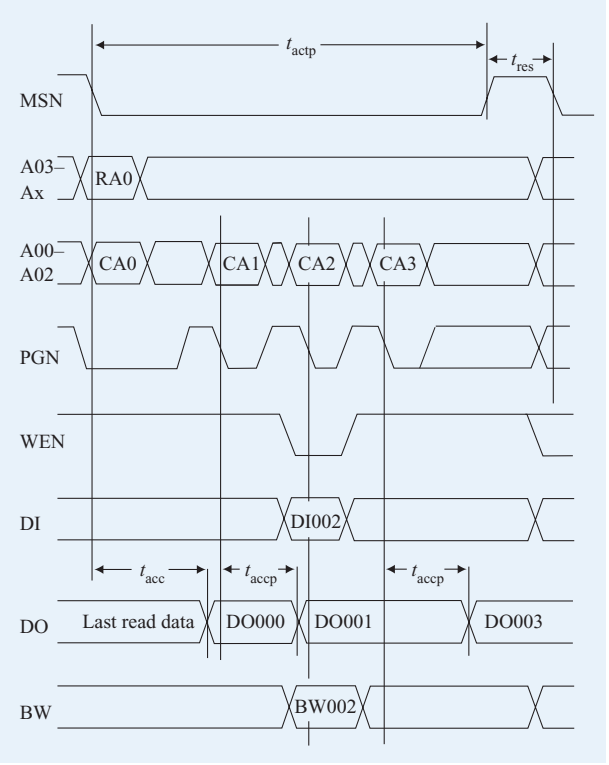

Figure 3

Single-bank timing diagram.

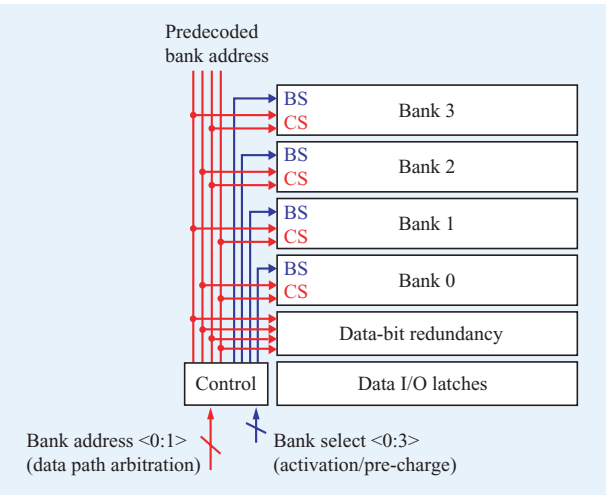

Figure 4

Multi-bank architecture.

To activate a bank, the corresponding BS signal must be low, and the row address for that bank must be supplied

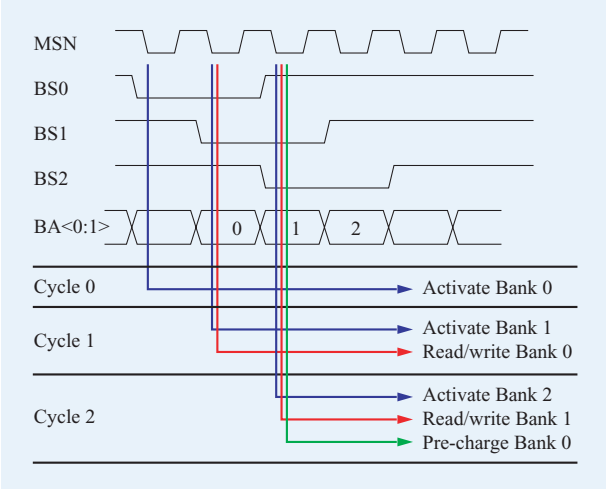

Figure 5

Multi-bank operation.

on address pins A3–A11 when MSN is clocked low. The row address for each active bank remains latched until the bank is pre-charged. This frees the row address bus to allow other banks to be activated at a different row address on subsequent MSN clock cycles. A bank is pre-charged by placing a high level on the corresponding BS input when the MSN is clocked low. No address information is required to pre-charge a bank.

Once a bank has been activated, a read or write operation can be performed to that bank by bringing PGN low, selecting the state of the WEN pin (read = high, write = low), specifying the column address on A0–A2, and specifying the bank address on A12–A15 as MSN is clocked low.

Macro organization

The embedded DRAM is constructed from building blocks: a 1Mb array core, a power system for generating boosted voltage levels used by the array core, a control system for buffering and generating the array core timing signals, column redundancy for replacing defect data bits, data I/O for receiving and transmitting off-macro data, and BIST for testing the embedded DRAM macro. BIST is composed of a microprocessor-based engine, instruction memory (ROM/SROM), a data comparator, and a redundancy allocation unit (Figure 2). The 1Mb array and its support circuitry are replicated to construct the desired macro size. Each embedded DRAM macro contains a single control system, a common power system, and a BIST.

The power system, for the scope of this paper, simply supplies the necessary voltage network levels required for biasing the DRAM cell matrix. The power system is located at the midpoint of the 1Mb arrays (or 1Mb offset in the case of odd-sized macros) to provide optimal power distribution. The row decoder selects one of 512 words, while the column decoder selects 256 of 2048 bits in the 1Mb array. The control system is the primary unit controlling signals to the array(s). It receives input signals from either primary macro inputs or the BIST. Signals to the array are selected by the system function mode: normal operation mode or test (BIST) mode. The control path also determines whether the embedded DRAM is operated as a single bank or as a multi-bank part.

The final block in the embedded DRAM macro is the BIST. The design goal of the BIST is to provide a test engine, operable in the logic test environment on low-cost, low-pin-count testers, that stimulates the control, data paths, and array of the embedded DRAM and provides fault coverage equivalent to that traditionally supplied by high-cost memory testers to discrete DRAM. The flexibility of the BIST system enables the test development engineers to alter the instruction memory to create new or modified test patterns or to change the sequence or number of patterns applied at each manufacturing test gate. Ultimately, the BIST locates all faults in each 1Mb array segment, calculates the two-dimensional redundancy solution required to repair these faults, and reports this solution via standard scan string methods [7]. The redundancy solution is permanently stored in a remote fuse memory (nonvolatile) programmed with a laser after testing.

Array core organization

The array core, shown in **Figure 6**, is the fundamental building block of the embedded DRAM macro; it includes the following:

1. DRAM cell array.
2. Row decoder for selecting one of many wordlines.
3. Level shift and driver for elevating the voltage level of the selected wordline.
4. Wordlines for coupling storage cells to bitlines.
5. Wordline stitch regions for reducing the wordline propagation delay with metal connections.
6. Bitlines for connecting storage cells to sense amplifiers (sense amps).
7. Restore devices and sense amps for reading the small signal level from bitlines.
8. Column decoder for selecting one of many sense amps.
9. Local buffers for driving the select sense amps to the edge of the macro on a data line.

Each 1Mb array core is organized as 512 wordlines (512 WL) by 2048 bitlines (2K BL). The 2048 bitlines are decoded from eight to one, producing a 256-bit data word

with an 8-bit page depth (additional bits available in page mode). The 292-parity option is implemented by adding 288 bitlines (or 36 data bits), extending the length of the wordline. The cell array and sense amplifiers are mirrored around a shared local buffer that drives the selected sense amps onto a global data line running over the cell array, parallel to the bitlines on third-level metal (M3). Each of these 1Mb array segments (or banks) contains eight redundant wordlines and eight redundant data bits that can be used to repair faults in that 1Mb array. The redundancy repair region is limited to the 1Mb bank to allow independent repair, simplifying test and redundancy allocation for macros constructed from multiple array cores.

This organization provides high array utilization while maintaining performance typically lost by creating longer wordlines or bitlines. Faster core performance can be achieved with shorter wordlines and bitlines, but at a cost: Every wordline requires a decoder/driver, and every bitline requires a sense amp. Cutting wordlines and bitlines in half requires four times as many word drivers and four times as many sense amps, resulting in a larger core area for an equivalent memory size. Although smaller arrays can run faster, there is a diminishing return in performance due to increased total area and resulting propagation delays, segment decoding, and data multiplexing. A metric for measuring efficiency is array utilization, which is calculated by multiplying the cell area by the number of bits and dividing by the total macro area. The large array block achieves a higher efficiency because it can amortize the support circuit overhead (word decoders/sense amps) across more memory cells.

Array core operation

The following sections describe in more detail the operation of the array core components including the row system for activating wordlines, the sense system for reading and writing data from the cell, and the data path for delivering data to and from the array core.

Row system

The logical function of a DRAM word system is simple: Select one wordline out of 512. However, the electrical and physical implementations are far from simple. Most DRAMs boost the wordline voltage above the bitline high level to increase the voltage written to the one transistor cell. The wordline high level is chosen to provide full write-back while staying within device reliability constraints. Overdriving the array cell allows the device designer to increase the array FET threshold voltage to achieve the low off-current required for data retention while maintaining a reasonable device performance. The physical implementation of the wordline driver is challenged by the aggressive wordline pitch (or periodicity). Each wordline must be decoded, level-shifted, and transmitted on minimum-pitch second-level wiring (M2). To minimize DRAM cell leakage, salicide used by the logic process to lower diffusion and polysilicon resistance must be blocked from the array. The resulting DRAM polysilicon wordlines are highly resistive and must be periodically connected to a parallel low-resistance M2 (or stitched) to meet the performance objectives. In the row decode system (Figure 2), addresses (A3–11) are received, and true–complement pairs are generated (Addr T/C) and simultaneously sent to the pre-decode system and redundancy compare circuits. The control block is activated with the bank select signal and enables address pre-decode and row latching. The redundancy circuits compare the current address with stored addresses of defective wordlines. If the incoming addresses match a stored defective address, the defective wordline is held inactive and a spare wordline is activated in its place. Redundancy enables repair of defective elements and improves yield.

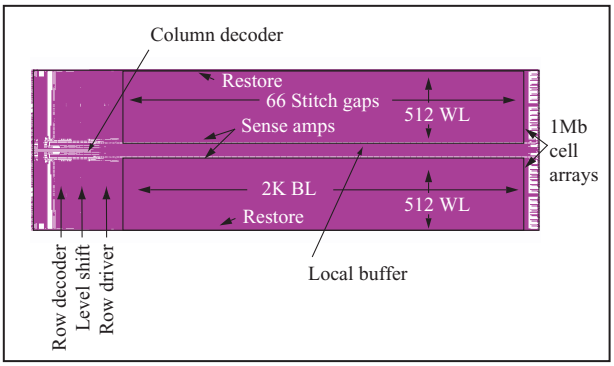

Figure 6

Physical layout of 2Mb array core.

Charge sensing

There are a variety of mechanisms for sensing charge stored on a capacitor. Conventional $V_{DD}/2$ sensing is reviewed for background, followed by the high-speed GND sensing scheme utilized by this work. The two schemes are contrasted, highlighting the benefits of GND sensing.

DRAMs operate on the principle of charge sharing, as shown in **Figure 7**. By activating a wordline (WL), charge from the cell storage capacitor (NODE) is transferred to the bitline (BL), altering the potential of the bitline. The change in bitline potential is limited by the transfer ratio of the cell capacitance (C_{cell}) to the sum of the bitline capacitance (C_{BL}) and cell capacitance:

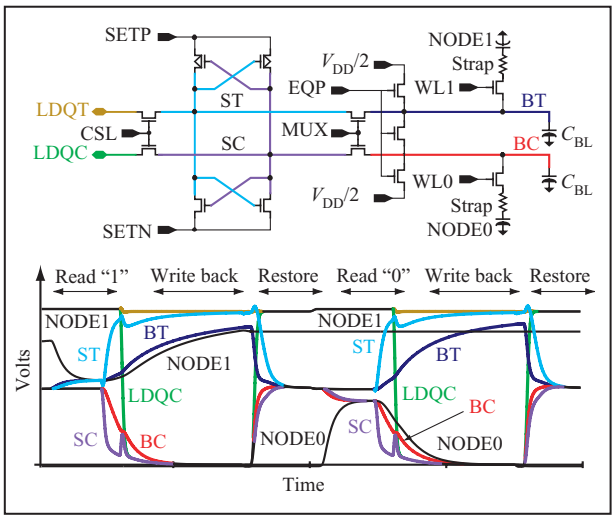

Figure 7

Schematic and simulation waveforms for conventional DRAM $V_{DD}/2$ pre-charge.

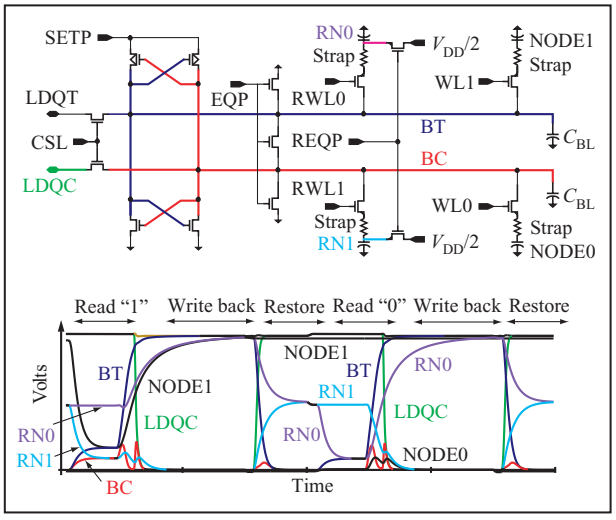

Figure 8

Schematic and simulation waveforms for embedded DRAM GND pre-charge.

$$\Delta V = (V_{BL} - V_{cell}) \left[\frac{C_{cell}}{C_{BL} + C_{cell}} \right].$$

For example, a transfer ratio of 1/5 and a bitline-to-cell voltage difference ($V_{BL} - V_{cell}$) of 600 mV would ideally result in an active bitline voltage change (ΔV) of 120 mV. The most reliable, low-power means to amplify this small signal is with a cross-coupled differential sense amplifier. Operation of the amplifier involves pre-charging the true and complement bitlines (BT/BC) to an equivalent voltage (pre-charge level) with an equalize pulse (EQP), activating the select wordline (WL) to transfer charge from the cell to the bitline, creating a small signal difference between BT and BC (reading the cell), then driving the common source lines of the p-FETs (SETP) to V_{DD} and common source lines of the n-FETs (SETN) to ground (GND). A multiplexor device, controlled by the MUX input signal, isolates the sense amp nodes (ST/SC) from the bitlines, allowing fast amplification. The bitline with the higher potential is driven to V_{DD} and the bitline with the lower potential is driven to GND. Once the sense amp has stabilized, the data can be transmitted off the macro through the local data lines (LDQT/LDQC) selected by the column-select line (CSL). Bitlines are then returned to their pre-charge level by reactivating the equalize signal (EQP).

With differential sense, the active bitline (connected to a selected cell) is compared to a reference bitline (not connected to the selected cell). Positive signal, as measured by the voltage difference between active bitline and reference bitline, is amplified to a logical "1." Negative signal is amplified to a logical "0." The reference bitline must be conditioned to allow the sense amp to reliably distinguish a low level stored in the cell from a high level stored in the cell. There are a variety of methods for preconditioning the reference bitline. Conventional DRAMs precondition both the active bitline and the reference bitline at $V_{DD}/2$. Reading a low level from the cell couples the active bitline below the reference, and a logical "0" is sensed. Reading a high level from the cell couples the active bitline above the reference bitline, and a logical "1" is sensed.

Pre-charge level

For $V_{DD}/2$ pre-charge, the pre-charge level itself provides an excellent reference. Reading a high level from the selected cell moves the active bitline above $V_{DD}/2$, creating positive signal. Reading a low level from the selected cell moves the active bitline below $V_{DD}/2$, creating negative signal. An alternative to $V_{DD}/2$ pre-conditioning, shown in **Figure 8**, is to pre-charge both the active and the reference bitline to GND. This scheme, however, cannot use the pre-charge level alone to provide a reference, because reading a low level from a cell does not move the active bitline ($V_{BL} - V_{cell} = 0$), resulting in zero signal and unpredictable amplifier operation. The most reliable means of generating a reference level for GND pre-charge is to condition the reference bitline with half charge from a reference cell (RN) activated by a reference wordline

(RWL). This can easily be accomplished by writing $V_{DD}/2$ into a reference cell with an additional access device controlled by a reference equalize signal (REQP). When the selected wordline is activated, an associated reference wordline is activated, placing the half charge on the reference bitline and resulting in a level exactly between reading a high level and reading a low level.

GND pre-charge deviates from conventional $V_{DD}/2$ pre-charge, but offers a wider operating range, and the improved read and pre-charge performance required by the ASIC environment. A difference to note is that GND pre-charge transfers charge to the active bitline only while reading a high level from the selected cell. Because there is no charge transfer when reading a low level from the active cell (cell and bitline are at the same potential), GND pre-charge relies on half charge transfer from the reference cell for reading a low level. In contrast, $V_{DD}/2$ sense schemes require charge transfer from the active cell for both a low and a high level for a read operation.

Cell read performance

When transferring a high level stored in a cell to the bitline, charge transfer does not start until the wordline is an array device threshold above the bitline. For $V_{DD}/2$, it takes more time for the wordline to reach $V_{DD}/2$ than the GND pre-charged scheme in which charge transfer begins when the wordline is a threshold above GND. This is critical for a longer wordline that will have a significant slew rate: a 1-V/ns slew introduces an extra 750-ps delay for $V_{DD}/2$ pre-charge (@ = 1.5 V). Additionally, as a high level is read out of the cell in the GND pre-charge case, device overdrive (defined as $V_{gs} - V_t$) increases and is always $V_{DD}/2$ greater than the $V_{DD}/2$ pre-charge case, resulting in faster charge transfer. Although reading a zero in the $V_{DD}/2$ case begins when the wordline reaches a threshold above the cell (GND), it loses overdrive as the cell charges to the bitline $V_{DD}/2$ potential. As a result of wordline slew and overdrive, GND pre-charge develops signal faster than the $V_{DD}/2$ pre-charge.

Reference cells

The reference cells and reference wordlines required by GND pre-charge increase core area; however, they provide many valuable features, including the following:

1. Bitline balance.
2. Equivalent WL-to-BL coupling for reference BL.
3. Lower sense amp operating point and more overdrive, avoiding stall at low voltage/low temperature.
4. Interlock signal to mimic circuit performance and generate sense amp timings.
5. SETN tied to GND, eliminating control and drive.

Implementation of reference cells provides both static and dynamic bitline balancing. Static bitline balancing is achieved by placing the equivalent capacitance of a reference cell on the reference bitline. Without a reference cell, the active bitline would see the extra capacitance of the storage capacitor of the selected cell, creating a 20% capacitance mismatch between bitline and reference bitline. Transient bitline balancing is accomplished by switching of the reference wordline, providing dynamic coupling to the reference bitline that is equivalent to the coupling from the selected wordline to the active bitline. The reference cell and reference wordline minimize the mismatch and coupling, allowing the sense system to operate on less stored charge, which enables increased performance and improved retention characteristics.

GND pre-charge simplifies sense amp control and provides faster amplification. With bitlines pre-charged to GND, gating the sense amp n-type cross-coupled pair source (SETN) is not required; consequently, SETN can be tied directly to GND, as shown in Figure 8. In contrast, $V_{DD}/2$ pre-charge requires controlling both SETP and SETN. GND sense achieves faster amplification by applying full overdrive to the p-cross-coupled pair during sense amp set (SETP driven to V_{DD}). In the $V_{DD}/2$ pre-charge case, SETN is driven from $V_{DD}/2$ to GND and SETP is driven from $V_{DD}/2$ to V_{DD}, resulting in less overdrive, essentially splitting the overdrive between the n-cross-coupled pair and the p-cross-coupled pair. At low voltage, this may not be enough overdrive to turn on either the n or p devices, resulting in sense amp stall. GND sensing was chosen for providing increased overdrive to enable low-voltage operation and improved performance at nominal voltage operation.

Bitline twisting

A key technology feature in achieving ASIC density and performance is back-end wiring characteristics, specifically metal pitch, resistance, and capacitance. Although a given design point may be good for logic, it does not necessary meet the ultralow-capacitance needs of the M1 bitline. DRAMs, which can tolerate higher-resistance bitlines, typically opt for metal half the thickness of the same-generation logic process, resulting in less line-to-line capacitance. The higher line-to-line capacitance not only increases total bitline capacitance (and signal; see the section on charge sensing) but also increases the noise created by a neighboring bitline pair, introducing data-pattern dependence. This data-pattern sensitivity must then be included when testing for signal margin. To counteract line-to-line noise, bitline twisting was implemented. Bitline twisting uses a series of twists to distribute the coupling effect of neighboring noise

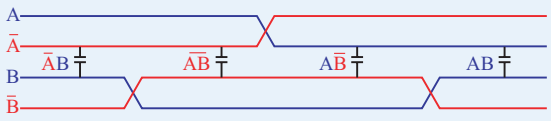

Figure 9

Bitline twisting.

equally into both the active bitline and reference bitline, converting the noise into common mode (see **Figure 9**). Bitline twisting has been shown to decrease the minimum raw signal required from 100 mV to 30 mV, allowing the sense amp to operate at a faster cycle time and lower latency without suffering manufacturing yield loss.

Data path

Once the selected cell data has been captured by the sense amplifiers, it is ready to be shipped to the edge of the macro. The data path is responsible for selecting a subset of sense amplifiers, sending data to the edge of the macro and latching and then transmitting data to customer logic. A wide I/O data path is desirable in order to maximize core bandwidth. For the area efficiency described in the section on array core organization, DRAM cores typically access 2048 bits simultaneously and naturally offer very wide I/O. Because of the minimum pitch of the bitlines, it is unrealistic to drive all of the bits on a wordline off the macro. A column decoder is placed adjacent to the sense amps, selecting 256 of the 2048 bitlines and reducing the I/O by 8:1. This reduction provides one bit of data on approximately the same physical pitch as a 12-track logic cell (4.8 μm). This feature reduces wiring congestion and facilitates bit-slice design within and outside the DRAM macro. Reducing the width from 2048 bits to 256 bits provides for enough wiring room to accommodate independent read and write data buses with ample tracks for power and noise shielding. Because of noise uncertainties in the surrounding ASIC environment and wide operating conditions (voltage and temperature), a robust, full-rail static data bus design was chosen. Dynamic data-line pre-charge schemes can offer higher performance, but typically burn more power and are less noise-tolerant. Separating the read and write paths eliminates design issues associated with bidirectional bus control and enables higher-frequency read-to-write performance. The timing specification in Figure 3 shows a read cycle followed by a write cycle in which write data is latched at the same time read data from the previous cycle is being transmitted. With a bidirectional data bus, this would typically require one dead cycle to allow for bus turnaround.

Now that we have presented the DRAM macro operation and core, we can turn our attention to methods for guaranteeing the embedded DRAM macro functional specifications in an ASIC environment.

Test strategy

As previously described, the cell arrays used in the embedded DRAM macro are very similar to those of their DRAM ancestors. We must assume that the cell arrays will have the same sensitivities as those well known from the development of DRAM, and require identification at test. Many of the interactions in the DRAM cell matrix are complex and are triggered only by certain combinations of defects and test patterns. To deliver the complex test patterns, commodity DRAMs use specialized test equipment with algorithmic pattern capability for generating the test sequences and large/fast-data-capture memory with redundancy allocation hardware for identifying and repairing faults. Considering how to test a DRAM embedded in logic creates a dilemma: logic tester or memory tester [8–10]? The logic test platform that has been developed for past generations of ASICs without embedded DRAM can be characterized as a low-cost, reduced-pin-count tester with no algorithmic pattern generation or redundancy allocation hardware; it is therefore unable to test DRAM without assistance. The logic test patterns implemented are automatically generated with software based on the customer's netlist. The test strategy comes down to either a two-tester solution (memory tester and logic tester) or comprehensive built-in self-test (BIST).

The two-tester approach suffers from the following issues: 1) Multiple test gates are required, with an associated increase in wafer handling; 2) cumbersome requirements are placed on the customer to multiplex the macro I/O to package pins; and 3) the required part-number-specific test-pattern development is typically difficult to automate. In the high-part-number ASIC environment, it is essential to implement a single tester platform utilizing BIST for memory test and automated test generation for logic test.

BIST engine design point

The use of BIST to test embedded SRAM in ASIC designs is not new. Much work since the late 1980s [11, 12] has proven this technique, which capitalizes on the similarities between the six-transistor SRAM memory cell and standard logic. The engine used was a state machine that simply ran through a predetermined sequence of standard patterns to uncover faults in either the memory array or the activation or decode support logic and determine pass/fail. Only in the most sophisticated cases

was redundancy even considered, and then only in the row dimension.

Since the same set of six to eight SRAM test patterns have remained constant for a number of generations, little, if any, capability to alter the SRAM test sequence was required. However, DRAM is sensitive to far more subtle systematic and random process variations and defects, requiring a more complicated and dynamic test sequence to identify and repair these faults. It is generally not possible to predict all data patterns or write/read patterns that will ultimately be needed to reduce the test escape rate to less than ten parts per million (10 ppm). The complicated dynamic pattern set and the multidimensional redundancy allocation (row and column) required by DRAM testing rendered earlier generations of BIST technology inadequate for DRAM testing. The new fail mechanism and pattern sensitivities specific to DRAM required reconsideration of the BIST engine dexterity. To address unique requirements of embedded DRAM test, a processor-based BIST engine with a high degree of program flexibility was developed [1]. A programmable BIST engine design provides a very flexible test pattern development system as well as a standard set of patterns for manufacturing test. This enables the test and characterization engineers to define the necessary patterns used at each test gate and the order in which they are applied. Test instructions that communicate with tester pin stimulation allow for external control of pauses used during retention-time testing and provide extended capability for exercises such as burn-in.

The BIST can be subdivided into eight major components: test multiplexor, instruction memory, sequencer, address generator, data generator, cycle generator, redundancy calculator, and clock generator, as shown in **Figure 10**.

Test multiplexor

The test multiplexor, controlled by the test input, enables the BIST to take control of the DRAM macro. With the test signal deactivated, the DRAM operates normally, responding to the user inputs. These two modes are referred to respectively as test mode and mission mode.

Instruction memory

The instruction memory is composed of a ROM and scannable read-only memory (SROM) and consists of 226 addressable words by 34 bits, with the SROM contributing 34 words. The ROM is programmed in the back end of the chip fabrication and contains instructions that represent predetermined patterns expected to be used during test of the embedded DRAM. The purpose of the SROM is to dynamically extend the programmability of the BIST. It allows new and unique patterns to be loaded into the program space in addition to the patterns hard-

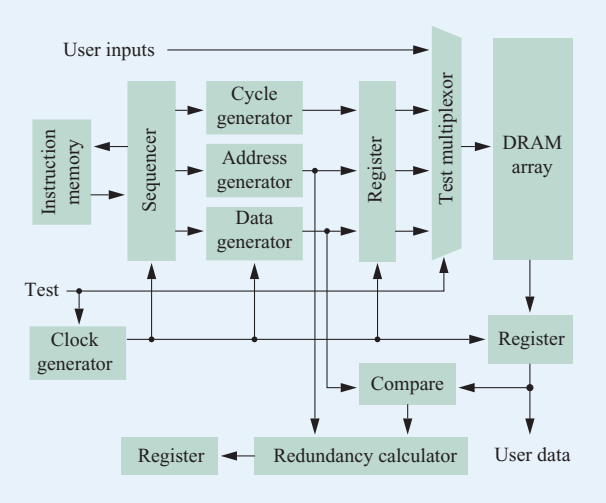

Figure 10

BIST block diagram.

coded into the ROM. If a pattern is continuously used within the register array, it can be programmed into the ROM on a new metal mask release. There are also various utility instructions programmed into the ROM that allow the processor to reorder, reconfigure, eliminate, or add test patterns. The index addresses of the instruction memory are partitioned such that the processor can operate out of hard-coded ROM, SROM initialized at the start of test, or a combination of both.

The recent improvement made to this scheme consists of the addition of the capability to allow the SROM to be reloaded an unlimited number of times in any given test pass without upsetting the state of the rest of the BIST processor. This is especially important for the generation and preservation of cumulative redundancy calculation results over all patterns tested. The use of this feature eliminates concern over the choice of the size of ROM/SROM included in the design being adequate to contain all patterns necessary for a particular test gate. The SROM need now be only large enough to store the longest pattern, and the ROM can be sized to improve test time and efficiency.

Sequencer

The sequencer is the backbone of the BIST engine. It is responsible for the execution flow of the programs stored in the BIST instruction memory. It contains a program counter used to fetch an instruction from the instruction memory. Once an instruction has been fetched, the sequencer calculates the next address for the program counter based on the instruction's branch and branch operation code (opcode) fields.

Two types of instructions are used in the BIST. The first is a normal instruction indicated when bit 31 of the instruction word is set to a logical "0." The second type of instruction is a test-mode instruction, indicated when bit 31 of the instruction word is set to a logical "1." It is used to set various latches in the BIST. These latches control modes of operation, set digital-to-analog converts (DACs) to offset internal voltages and timings for margin testing, program the data scramble, and set the utility counter.

Address generator

The address generator is responsible for generating the addressing for the array during self-test. This is done with the aid of three counters: row, column, and block, which are controlled by the appropriate fields of the BIST instruction. The counters can be incremented or decremented as defined by the increment/decrement bit of the instruction.

Data generator

Because of the difficulty in predicting what actual data patterns will be necessary in the embedded DRAM array matrix to activate and expose subtle faults, a flexible data generator was introduced. The previous versions allowed only a short list of pre-coded patterns that provided blanket lows or highs, checkerboard lows/highs, row stripes, and column stripes to the matrix. This approach was found to be inadequate for proper test-pattern development and not directly reusable for varying array architectures. Thus, a design that accounts for current row and column address, adjacent data bit, and physical or logical data state stipulated in the pattern was added [4, 13]. The algorithm to be used by the data generator for a given test is specified with a mode-set instruction prior to executing any writes or reads to the DRAM. This instruction alters the states of latches in the design that govern the operation of the generator. Odd and even data bits are generated and replicated across the 256-bit data word. Odd and even data bits are a function of the program data scramble (PDS), the current address, and the data state specified in the read or write instruction. The PDS contains latches that are programmed with a test-mode instruction. These latches allow the test pattern to generate stripes of various sizes, checkerboards, or other desired patterns in the 1Mb array(s).

The term *logical data* refers to the state of the data at the macro inputs, logical "1" being a high voltage level and logical "0" being a low voltage level. Logical data is simply written as is, regardless of the bitline connection to the sense amp and its location in the array. The term *physical data* refers to the voltage level stored on the DRAM cell capacitor. DRAM architectures routinely invert logical data when writing to the cell; i.e., a logical "1" may be stored as a physical "0." Inversion is address-dependent and is acceptable as long the architecture reinverts the data when it is read. Controlling physical data is required because many defect mechanisms are sensitive to the physical data state stored. When physical data is desired, the data is modified according to the bitline connection to either the true or complement side of the sense amp. Also taken into consideration is the location of the bit with respect to array topography. During a read or write operation, data is generated at the time of instruction execution. In a write operation, data is sent directly to the data path latches. During a read operation, the data generator produces expected data for the comparator.

Cycle generator

The cycle generator is responsible for generating the row and column signals used in single-bank mode for the 1Mb array(s). When in multi-bank mode, the cycle generator must generate the bank select signals as well. The protocols for the single-bank and multi-bank modes of operation are different. Examples of the signals required in each of these modes are shown in Figure 3 and Figure 5. These signals are created as a function of the internal BIST clocks and the cycle opcode defined by the instruction.

When running BIST prior to redundancy repair, the control signals, generated to stimulate the array, are restricted to a single 1Mb array, so only the single-bank operating mode is required. The logical addressing to the 1Mb array under test is defined by a group of latches in the BIST. This restriction is enforced because the redundancy calculator can compute a solution for only a single 1Mb array at a time.

Once a redundancy solution has been computed for each 1Mb array, the macro can be repaired. The macro can then be stimulated either through the single-bank or the multi-bank protocol as a complete macro. If a fail is found at any point after the macro has been repaired, the macro is considered a reject.

Redundancy calculator

A key enhancement to BIST schemes used previously for embedded SRAM macros is the inclusion of a redundancy calculator, also referred to as redundancy allocation logic (RAL), for two-dimensional redundancy. The function of the redundancy calculator is to compare data read from the array with expect data from the BIST engine and optimally allocate redundancy for array fails. In this system [1], the BIST processor calculates row and data-bit redundancy for wide-I/O embedded DRAM macros. Each 1Mb array contains its own redundant elements, which may not be shared with other 1Mb arrays. For this reason, BIST calculates and stores only 1Mb worth of a redundancy solution at a time. Calculation of the full

16Mb repair solution would require 16 times the number of fail counters and address registers, increasing the silicon area required for the BIST to an unacceptable level.

Several further enhancements have been made to this system to heighten manufacturability. The first of these enhancements deals with the definition of a "must-fix" data bit. A fault was previously defined as a "must-fix" data bit if there were more row addresses failing within a data bit than there were redundant row elements available to fix them. This was hard-coded into the calculator logic as the true must-fix limit. The redundancy calculator was modified to allow the option of specifying, at the beginning of test, what definition of must-fix data bit the calculator will use. The choices supported are one quarter, one half, or the full number of available redundant row elements. The choice is communicated to the redundancy calculator via scan initialization. With this option, decisions can be made as to how faults will be fixed if the fabricator may be prone to a certain mix of fault types.

The second enhancement to the RAL portion of the BIST is the integration of redundant row testing. To ensure high final module yield, redundant rows must be tested prior to use. In the previous BIST design [1], redundant rows were tested with a separate set of patterns. The results from redundant row test were then merged offline with the RAL solution from the normal test patterns, producing a modified laser fuse solution when defective redundant rows were identified. To eliminate the need for multiple-pass testing and offline data manipulation, the failing row address calculator portion of the RAL was modified along with the row address generator in the processor. The row address was extended to a nonbinary value to include address space for selecting the redundant rows in each address schedule. The choice to include redundant addresses in the pattern is optional via test mode. The content-addressable memory (CAM) in the failing row address calculator recognizes failing redundant rows and eliminates them from the final redundancy solution reported at the end of test. With this combination of modifications, a single test pass provides a redundancy solution that accounts for failing redundant elements and their effect on total macro fixability.

The third and final enhancement to the redundancy calculator comes with the capability to reload the SROM instruction memory discussed earlier. By segmenting the scan string and reloading only the SROM, neither the state of the BIST engine nor the current state of the RAL calculated redundancy solution is upset. Thus, additional patterns can continue to be applied one after another, with the redundancy solution being calculated on the cumulative failure set from all patterns.

Clock generator

The final component of the BIST is the clock generator, which is responsible for generating all of the necessary clocks needed for self-test of the DRAM. It has the ability to receive external clocks and pass them directly to the BIST, or generate its own clocks from an external oscillator.

To maximize system performance, 250-MHz timing rules are provided to describe the RAM core performance. Testing at speed (high-frequency or ac testing) becomes essential to ensure the high-performance timings, especially in the presence of speed-sensitive fails commonly found in DRAMs. As the complexity of chip-level integration grows, so do the challenges for ac test. The DRAM core performance specifications include only those delay elements within the core boundary. Accurate evaluation of these delays requires that they be measured either directly at the core boundary or from test points where the delay from the macro boundary to said test point is known.

Effective ac testing requires that the stimuli be provided with sufficient speed and accuracy, and efficient ac testing requires that constraints for cost-competitive manufacturing be met. A basic BIST design can be enhanced to provide an effective, efficient means of ensuring ac performance specifications for cycle and access times.

Placing BIST units in such a way that the delays between the BIST and the RAM core boundary are minimized and predictable greatly simplifies ac test development. An example of such a design is one in which each instance of a RAM core comes with its own BIST. While this approach may use more silicon area than others, savings in design, test development, and test cost can be realized through the reuse of the same integrated RAM/BIST core across a wide range of applications. In some environments, the results that can be derived from a finite design and test development resource are enhanced by focusing on the development of a reusable core rather than the development of a specific customer part number. The RAM/BIST integrated core design point enables effective ac test by reducing the delay between the RAM and the BIST that minimizes the timing uncertainties introduced by process variability and typical tester hardware. Efficient ac test is realized by leveraging the self-test concept. Performing data compare and storing fail locations in the BIST reduce the demands on tester resources to input only. Eliminating the need for expensive high-speed data capture hardware in the test head greatly simplifies requirements from the point of view of both bandwidth and calibration. Another large step toward reducing tester demands by containing critical timings within a BIST is taken when clock multiplication capability is added.

Clock multiplier

For example, with very modest test equipment a 10-MHz clock can be used as input to a 20:1 clock multiplier, creating a 200-MHz BIST test environment. The clock multiplier receives a single clock from a tester and multiplies it by a factor (MULT) ranging from 2 to 32. The multiplier is designed such that the only restriction on the tester clock cycle, $t(\mathrm{CYC}_{\mathrm{TESTER}})$, is that

$$\frac{t(\mathrm{CYC}_{\mathrm{TESTER}})}{MULT} = t(\mathrm{CYC}_{\mathrm{TARGET}}).$$

The range of target cycle, $t(\mathrm{CYC}_{\mathrm{TARGET}})$, is limited by silicon area, process, and operating conditions. If an internal target of a 2.5-ns cycle is desired and a 50-ns tester clock cycle is used, *MULT* should be 20.

DRAM random-access cycle and data path cycle times typically differ by factors of three to eight times. The faster data path cycle time allows for higher bandwidth in spite of the relatively slow random-access cycle time. A clock-shaping function can simplify the creation of appropriate DRAM test patterns from a single high-speed clock. Within the BIST, the multiplied clock is "shaped" to create control signals with the desired duty cycle. Specifically, the BIST shapes the clock into row and column controls that are used to stress row cycles and column cycles. For example, the BIST can shape the clock to generate a six-cycle row active, two-cycle row pre-charge, one-cycle column low, and one-cycle column high. As another example, a 200-MHz input to a clock shaper can be used in such a way that data path operations occur every 5 ns, while random-access events occur at 15-ns intervals. The random-access duty factor for this example would be 2/3 active, 1/3 pre-charge. Allowing selection from any number of high-speed clock periods from one to 16 for each basic DRAM core event enables the test engineer to selectively stress individual timing values. The combination of clock-shaping and clock-multiplication capabilities provides a very flexible and highly accurate test environment for DRAM cores with minimal demands on test equipment and device interface hardware.

dc test

DRAMs require voltages above V_{DD} and below GND to provide a sufficient difference between array transistor on- and off-currents. If the off-current is not low enough, DRAM retention time is compromised; if the on-current is not high enough, cycle time is compromised. Charge pump circuits are used to generate high-voltage supplies and are delivered to the DRAM array via distribution networks. For efficiency, from the point of view of both silicon area and power consumption, these pump systems are designed with modest overhead and standby operating modes. Defects in the pumps or on the networks can inhibit proper operation and must be detected. While functional patterns could be created to detect these defects, it is often more efficient to use dc test modes. For example, a functional failure can occur when a parasitic network leakage exceeds the capacity of the standby portion of the pump system. The functional test to detect this failure would require putting the DRAM into a standby state prior to activating each address. The time required for this type of functional test is much longer than the time required to perform the dc test that checks the network for parasitic leakage. This type of dc testing requires a test-mode control that turns the pump system off and provides access to the network from the tester such that the network leakage can be measured. Another means of enhancing DRAM test coverage is to check for operating margins around the nominal set-points for these network voltages. Methods selected for exciting these margin tests could use the same tester connection as the leakage test, or the pump output voltage set-point can be modified. Providing these types of test modes and enabling test-mode activation through BIST instructions improve test effectiveness and efficiency.

Test and diagnostic capability

In the ASIC environment, where the BIST has all of the capability and flexibility to test the DRAM with minimal tester interaction, the tester must still obtain the results from the BIST. Upon completion of a functional pattern, several types of data must be acquired from the BIST. Each of these data types has been considered in designing the BIST and the rules for integration of the DRAM/BIST core into an ASIC design system. The most frequently needed type of information is pass/fail data, which is supplied by three bits: one bit to flag successful completion of the BIST sequence; one bit to separate "perfect" from "not perfect"; one bit to separate "fixable" from "not fixable." Since the design system integration rules are set up in such a way that these bits are immediately accessible, overhead for the results-unload operation is minimized. Another data type to be considered is the repair data needed for fusing. The three pass/fail bits are the first out during a serial unload operation. If the part is fixable, the redundancy solution is unloaded from the BIST and passed on to the fusing tool for permanent redundancy implementation. If the part is either perfect or not fixable, no further BIST unloading is necessary. Skipping the redundancy unload step for perfect and non-fixable parts further reduces overhead. Another data type that deserves design-for-test attention is bit-map results. The essential elements for BIST bit-mapping are the ability to identify failing cycles and acquire the data-out states for the failing cycles. Enhancements to this minimum requirement can significantly reduce the effort required to create bit-

maps. Specifically, the failing address is acquired directly, eliminating the need to create and query a cycle count to address cross-reference. The failing data-out state is flagged and positioned for easy acquisition from the BIST comparator, so the need to create and query a cycle count to expected data cross-reference is eliminated. Providing easy access to a synchronized set of address and compare states makes it possible to create bit-maps for any BIST pattern with a minimal amount of offline processing. In this scheme, BIST design-for-test efforts have had a significant positive impact on both test overhead costs and diagnostic capabilities.

In the early stages of embedded DRAM development, painful steps involving an iterative process of trial and error were taken to ensure the validity of the BIST patterns for manufacturing test. This process required the interactions of several individuals and many hours because of the lack of a methodology and an environment for test-pattern developers to use when coding and verifying patterns. Patterns were initially developed as binary vectors or strings of 1s and 0s that were loaded into the BIST scan chains. It was virtually impossible to ensure that the correct bits were set without sitting at a tester and debugging the vectors. This led to the development of a microcode for constructing ROM patterns and a software package to ensure the correctness of the binary vector and of the pattern developed.

The test-pattern development methodology involves three steps centered around logic simulation. These steps form a cyclical path of verification performed by a pattern developer. The first step involves writing the test-pattern file (TPF). Once the TPF is written, it is translated into a vector that can then be applied to the macro using a simulator. When the simulator has successfully completed, the final step involves verifying the results and extracting the initialization vector and expect vector required by manufacturing test. The final step required to produce a valid manufacturing test vector is to extract the data from the simulation graphics file. The data extractor takes as input the same scan-chain files used by the compiler and the graphics file generated from simulation. The extractor parses the graphics file and extracts the initialization vector and expect vector from a desired trigger provided as an argument. Once the data has been extracted, the vectors can be written to a file format appropriate for the tester platform to be used during manufacturing test.

Future work

Future work in this area includes development of a compiler system that expands the configurations offered and provides the customer with the ability to optimize performance, density, power, and aspect ratio. The compiler also facilitates support of multiple core designs in which each core can be tailored for the performance/density tradeoff. This will open the door for novel architectures that target specific design goals such as fast cycle or low latency. BIST enhancements include enabling the BIST to test the entire memory space without unloading the redundancy solution between repair regions. Eliminating redundancy unload is key to reducing test time and minimizing tester interaction. Maintaining the entire repair solution on-chip, combined with electrically blown fuses, enables module-level repair, further improving manufacturing productivity.

Summary

This paper has presented an overview of the macro design, architecture, and BIST implementation as part of the IBM third-generation embedded DRAM for the 0.11-μm ASIC design system (CU-11). A growable embedded DRAM architecture with a simple SRAM-like memory interface was chosen to provide customers with the highest degree of flexibility while maximizing design and characterization reuse. A robust GND sensing array core has enabled the wide range of operating conditions required by a diverse set of ASIC applications. The advantages of the built-in self-test implementation have been described, along with a series of features designed to satisfy the demanding requirements of DRAM test in a logic environment. We expect to see continuing growth of embedded DRAM utilization as performance and density continue to enable new applications.

Note: Portions of this paper are based on "A 300 MHz Multi-Banked, eDRAM Macro Featuring GND Sense, Bit-Line Twisting and Direct Reference Cell Write," by J. Barth, D. Anand, J. Dreibelbis, and E. Nelson, *2002 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, Vol. 45, © 2002 IEEE.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. J. Dreibelbis, J. Barth, H. Kalter, and R. Kho, "Processor Based Built-In Self Test for Embedded DRAM," *IEEE J. Solid-State Circuits* **33,** No. 11, 1731–1740 (November 1998).
2. T. Yabe, S. Miyano, K. Sato, M. Wada, R. Haga, O. Wada, M. Enkaku, T. Hojyo, K. Mimoto, M. Tazawa, T. Ohkubo, and K. Numata, "A Configurable DRAM Macro Design for 2112 Derivative Organizations to be Synthesized Using a Memory Generator," *IEEE J. Solid-State Circuits* **33,** No. 11, 1752–1757 (November 1998).
3. S. Crowder, R. Hannon, H. Ho, D. Sinitsky, S. Wu, K. Winstel, B. Khan, S. R. Stiffler, and S. S. Iyer, "Integration of Trench DRAM into a High Performance 0.18-μm Logic Technology with Copper BEOL," *International Electron Devices Meeting, Digest of Technical Papers*, 1998, pp. 1017–1020.
4. T. Obremski, "Advanced Non-Concurrent BIST Architecture for Deep Sub-Micron Embedded DRAM Macros," Ph.D. Dissertation, University of Vermont, Burlington, May 2001.

5. N. Watanabe, F. Morishita, Y. Taito, A. Yamazaki, T. Tanizaki, K. Dosaka, Y. Morooka, F. Igaue, K. Furue, Y. Nagura, T. Komoike, T. Morihara, A. Hachisuka, K. Arimoto, and H. Ozaki, "An Embedded DRAM Hybrid Macro with Auto Signal Management and Enhanced on Chip Tester," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, 2001, pp. 388–389, 469.
6. R. H. Dennard, "Field Effect Transistor Memory," U.S. Patent 3,387,286, June 4, 1968.
7. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *J. Design Automat. Fault-Tolerant Comput.* **2,** 165–178 (May 1978).
8. J. Dreibelbis, J. Barth, H. Kalter, and R. Kho, "Built-In Self Test for Embedded DRAM," *Proceedings of the IEEE North Atlantic Test Workshop*, West Greenwich, RI, 1997, pp. 19–27.
9. R. McConnell, U. Moller, and D. Richter, "How We Test Siemens' Embedded DRAM Cores," *Proceedings of the International Test Conference*, 1998, pp. 1120–1125.
10. R. Aitken, "On-Chip Versus Off-Chip Test: An Artificial Dichotomy," *Proceedings of the International Test Conference*, 1998, p. 1146.
11. J. Dreibelbis, J. Barth, Jr., R. Kho, and T. Kalter, "An ASIC Library Granular DRAM Macro with Built-In Self Test," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, 1998, pp. 74–75.
12. H. A. Bonges III, R. D. Adams, A. J. Allen, R. Flaker, K. S. Gray, E. L. Hedberg, W. T. Holman, G. M. Lattimore, D. A. Lavalette, K. Y. T. Nguyen, and A. L. Roberts, "A 576K 3.5ns Access BiCMOS ECL Static Ram with Array Built-in Self Test," *IEEE J. Solid-State Circuits* **27,** No. 4, 649–656 (April 1992).
13. P. Jakobsen, J. Dreibelbis, G. Pomichter, D. Anand, J. Barth, M. Nelms, J. Leach, and G. Belansek, "Embedded DRAM Built In Self Test and Methodology for Test Insertion," *Proceedings of the International Test Conference*, 2001, pp. 975–984.

Received October 23, 2001; accepted for publication July 26, 2002

John E. Barth, Jr. *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (jbarth@us.ibm.com).* Mr. Barth received the B.S.E.E. degree from Northeastern University in 1987, and the M.S.E.E. degree from National Technological University in 1992. During his B.S. degree work he was a co-op student from 1984 to 1985 at the Timeplex Development Laboratory in Rochelle Park, New Jersey, where he wrote data communications and network monitoring software. He was also a co-op student in 1986 at the IBM Development Laboratory in Essex Junction, Vermont, where he was involved in the design and characterization of the 1Mb DRAM product. After receiving his B.S. degree, Mr. Barth joined IBM at the Development Laboratory in Essex Junction, and was involved in the design of a 16Mb DRAM product featuring embedded ECC and SRAM cache. Following this, he worked on array design for the 16/18Mb DRAM product. In 1994 he began work in his current field of wide-I/O, high-performance DRAM macros for embedded applications.

Jeffrey H. Dreibelbis *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (jdreib@us.ibm.com).* Mr. Dreibelbis received the B.S. degree in electrical engineering from Lehigh University in 1973. Upon graduation, he joined the U.S. Air Force, where he served as a Communications Electronics Engineer for the USAF Communications Service at Griffiss AFB, Rome, New York, until 1977. In 1977 he joined the semiconductor development laboratory of the IBM Microelectronics Division in Essex Junction, Vermont, where he first worked in n-MOS SRAM and DRAM product design. He later became a key developer in CMOS SRAM projects, CMOS commodity DRAM designs, and embedded SRAM macros. He is currently a Senior Technical Staff Member in the IBM Advanced Memory Design group, working on the development of embedded DRAM macros for IBM ASIC offerings. Mr. Dreibelbis is a member of Tau Beta Pi, Eta Kappa Nu, and the IEEE.

Erik A. Nelson *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (eanelson@us.ibm.com).* Mr. Nelson received the B.S.Ch.E. degree from Cornell University. He joined IBM in East Fishkill, New York, in 1982, working on wafer fabrication process development for bipolar transistor products. In addition to process development, he has experience in electrical characterization and wafer manufacturing. Since 1993, Mr. Nelson has worked on DRAM product development in Essex Junction, Vermont. After working with IBM development partners on 64Mb to 256Mb commodity DRAMs, he turned his attention to embedded DRAM, contributing to product development and introduction to manufacturing for SA-27E and CU-11 products.

Darren L. Anand *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (danand@us.ibm.com).* Mr. Anand received a B.S.C.E. degree from Clarkson University in 1997. After his junior year at Clarkson, he interned with IBM Microelectronics in Essex Junction, Vermont, in 1996, working in the PowerPC development group. In 1997, he became a full-time employee of IBM Microelectronics in Essex Junction, working on high-performance 16Mb SGRAM design. In 1998, he began work in his current field, high-performance embedded DRAM macro design. In addition to eDRAM data path and array

design, Mr. Anand has worked on a 144Mb SRAM replacement based on the eDRAM macro and has done extensive work on eFuse development.

Gerald (Gary) Pomichter *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (garypp@us.ibm.com).* Mr. Pomichter received his B.S.C.E. degree from Clarkson University in 1990. He subsequently joined the IBM Workstation Division in Kingston, New York, where he was involved with high-speed 3D workstation graphics controller design. After moving to the IBM Microelectronics Division in Essex Junction, Vermont, in 1994, he helped to design the IBM family of synchronous graphics RAMs. Since 1998, he has been working on the IBM wide-I/O, high-performance DRAM macros for embedded applications.

Peter Jakobsen *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (jakobsp@us.ibm.com).* Mr. Jakobsen received the B.S.C.E. degree from Pennsylvania State University in 1997. During his B.S. degree work he was a co-op student with IBM in Essex Junction, Vermont, where he tested 4Mb SGRAM and wrote data extraction software to analyze circuit simulation data. In 1998 he joined the IBM SDRAM development group which later developed RAMBUS. In 2000 he began work in his current field of wide-I/O, high-performance DRAM macros for embedded applications.

Michael R. Nelms *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (mnelms@us.ibm.com).* Mr. Nelms received a B.S.E.E. degree from the University of Vermont in 1998. During his B.S. degree work, he was an intern at Game Financial Corporation in Plymouth, Minnesota, writing and maintaining software for a customer call center. He also was a co-op student at IBM in Essex Junction, Vermont, where he modeled 256MB DIMMs and motherboard configurations. After graduation, he joined IBM in Essex Junction, where he now works on developing embedded DRAM macros for the IBM ASICs program.

Jeffrey Leach *(current address unavailable).* Mr. Leach received a B.S. degree in computer engineering from Clarkson University in 2000. In 1998 he was a co-op student at GE Power Systems as a Y2K database design engineer. In 1999 he was a co-op student with IBM on the ASIC ROM development team writing software to automate schematic verification. At Clarkson he was a software systems architecture teaching assistant. After receiving his bachelor's degree, Mr. Leach joined the IBM Microelectronics embedded DRAM design team in Essex Junction, Vermont, as a circuit design engineer. He left IBM in November 2001.

George M. Belansek *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (georgeb@us.ibm.com).* Mr. Belansek received his B.S.E.E. degree from Pennsylvania State University in 1983, and his M.S.E.E. degree from the University of Vermont in 1990. While an undergraduate, he twice interned with IBM in the Essex Junction, Vermont, facility working in final test and DRAM design. Upon graduation he joined the IBM Microelectronics Division in Essex Junction. Mr. Belansek has held various assignments at IBM, including lead characterization engineer for IBM SRAM products, Project Manager for the IBM graphic memory (SGRAM) product line, and, currently, Design Manager of the high-performance embedded DRAM and SRAM development team.