Pseudorandom built-in self-test methodology and implementation for the IBM RISC System/6000 processor

by I. M. Ratiu
H. B. Bakoglu

This paper describes a unified self-test and system bring-up methodology. The components involved include a common on-chip processor (COP) that executes the chip self-test sequence and provides an interface to the COP bus, a serial bus (COP bus) that links the chips to OCS and ESP, an on-card sequencer (OCS) that controls the self-test and system initialization sequences, and an engineering support processor (ESP) that is used for system verification, bring-up, and debug. Almost all RISC System/6000* chips contain embedded RAMs such as register files, caches, and directories; therefore, the self-test methodology described here is particularly suitable for logic chips that contain embedded arrays. Logic and RAM self-test is executed by a control processor (COP) integrated on the chips. The COP controls the self-test sequence, generates pseudorandom test vectors, scans them into chip registers, and provides the select lines that establish a one-to-one correspondence between RAM input/output and chip registers. The COP also drives RAM read/write lines during self-test, scans the captured RAM outputs, and compresses them to obtain a signature. After the vectors are scanned in, the chip runs for one or two system cycles, the logic outputs are captured in registers, and the chip state is scanned back into the COP, where it is compressed to obtain a signature. This procedure is repeated many times, and the final signature is then compared with a predetermined "good" signature to establish whether the chip is good or bad. Special techniques are developed to improve the coverage of logic that feeds RAMs or receives its inputs from RAMs. Both ac and dc self-test are described. The self-test sequence is controlled by a program stored in the OCS, and ESP is used during system bring-up to set up break-points and to display and modify the machine state.

*RISC System/6000 is a trademark of International Business Machines Corporation.

Introduction

Pseudorandom built-in self-test (BIST) provides a common test strategy for ac and dc testing of the IBM RISC System/6000* CPU. A dedicated serial bus linking the chips to a microcontroller and a workstation allows for distributed control of the test process across the system and for the implementation of other architectural functions. The BIST architecture expands on earlier techniques for LSSD pseudorandom testing [1, 2], boundary-scan latches [3], built-in monitoring hardware for testing [4], ac testing [5], and testing of embedded memory [6–8]. An introductory overview is presented in [1], and [9] gives a detailed description.

Design approach: An overview

The BIST approach described in this paper has been applied to the chips of the IBM RISC System/6000 CPU, all of which are LSSD (level-sensitive scan design) compatible. All of these chips contain boundary-scan latches that can electrically isolate the chip from the fixture on which it resides. They also include built-in hardware that can communicate over a serial bus using a simple protocol. For BIST to work, special design rules must be followed in chip design.

The BIST architecture for the CPU board is shown in **Figure 1**. It consists of four components that are presented in detail in the rest of this paper: the common on-chip processor (COP); the serial bus linking the chips (COP bus); the on-card sequencer (OCS); and the engineering support processor (ESP).

Traditional LSSD methods suffice for chip testing but not for board testing. Isolating the failing chip during board test is impractical because of the high volume and cost of the board-test diagnostic process. Moreover, individual chips cannot be tested when mounted on the board, since surface mounting and inner signal layers in the board drastically reduce the number of access points. Finally, the described BIST approach is capable of ac-testing both logic and memory on the CPU chips at the targeted system speed.

In addition to the standard LSSD rules, the design of the CPU chips must follow a set of self-test design rules. First, every chip must contain a COP connected to the COP bus. Second, all chips must contain boundary-scan latches for all I/O pins, and these latches should operate such that under no circumstance will spurious data be latched into or triggered from these boundary-scan latches. Third, no indeterminate state should ever be latched during functional or nonfunctional operation at normal system speed, and all chip LSSD scan latches must be able to switch from scan mode to normal system mode and back in less than a cycle. Finally, embedded memories and their adjacent logic must allow for special control signals from the COP that govern read and write operations.

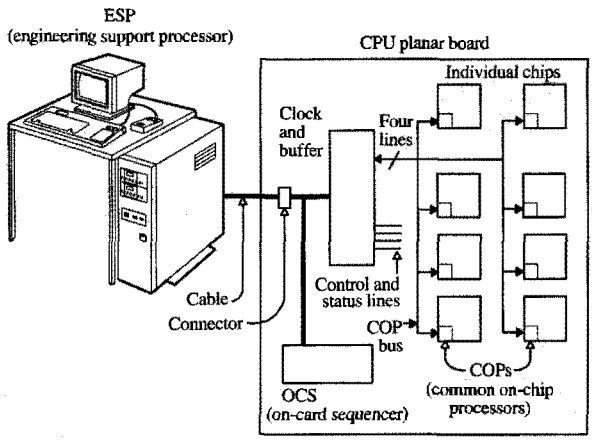

Figure 1

Built-in self-test architecture.

The embedded-RAM self-test rules are the least general in terms of implementation, since embedded RAMs are least standardized in terms of size, aspect ratio, and access time. Because the implementation of these rules can substantially degrade system performance, custom implementation is the norm.

The built-in self-test for embedded RAMs must deal with and overcome some fundamental topological and timing limitations of the memory. A memory acts as both a source and a sink for signals, and its timing in a nonfunctional environment is a random variable. Two memory operations pose serious timing problems: dynamic write-through and hot read. *Dynamic write-through* is a write to an address followed by a read from the same address that starts before the write operation is completed. *Hot read* is a memory-read cycle initiated by any change of data on its address bus. When used in normal system functions, both operations save a few nanoseconds and increase the speed of the CPU. Unfortunately, carefully architected system rates are not likely to be present during pseudorandom self-test, and embedded memories can act as sources of spurious data released at random time intervals into the system. The presence of multi-port RAMs and the logic surrounding an embedded RAM only compounds the problem. The customized implementation of the self-test design rules for RAMs prevents such data from ever being latched, and brings discipline to the read and write cycles.

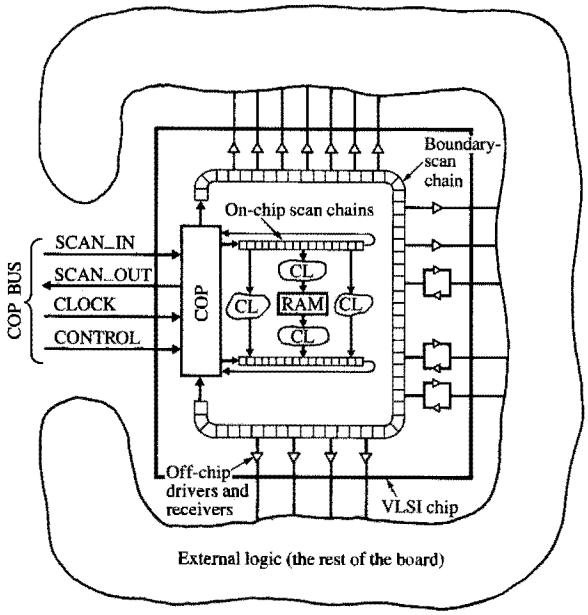

Figure 2

The common on-chip processor interface.

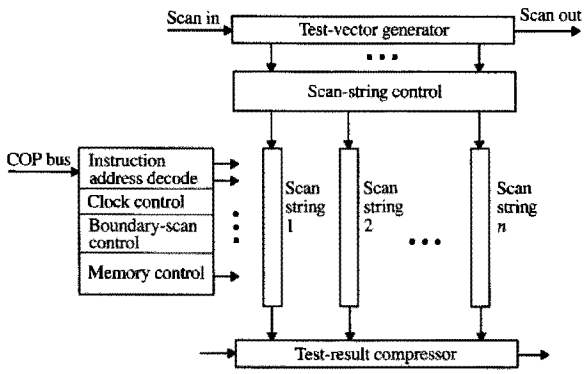

Figure 3

The common on-chip processor functional blocks.

The common on-chip processor

The common on-chip processor (COP) is a technology-independent processor that controls the built-in self-test sequence. There is a COP on every CPU chip, and a COP takes less than 3 percent of a CPU chip in this particular implementation. The COP communicates with the outside world via a four-bit serial bus according to a simple bus protocol. Together with the on-card sequencer, the COP isolates faulty CPU chips. Together with the engineering support processor, the COP also aids in the bring-up and test of CPU chips not previously tested elsewhere.

The COP interface is shown in **Figure 2**, and the major COP functional blocks are shown in **Figure 3**. The COP consists of the hardware for pseudorandom test-vector generation and test-result compression and a small processor controlling its operation. The test-vector generator is a 31-bit linear-feedback shift register that feeds the parallel LSSD scan strings with pseudorandom data. A similar register is loaded in parallel by all scan strings with the test results: Thousands (or millions) of output bits are compressed into 31-bit signatures. What happens between unloading the test-vector generator and loading the test-result compressor—dc or ac test, memory test, reset, load, etc.—is dictated by the instructions received by the small processor via the COP bus.

The COP has an internal and an external interface. The internal interface consists of a few dozen signals linking the COP to the other modules of the same chip. For example, the COP controls the data traffic to and from all LSSD scan strings, the address register of all embedded memories, and the operation of the boundary-scan latches. The external interface of the COP is the four-pin serial COP bus. Commands issued on this bus by either the on-card sequencer or the engineering support processor are processed by the COP and appropriate action is taken. The bus protocol supports both broadcast mode and individual addressing of a chip.

The COP is a versatile component employed in both system test and maintenance functions. It can perform dc and ac logic self-testing, embedded memory self-testing, dynamic burn-in testing, performance sorting, and module I/O parametric testing. For system maintenance operations, the COP is used to dump the state of the system (all LSSD latches) into nonvolatile RAM, to reset all hardware during power-up, and to debug the failing system.

For self-test, the COP controls initialization, pseudorandom data generation and test-result compression, and the clocking of the chip scan strings. It also controls the length of the self-test sequence and handles the dialogue with the COP bus controller (self-test status, seed and signature transmission). For hardware reset, the COP resets all scan strings and initializes all embedded memories. Finally, for chip bring-up and debug, the COP loads and unloads scan strings and embedded memories, halts and resumes CPU

operation, steps n cycles, and stops execution on a predefined condition (for example, parity-check error, set breakpoint, etc.).

Logic self-test

Figure 4 shows a simplified block diagram of logic self-test hardware. Pseudorandom test vectors are generated by the COP using a linear-feedback shift register (LFSR). During self-test, the COP takes over the control of the scan chains required by level-sensitive scan design (LSSD). (The COP drives the scan-control inputs of the registers and can force them into scan mode when necessary.) Accordingly, random patterns of ones and zeros from the LFSR are scanned into the chip registers under control of the COP. Once a test vector is scanned into the registers, the COP lets the chip run for one or two "function-like" cycles. During these cycles, the pseudorandom state scanned in the chip stimulates the logic, and the result is captured by the registers. Next, this new chip state is scanned back into the COP and compressed by a multiple-input signature register (MISR). This is repeated many times to ensure high fault-coverage. The final value of the MISR is the signature.

As shown in the block diagram in Figure 4, almost all RISC System/6000 chips contain RAMs embedded in combinatorial logic. Because one cannot scan test vectors directly in a RAM, a scheme must be provided to prevent the RAMs from injecting indeterminate or unrepeatable states into the combinatorial logic during a test. A convenient way to achieve this is to force the RAMs into write-through mode during logic self-test. In write-through mode, RAM output is equal to input and, as a result, the RAM behaves like a buffer. With this method, RAMs are effectively eliminated and only the combinatorial logic is tested. The COP supplies a signal labeled LOGIC_SELF_TEST, which is used to force the RAMs into write-through mode during logic self-test. This eliminates the need to initialize the RAMs with test patterns prior to logic BIST.

Built-in logic self-test can be performed at functional speed to perform ac testing to detect path-delay faults. These are faults that cause a signal to miss its prescribed timing window due to defects or parameter variations in transistors and interconnections. These defects can be introduced during fabrication or in the field (for example, threshold potential degradation due to hot electrons).

An ac logic self-test sequence is shown below.

•
•
•
SCAN
SCAN
FREEZE

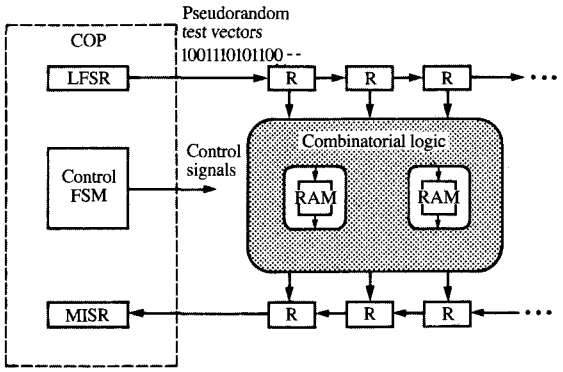

Figure 4
Logic self-test block diagram.

RUN
RUN
SCAN
SCAN
•
•
•

During ac logic self-test, vectors are scanned into the registers, and the logic is frozen for one cycle by stopping the master clock of the registers. The COP provides a signal labeled STOP_L1_CLOCK for this purpose. The freeze cycle allows for the transients after the last scan to settle down. Then the chip runs for two system cycles. The first run cycle captures the dc response of the test vector scanned into the chip, and the ac testing is performed during the second cycle. Self-test, of course, can be converted into a dc test by simply having only one run cycle instead of two.

Embedded RAM self-test

Figure 5 shows a simplified block diagram of the RAM self-test hardware. During RAM self-test, the COP provides a RAM address generated by a counter, and read/write control signals. Once a test vector is scanned into the chip registers, the COP activates the RAM write signal, and the scanned bits are written into the RAMs. This sequence is repeated multiple times as the address counter is incremented. Because the same counter is used for testing *all* the RAMs simultaneously, the counter should be big enough to cover the address space of the largest RAM on the chip. Test data is repeatedly written

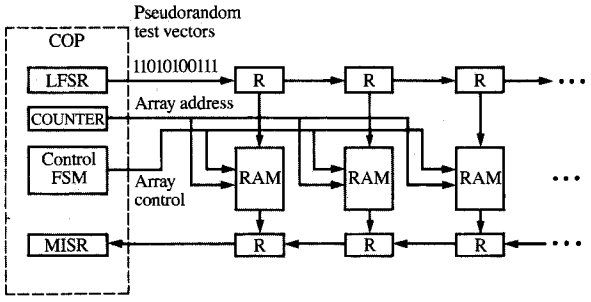

Figure 5

RAM self-test.

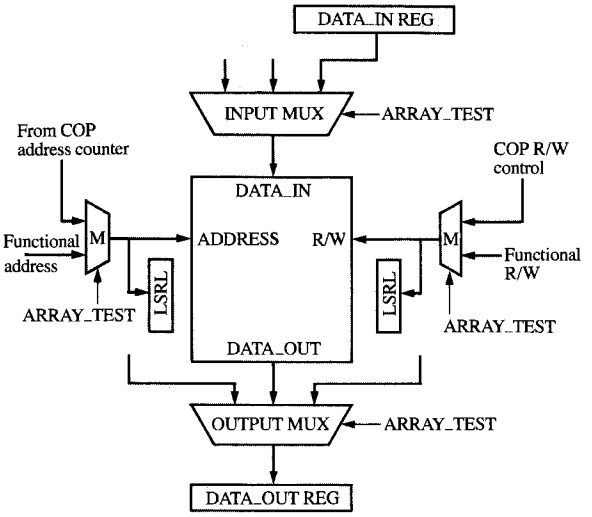

Figure 6

Built-in self-test of a RAM surrounded with combinatorial logic.

into small RAMs. After all the RAM locations are initialized with pseudorandom values, a read sequence starts. RAM read is performed similarly to RAM write, but instead of the output of the LFSR being loaded into the RAM, this time RAM outputs are captured in registers, scanned into the COP, and compressed by the MISR.

In most designs, RAMs are not necessarily bounded by registers, and functional as well as self-test address and control signals must be fed to RAMs. Control signals from the COP are used to establish a one-to-one correspondence between the RAM data input/output and the chip registers during self-test. In addition, the COP address counter and read/write controls for self-test are multiplexed with functional signals.

A typical RAM and the surrounding logic are shown in **Figure 6**. The COP supplies the following signals for RAM self-test:

- ARRAY_ADDRESS This is the address used during RAM self-test.
- ARRAY_READ/WRITE This is the read/write control for RAM self-test.
- ARRAY_TEST/FUNCTIONAL This is used as the select line for the multiplexor that picks the address and read/write signals broadcast by the COP during RAM self-test.
- INHIBIT_ARRAY_CLOCK This prevents writing into the RAMs during scanning. When this signal is activated, internal RAM clocks are disabled.
- LOAD_ARRAY_ADDRESS This is used when RAM address and read/write control are fed directly from a latch. By using this signal, one port of the address and R/W registers can be loaded with COP outputs rather than going through a multiplexor right before the RAM. This permits the flexibility of moving the multiplexor behind the latch if RAM address and R/W control lines are in the critical path.

The built-in RAM self-test can be executed at functional speed to perform ac testing. Pseudorandom vectors are scanned into the registers, and a RAM read or write operation is activated in the cycle immediately following. This way, the internal circuitry of the RAM and the functional paths between the RAM and the data in/out registers are tested at system speed. Self-test, of course, can be converted into a dc test by simply waiting for one additional cycle before reading or writing into the RAM. Then, the RAM input signals settle down during the wait cycle, and the test of the path between the scan latches and the RAM essentially becomes dc.

On-card sequencer

The on-card sequencer (OCS) is an 8-bit 8051 microcontroller with a 4-Kbyte on-chip ROM and 128 bytes of on-chip RAM. The ROM stores the "seed-good" signature pair for a pseudorandom test sequence. The good signature is obtained either through simulation or

by using a "golden model" approach. Both seed and signature for a sequence are 31 bits long; hence, several hundred can be stored in the 4-Kbyte on-chip ROM. For system function, the OCS addresses external memory on the CPU board for system operation and maintenance: 64 Kbytes of ROM and 16 Kbytes of nonvolatile RAM. It responds to the reset button on the operator panel and reports errors on the operator panel display.

The main function of the OCS is CPU self-test and reset. Because the OCS contains both seed and good signatures for a test sequence, it can control an entire self-test sequence by sending the proper commands to the COP of each chip. At power-up, after both supply voltage and system clock are stable, the OCS broadcasts a set of commands from its on-chip ROM onto the COP bus. First, all three-state output drivers for the CPU chips are disabled, thus electrically isolating each chip from its neighbors. From this point on, all CPU chips operate in parallel. The following sequence is executed:

1. Initalize the COPs.
2. Initialize all embedded memories.
3. Test embedded memories.
4. Self-test the dc logic.
5. Self-test the ac logic.

After each operation, the OCS polls the individual COPs. If all signatures match the stored ones in the OCS ROM, the CPU is reset for system initialization.

If the signature generated by a sequence does not match the "golden" one in the OCS ROM, a suitable error code is flashed on the operator panel display. The size of the sequence and the number of stored golden signatures determine the resolution of the OCS self-test. For example, self-test can identify a specific failing embedded memory on a chip.

Each test operation in the above sequence may involve several million system cycles. Regardless of how many cycles are executed, the test output data is always compressed to a 31-bit signature. Since seed and signature are less than four bytes each, the speed and bandwidth of the four-pin serial COP bus are adequate.

The CPU board self-test unequivocally identifies all failing chips. Since the I/O circuitry is not exercised, the chip-to-chip connections must be tested later (during the functional self-test for the entire system).

Engineering support processor

The engineering support processor (ESP) is a stand-alone processor used for verification, bring-up, and debug. Unlike the OCS, which is part of the product, the ESP is not shipped with the product. It consists of an IBM RT System equipped with a COP bus interface adapter and the application software that controls its operation. The COP bus interface adapter acts as a protocol converter between the native AT bus of the RT and the serial COP bus of the RISC System/6000 CPU board. The ESP is intended to operate at a debug station in manufacturing or in a laboratory; it can be up to 30 meters away from the CPU board under test.

Because the ESP implements the full set of COP commands, it can run all the functions of the OCS. The additional functions of the ESP allow it to aid in the isolation of faults during system verification and debug, operations that require larger memory and disk storage than are available in the OCS:

- Control of architectural verification programs.
- Hardware debug for chip, board, and system.
- Software debug of the operating system.

The ESP software runs under the AIX operating system and consists of 25 000 lines of C code. It takes about 15 Mbytes of file space on the RT and runs in 800 Kbytes of RAM. The interface makes extensive use of windowing and can be tailored by the user.

To support a new chip with a COP, only three parameters need to be passed to the ESP software: the LSSD scan string tables, the embedded memory definition, and the new screen definition for the user. Once the new breakpoints are set, the debug process can start exactly as it does for any other chip with a COP. This remarkable consistency has resulted in a reduction in CPU chip bring-up time from a few months to a few weeks.

Conclusions

A novel pseudorandom built-in self-test approach has been designed and implemented for the IBM RISC System/6000 CPU. The presence of the common on-chip processor on each CPU chip, the serial bus linking all CPU chips to the on-card sequencer, and the simple bus protocol allow the test environment to be replicated from chip to field, and the test process is therefore easy to set up and run in the field. The actual self-test isolates the failing chip, and, if desired, can identify failing embedded memories within the chip. The logic on the chip is tested for both dc stuck faults and ac faults at system speed. The built-in self-test architecture and approach described here can easily be applied to any board containing several VLSI LSSD chips.

Acknowledgments

The built-in self-test approach described in this paper, and its implementation, reflect the contributions of several of our colleagues, whose assistance is gratefully acknowledged: T. Jaber for the common on-chip

processor, W. Tuten for the on-card sequencer and bus protocol, D. Gregoire and K. Shah for the implementation of logic and embedded memory self-test, and D. Cawthron for the engineering support processor software.

References

1. E. J. McCluskey, *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1986.
2. P. H. Bardell and W. H. McAnney, "Parallel Pseudorandom Sequences for Built-In Test," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1984, IEEE Catalog No. 84CH2084-2, pp. 302–308.
3. D. Komonytsky, "LSI Self-Test Using Level Sensitive Scan Design and Signature Analysis," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1982, IEEE Catalog No. 82CH1808-5, pp. 414–424.
4. J. J. LeBlanc, IBM Advanced Workstations Division, Austin, TX, "LOCST," private communication.
5. G. L. Smith, "Model for Delay Faults Based upon Paths," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1985, IEEE Catalog No. 85CH2230-1, pp. 342–349.
6. W. H. McAnney, P. H. Bardell, and V. P. Gupta, "Random Testing for Stuck-At Storage Cells in an Embedded Memory," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1984, IEEE Catalog No. 84CH2084-2, pp. 157–166.
7. W. H. McAnney, J. Savir, and S. R. Vecchio, "Random Pattern Testing for Data-Line Faults in an Embedded Memory," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1985, IEEE Catalog No. 85CH2230-1, pp. 100–105.
8. J. Savir, W. H. McAnney, and S. R. Vecchio, "Random Pattern Testing for Address-Line Faults in an Embedded Memory," *Proceedings of the International Test Conference*, IEEE Computer Society, Philadelphia, 1985, IEEE Catalog No. 85CH2230-1, pp. 105–114.
9. P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI Pseudorandom Techniques*, John Wiley & Sons, Inc., New York, 1987.

Received February 12, 1989; accepted for publication January 29, 1990

Ion M. Ratiu *IBM Advanced Workstations Division, 11400 Burnet Road, Austin, Texas 78758.* Dr. Ratiu studied electrical engineering at the Polytechnic Institute of Timisoara, Romania. He received his M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley in 1980 and 1983, respectively, for work in semiconductor process simulation and testability analysis and redundancy identification in VLSI circuits. After a brief postdoctoral fellowship at the University of Hannover, FRG, he joined IBM in 1983 in Advanced Engineering in Manufacturing at Poughkeepsie, New York, where he worked on pseudorandom ac self-test of digital networks. Since 1986, Dr. Ratiu has been with the Advanced Workstations Division in Austin, Texas, where he has led a second-generation self-test approach and has managed a high-end architectural simulation group. Dr. Ratiu is an active member of the IEEE and of the IEEE Test Technology Committee of the Computer Society. He currently serves as coordinator of the tutorial program of the International Test Conference.

H. B. Bakoglu *IBM Advanced Workstations Division, 11400 Burnet Road, Austin, Texas 78758 and IBM Research Division. Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Bakoglu received the B.S.E. degree in electrical engineering and computer science in addition to engineering physics from Princeton University in 1982, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1984 and 1986, respectively. He joined the IBM Thomas J. Watson Research Center in 1986, and is currently on assignment with the IBM Advanced Workstations Division in Austin, Texas, where he is a Systems Architect in the Systems Engineering and Architecture area. Dr. Bakoglu has received many awards, including the IBM graduate fellowship, the Jeffrey O. Kephard Prize in engineering physics, the Treen Scholarship, and the Princeton University Jadwin and Hall-Mercer scholarships. He has served on the program committee of the IEEE International Solid-State Circuits conference, has published numerous papers on VLSI design, and holds patents on semiconductor devices. Dr. Bakoglu is a member of Phi Beta Kappa, Sigma Xi, and Tau Beta Pi, and is the author of *Circuits, Interconnections, and Packaging for VLSI* (Addison-Wesley, 1989).