MOTOROLA
SEMICONDUCTOR APPLICATION NOTE

Order this document by AN1554/D

AN1554

SRAM Built-in Self Test

Prepared by: Steve Bradley
1300 N. Alma School Rd.
Chandler Arizona

INTRODUCTION

Built-in Self Test (BIST) is a test methodology in which on-chip test circuitry automatically and efficiently tests memories and other devices for various types of faults.

SRAM BIST can provide many advantages:

1. Cycle time is reduced by greatly decreasing the test pattern generation time of specialized vectors for embedded RAM verification.
2. Embedded RAMs can be quickly tested with a very high fault coverage. High test coverage can be obtained without having direct access to the RAMs.
3. Cost is reduced by minimizing tester requirements and reducing vector counts.

Motorola ASIC has developed two implementations of SRAM BIST. A Comparator BIST provides the highest fault coverage and best test capabilities. The Comparator BIST implements a deterministic test algorithm capable of 100% fault detection. The Comparator BIST provides this superior performance with the penalty of greater circuit complexity and a high gate count. The Pseudo-Random BIST uses a Linear Feedback Shift Register (LFSR) to generate a unique test signature. The Pseudo-Random BIST provides adequate fault coverage with minimum complexity and a low gate count.

Motorola ASIC requires that all embedded RAMs have BIST. The Pseudo-Random BIST provides sufficient fault coverage to fulfill the RAM-BIST requirement. Table 1 identifies differences between Pseudo-Random BIST and Comparator BIST.

Motorola's ASIC BIST Features:

1. Significantly reduces tester use and cost
2. 100% fault detection (using Comparator BIST)
3. At-speed test to 40 MHz (using Comparator BIST)
4. Independent of memory architecture and technology
5. Verifies both memory and BIST circuitry
6. Configurable to any memory size
7. JTAG scan, ATPG compatible
8. BIST test vector count can be compressed by 1000 to 1 ratio

Table 1. Comparison of Pseudo-Random BIST and Comparator BIST

Feature	Pseudo-Random BIST	Comparator BIST
Application	meets RAM BIST requirement	high reliability/performance
Algorithm	LFSR pseudo-random signature	Deterministic 21 N march
Gate Requirement *	907	1282
Maximum Fault Coverage	99%	100%
At Speed Testing (40 MHz)	< 64 words	Total Memory
Test Time *	16 msec.	4 msec.
Simulation Time **	0.8 hr.	3 hrs.
RAM Diagnostic Capability	No	Yes
Clear Memory	No	Yes
Fault Location	No	Yes
Address Stress Test	No	Yes

* Data based on 1024x8 diffused SRAM for comparison

** Performance based on 2MIPs.

This section is intended to provide a general understanding of the capabilities of the two BIST implementations. For complete descriptions please refer to the application note: Embedded RAM BIST (AN1502).

1. Pseudo-Random BIST Description

The Pseudo-Random BIST design is based upon two soft macros (ADDR_CELL, DATA_CELL) which are included in the H4CP & H4EP library. See Figure 1. These one bit macros can be used to construct a pseudo-random BIST circuit for any Motorola supplied RAM block.

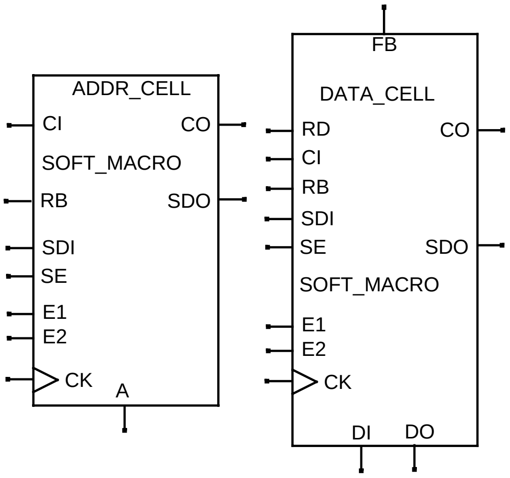


Figure 1. Pseudo-Random BIST Soft Macro Symbols

ADDR_CELL macro I/O signal description:

ADDR_CELL-CK	input	BIST Clock
ADDR_CELL-RB	input	BIST reset - active low
ADDR_CELL-CI	input	Counter carry in - daisy chain carry
ADDR_CELL-SDI	input	Scan path serial data input
ADDR_CELL-SE	input	Scan path scan enable
ADDR_CELL-E1	input	Scan path enable 1
ADDR_CELL-E2	input	Scan path enable 2
ADDR_CELL-A	output	BIST Address bit
ADDR_CELL-C0	output	Counter carry out - daisy chain carry
ADDR_CELL-SDO	output	Scan path data out

DATA_CELL macro I/O signal description:

DATA_CELL-DI	input	BIST data input - from RAM
DATA_CELL-RB	input	BIST reset - active low
DATA_CELL-CK	input	BIST clock
DATA_CELL-RD	input	LFSR read mode operation
DATA_CELL-FB	input	LFSR feedback input
DATA_CELL-CI	input	LFSR carry input
DATA_CELL-SDI	input	Scan path serial data In
DATA_CELL-SE	input	Scan path scan enable
DATA_CELL-E1	input	Scan path enable 1
DATA_CELL-E2	input	Scan path enable 2
DATA_CELL-DO	output	BIST data output - to RAM
DATA_CELL-CO	output	LFSR carry output
DATA_CELL-SDO	output	Scan path serial data out

The customer, by the BIST module construction, will control: test fault coverage, probability of aliasing the data signature, and circuit performance. Connection to scan based architecture or JTAG will require additional customer consideration. Extra logic will be required if the BIST is to test multiple RAMs or various sized RAMs.

1.1 Pseudo-Random BIST Functional Block Description

1. Address Block- address counter
2. Address Block- read/write bit
3. Address Block- pass counter
4. Address Block- stop bit
5. Data Block- data generator/analyzer
6. RAM mux- external mux for RAM address, data, and control lines

See Figure 2.

The address counter will increment the RAM address from zero to the number of RAM words. For RAMs with non-binary word counts (NOT 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, etc.) additional "end of count" carry/reset logic will be required.

The read/write bit will toggle after the address counter has reached its maximum count. Initially the read/write bit will be reset to "write". After the entire RAM has been written, the bit toggles and the next count through the RAM address field will "read". A single write/read cycle through the entire RAM address field is considered one pass.

The pass counter increments at the completion of each complete RAM write/read cycle pass. The level of fault coverage is determined by this block. The fault coverage increases with the number of test patterns written to and read from the RAM. Motorola ASIC recommends a count of 64 for the pass counter limit.

The stop bit will toggle after the final pass through the ram. Initially this bist is reset to the "test", toggling to the "test_complete" state at the end of BIST test.

The data generator/analyzer will generate a pseudo-random pattern in its Linear Feedback Shift Register (LFSR). During the BIST "write" mode, the LFSR generates a pseudo-random data pattern for the RAM. During the BIST "read" mode, the data from the RAM will be fed into the same LFSR to build the RAM test signature. At the completion of BIST testing the value of this signature can be serially shifted from the BIST for comparison. If a serial shift is not desired, comparing the carry out pins of the data generator/analyzer DATA_CELLs against a known good value can generate a "test_pass/test_fail" signal.

The RAM mux is required for H4CP & H4EP diffused and metallized RAMs. The RAM mux will select RAM input signal from either the BIST or the customer's design. The "BIST select" signal from the stop bit will select between the RAM receiving signals from the pseudo-random BIST or the customer's design.

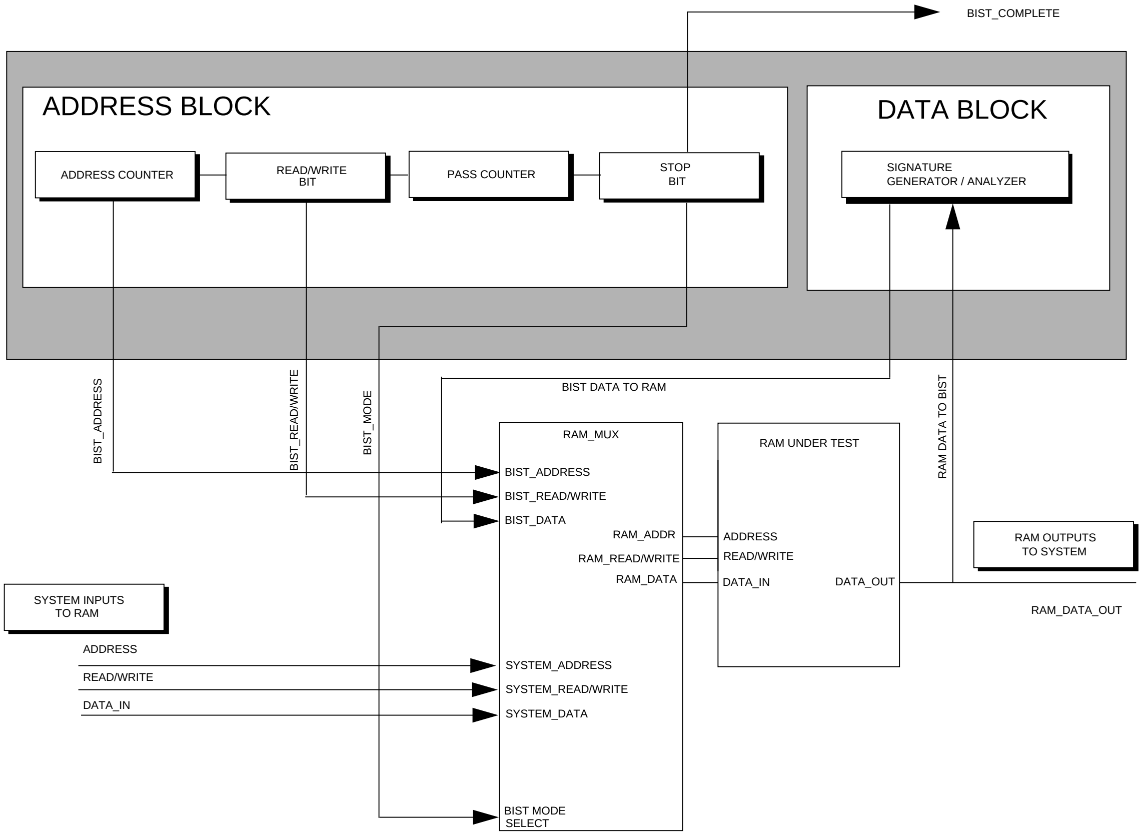


Figure 2. PSEUDO-RANDOM BIST FUNCTIONAL BLOCK DIAGRAM

1.2 Pseudo-Random BIST Algorithm

RAM Address	read/write	pass_counter	stop_bit	Data Generator/Analyzer		
					Pass 1	
0	0 (write)	0	0 (testing)	Reset data value		write
1	0 (write)	0	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	0 (write)	0	0 (testing)	pseudo-random signature		
Nword-1	0 (write)	0	0 (testing)	pseudo-random signature		
0	1 (read)	0	0 (testing)	pseudo-random signature		read
1	1 (read)	0	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	1 (read)	0	0 (testing)	pseudo-random signature		
Nword-1	1 (read)	0	0 (testing)	pseudo-random signature		
					Pass 2	
0	0 (write)	1	0 (testing)	pseudo-random signature		write
1	0 (write)	1	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	0 (write)	1	0 (testing)	pseudo-random signature		
Nword-1	0 (write)	1	0 (testing)	pseudo-random signature		
0	1 (read)	1	0 (testing)	pseudo-random signature		read
1	1 (read)	1	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	1 (read)	1	0 (testing)	pseudo-random signature		
Nword-1	1 (read)	1	0 (testing)	pseudo-random signature		
:	:	:	:	:		
:	:	:	:	:		
:	:	:	:	:		
:	:	:	:	:		
:	:	:	:	:		
					Pass 64	
0	0 (write)	final (63)	0 (testing)	pseudo-random signature		write
1	0 (write)	final (63)	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	0 (write)	final (63)	0 (testing)	pseudo-random signature		
Nword-1	0 (write)	final (63)	0 (testing)	pseudo-random signature		
0	1 (read)	final (63)	0 (testing)	pseudo-random signature		read
0	1 (read)	final (63)	0 (testing)	pseudo-random signature		
:	:	:	:	:		
Nword-2	1 (read)	final (63)	0 (testing)	pseudo-random signature		
Nword-1	1 (read)	final (63)	0 (testing)	pseudo-random signature		
0	0 (write)	0	1 (done)	BIST test signature		

1.3 Fault Coverage

The pseudo random BIST uses a Linear Feedback Shift Register (LFSR) to generate the data pattern stimulus for the RAM test and to store the RAM test signature. This is a statistical method where pseudo-random data patterns are used to test for hard memory faults. The test patterns do not test for specific types of hard faults, but are used to exercise the memory devices for correct functionality. As a result the fault coverage is controlled by two factors,

(a) the probability of the stimulus testing for a particular fault,

(b) the probability of the faulty signature not being the same as the good one.

The total fault coverage is found by multiplying these two fault coverages together.

1.3.1 Stimulus Fault Coverage

Three types of RAM faults are generally considered:

1. Stuck at - The value of a data bit is stuck either high or low and cannot be changed.
2. State coupling - The value of one data bit A is controlled by the state of another data bit B. This fault is unidirectional so that while A is affected by B, B is unaffected by A. This fault will cause A to always be a constant function of data bit B. The function may either be normal (such that A = B) or inverting (such that A = not B).

3. Transition coupling - Transition of one data bit A cause a state change in data bit B. This is similar to State coupling, except that it is the actual transition of data bit B that changes the contents of data bit A.

The Pseudo Random BISTs data generator does not create test patterns specifically to test each fault type. The "randomness" of the test pattern prevents a deterministic calculation of fault coverage. Instead the fault coverage is derived statistically.

The fault coverage increases with the number of passes through the RAM.

Table 2 below shows the probabilistic fault coverage for each type of fault over a range of passes.

Table 2. Pseudo Random BIST estimated fault coverage

Passes	1	2	4	8	16	32	64
Stuck At	50%	75%	94%	99%	100%	100%	100%
State Coupling	25%	37%	47%	49%	50%	50%	50%
Transition Coupling	6%	11%	20%	32%	44%	49%	50%

The coupling faults have a maximum coverage of 50% because the address counter only increments, so no faults can be found which effect higher addresses.

The apparently low fault coverage for coupling faults is not a severe limitation because of the fault clustering that is observed during manufacture.

Table 2 was constructed using the standard simplistic assumption that one (1) fault will exist. In practice a physical defect will normally cause multiple faults which are clustered within a small area. The detection of any one of these faults is sufficient to determine that the RAM is failing.

1.3.2 Signature Aliasing

Aliasing occurs when a signature resulting from a "failed" test matches the expected "good" signature of a passed test. The probability of aliasing is determined by the formula:

$$P(alias) = \frac{1}{(2^n)-1}$$

where "n" is the number of signature bits. The probability of signature aliasing decreases as the number of signature bits in the data generator/analyzer section increases. The probability of signature aliasing is determined by the BIST construction (i.e. the customer). **Table 3** shows the aliasing probability for different signature lengths.

Motorola ASIC is currently recommending using a minimum of 12 signature bits in the data generator/analyzer section.

Table 3. Pseudo Random BIST aliasing probability for different signature lengths

signature bits	aliasing probability	Probability of NOT aliasing
1	50.000%	50.000%
2	33.333%	66.667%
3	14.286%	85.714%
4	6.667%	93.333%
5	3.226%	96.774%
6	1.588%	98.412%
7	0.787%	99.213%
8	0.392%	99.608%
9	0.196%	99.804%
10	0.098%	99.902%
11	0.049%	99.951%
12	0.024%	99.976%
13	0.012%	99.988%
14	0.006%	99.994%
15	0.003%	99.997%
16	0.002%	99.998%
17	0.001%	99.999%

1.3.3 Pseudo-Random BIST construction.

Gate count estimation of a Pseudo-Random BIST

A gate count estimate can be generated by counting the required ADDR_CELL and DATA_CELL macros in the pseudo-random BIST. Logic for generating BIST control signals and the RAM mux have been ignored.

	Number of Macros Required	**ADDR_CELL**	**DATA_CELL**	
one	ADDR_CELL for each address line of the RAM			
one	ADDR_CELL for the read/write control bit	1		
six	ADDR_CELL for the pass counter (64 passes)	6		
one	ADDR_CELL for the stop bit	1		
	choose the greater number of DATA_CELLs a) one DATA_CELL for each RAM data line b) 12 DATA_CELLs to minimize signature aliasing			
	sum total macros			
	equivalent gates per macro	x20	x27	
	sum of equivalent gates	+	=	
				Total

Construction of a Pseudo-Random BIST

The construction of a pseudo-random BIST will be described by drawing its schematic. Associated diagrams are of a BIST for testing a generic 32 word x 8 bit RAM. Please refer to Figure 5.

Placement of ADDR_CELL macros for the Address Block. Please refer to **Figure 3**.

1. Address_counter place one ADDR_CELL macro for each address bit of the RAM in a row from left to right. The left macro will be for the RAM address LSB. The right macro will be for the RAM address MSB.
2. Read/write controller place one ADDR_CELL to the right of the address counter MSB ADDR_CELL macro.
3. Pass_counter place six ADDR_CELLs to the right of the read/write controller ADDR_CELL. This will build a 64 test pass counter.
   The number of test passes is defined by:
   $$N_{(passes)} = (2^n)$$
   where "n" is the number of ADDR_CELLs macros in the pass_counter.
4. Stop_bit places one ADDR_CELL macro to the right of the pass_counter.

Connecting signals for the Address Block

1. Connect all ADDR_CELL-SE inputs to a common BIST Scan Enable control signal.
2. Connect all ADDR_CELL-E1 inputs high (VDD). (The pseudo-random BIST doesn't use this scan control.)
3. Connect all ADDR_CELL-E2 inputs high (VDD). (The pseudo-random BIST doesn't use this scan control)
4. Connect all ADDR_CELL-CK inputs to a common clock node. The BIST clock should be a separate clock and not tied to a system wide clock tree. It may be necessary to delay this clock to fulfill RAM address hold time constraints. This delay will be design dependent.
5. Connect the ADDR_CELL-SDI input and ADDR_CELL-SDO output.

5a) Address_counter LSB ADDR_CELL-SDI is the serial data input to the pseudo-random BIST block (BIST_SDI). It may be necessary to delay this signal due to scan path hold time considerations.

5b) Connect the ADDR_CELL-SDO output to the ADDR_CELL-SDI input of ADDR_CELL macro to the right.

5c) The stop_bit ADDR_CELL-SDO will later connect to the SDI input of the "data_generator/ analyzer".

6. Connect the ADDR_CELL-CI input and ADDR_CELL-CO output to configure the carry chain.

6a)The ADDR_CELL-CI input of the address_counter LSB macro (left most) will be driven by the signal "ADRBLK_BC".This signal will be defined in step 11.

6b)The remaining ADDR_CELL macros in the address counter will have their ADDR_CELL-CI inputs driven by the ADDR_CELL-CO output of the macro to the left.

6c)The Read/write_controller ADDR_CELL-CI input may be driven by one of two sources. First determine if the RAM has an "even power of 2" number of words (i.e. 32, 64, 128, 256, 512, 1024...).

If so,the read/write_controller ADDR_CELL-CI input will be driven by the address_ counter MSB macro's ADDR_CELL-CO output.

If not,the read/write_controller ADDR_CELL-CI input will be driven by logic which detects the address count reaching "number of RAM words -1".

6d)For the ADDR_CELL-CI inputs of the pass_counter and stop_bit, drive the CI inputs with the CO output from the ADDR_CELL macro on the left.

7. Connecting the ADDR_CELL-RB input for address block reset and end of count functions.

7a) Connect the ADDR_CELL-RB input of the read/ write_controller, pass_counter, and the stop_bit macros to a common node driven by the BIST reset signal.

7b) The address counter ADDR_CELL-RB input may be driven by one of two sources. First determine if the RAM has an "even power of 2" number of words (i.e. 32, 64, 128, 256, 512, 1024...).

If so, the address counter ADDR_CELL-RB input will be driven by the BIST reset signal.

If not, the address counter ADDR_CELL-RB input will be driven by the BIST reset signal - AND-logic which detects the address count reaching "number of RAM words -1"
(active low)

8. Identify the BIST address lines. The address_counter ADDR_CELL-A outputs will go to the RAM_mux as the BIST address bus.

9. Generate the output signal "RWB" (read/ write_bar) by "OR"ing the read/write_controller ADDR_CELL-A output with the BIST Scan Enable control signal to generate the BIST RWB signal.

10. Generate the output signal "BSB" (BIST select_bar) by "AND"ing the stop bit ADDR_CELL-A output with the inverse of the BIST Scan Enable control signal.

11. Generate the output signal "ADRBLK_BC" (BIST complete) by inverting the stop_bit ADDR_CELL-A output. This signal was reference in step "6a".

Note:

If the carry chain used to enable the address counters limits the high speed operation of the BIST, the use of carry look ahead logic in the address_counter and pass_counter may be useful.

The above BIST configuration will drive the RAM inputs from the BIST circuit during scan mode. This is used when scan chains, other than the BIST, exist in the design. Scan operation forces the BIST control signal "BSB" into "BIST_mode" and the BIST control signal "RWB" to "read". This will prevent spurious writes and corruption of the RAM content during scan. If it is desirable to drive the RAM by the standard system inputs

a) generate the BIST control signal "RWB" directly from read/write_controller ADDR_CELL-A output.

b) generate the BIST control signal "BSB" by "OR"ing the stop bit ADDR_CELL-A output with the BIST Scan Enable control signal.

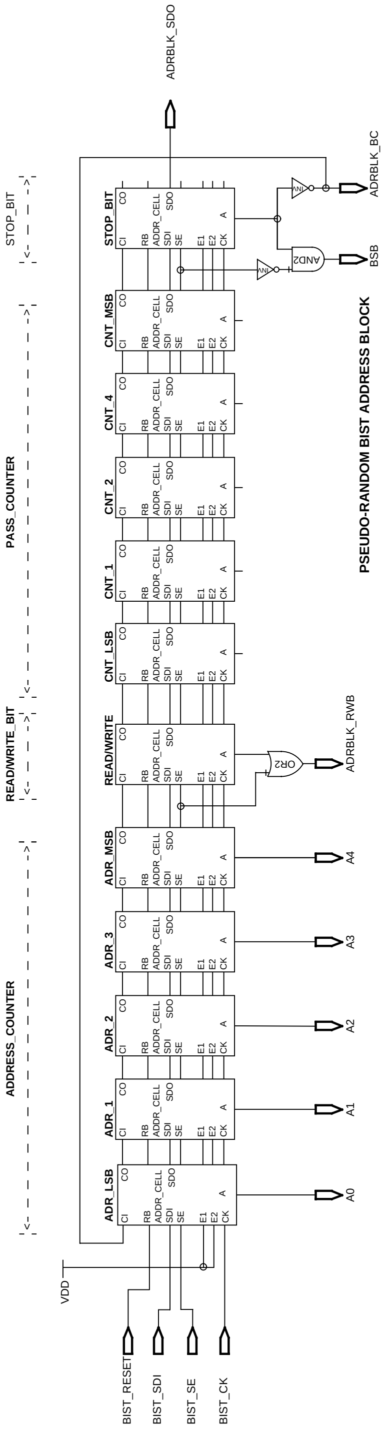


Figure 3. Pseudo Random BIST Address_Block

Placement of DATA_CELL macros for the Data Generator/Analyzer. Please refer to Figure 4.

1. Place twelve DATA_CELL macros in a row from left to right. The probability of signature aliasing is considered acceptable with twelve bits. (0.0024%) refer to Table 3.
2. If RAM has more than twelve data bits, place one DATA_CELL for each RAM data bit greater than 12 to the right of the previously placed DATA_CELLs.
3. For RAMs with more than 32 data bits, split data_generator/analyzer block into multiple LFSR shift registers with separate MLLFSR feedback connections. This will make it possible to predict the proper signature before simulations. *Contact ASIC Option Development Engineering for details.*

Note:

There should always be 12 DATA_CELL macros or one DATA_CELL macro for each RAM data bit, whichever is greater.

Connecting signals for the Data_Generator/Analyzer Block

1. Connect all DATA_CELL-RD inputs to a common node. Connect to the signal "ADRBLK_RWB".
2. Connect all DATA_CELL-RB inputs to a common node. Connect this node to the BIST reset signal (active low).
3. Connect all DATA_CELL-SE inputs to a common node. Connect to the BIST scan path enable signal (active high).
4. Connect all DATA_CELL-E1 inputs to a common node. Connect to the signal "ADRBLK_BC". It selects function of the data_generator/analyzer during BIST testing.
5. Connect all DATA_CELL-E2 inputs high (VDD). This function is not used in the standard pseudo-random BIST circuit.
6. Connect all DATA_CELL-CK inputs to a common clock node. The BIST clock should be a separate clock and not tied to a system wide clock tree. It may be necessary to delay this clock to meet RAM address hold time constraints. Required delay will be design dependent.
7. Connect the data_generator/analyzer carry chain.

7a) Connect the left most DATA_CELL-CI input low (VSS).

7b) Connect DATA_CELL-CO output to the adjacent DATA_CELL-CI input of the macro to the right.

7c) The right most DATA_CELL-CO output will be inverted and used to drive the LFSR feedback tap connections DATA_CELL-FB.

8. Connection the data_generator/analyzer scan path.

8a) Connect the left most DATA_CELL-SDI input to the signal "ADRBLK_SDO".

8b) Connect DATA_CELL-SDO output to the adjacent DATA_CELL-SDI input of the macro to the right.

8c) The right most DATA_CELL-SDO output will be the BIST Scan Data Out signal "BIST_SDO".

9. Connection the data_generator/analyzer data out bus "DO" (from BIST to RAM_MUX)

9a) Connect the left most DATA_CELL-DO output to the RAM mux LSB BIST data input.

9b) Continue right connecting DATA_CELL-DO output to the next highest RAM mux BIST data input.

9c) If the LFSR is wider than the RAM data bus, leave extra DATA_CELL-DO output unconnected.

10. Connecting the data_generator/analyzer data in bus "DI" (from RAM to BIST)

10a) Connect the right most DATA_CELL-DI input to the RAM's LSB data out signal. This is the reverse direction as the BIST DO bus.

10b) Continue left connecting DATA_CELL-DI inputs to the next highest RAM data out signal.

10c) If the LFSR is wider than the RAM data bus, connect the extra DATA_CELL-DI inputs low (VSS).

11. Connect of the data_generator/analyzer LFSR feedback

11a) Connect the appropriate DATA_CELL-FB inputs to the inverted right most macro DATA_CELL-CO output. See Table 4 for LFSR feedback tap connections for the sizes of Data_Generator/Analyzer registers.

11b) Connect the remaining unconnected DATA_CELL-FB inputs low (VSS).

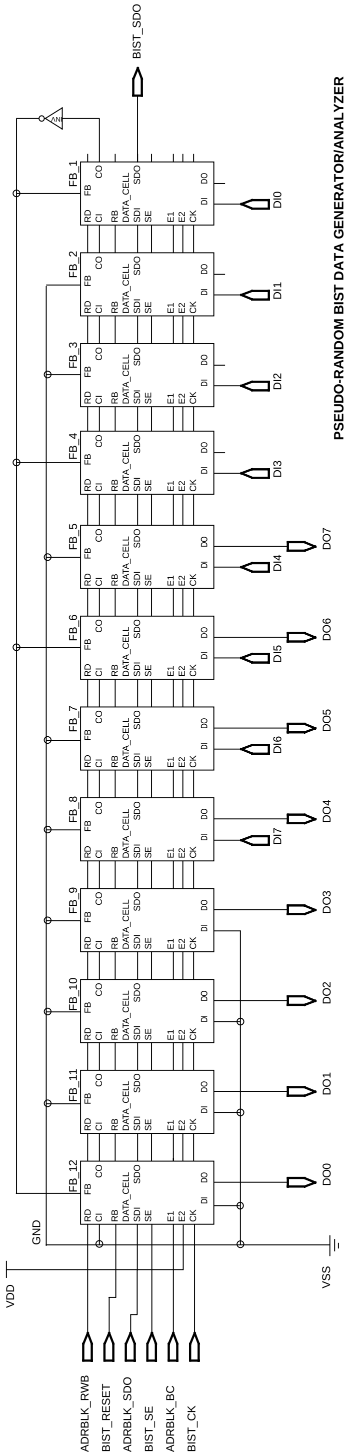


Figure 4. Pseudo Random BIST Data_Generator/ Analyzer Block

Table 4. LFSR Feedback Connections Table

LFSR bits	Sequence Length	Feedback Taps	Feedback Taps
12	4095	[12,6,4,1]	[12,9,3,2]
13	8191	[13,4,3,1]	[13,10,9,7,5,4]
14	16384	[14,12,2,1]	[14,12,11,1]
15	32767	[15,1]	[15,4]
16	65535	[16,12,3,1]	[16,12,9,6]
17	131071	[17,3]	[17,3,2,1]
18	262143	[18,7]	[18,10,7,5]
19	524287	[19,5,2,1]	[19,13,8,5,4,3]
20	1048575	[20,3]	[20,9,5,3]
21	2097151	[21,2]	[21,14,7,2]
22	4194303	[22,1]	[22,9,5,1]
23	8388607	[23,5]	[23,17,11,5]
24	16777215	[24,7,2,1]	[24,4,3,1]
25	33554431	[25,3]	[25,3,2,1]
26	67108863	[26,6,2,1]	[26,22,21,16,12,11,10,8,5,4,3,1]
27	134217727	[27,5,2,1]	[27,18,11,10,9,5,4,3]
28	298453455	[28,3]	[28,13,11,9,5,3]
29	536870911	[29,2]	[29,20,11,2]
30	1073741823	[30,23,2,1]	[30,6,4,1]
31	2147483647	[31,3]	[31,3,2,1]
32	4294967265	[32,22,2,1]	[32,7,5,3,2,1]

Note:

Table 4 has been copied with corrections from: 'Spread Spectrum Systems', by Robert C. Dixon.

For any particular length MLLFSR there is more than one set of connections that will create a maximal length sequence. The connections vary depending upon the MLLFSR length. This table contains a list of two possible feedback taps for a number of different length MLLFSRs.

WARNING: The Feedback Taps numbers are ordered from right to left. This is the same direction as the DATA_IN numbering order, however, Feedback Taps numbers start with "1" not "0" as DATA_IN.

Complete BIST connections - Address Block to Data_Generator/Analyzer.

These connections are listed for completion. Many of them have already been made.
Please refer to Figure 5.

1. Connect the BIST reset "BIST_RESET" to the appropriate node in the address and data block.
2. Connect the BIST scan enable "BIST_SE" to the appropriate node in the address and data block.
3. Connect the BIST serial data input "BIST_SDI" the address counter's LSB ADDR_CELL-SDI input.
4. Connect the BIST clock "BIST_CK" to the ADDR_CELL-CK and DATA_CELL-CK inputs.
5. Connect the address block stop_bit ADDR_CELL-SDO output "ADRBLK_SDO" to the data block left most DATA_CELL-SDI input.
6. Connect the address block output signal "ADRBLK_BC" to the data block DATA_CELL-E1 inputs. The "ADRBLK_BC" will also serve as the Bist output "BC".
7. Connect the address block output signal "ADRBLK_RWB" to the data block DATA_CELL-RD input. The "ADRBLK_RWB" will also serve as the BIST output "RWB".
8. Identify the BIST address bus as the address counters ADDR_CELL-A outputs.
9. Identify the BIST data out bus as the data block DATA_CELL-DO outputs. Ignore unconnected DATA_CELL-DO outputs.
10. Identify the BIST data in bus as the data block DATA_CELL-DI inputs. Ground unused DATA_CELL-DI inputs.
11. The data block right most DATA_CELL-SDO output will serve as the BIST serial data output "BIST_SDO".

This construction has ignored additional gate requirements for signal buffers and delay insertion. These values are dependent on the customer's design. Expect to delay the BIST_SDI and BIST_CK by a similar amount. Heavily loaded signals will require additional buffering to meet edge rate and performance requirements. RAM interface signals must all meet RAM setup and hold requirements. Timing requirements must be verified with Verilog^{TM} simulation.

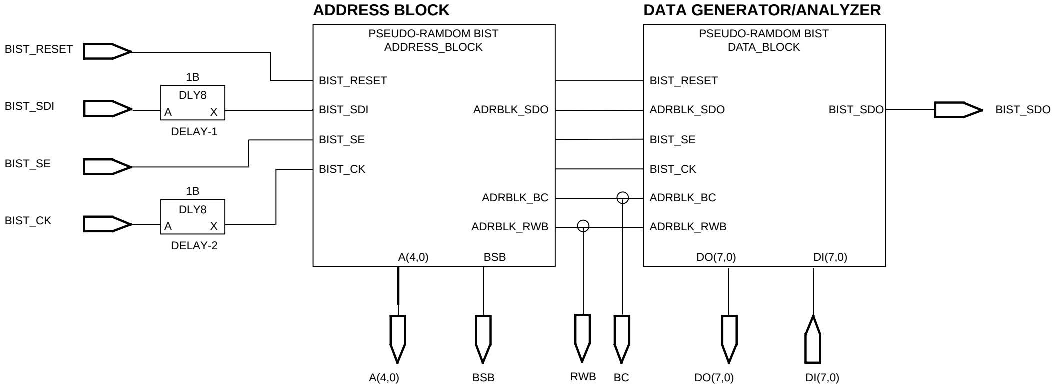


Figure 5. Pseudo Random BIST Block

Pseudo-Random BIST circuit I/O signal description:

BIST_RESET	input	BIST Reset	(low-reset)
BIST_SDI	input	BIST Serial Data Input	scan chain
BIST_SE	input	BIST Scan Enable	(high-scan)
BIST_CK	input	BIST Clock	(rising edge triggered)
DI<7..0>	input	Data In bus	(from RAM to BIST 8 bit word)
ADRBLK_SDO	internal	scan data	connection from address to data block
ADRBLK_BC	internal/output	BIST complete	signals data block of BIST operation
ADRBLK_RWB	internal /output	equivalent to RWB	
A<4..0>	output	RAM Address bus	(32 word memory)
BSB	output	BIST Select Bar	(high-system, low-BIST)
RWB	output	Read/Write_Bar	(high-read, low-write)
DO<7..0>	output	Data Out bus	(from BIST to RAM 8 bit word)
BIST_SD0	output	BIST Serial Data Out	scan chain
BC	output	BIST Complete	signals Bist test line active/complete

BIST Advisor

Motorola ASIC Option Development Engineering can assist in developing a pseudo-random BIST for various RAM sizes with the use of the "BIST_ADVISOR". This program will graphically show connection of the LFSR feedback tap and list the correct test data signature. This can predict the test signature determined by Verilog^{TM} simulation. BIST_ADVISOR can generate signatures for LFSR up to 32 bits in length.

Note:

The schematics of BIST Advisor will drive the RAM inputs from the system inputs during scan mode. This is commonly used when the BIST contains the design's only scan path.

BIST Advisor

```
              ============
              Version 1.01

Which type of BIST do you want?
       0 = Pseudo Random BIST
       1 = Comparator BIST
BIST type: 0

BIST Data
  Number of data register bits : 12
  Number of passes             : 64

Ram Data
  Number of data bits : 8
  Number of address   : 32


    BIST Address Counter Configuration
    ==================================

 The connections to the BIST address counter configuration should be made, in accordance with
the application note.  The diagram below shows how to make the connections:

    +---------------------------------------------------------------------+
    |                                                                     |
    +-XXX XXX XXX XXX XXX    XXX    XXX XXX XXX XXX XXX XXX     XXX       |
SDI --XXX-XXX-XXX-XXX-XXX----XXX----XXX-XXX-XXX-XXX-XXX-XXX-----XXX-------|--> SDO
SE -+-XXX XXX XXX XXX XXX    XXX    XXX XXX XXX XXX XXX XXX     XXX  |\   |
    |  |   |   |   |   |      |                                   +--| >0-+
    +--|---|---|---|---|------|--------------------------------+  |  |/
       |   |   |   |   |      |                               |\_/|
       |   |   |   |   |      |                               |   |
       |   |   |   |   |      |                                \_/
       |   |   |   |   |      |                                 |
TO     V   V   V   V   V      V                                 V
RAM   A00 A01 A02 A03 A04    RWB                               BSB


    BIST Data Register Configuration
    ================================

 The diagram below shows how the feedback connections to the BIST should be made, in accordance
with the application note.  In order to achieve a maximal sequence the following feedback taps
should be used :
                [12,6,4,1]
 The output from the last macro in the data register should be connected to these feedback pins
as shown below. All other feedback pins should be tied to ground.

 The data bits to the RAM should be connected from the BIST starting from the left most data
register. The data bits from the RAM should be connected to the BIST in REVERSE order starting
at the rightmost data register. Any unused data input pins should be tied to ground.

      +-----------------------+-------+-----------+--+
      |                       |       |           |  0
GND --|---+---+---+---+---+---|---+---|---+---+   | / \
      |   |   |   |   |   |   |   |   |   |   |   | ---
 FB   V   V   V   V   V   V   V   V   V   V   V   V  |
     XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX-+  (CO)
SDI -XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX---> SDO
     XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
     ^ | ^ | ^ | ^ | ^ | ^ | ^ | ^ | ^   ^   ^   ^
FROM | | | | | | | | | | | | | | | | |   |   |   |
RAM GND|GND|GND|GND|D07|D06|D05|D04|D03 D02 D01 D00
       |   |   |   |   |   |   |   |
TO     V   V   V   V   V   V   V   V
RAM   D00 D01 D02 D03 D04 D05 D06 D07

  TEST RESULTS
  ============

test length = 4096 clock cycles
signature :
SDI-> 0   0   1   1   0   1   0   0   1   0   1   1  -> SDO
```

1.3.4 Pseudo-Random BIST connection to Metallized RAMs.

To test a H4CP or H4EP metallized RAM with the pseudo-random BIST, a RAM_MUX block will be necessary. The RAM_MUX will select the RAM inputs from either the BIST circuitry or the customer's system. Please refer to Figure 6.

1. Use the pseudo-random BIST signal "BSB" as the RAM_MUX select. A "BSB" low value indicates BIST testing. A "BSB" high value will indicate normal system operation.
2. The pseudo-random BIST output signal "RWB" must be "OR"ed with the BIST clock to generate the proper RAM write cycle timing.
3. Construct the RAM_MUX of 2:1 mux cells which select between:
   a) system address lines and BIST address lines
   b) system data _in lines and BIST data_in lines
   c) system read/write_bar (RWB) and BIST read/write_bar (RWB).
4. The output of the RWB mux must be buffered sufficiently to drive all of the metallized RAM's "R/WB" input signals (one per data bit).
5. Connect the RAM "DO" (data out) bus to the pseudo-random BIST "DI" (data in) bus.

<u>Note:</u>

The BIST clock polarity is the same at the BIST as the RAM, unlike the diffused ram, where the BIST uses the inverse of the RAM clock.

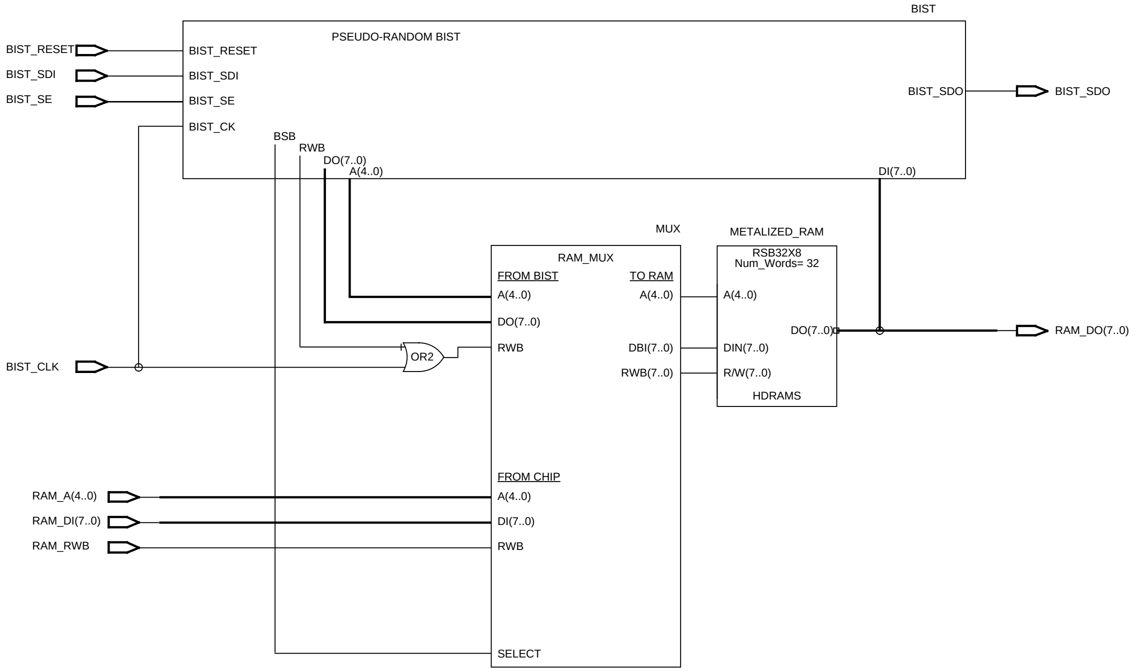


Figure 6. Pseudo Random BIST connections to H4CP Metallized RAM

1.3.5 Pseudo-Random BIST connection to Diffused RAMs

To test a H4CP or H4EP diffused RAM with the pseudo-random BIST, a RAM_MUX will be necessary. The RAM_MUX will select the RAM inputs from either the BIST circuitry or the customer's system. Please refer to Figure 7.

1. Use the pseudo-random BIST signal "BSB" as the RAM_MUX select. A “BSB” low value indicates BIST testing. A "BSB" high value will indicate normal system operation.
2. Construct the RAM_MUX of 2:1 mux cells which select between:
   a)system address lines and BIST address lines
   b)system data _ in lines and BIST data_in lines
   c)system RAM_ST (strobe) signal and the inverse BIST input signal BIST_CK. The BIST clock and system strobe timing should be the opposite polarity. (i.e. the BIST switches on the opposite edge as the RAM)
   d)system read/write control and BIST control signal RWB.
   e)system CSB (RAM DBO tristate control enable) and hard wired enable (VSS) for the BIST.
3. Connect the RAM "DBO" (data bus out) bus to the pseudo-random BIST "DI" (data in) bus.

1.3.6 Pseudo-Random BIST testing of Diffused dual port RAMs

Pseudo-Random BIST testing will need port arbitration logic to prevent simultaneous accessing of both ports. This will prevent testing in a true parallel fashion. One port then the other will be tested. It is possible to interleave the port addresses, if desired. Gating the port inputs may be necessary. There may also be restrictions on the inactive port's address bus and data bus preventing these BIST inputs from being directly wired together.

1.3.7 Pseudo-Random BIST testing of Multiple RAMs

One BIST module may be used to test several RAMs to minimize the gate overhead for BIST. The two methods of sharing BIST between multiple RAMs are parallel and serial testing.

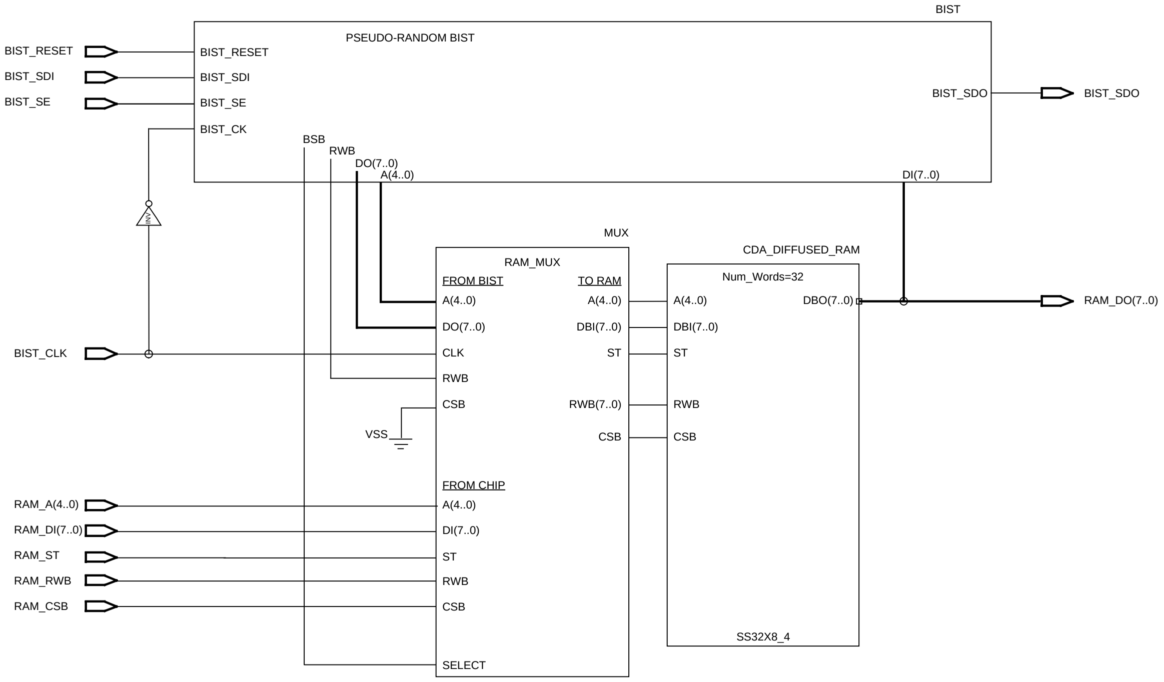


Figure 7. Pseudo Random BIST connections to H4CP CDA diffused RAM

Parallel BIST test all RAMs in one pass. This method is fast but the test time is still determined by the time it takes the BIST to test the largest RAM. All address and control signals are shared and must support the largest RAM. (Special considerations are required for testing dual port RAMs) This method requires:

1. Pseudo-Random BIST address block is designed to access the largest RAM.
2. Pseudo-Random BIST data block will have one DATA_CELL for each data bit on every RAMs under test.

Serial BIST tests one RAM at a time. It is useful for testing groups of RAMs which are configured into larger RAM blocks. This method is slower than parallel testing. All input and output data bussed are shared and must be as wide as the widest RAM. This method requires:

1. Pseudo-Random BIST address block be able to exactly address each RAM sequentially. This will require some additional address counter carry logic.
2. Pseudo-Random BIST data block be wide enough to support the widest RAM. In the other RAM test the unused data bus in signals will be tied low (VSS).

1.3.8 Pseudo-Random BIST Test Vector and Signature Generation

Simulation vectors are used as test vectors and to generate the “test” signature. The stimulus for the simulation is contained within three basic steps: initialization, BIST run, and signature scan. Please refer to Figure 8.

Pseudo-Random BIST Test Procedure:

1. **Initialization:** RB should be low for at least 2 cycles.
2. **Run BIST**: Force RB high. The end of test is signalled when BSB goes high. The number of cycles required to complete a test is

   $$\text{Cycles} = 2(\text{addresses})(\text{passes}).$$
3. **Scan Signature:** Set SDI to "1" (HIGH), set SE to "1" (HIGH), and pulse CLK to scan out signature through SDO.
4. **Exit Scan Mode:** Force SE low to exit scan mode. The "1"s scanned into the BIST through SDI will put the BIST into an inactive mode.

1.3.9 Pseudo-Random BIST On-chip operation

Once the customer has constructed the Pseudo-Random BIST circuit, decisions must be made on how the SRAM BIST testing is to be done in the customer’s application. Section 1.3.8 describes how the BIST testing is activated using the BIST_RESET signal. The BIST_RESET could be activated by a "Power_On_Reset" signal or a dedicated "Execute_BIST" or JTAG command. It is feasible to activate the BIST testing using the scan-path.

The standard use of the scan path is to disable the BIST during simulations used to create the UTIC.FINAL test program. By scanning in the inactive state into the BIST, the Pseudo-Random BIST testing cycle can be avoided. In this way, the customer will avoid a BIST test at the beginning of each test block.

Compression of Final Test Vector count

The OACS test flow uses the "repeat vector" tester mode to compress the BIST test vectors. This allows vector compression up to 1000:1. This is done one of two ways:

1) Have the input BIST clock be the only switching I/O on the chip. CONTEST will automatically compress the vectors. It may be necessary to tristate or gate switching outputs, including the BIST SDO signal.

2) Using the "Input Clock Burst" mode in Verilog^{tm} simulation. The simulation must include the variables "UTIC_PERIOD_DEFINITION" and "BURST_EN".

Please contact your Motorola ASIC representative for specific information on test vector compression.

Figure 8. Pseudo-Random BIST SImulation

2. Comparator BIST

The Comparator BIST uses the 21N march, a deterministic test algorithm, designed to test for hard memory faults. Complete 100% fault coverage is possible using this algorithm. The Comparator BIST also provides additional capabilities: memory diagnostics, memory clear, fault location, and address stress test. This additional capability makes the Comparator BIST much more powerful and complex than the Pseudo Random BIST. The Comparator BIST provides this superior performance with the penalty of greater circuit complexity and a high gate count. Construction of the Comparator BIST requires additional information on the physical implementation of the RAM. In addition to the number of RAM words and word width; it will be necessary to know the number of word lines in the row decoder, the width of column decoder, and number of blocks in the final RAM. Please refer to Figure 9.

The Comparator BIST design utilizes three soft macros (COMPBISTCNTL, COMPACELL, & COMPDCELL) found in the H4CP library. See Figure 10, 11, & 12. Using these soft macros and additional logic, a comparator BIST circuit can be built for any Motorola supplied RAM block.

COMPBISTCNTL controller block of the Comparator BIST.

COMPACELL up/down counter macro in the Comparator BIST "address block".

COMPDCELL data generator/analyzer macro in the Comparator BIST "data block".

A Comparator BIST design has three functional blocks: See Figure 9.

The **"controller block"** orders and synchronizes BIST testing. The "controller block" uses a single COMPBISTCNTL soft macro of the H4CP library. The COMPBISTCNTL contains three functional units: controller, pass control, cycle control.

controller - interfaces between the control lines and the operation of the comparator BIST module. It determines the type of operation being performed, the number of passes, the type of data analysis, status of comparison test, and if BIST test is complete.

pass controller - manages the seven cycles in each pass of 21N march and controls the data polarity. It also determines which cycles should use the wait count in the data retention test.

cycle controller - manages the three phases in each cycle and controls the data polarity. It also provides an override for the "RDB" and "WRB" lines allowing direct memory access. The cycle consists of (read_X, write_Y, read_Y) where "X" is the inverse pattern of "Y". The first read is suppressed in cycles 1 & 2.

The **"address block"** contains the RAM address up/down counters. The "address block" uses the COMPACELL soft macro from the H4CP library. The "address block" will have a separate counter to test each type of address decoder individually (i.e. row decoder, column decoder, & block decoder). See Note*

The **"data block"** contains the RAM data generator/analyzer. The "data block" uses the COMPDCELL soft macro from the H4CP library. The "data block" will generate the deterministic 21N march data patterns and act as a Linear Feedback Shift Register (LFSR) during BIST signature generation.

Note:

Currently all RAMs supplied by Motorola ASIC are partitioned into only rows and columns. The block testing capability will not be necessary unless the design utilizes multiple block memories.

Additional logic will be required at the BIST's top hierarchy level to assure proper circuit timing.

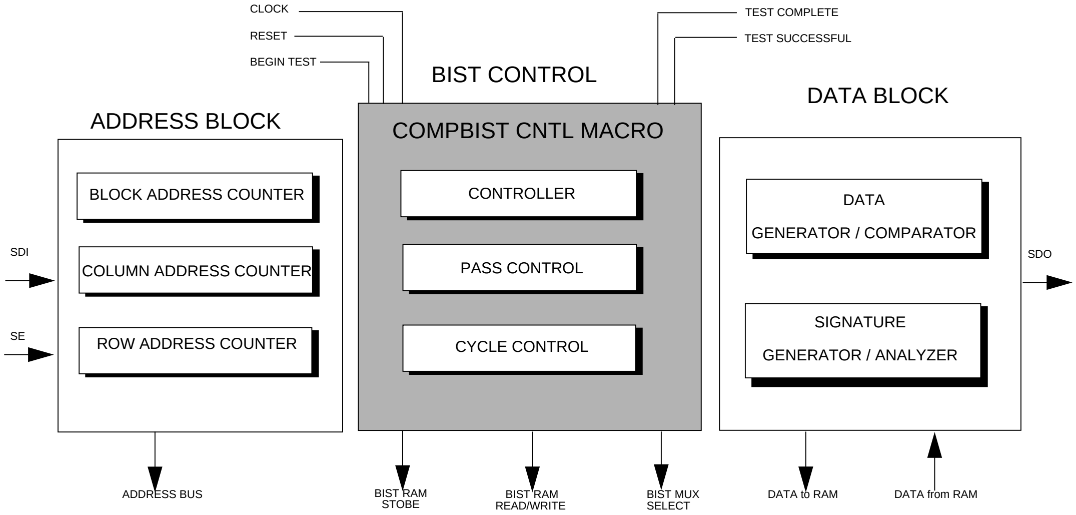


Figure 9. Comparator BIST Functional Diagram

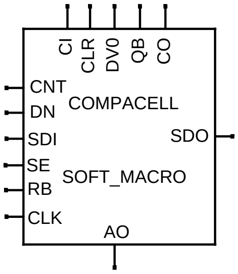


Figure 10. COMPACELL Soft Macro

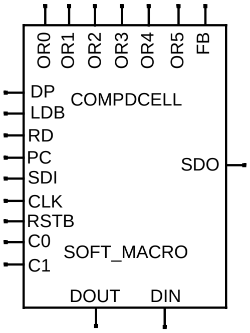


Figure 11. COMPDCELL Soft Macro

I/O description of COMPACELL

CNT	input	count enable
DN	input	count direction
RB	input	synchronous reset
CLR	input	count clear
CI	input	carry in
CLK	input	clock
DVO	input	highest bit value
SDI	input	scan data in
SE	input	scan enable
CO	output	carry out
AO	output	address bit out
SDO	output	scan data out
QB	output	inverted A0

I/O description of COMPDCELL

LDB	input	load RAM data	
RD	input	read	
PC	input	previous data bit	
RSTB	input	synchronous reset	
CLK	input	clock	
DIN	input	BIST data input	
SDI	input	scan data in	
DP	input	data pattern	
OR0	input	data bit order 0	
OR1	input	data bit order 1	
OR2	input	data bit order 2	
OR3	input	data bit order 3	
OR4	input	data bit order 4	
OR5	input	data bit order 5	
FB	input	LFSR feedback	
C0	input	ctrl 0	0 - signature analysis
C1	input	ctrl 1	1 - analyze RAM data
			2 - generate RAM data
			3 - scan enable
DOUT	output	BIST data output	
SDO	output	scan data out	

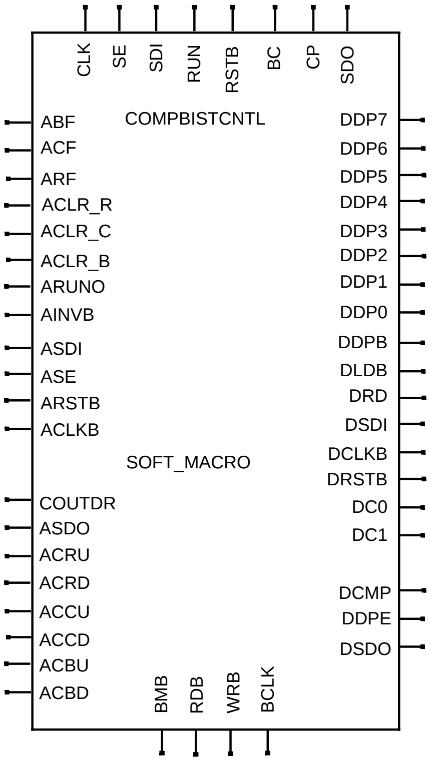


Figure 12. Comparator BIST Soft Macro Symbol COMPBISTCNTL

I/O description of COMPBISTCNTL

ABF	output	address block fast
ACF	output	address column fast
ARF	output	address row fast
ACLR_R	output	address clear row
ACLR_C	output	address clear column
ACLR_B	output	address clear block
ARUN0	output	address run
AINVB	output	count up/down control
ASDI	output	address scan data in
ASE	output	address scan enable
ARSTB	output	address reset
ACKB	output	address clock
COUNTDR	input	carry out data retention
ASDO	input	address scan data output
ACRU	input	address carry row up
ACRD	input	address carry row down
ACCU	input	address carry column up
ACCD	input	address carry column down
ACBU	input	address carry block up
ACBD	input	address carry block down
DDP7	output	data position 7
DDP6	output	data position 6
DDP5	output	data position 5
DDP4	output	data position 4
DDP3	output	data position 3
DDP2	output	data position 2
DDP1	output	data position 1
DDP0	output	data position 0
DDPB	output	data polarity bar
DLDB	output	data load bar
DRD	output	data read
DSDI	output	data serial data in
DCLKB	output	data clock
DRSTB	output	data reset
DC0	output	data encoded control 0
DC1	output	data encoded control 1 0 - signature analysis 1 - analyze RAM data 2 generate RAM data 3 - scan enable
DCMP ***	input	data compare
DDPE ***	input	data pattern end
DSDO	input	data serial data out
BMB	output	BIST mode
RDB	output	RAM read
WRB	output	RAM write
BCLK	output	BIST clock

*****WARNING** -- RAMs having only a 1 or 2 bit word length cause a DDPE generation exception due to limitations in the COMPBISTCNTL soft macro. DDP1 = DDPE is technically the correct choice for 1 and 2 bit wide RAMs. Simulation has shown that using DDP1 = DDPE causes unpredictable behavior of COMPBISTCNTL. The COMPBISTCNTL state machine cannot take an pattern end that early. For 1 and 2 bit wide RAM, use DDP2 = DDPE and use the logic "or" of C(0) and C(1) for generation of DCMP.

2.1. Comparator BIST Test Description

The comparator BIST uses multiple test methods to verify the RAM operation. Multiple data patterns are used to check for interactions between bits. Address decoders are tested at full speed with three "fast" tests. The RAM data array is checked using a deterministic 21N march. A statistical signature analysis is done to verify both the BIST and RAM. The comparator BIST also supports additional debug and characterization capabilities including stop on fault, RAM control through scan path control, and limited data retention testing.

The standard testing order of the comparator BIST is:

- Data pattern #1
  - block fast
    - 21N march with "block" addressing
  - column fast
    - 21N march with "column" addressing
  - row fast
    - 21N march with "row" addressing
- Data pattern #2
  - block fast
  - column fast
  - row fast
- Data pattern #3
  - ...
- Data pattern #FINAL
  - block fast
    - 21N march with "block" addressing
  - column fast
    - 21N march with "column" addressing
  - row fast
    - 21N march with "row" addressing

  - signature test

21N march RAM test pattern sequence

The comparator BIST utilizes a 21N march pattern to locate RAM coupling faults. The 21N march consists of 7 phases, each phase with 3 cycles. In phase 1, 2, 3, & 4 the address is incremented from its lowest to highest value. In phase 5, 6, & 7, the address is decremented from its highest to lowest value. Each of the seven phases has 3 cycles consisting of a Read, Write, Read pattern. All three cycles are completed at a data location before the address is changed. At address 0, the three cycles (R,W,R) are completed before the address is incremented (decremented). At the second address, the three cycles (R,W,R) are again completed before the address is incremented (decremented). The cycles continues until the reaching highest (lowest) address. There the current phase completes and the address is initialized for the next phase. At the end of phase seven, the 21N march algorithm is complete.

Table 5. 21N March Algorithm

Phase	(CY1,CY2,CY3)	first address	last address	address counter
Phase 1	(--, W0, R0)	lowest "0"	highest	increment
Phase 2	(--, W1, R1)	lowest "0"	highest	increment
Phase 3	(R1, W0, R0)	lowest "0"	highest	increment
Phase 4	(R0, W1, R1)	lowest "0"	highest	increment
Phase 5	(R1, W0, R0)	highest	lowest "0"	decrement
Phase 6	(R0, W1, R1)	highest	lowest "0"	decrement
Phase 7	(R1, W0, R0)	highest	lowest "0"	decrement

Notes:
1. The first read is ignored in phase 1&2 because the RAM is not considered initialized.
2. W0 - indicates writing the data pattern from the data generator (no inversion)
W1 - indicates writing the inverted data pattern from the data generator

Block Fast, Column Fast, Row Fast - Test Addressing

RAMs commonly have up to three types of address decoders: block select, column multiplexer, and row decoder. The comparator BIST can control each address decoder individually to test each at full speed.

The standard 21N march algorithm moves directly through the address field. Depending on the address multiplexing scheme, it could be several RAM cycles before an address decoder changes state. In the three "fast" tests the address will be changed in some "modulo" fashion so that in block_fast testing, the block address decoder changes state every RAM cycle. In row_fast testing the row decoder changes state every RAM cycle and in column_fast testing the column decoder changes state each RAM cycle.

Example: An 8 word RAM has 2 blocks, 2 rows, and 2 columns. The three address bits are in the following fields: (blk, row, column) {msb-lsb}.

block_fast addressing order:	000, 100, 001, 101, 010, 110, 011, 111	(0 4 1 5 2 6 3 7 modulo 4)
row_fast addressing order:	000, 010, 100, 110, 001, 011, 101, 111	(0 2 4 6 1 3 5 7 modulo 2)
column_fast addressing order:	000, 001, 010, 011, 100, 101, 110, 111	(0 1 2 3 4 5 6 7)

These three address decoder tests are hard coded into the "COMPBISTCNTL" macro. The customer will configure a counter in the comparator BIST address block for each address decoder to be tested. Current H4CP diffused RAMs have only column and row decoders. H4CP metallized RAMs have only row (word) decoders. Unused "fast" tests will be repeated on one of the RAMs address decoder.

Comparator BIST data pattern generation

In order to fully test RAM’s data array, multiple passes of the block_fast, row_fast, and column_fast tests are performed using different data patterns. The background patterns are used to prove adjacent RAM bits will not affect the logic state of each other. The COMPBISTCNTL/COMPDCELL logic can provide these independent test patterns for RAMs up to 128 bits wide.

The comparator BIST data pattern generator will use up to eight background patterns. The comparator BIST's 21N march algorithm uses the data patterns for the "R0" and "W0" data. It uses the inverted data pattern for the "R1" and "W1" data.

The number of background used depend on the RAM data width. The final data pattern always has all bits equal zero. For a 8 bit RAM the four backgrounds used are:

10101010	pattern 1
11001100	pattern 2
11110000	pattern 3
00000000	pattern 4

The data pattern backgrounds are described in the following information.

Data Generator Background Pattern

State		(maximum of 128 bits wide - 64 bits shown)	data LSB	PATTERN
000	10101010	10101010 10101010 10101010 10101010 10101010 10101010	10101010	1(1's), 1(0's) repeat
001	11001100	11001100 11001100 11001100 11001100 11001100 11001100	11001100	2(1's),2(0's) repeat
010	11110000	11110000 11110000 11110000 11110000 11110000 11110000	11110000	4(1's), 4(0's) repeat
011	11111111	00000000 11111111 00000000 11111111 00000000 11111111	00000000	8(1's), 8(0's) repeat
100	11111111	11111111 00000000 00000000 11111111 11111111 00000000	00000000	16(1's), 16(0's) repeat
101	11111111	11111111 11111111 11111111 00000000 00000000 00000000	00000000	32(1's), 32(0's) repeat
110	00000000	00000000 00000000 00000000 00000000 00000000 00000000	00000000	64(1's), 64(0's)
111	00000000	00000000 00000000 00000000 00000000 00000000 00000000	00000000	128(0's)

2.2. Comparator BIST data analyzer

The data analyzer compresses the memory check information. The data analyzer has two distinct operating modes: data comparison and signature analysis.

During the data comparison testing a "1" is stored in the corresponding data analyzer bit for any RAM bit that doesn't match the expected pattern. In a characterization mode the Comparator BIST can be initialized to halt on the first data miss-match. Normally the memory is checked and the failing bits are marked. If any data errors are found, the CP (comparator pass) line will be set low and the test will stop before signature analysis is run.

During the signature analysis testing the data analyzer is configured into a Linear Feedback Shift Register (LFSR). During the "write" mode, the LFSR generates a pseudo-random data pattern for the RAM. During "read" mode, the RAM data will be fed into the same LFSR to build the RAM test signature. At the completion of the signature analysis the final signature is stored in the data analyzer. No comparison is done on the signature. External comparison logic must be added or the signature must be scanned out for comparison. The signature analysis verifies the operation of both the RAM and BIST circuitry. The LFSR must be sufficient length to avoid signature aliasing.

2.3. Signature Aliasing

see section 6.3.2 on page 5 for Signature Aliasing

2.4. Comparator BIST Construction

2.4.1. Gate count estimate of the Comparator BIST

A gate count estimate can be generated by counting the required COMPACELL, COMPDCELL, and COMPBISTCNTL macros in the comparator BIST. Logic for generating BIST control signals and the RAM mux have been ignored.

Number of macros required		COMPBISTCNTL	COMPACELL	COMPDCELL	
one	COMPBISTCNTL macro	1			
one	COMPACELL for each RAM address line				
choose the greater number of COMPDCELLs a) one COMPDCELL for each RAM data line b) 12 COMPDCELLs to minimize signature aliasing					
	Multiply by equivalent gates per macro x	x 780	x 34	x 54	
	Sum number of equivalent gates total	780	+	+	= Total

2.4.2. Construction of a Comparator BIST

The construction of the comparator BIST will be described by drawing its schematic. Please refer to Figure 13, 14, & 15. The comparator BIST will be for testing a 192x8 RAM. The addressing is configured as 8 columns and 24 rows. (A full 256x8 ram in this configuration would have 8 columns and 32 rows) The comparator BIST will consist of:

1. an address block with a column counter and a row counter to generate A(0,7).
2. a controller block with a single COMPBISTCNTL macro
3. a data block with a 12 bit data generator/analyzer to evaluate DI(0,7) & DO(0,7).

Note:

The signal names given in this description are used to identify nodes on the illustrations. It is not a requirement that customer designs use this naming convention.

2.4.3. Construction of Comparator BIST address block

Please contact your Motorola ASIC representative for specific information on RAM implementation. Before a comparator BIST can be constructed certain information must be known:

1. Size of RAM, number of words, bits per word.
2. RAM memory array structure; use of block, row, & column decoders
3. the correspondence between the RAM address lines and the decoder they control

The diffused RAMs supplied by Motorola ASIC currently use only row and column address decoders. There are no block decoders in the diffused RAMs (this might be implemented in the customer design). Current Motorola supplied RAMs have the column decoder driven by the lowest address bits. The row decoder is driven by the higher address bits. The COMPBISTCNTL macro is hardwired to execute all three tests, block_fast, column_fast, and row_fast. In this example the column counter will drive both the block_fast & column_fast control signals. Therefore the comparator BIST will run the column_fast test twice, and the row_fast test once. The choice to repeat the column_fast test was an arbitrary one. There will be no difference in test time. Please refer to Figure 13.

Placement of macros for address block.

1. For the column counter place one COMPACELL for each RAM address line going to the column decoder. In this example, three address bits are required for an 8:1 column decoder, and three COMPACELLs will be needed. The COMPACELL on the left will generate the A0 signal, the middle COMPACELL will generate the A1 signal, the right COMPACELL will generate the A2 signal.
2. For the row counter place one COMPACELL for each RAM address line going to the row decoder. In this example, five address bits required to decode the 24 rows, five COMPACELLs macros will be required. The left most row counter COMPACELL will generate the A4 signal. The right most COMPACELL will generate the A7 signal.

Connection of the address block.

1. Connect all COMPACELL pin "CNT" to a common node named "RUN". This will later connect to the COMPBISTCNTL signal "ARUN0".
2. Connect all COMPACELL pin "DN" to a common node named "INV". This will later connect to the COMPBISTCNTL signal "RINVB"
3. Connect all COMPACELL pin "SE" to a common node named "SE". This will later connect to the COMPBISTCNTL signal "ASE".
4. Connect all COMPACELL pin "RB" to a common node named "RSTB". This will later connect to the COMPBISTCNTL signal "ARSTB".

5. Connect all COMPACELL pin "CLK" to a common clock node named "CLK". Timing analysis may show this clock will need to be delayed from the COMPBISTCNTL signal "ACLKB" to eliminate hold time violations on the RAM address lines. If so, rename the signal "CLK" to "CLK_D" (delayed), connect the COMPBISTCNTL signal "ACLKB" to a delay macro (i.e. DLY8), and use the output to drive the common node "CLK_D".
6. Connect the column counter's COMPACELL pin "CLR" to a common node named CLR_C. This will later connect to the COMPBISTCNTL signal "ACLR_C".
7. Connect the row counter's COMPACELL pin "CLR" to a common node named CLR_R. This will later connect to the COMPBISTCNTL signal "ACLR_R".
8. Drive the column counter's LSB COMPACELL pin "CI" (example's "A0") with the logical "OR"ing of the COMPBISTCNTL signals "ACF" and "ACLR_R". The effect is that the column counter will be enabled during the column_fast testing and toggled at the end count of a row_fast test. (by default it will also be toggled at the end of the block_fast test).Connect the remaining column counter COMPACELL pin "CI" by the "CO" pin on the COMPACELL macro on the left. (i.e. "A1_CI" is driven by "A0_CO" and "A2_CI" is driven by "A1_CO")
9. The column counter's MSB COMPACELL pin "CO" (example's "A2") will drive the COMPBISTCNTL signals "ACCU", "ACCD", "ACBU", and "ACBD". This is possible since the "CO" signal is active both when the column counter reaches its maximum count "7" and its minimum count "0". If the number of column were not a "power of 2" number (i.e. 2,4,8, or 16), additional maximum count logic would be required. This maximum count logic would then drive the "ACCU" and "ACBU" signals. Still, the signals "ACCD" and "ACBD" would be driven by the "CO" signal.
10. Drive the row counter LSB COMPACELL pin "CI" (example's "A3") with the logical "OR"ing of the COMPBISTCNTL signals "ABF", "ARF", and "ACLR_C". The effect is that the row counter will be enabled during both the block_fast and the row_fast testing or toggled at the end count of a column_fast test.
11. Connect the remaining row counter COMPACELL pin "CI by the "CO" output pin on the COMPACELL macro on the left. (i.e. "A4_CI is driven by "A3_CO", "A5_CI" is driven by "A4_CO", etc.)
12. The row counter's MSB COMPACELL pin "CO" (example's "A7") will drive the COMPBISTCNTL signal "ACRD". In the example "24" is defined as the number of rows. Logic which decodes the COMPACELL "QB" output for the maximum row address, generates the COMPBISTCNTL "ACRU". If the row counter was addressing a "power of 2" number of rows (i.e. 2,4,8,16,32,64, etc.), the row counter's MSB COMPACELL pin "CO" (A7) could have driven both the COMPBISTCNTL signal "ACRU" & "ACRD".
13. The following connection is for a special data retention test mode, use the row counter's MSB COMPACELL pin "CO" (A7) signal to drive the COMPBISTCNTL signal "COUTDR".
14. The COMPACELL input "DV0" pin defines the macro's maximum count value. If the maximum count value is a "power of 2" value (2, 4, 8, 16, 32, 64, etc.) ground all COMPACELL "DV0" pins. This will load all "1's" into the counter. If the maximum count value is not a "power of 2" value, the maximum count bit state is loaded in by the "DV0" pin. The "DV0" loads the inverse of the maximum count bit state (beware).
15. Scan path connection: Connect the COMPBISTCNTL "ASDI" signal to the RAM's LSB address bit's COMPACELL (example's "A0") "SDI" input. It may be necessary to delay this signal as the CLK signal to avoid scan path timing problems. Connect the "SDO" pin to the next greatest address bit's COMPACELL "SDI" pin. ("A0_SDO" to "A1_SDI", "A1_SDO" to "A2_SDI", "A2_SDO" to "A3_SDI", etc.) The highest address bit's COMPACELL (example's "A7") "SDO" pin will drive the COMPBISTCNTL "ASDO" pin.
16. BIST address line connections: Connect the column counter LSB to the BIST A(0) pin. Continue connecting all the COMPACELL's "A0" pins to the appropriate BIST address lines. The actual connections depend on the exact implementation of the RAM's address decoding scheme. In this example the column counter connects to the BIST address lines first. Then the row counter connects to the upper bits in the BIST address lines.

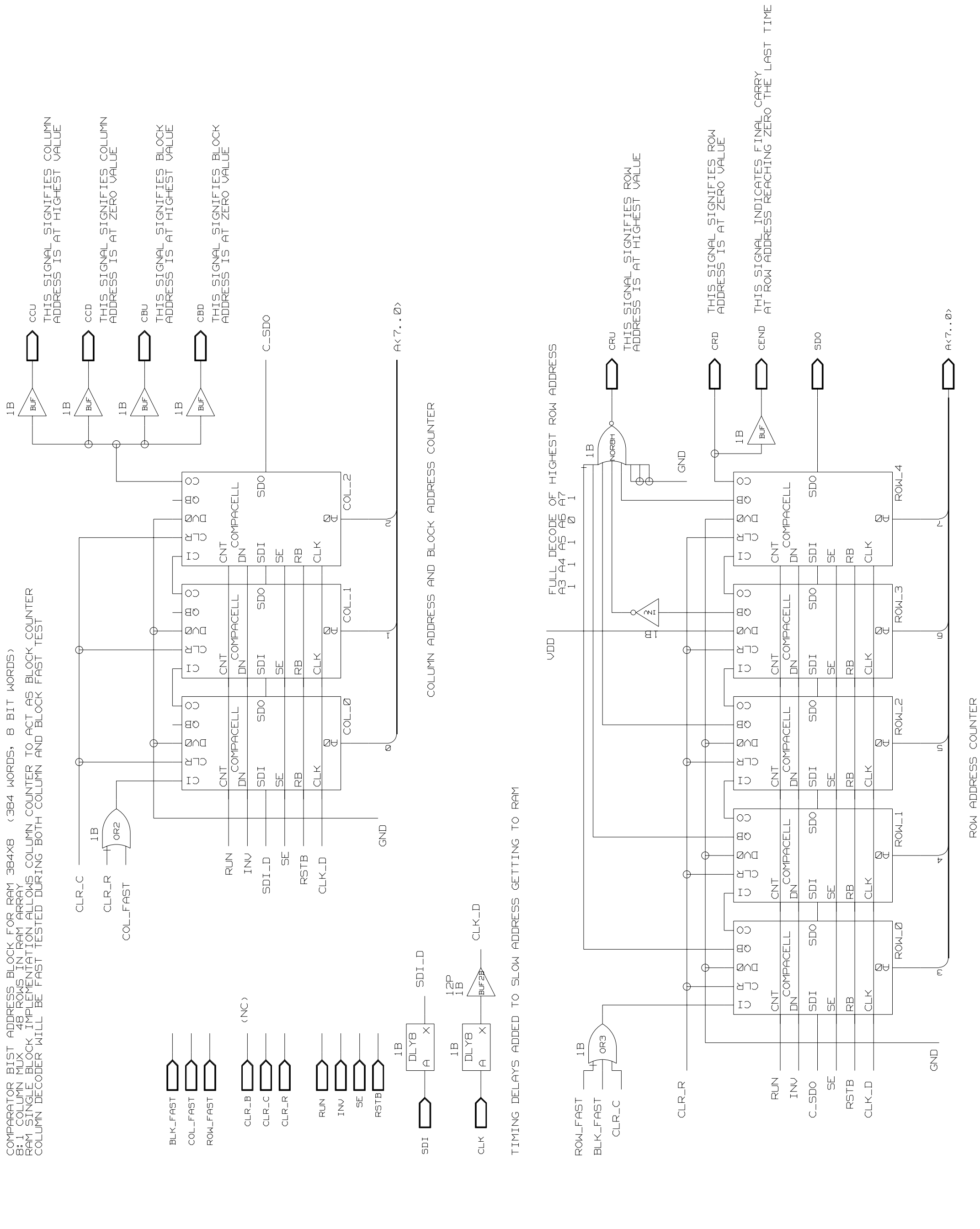

COMPARATOR BIST ADDRESS BLOCK FOR RAM 384X8 (384 WORDS, 8 BIT WORDS)
8:1 COLUMN MUX 48 ROWS IN RAM ARRAY
RAM SINGLE BLOCK IMPLEMENTATION ALLOWS COLUMN COUNTER TO ACT AS BLOCK COUNTER
COLUMN DECODER WILL BE FAST TESTED DURING BOTH COLUMN AND BLOCK FAST TEST
CCU
THIS SIGNAL SIGNIFIES COLUMN ADDRESS IS AT HIGHEST VALUE
CCD
THIS SIGNAL SIGNIFIES COLUMN ADDRESS IS AT ZERO VALUE
CBU
THIS SIGNAL SIGNIFIES BLOCK ADDRESS IS AT HIGHEST VALUE
CBD
THIS SIGNAL SIGNIFIES BLOCK ADDRESS IS AT ZERO VALUE
C_SDO
A<7..0>
BUF
1B
COMPACELL
COL_0
COL_1
COL_2
CI
CLR
CNT
DN
SDI
SDO
SE
RB
CLK
CU
CD
CLR_C
CLR_R
COL_FAST
OR2
RUN
INV
SDI_D
SE
RSTB
CLK_D
GND
COLUMN ADDRESS AND BLOCK ADDRESS COUNTER
BLK_FAST
COL_FAST
ROW_FAST
CLR_B
CLR_C
CLR_R
(NC)
RUN
INV
SE
RSTB
SDI
DLY8
SDI_D
CLK
12P
BUF2P
CLK_D
TIMING DELAYS ADDED TO SLOW ADDRESS GETTING TO RAM
FULL DECODE OF HIGHEST ROW ADDRESS
A3 A4 A5 A6 A7
1 1 1 0 1
VDD
INV
NOR5H
GND
CRU
THIS SIGNAL SIGNIFIES ROW ADDRESS IS AT HIGHEST VALUE
CRD
THIS SIGNAL SIGNIFIES ROW ADDRESS IS AT ZERO VALUE
CEND
THIS SIGNAL INDICATES FINAL CARRY AT ROW ADDRESS REACHING ZERO THE LAST TIME
SDO
A<7..0>
ROW_0
ROW_1
ROW_2
ROW_3
ROW_4
ROW_FAST
BLK_FAST
CLR_C
OR3
CLR_R
RUN
INV
C_SDO
SE
RSTB
CLK_D
GND
ROW ADDRESS COUNTER

Figure 13. Comparator BIST address block.

2.4.4. Construction of Comparator BIST data block

Before the Comparator BIST can be constructed it will be necessa4ry to know.Please refer to Figure 14**.**

1. number of bits in RAM data word.
2. signature length required to obtain the desired aliasing protection. Motorola ASIC recommends minimum of 12 bits.

Placement of macros for data block.

1. Determine the number of COMPDCELLs necessary for the data block. Choose the larger of the two numbers:
   a) 12, chosen to minimize chance of signature aliasing.
   b) the number of data bits in the RAM's word.
2. Place the COMPDCELLs in a ROW from left to right. To assist in connecting the feedback signal for the signature Linear Feedback Shift Register (LFSR), name the COMPDCELLs from right to left, FB_1, FB_2, FB_3, FB_4, etc.

Connection of the data block.

1. Connect all COMPDCELL pin "DP" to a common node named "DP". This will later connect to the COMPBISTCNTL signal "DDPB".
2. Connect all COMPDCELL pin "LDB" to a common node named "LDB". This will later connect to the COMPBISTCNTL signal "DLDB".
3. Connect all COMPDCELL pin "RD" to a common node named "RD". This will later connect to the COMPBISTCNTL signal "DRD".
4. Connect all COMPDCELL pin "CLK" to a common node named "CLK". This will later connect to the COMPBISTCNTL signal "DCLKB". This clock signal does not usually need to be delayed.
5. Connect all COMPDCELL pin "C0" to a common node named "C0". This will later connect to the COMPBISTCNTL signal "DC0".
6. Connect all COMPDCELL pin "C1" to a common node named "C1". This will later connect to the COMPBISTCNTL signal "DC1".
7. Connection of the COMPDCELL pin "FB"; See Table 4 for the LFSR feedback tap connection for various lengths of LFSR. Connect the appropriate "FB" inputs to the inverted "SDO" output of the right most COMPDCELL. Any unconnected "FB" inputs will be grounded. In the example the COMPDCELLs have been named FB_1, FB_2, to correspond to Table4.
8. The six inputs on the COMPDCELL (OR5, OR4, OR3, OR2, OR1, OR0) will indicate the bit position in the RAM data word. These signals select the bit value for the test's background pattern. To connect the "ORx" signals the value of "0" indicates connecting to VSS. The value of "1" indicates connecting the COMPDCELL "ORx" signal to the corresponding COMPBISTCNTL "DDPx" signal. The COMPDCELL which corresponds to RAM bit "D0" bit will be hardwired to "000000". The COMPDCELL which corresponds to RAM bit "D1" is hardwired to "000001". The COMPDCELL which corresponds to RAM bit "D2" is hardwired to "000010". The COMPDCELL which corresponds to RAM bit "D3" is hardwired to "000011". The COMPDCELL which corresponds to RAM bit "D4" is hardwired to "000100". This pattern will continue until all RAM data bits are located. If the COMPDCELL does not correspond to a RAM data bit (i.e. added for signature aliasing) tie all "ORx" signals to VSS.
9. The left most COMPDCELL (corresponds to RAM bit D0) SDI input will be driven by the COMPBISTCNTL "DSDI" signal.
10. The left most COMPDCELL (corresponds to RAM bit D0) PC input will be tied to VSS.
11. All other COMPDCELL "SDI" and "PC" inputs will be driven by the COMPDCELL "SDO" signal to its left.
12. The right most COMPDCELL (named FB_1) SDO signal will drive the COMPBISTCNTL "DSDO" input.
13. To connect the COMPDCELL "DIN" pin; start at left most COMPDCELL and connect the "DIN" pin to RAM Dout(0) signal. The COMPDCELL immediately to the right will connect "DIN" to RAM Dout(1). Continue moving right until all RAM Dout bits are connected to a COMPDCELL "DIN" pin. If the COMPDCELL does not correspond to a RAM data bit (i.e. added for signature aliasing) connect the "DIN" pin to VSS.
14. To connect the COMPDCELL "DOUT" pin; start at left most COMPDCELL and connect the "DOUT" pin to the RAM DI(0) signal. The COMPDCELL immediately to the right will connect its "DOUT" pin to the RAM DI(1). Continue moving right until all RAM DI signals are connected to a COMPDCELL "DOUT" pin. If the COMPDCELL does not correspond to a RAM "DIN" signal (i.e. added for signature aliasing) leave the "DOUT" pin unconnected.
15. To generate the COMPBISTCNTL signal DDPE, invert the lowest numbered COMPBISTCNTL signal "DDPx" which is not connected to a COMPDCELL (example's "DDP3").
16. To generate the COMPBISTCNTL signal "DCMP", logic "OR" all COMPDCELL "SDO" pins which correspond to a RAM data bit. (i.e. NOT added just for signature aliasing).

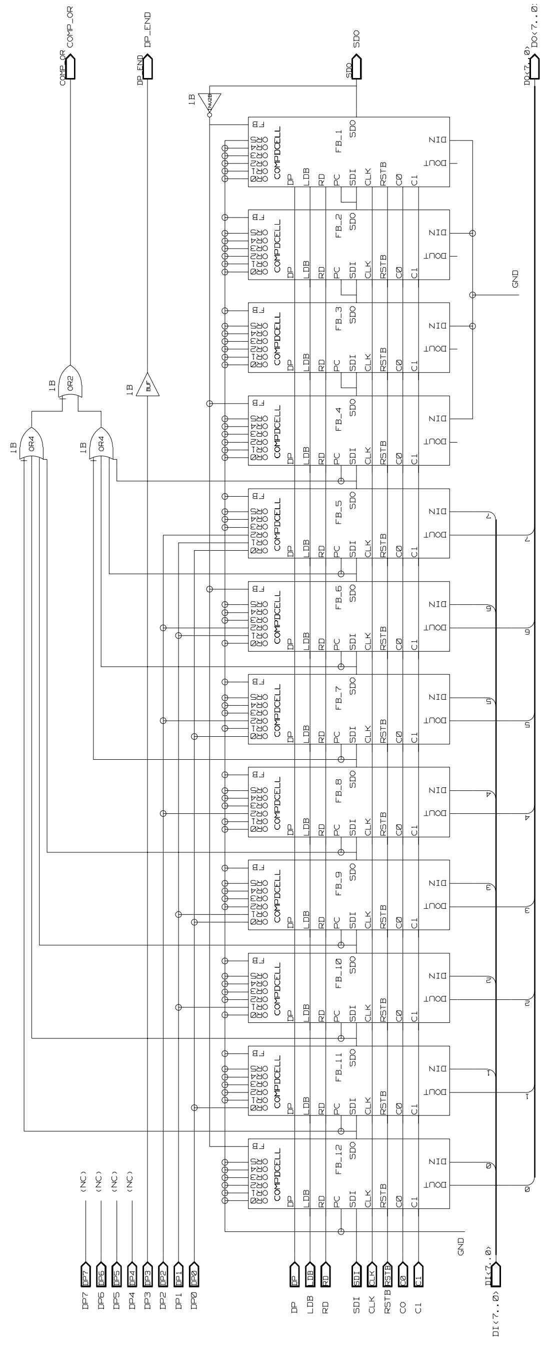


Figure 14. Comparator BIST data block.

2.5. Construction of Comparator BIST block

After the comparator BIST address block and data block are complete, construct the hierarchical icons/bodies for both blocks. The three functional blocks, data, address, and COMPBISTCNTL will now be assembled for the final Comparator BIST schematic. Please refer to Figure 15**.**

Placement of macros for Comparator BIST.

1. Place the hierarchical icon/body for the comparator BIST address block.
2. To its right place the COMPBISTCNTL macro.
3. To their right place the hierarchical icon/body for the comparator BIST data block.

Connection of the comparator BIST block.

1. Connection between the address block and the COMPBISTCNTL macro were discussed in 2.5.1 Construction of Comparator BIST address block. Use the following connections.

ADDRESS BLOCK	**to**	**COMPBISTCNTL**
BLK_FAST	ABF	block fast test
COL_FAST	ACF	column fast test
ROW_FAST	ARF	row fast test
CLR_R	ACLR_R	clear row counter
CLR_C	ACLR_C	clear column counter
CLR_B	ACLR_B	clear block counter
RUN	ARUN0	All counter enable
INV	RINBB	Increment/Decrement
SDI	ASDI	scan data input
SE	ASE	scan enable
RSTB	ARSTB	reset
CLK	ACLKB	clock
CEND	COUTDR	data retention test carry
SDO	ASDO	scan data output
CRU	ACRU	row count up carry
CRD	ACRD	row count down carry
CCU	ACCU	column count up carry
CCD	ACCD	column count down carry
CBU	ACBU	block count up carry
CBD	ACBD	block count down carry

2. Connections between the COMPBISTCNTL macro and the data block were discussed in 2.6 Construction of Comparator BIST data block. Use the following connections.

DATA BLOCK to	**COMPBISTCNTL**	
DP7	DDP7	data position 7
DP6	DDP6	data position 6
DP5	DDP5	data position 5
DP4	DDP4	data position 4
DP3	DDP3	data position 3
DP2	DDP2	data position 2
DP1	DDP1	data position 1
DP0	DDP0	data position 0
DP	DDPB	data polarity
LDB	DLDB	load RAM data for compare
RD	DRD	load RAM data for signature
SDI	DSDI	scan data input
CLK	DCLKB	clock
RSTB	DRSTB	reset
C0	DC0	encoded function command
C1	DC1	encoded function command
COMP_OR	DCMP	data comparison (passing test)
DP_END	DDPE	data pattern end
SDO	DSDO	scan data out

3. Connect COMPBISTCNTL "RSTB" to an appropriate system reset or BIST reset signal.
4. Connect COMPBISTCNTL "RUN" to an appropriate system BIST RAM testing control signal.
5. Connect COMPBISTCNTL "SDI" to the appropriate location in the customer scan path.
6. Connect COMPBISTCNTL "SE" to the appropriate system Scan Enable signal.
7. Connect COMPBISTCNTL "CLK" signal to an appropriate BIST system clock. Care must be taken this node actually exists as a BIST input. Do not merge all BIST's clocks into a larger clock tree structure. That would remove a number of timing delays in the COMPBISTCNTL that minimize potential timing problems.
8. Connect COMPBISTCNTL "BC" to an appropriate location to indicate "BIST test Complete".
9. Connect COMPBISTCNTL "SDO" to the appropriate location in the customer scan path.
10. Connect the address block's "A(7,0)" address bits through the RAM's BIST/system mux to the RAM's address bus.
11. Connect the data block's "DOUT(7,0)" data bits to through the RAM's BIST/system mux to the RAM's data_in bus.
12. Connect the RAM's data_out bus to the comparator BIST data block "Din(7,0)" data inputs.
13. Connect the COMPBISTCNTL "BCK" through the RAM's BIST/system mux to the RAM's strobe input.
14. The COMPBISTCNTL "RDB" remains unconnected. Current Motorola ASIC supplied RAMs do not utilize separate READ and WRITE control signals.

Scan path considerations.

The Comparator BIST signals "BMB" and "RWB" can be generated in two ways depending on the intended scan path usage. If the customer wants the system bus to control the RAM under scan conditions use the method A. If it is desired to control the RAM from the BIST circuitry during scan use the method B.

Method A - RAM controlled by system bus during scan.

This could be used if the customer prefers to control the RAM inputs from the system interface during scan of the BIST circuitry. By design, the COMPBISTCNTL will hold "BMB" so it will select the system inputs for the RAM during scan. All other BIST signals wiggle according to the data being shifted through the scan path at that instant.

15A. Connect the COMPBISTCNTL "WRB" signal through the RAM's BIST/system mux to the RAM's "RWB" signal.
16A. Connect the COMPBISTCNTL "BMB" signal through the RAM's BIST/system mux to the RAM's "BMB" signal. The COMPBISTCNTL macro will hold "BMB" high during scan.

Method B - RAM controlled by BIST during scan.

This method would drive the RAM from the comparator BIST inputs during scan operation. The signal "BMB" will select the BIST inputs for the RAM during scan. The signal "RWB" is held in the read state during the scan operation to avoid corrupting the RAM data by spurious writes during scan operation.

15B. Logically "OR" the COMPBISTCNTL "WRB" with the comparator BIST "SE" signal. This will force the RAM RWB signal to remain in "read" during the scan path operation.
16B. Logically "AND" the COMPBISTCNTL "BMB" with the inverse of the comparator BIST "SE" signal. This will force the RAM's BIST/system mux to drive the RAM from the BIST generated signals.

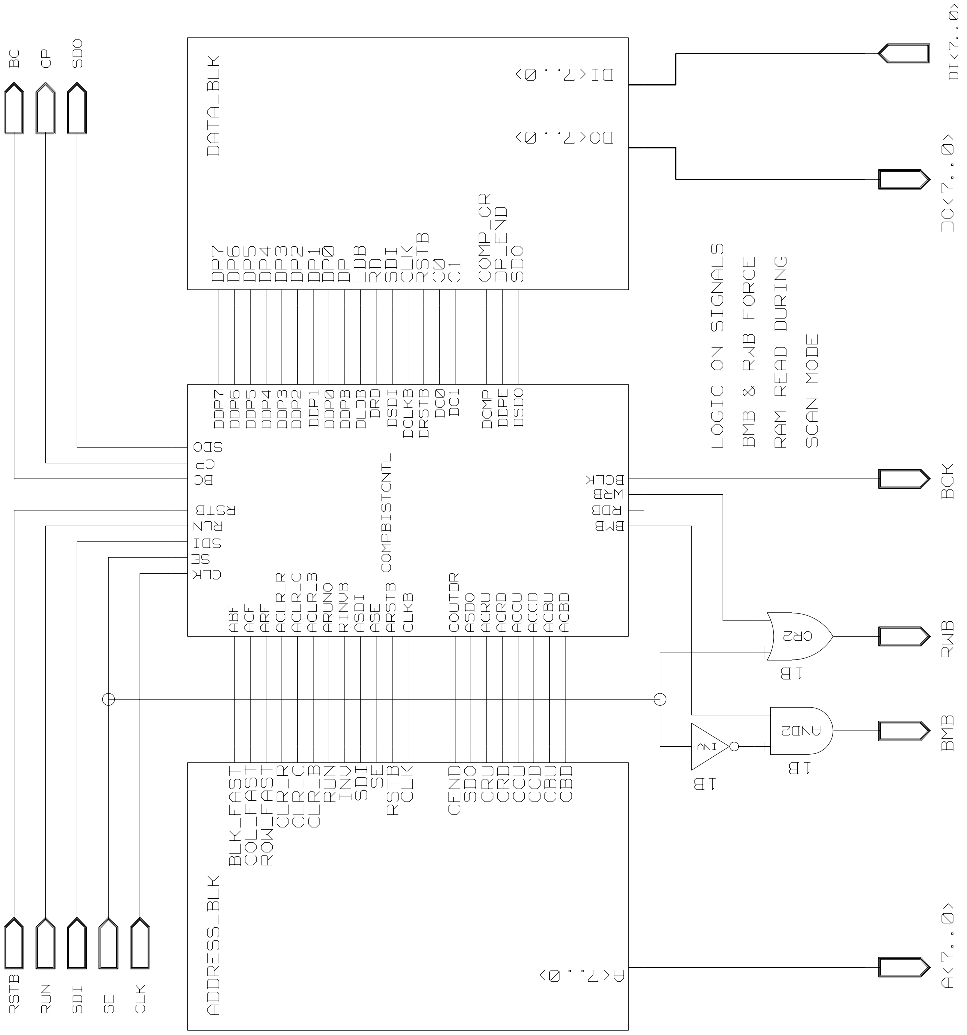


Figure 15. Comparator BIST block.

2.6. BIST Advisor

Motorola ASIC Option Development Engineering can assist in developing a comparator BIST for various ram sizes with the use of the "BIST_ADVISOR". "BIST_ADVISOR" will not generate a graphical representation of the comparator ADDRESS_BLOCK. This program will graphically show connection of the LFSR feedback tap in the comparator DATA_BLOCK and list the correct test data signature. This can predict the test signature determined by Verilog^{TM} simulation. BIST_ADVISOR can generate signatures for LFSR up to 32 bits in length.

```
              BIST Advisor
              ============
              Version 1.01

Which type of BIST do you want?
       0 = Pseudo Random BIST
       1 = Comparator BIST
BIST type: 1

BIST Data
 Number of data register bits: 12

Ram Data
 Number of data bits: 8
 Number of address: 192

    BIST Data Register Configuration
    ================================

 The diagram below shows how the feedback connections to the BIST should be made, in
accordance with the application note. In order to achieve a maximal sequence the following
feedback taps should be used:
                [12,6,4,1]
 The output from the last macro in the data register should be connected to these feedback
pins as shown below. All other feedback pins should be tied to ground.

 The data bits to and from the RAM should be connected in the same order starting at the
left most data register. Any unused data input pins should be tied to ground.

      +-----------------------+-------+-----------+--+
      |                       |       |           |  0
GND --|---+---+---+---+---+---|---+---|---+---+   | / \
      |   |   |   |   |   |   |   |   |   |   |   | ---
 FB   V   V   V   V   V   V   V   V   V   V   V   V  |
     XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX-+  (CO)
SDI -XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX-XXX---> SDO
     XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
     ^ | ^ | ^ | ^ | ^ | ^ | ^ | ^ | ^   ^   ^   ^
FROM | | | | | | | | | | | | | | | | |   |   |   |
RAM D00|D01|D02|D03|D04|D05|D06|D07|GND GND GND GND
       |   |   |   |   |   |   |   |
TO     V   V   V   V   V   V   V   V
RAM   D00 D01 D02 D03 D04 D05 D06 D07


end of pass 1: (4036 clks)
SDI-> 1   1   1   0   1   1   0   1   0   1   0   0  -> SDO

end of pass 2: (8072 clks)
SDI-> 0   0   0   0   1   1   0   1   1   1   1   0  -> SDO

end of pass 3: (12108 clks)
SDI-> 0   1   0   0   1   0   1   0   0   1   0   0  -> SDO

  TEST RESULTS
  ============

test length = 16144 clock cycles
signature :
SDI-> 1   1   0   1   1   1   0   1   0   0   0   1  -> SDO

 Note : Clock count is from the start of signature analysis.  Total test takes twice as
long.
```

2.7. Comparator BIST connection to Diffused RAMs.

To test a H4CP diffused RAM with the comparator BIST, a RAM_MUX will be necessary. The RAM_MUX will select RAM inputs from either the BIST circuitry or the customer's system RAM interface. Please refer to Figure 16.

1. Use the comparator BIST signal "BMB" to select between the RAM receiving input from either customer's RAM interface (BMB = high) or the comparator BIST (BMB = low). Usually the "BMB" signal is buffered to drive the select pin on a bank of 2:1 mux macros.
2. Generate the RAM's "A<4..0>" address bus by muxing the system interface address bus and comparator BIST address bus. Select system "A<4..0>" if comparator signal "BMB = high". Select BIST "A<4..0>" if comparator signal "BMB = low".
3. Generate the RAM's "DBI<8..0>" address bus by muxing the system interface data bus and comparator BIST data bus. Select system "DI<8..0>" if comparator signal "BMB = high". Select BIST "DO<8..0>" if comparator signal "BMB = low".
4. Generate the RAM's "ST" clock strobe by muxing the system interface strobe and comparator BIST "BCK" signal. Select system "RAM_ST" if comparator signal "BMB = high". Select BIST "BCK" if comparator signal "BMB = low".
5. Generate the RAM's "RWB" signal by muxing the system interface read/write signal and comparator BIST signal "RWB". Select system "RWB" if comparator signal "BMB = high". Select BIST "RWB" if comparator signal "BMB = low".
6. Generate the RAM's "CSB" chip select by muxing the system interface select signal and hardwired enable for the BIST. The comparator BIST does not generate a select signal to enable the RAM. Select system "CSB" if comparator signal "BMB = high". Select the hardwired chip select if comparator signal "BMB = low".
7. Connect the RAM's data out bus to the comparator BIST "DI<8..0>" inputs for data comparison and signature generation.
8. The Comparator BIST dedicated read signal "RDB" will not be used for the diffused RAM application.

Timing considerations. The comparator BIST can cause setup errors on the RAM's address lines. The root cause is after the clock edge, the comparator BIST COMPACELL address output changes before the RAM's setup time requirement is satisfied. This is a problem in the larger CDA RAMs whose setup times are much larger than the "clock to Q" time of the COMPACELL. A relative easy fix is to delay the COMPBISTCNTL "ACLKB" clock to the address block. To allow proper scan operation the address block "ASDI" must be delayed by a similar amount.

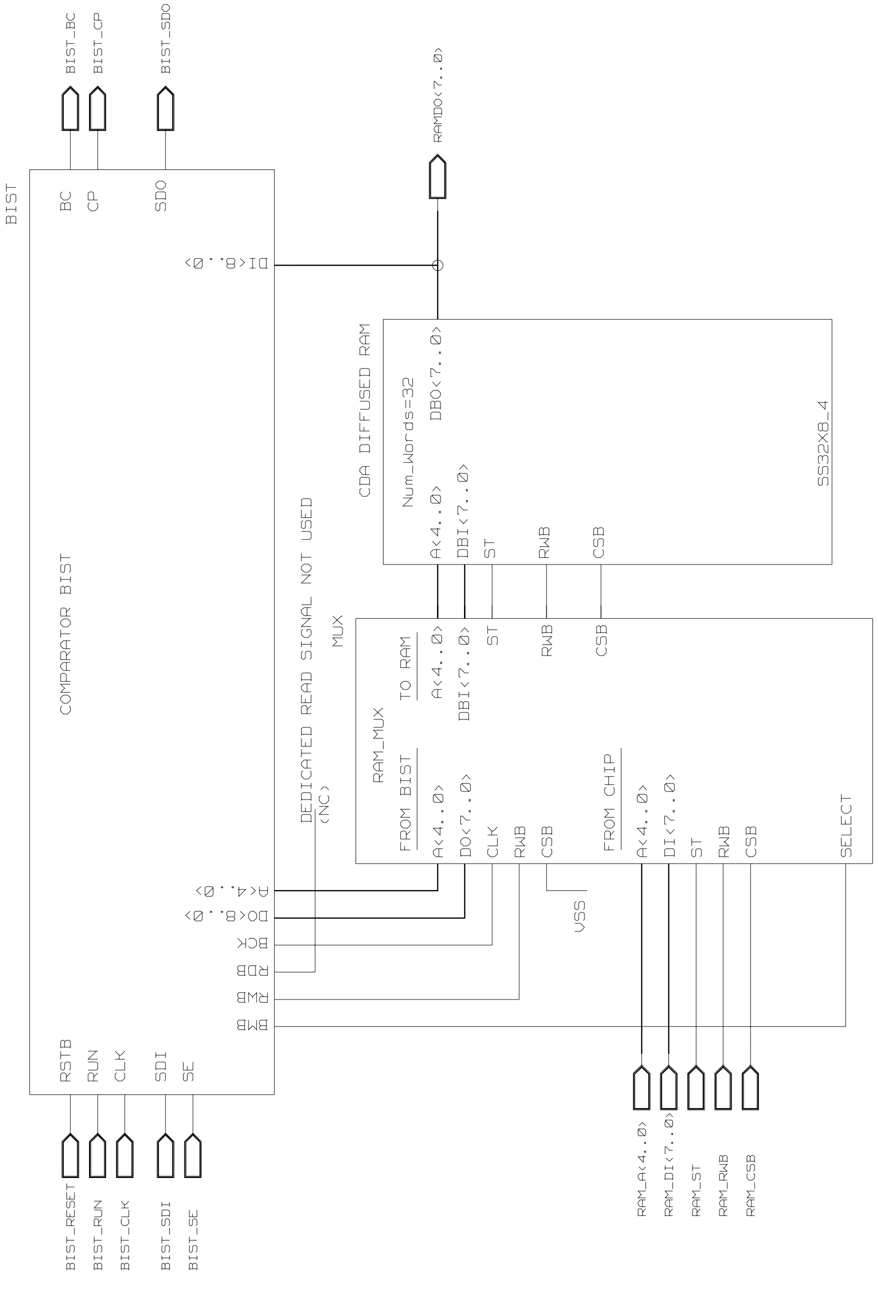


Figure 16. Comparator BIST connections to H4CP CDA RAM

2.8. Comparator BIST connection to Metallized RAM blocks.

To test a H4CP metallized RAM with the comparator BIST, a RAM_MUX will be necessary. The RAM_MUX will select RAM inputs from either the BIST circuitry or the customers system RAM interface. Please refer to Figure 17**.**

1. Use the comparator BIST signal "BMB" to select between the RAM receiving input from either customer’s RAM interface (BMB = high) or the comparator BIST (BMB = low). Usually the "BMB" signal is buffered to drive the select pin on a bank of 2:1 mux macros.
2. Generate the RAM's "A<4..0>" address bus by muxing the system interface address bus and comparator BIST address bus. Select system "A<4..0>" if comparator signal "BMB = high". Select BIST "A<4..0>" if comparator signal "BMB = low".
3. Generate the RAM's "DBI<8..0>" address bus by muxing the system interface data bus and comparator BIST data bus. Select system "DI<8..0>" if comparator signal "BMB = high". Select BIST "DO<8..0>" if comparator signal "BMB = low".
4. Generate the RAM's "RWB" signal by muxing the system interface read/write signal and comparator BIST read/write signal. To generate the BIST read/write signal "OR" the comparator "RWB" with "BCK". This logic is required due to the asynchronous operation of the metallized RAMs. Select system read/write signal if comparator signal "BMB = high". Select BIST read/write signal if comparator signal "BMB = low". The RAM_MUX must have additional buffering to drive the multiple read/write lines "R/W<7..0>" of the metallized RAM inputs.
5. Connect the RAM's data out bus to the comparator BIST "DI<8..0>" inputs for data comparison and signature generation.
6. The Comparator BIST dedicated read signal "RDB" will not be used for the metallized RAM application.

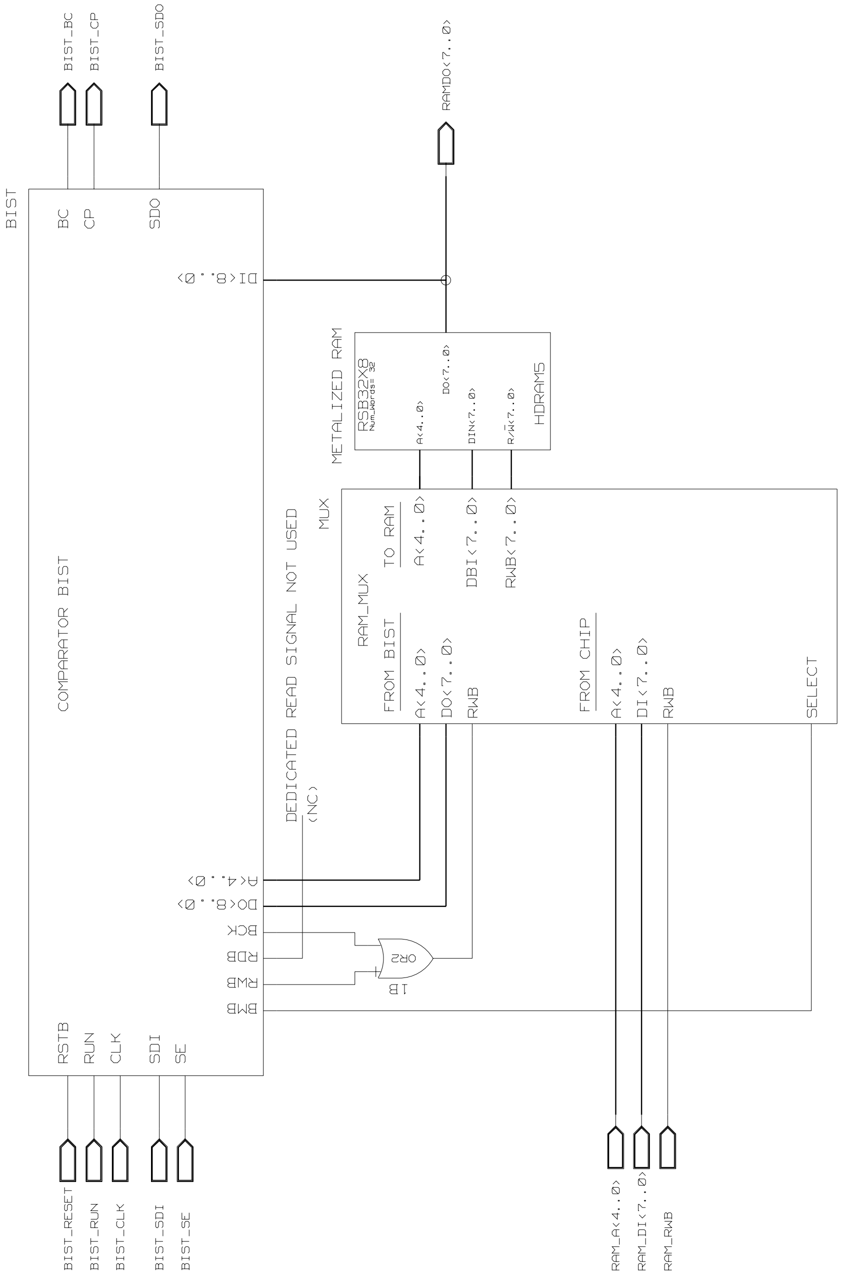


Figure 17. Comparator BIST connections to H4CP Metalized RAM

2.9. Comparator BIST Test Vector and Signature Generation

Simulation vectors are used as test vectors and to generate/verify the "test" signature. The comparator BIST can be controlled by two modes: control signals and scan path control. Both simulations contain the three basic steps: initialization, BIST execution, and extract test signature. Please refer to following Verilog^{TM} HDL simulation.

2.9.1. Simulation with control signals

initialization: The first step is to reset the BIST. The "RSTB" line must be held low for at least two clock cycles. This resets the BIST internal registers and output status flags. BMB (BIST mode) will become high, CP (comparator pass) will become high, BC (BIST complete) will become low, and SDO (serial data out) will become low. After reset return "RSTB" line high. The Comparator BIST is now in a idle reset mode.

execution: To start the Comparator BIST, raise the "RUN" line. On the next falling edge of "CLK", the "BMB" line will go low and testing begins. During execution "CP" should remain high. "CP" goes low only if the BIST test fails. The BIST test runs until completion. At completion of BIST test "BC" and "BMB" will go high. A successful test will have "CP" high.

signature: Scanning out the correct test signature gives a high degree of confidence that the Comparator BIST, RAM_MUX, and RAM are operating correctly. To scan the signature, raise the "SE" pin and use "CLK" to clock out the data from "SDO". The contents of the BIST internal registers will be changed and all BIST control lines except "BMB" may toggle. It is a good practice to use "SDI" and scan in an idle state into the BIST. After the signature has been scanned out (or entire state of the BIST and re-initialize) force "SE" and "RUN" low. BIST should now be idle.

2.9.2. Simulation with scan chain registers

The comparator BIST can also be controlled using only the scan chain. This provides a means of running the BIST without having external control of "RSTB" and "RUN" signals. The operation of each of the registers is illustrated in a sample comparator BIST scan chain. The comparator BIST can be reset and run by shifting in the following pattern into the COMPBISTCNTL controller block: SDI->"11111110000000"->SDO this pattern is shifted in starting with the right bit first and the left bit last. The test will run to completion ("BC"=1) and the test signature can be scanned out as previously described.

A sample Verilog^{TM} HDL simulation patterns and a sample Verilog^{TM} output log are included to illustrate a Comparator BIST simulation using control signal inputs.

2.9.3. Compression of Final Test Vectors

see section 1.3.10 on page 16 for Compression of Final Test vectors.

2.10. Comparator BIST Verilog^{TM} HDL simulation using control signals

```
module stim_hdl;

`define CYCLE 1000

`define BIST_CLK stim.BIST_CLK  // input signal
`define BIST_RSTB stim.BIST_RSTB // input signal
`define BIST_RUN  stim.BIST_RUN  // input signal
`define BIST_SDI  stim.BIST_SDI  // input signal
`define BIST_SE   stim.BIST_SE   // input signal
`define BIST_BC   stim.BIST_BC   // output signal
`define BIST_CP   stim.BIST_CP   // output signal
`define BIST_SDO  stim.BIST_SDO  // output signal

/* -------------------- for Clock Generation Module -- */
integer NEGEDG ;   /* Offset of the falling clock edge */
integer POSEDG ;   /* Offset of the rising clock edge  */
integer PULSEWID ; /* Width of the clock (hi) pulse    */
integer HOLDTIME ; /* delay from clock to input edge   */

/************************
  Comparator BIST Module
************************/
initial
  begin
  HOLDTIME = (`CYCLE/10) ;
     `BIST_CLK  = 0;   // input signal
     `BIST_RSTB = 1;   // input signal
     `BIST_RUN  = 0;   // input signal
     `BIST_SDI  = 0;   // input signal
     `BIST_SE   = 0;   // input signal


  #`CYCLE ;
  #`CYCLE ;
  #`CYCLE ;
  $display(" ");
  $display(" --------------------------------------------------------- ");
  $display(" ");
  $display(" Begin Comparator BIST testing") ;

  @(negedge `BIST_CLK) #HOLDTIME ; `BIST_RSTB = 0 ;   Reset Compartor BIST
  @(negedge `BIST_CLK) ;
  @(negedge `BIST_CLK) ;
  @(negedge `BIST_CLK) #HOLDTIME ; `BIST_RSTB = 1 ;   Reset Compartor BIST
  @(negedge `BIST_CLK) #HOLDTIME ; `BIST_RUN  = 1 ;   Start Compartor BIST test

                                                   Execution of Comparator BIST RAM test

  @(posedge `BIST_BC)$display("  ") ;                 End of Comparator BIST test
                         display(" Comparator BIST_COMPLETE = ",`BIST_BC," (complete) at : ",$realtime);
                         $display(" Comparator BIST_COMPARISON = ",`BIST_CP);
                         $display(" ") ;

                     #HOLDTIME ;  `BIST_SE  = 1 ;   Comparator BIST scan mode

  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_11 signature bit 1") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_10 signature bit 2") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_9  signature bit 3") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_8  signature bit 4") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_7  signature bit 5 or BIST Dout (7)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_6  signature bit 6 or BIST Dout (6)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_5  signature bit 7 or BIST Dout (5)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_4  signature bit 8 or BIST Dout (4)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_3  signature bit 9 or BIST Dout (3)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_2  signature bit 10 or BIST Dout (2)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_1  signature bit 11 or BIST Dout (1)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  Dout_0  signature bit 12 or BIST Dout (0)") ;
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Data Generator register_0 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Data Generator register_1 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Data Generator register_2 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Cycle Controller register_0 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Cycle Controller register_1 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (7)  -or- Row Address (4)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (6)  -or- Row Address (3)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (5)  -or- Row Address (2)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (4)  -or- Row Address (1)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO," BIST Address(3)  -or- Row Address (0)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (2)  -or- Column Address (2)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (1)  -or- Column Address (1)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  BIST Address (0)  -or- Column Address (0)");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Pass Controller Reg 5 pass complete ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Pass Controller Reg 4 3 clk delay ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Pass Controller Reg 3 cycle count ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Pass Controller Reg 2 cycle count ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Pass Controller Reg 1 cycle count ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_14 test passed ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_13 data retention active ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_12 BIST test complete ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_11 test type CMP vs.SIG ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_10 test type code bit_1 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_9  test type code bit_0 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_8  pass complete carry ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_7  Scan BIST reset bit ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_6  BIST diagnostic mode ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_5  Scan BIST r/w control bit_ 1 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_4  Scan BIST r/w control bit_ 0 ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_3  Scan BIST RUN ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_2  Scan BIST SE ");
  @(negedge `BIST_CLK) $display(" BIST_SDO = ",`BIST_SDO,"  COMPBISTCNTL Controller Reg_1  Scan BIST RSTB ");

  @(posedge `BIST_CLK) #HOLDTIME   ;
                          `BIST_SE  = 0 ;BIST scan mode exit
                          `BIST_RUN = 0 ;disable BIST
  @(negedge `BIST_CLK) ;

end

/**************************
  Clock Generation Module
**************************/
initial
  begin
       NEGEDG   = (7 * `CYCLE/10) ;
       POSEDG   = (2 * `CYCLE/10) ;
       PULSEWID = (5 * `CYCLE/10) ;
  #`CYCLE ;
  #POSEDG ;
            `BIST_CLK = 1 ;
            $display(" Initial Rising Edge of BIST_CLK = ",`BIST_CLK,"   at time:",$realtime);
   forever
     begin
       #PULSEWID `BIST_CLK = 0 ;
       #(`CYCLE - PULSEWID) `BIST_CLK = 1 ;
     end
end
```

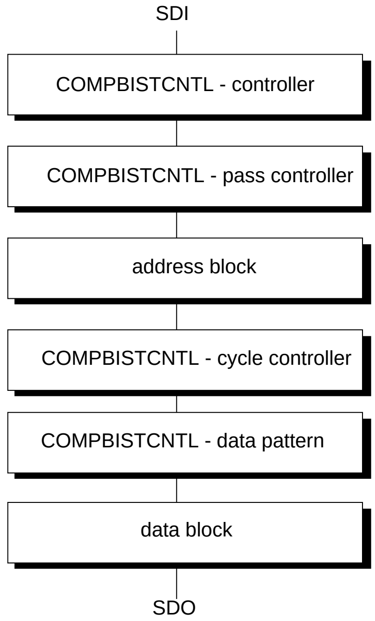
2.11. Comparator BIST Verilog^{TM} simulation output

```
Initial Rising Edge of BIST_CLK = 1   at time:1200
---------------------------------------------------------------
Comparator BIST Memory
Begin Comparator BIST testing   at time: 3000

BIST_COMPLETE   = 1 (complete)
BIST_COMPARISON = 1

BIST_SDO = 1  Dout_11 signature bit 1
BIST_SDO = 0  Dout_10 signature bit 2
BIST_SDO = 0  Dout_9  signature bit 3
BIST_SDO = 0  Dout_8  signature bit 4
BIST_SDO = 1  Dout_7  signature bit 5  or BIST Dout(7)
BIST_SDO = 0  Dout_6  signature bit 6  or BIST Dout(6)
BIST_SDO = 1  Dout_5  signature bit 7  or BIST Dout(5)
BIST_SDO = 1  Dout_4  signature bit 8  or BIST Dout(4)
BIST_SDO = 1  Dout_3  signature bit 9  or BIST Dout(3)
BIST_SDO = 0  Dout_2  signature bit 10 or BIST Dout(2)
BIST_SDO = 1  Dout_1  signature bit 11 or BIST Dout(1)
BIST_SDO = 1  Dout_0  signature bit 12 or BIST Dout(0)
BIST_SDO = 0  COMPBISTCNTL Data Generator register_0
BIST_SDO = 1  COMPBISTCNTL Data Generator register_1
BIST_SDO = 0  COMPBISTCNTL Data Generator register_2
BIST_SDO = 0  COMPBISTCNTL Cycle Controller register_0
BIST_SDO = 0  COMPBISTCNTL Cycle Controller register_1
BIST_SDO = 0  BIST Address(7) -or- Row Address(4)
BIST_SDO = 0  BIST Address(6) -or- Row Address(3)
BIST_SDO = 0  BIST Address(5) -or- Row Address(2)
BIST_SDO = 0  BIST Address(4) -or- Row Address(1)
BIST_SDO = 0  BIST Address(3) -or- Row Address(0)
BIST_SDO = 0  BIST Address(2) -or- Column Address(2)
BIST_SDO = 0  BIST Address(1) -or- Column Address(1)
BIST_SDO = 0  BIST Address(0) -or- Column Address(0)
BIST_SDO = 0  COMPBISTCNTL Pass Controller Reg 5 pass complete
BIST_SDO = 0  COMPBISTCNTL Pass Controller Reg 4 3 clk delay
BIST_SDO = 0  COMPBISTCNTL Pass Controller Reg 3 cycle count
BIST_SDO = 0  COMPBISTCNTL Pass Controller Reg 2 cycle count
BIST_SDO = 0  COMPBISTCNTL Pass Controller Reg 1 cycle count
BIST_SDO = 0  COMPBISTCNTL Controller Reg_14 test passed
BIST_SDO = 0  COMPBISTCNTL Controller Reg_13 data retention active
BIST_SDO = 1  COMPBISTCNTL Controller Reg_12 BIST test complete
BIST_SDO = 0  COMPBISTCNTL Controller Reg_11 test type CMP vs.SIG
BIST_SDO = 0  COMPBISTCNTL Controller Reg_10 test type code bit_1
BIST_SDO = 1  COMPBISTCNTL Controller Reg_9  test type code bit_0
BIST_SDO = 0  COMPBISTCNTL Controller Reg_8  pass complete carry
BIST_SDO = 0  COMPBISTCNTL Controller Reg_7  Scan BIST reset bit
BIST_SDO = 1  COMPBISTCNTL Controller Reg_6  BIST diagnostic mode
BIST_SDO = 1  COMPBISTCNTL Controller Reg_5  Scan BIST r/w control bit_ 1
BIST_SDO = 1  COMPBISTCNTL Controller Reg_4  Scan BIST r/w control bit_ 0
BIST_SDO = 1  COMPBISTCNTL Controller Reg_3  Scan BIST RUN
BIST_SDO = 1  COMPBISTCNTL Controller Reg_2  Scan BIST SE
BIST_SDO = 1  COMPBISTCNTL Controller Reg_1  Scan BIST RSTB
```

2.12. Comparator BIST scan registers

The scan path can be used to access the internal control registers in the COMPBISTCNTL. This can be used to invoke the more complex and powerful operating modes of the comparator BIST. These functions were not intended for normal production testing but have proved valuable in some specialized analysis (i.e. characterization). Contact your Motorola ASIC representative for more information.

This description will help identify the registers as observed through scan path. The scan path is routed through the functional blocks of the Comparator BIST in the following order.


SDI
COMPBISTCNTL - controller
COMPBISTCNTL - pass controller
address block
COMPBISTCNTL - cycle controller
COMPBISTCNTL - data pattern
data block
SDO

COMPBISTCNTL controller .... interfaces between the control lines and the operation of the comparator BIST module. It determines the type of operation being performed, the number of passes, the type of data analysis, status of comparison test, and if BIST test is complete.

COMPBISTCNTL pass controller .... manages the seven cycles in each pass of 21N march and controls the data polarity. It also determines which cycles should use the wait count in the data retention test.

Comparator BIST address block .... Customer configured block. The RAM's LSB address bit should be first while the RAM's MSB address line should be last.

COMPBISTCNTL cycle controller .... manages the three phases in each cycle and controls the data polarity. It also provides an override for the "RDB" and "WRB" lines allowing direct memory access. The cycle consists of (read_X, write_Y, read_Y) where "X" is the inverse pattern of "Y". The first read is suppressed in cycles 1 & 2.

COMPBISTCNTL data pattern............. manages the selection of which one of eight data patterns is run during this test pass.

Comparator BIST data block .... Customer configured block. The RAM's LSB data bit should be first while the RAM's MSB data line should be last. Additional registers may exist after the RAM's MSB to prevent data signature aliasing.

COMPBISTCNTL controller

register 1 Clocked version of signal "RSTB". A "0" state indicates the reset line is active. This register provides access to the current RSTB status. It should always be loaded with "1".

register 2 Stored status of signal "SE". It should always be loaded with a "1".

register 3 Stored status of signal "RUN". It can be loaded with "1" to start BIST testing. It can be loaded with "0" to disable BIST testing.

register 4 See 5

register 5 Registers 4&5 contain a coded state of the read/write/modify function. Using the order (#4, #5) state 11 indicates normal comparator BIST operation. State 10 indicates a read to address. State 01 indicates a write to address. State 00 indicates a modification of address. Note that the modify command consist of a write followed by a read of address.

register 6 Controls the diagnostic mode. To enter diagnostic mode load bit with "0". The BIST will then halt during comparison test if an error is found. The BIST will stop in the cycle after the error was detected. In normal BIST operation this bit's value is "1".

register 7 Allows the comparator BIST be reset from the scan chain. This reset will affect only the COMPACELLs and COMPDCELLs in the address and data blocks. It has no affect on registers in COMPBISTCNTL macro.

register 8 Contains the carry signal when a pass is completed. This bit is in the scan path to allow observation into the COMPBISTCNTL control logic.

register 9 See 10

register 10 Registers 9&10 contain a coded state of the test being run by the comparator BIST. Using the order (#9,#10) state 00 is block_fast. State 10 is column_fast. State 01 is row_fast. State 11 is a unsupported data retention mode. During normal Comparator BIST operation the initial code will be 00 and at the end of the test it will be 11. These registers should be normally loaded with 00.

register 11 Determines the types of analysis. A value of "0" indicates comparator test. A value of "1" indicates signature analysis. During normal BIST operation the initial code is "0" and at the end of testing it will be "1". It should normally be loaded with "0".

register 12 Indicates completion of test. A "1" indicates the test is finished. When this register reaches the "1" state all COMPBISTCNTL registers are put into halt mode. This should normally be loaded with "0".

register 13 Indicates the comparator BIST is running the data retention test. This test is not part of the production flow, but used for characterization. When this register is "1" all the registers are put into halt mode. This should normally be loaded with "0".

register 14 Indicates if the comparison test passed. A "0" means no errors were detected. This should normally be loaded with "0". Note this BIST does not indicate status of the signature analysis test.

COMPBISTCNTL-pass controller

register 1 See register 3

register 2 See register 3

register 3 Registers 1,2,&3 determine 21N march cycle to be run. Using the order(#1,#2,#3) the state are 000=cycle1, 100=cycle2, 010=cycle3, 110=cycle4, 001=cycle5, 101=cycle6, 111=cycle7.

register 4 Introduces a three clock cycle delay when the count direction is changed from count up to count down. It is normally "0" but becomes "1" for one Read/Write/Read cycle at the end of cycle 4. It should normally be loaded with "0".

register 5 Indicates that the pass is complete. When all seven cycles have been performed this becomes "1" and stops the address counter. It should normally be loaded with "0".

Comparator BIST address block

Customer configured block. The RAM's LSB address bit should be first while the RAM's MSB address line should be last.

COMPBISTCNTL-cycle controller

register 1 Controls the read/write operation. "0" performs read if not suppressed. "1" performs write.

register 2 Controls the data polarity. "0" indicates that "X" pattern is used. A "1" indicates the inverse or "Y" pattern is used. Normal sequence of register 1 & 2 during a cycle is 00, 11, 01 corresponding to read_X, write_Y, read_Y.

COMPBISTCNTL - data pattern

register 1 see register 3

register 2 see register 3

register 3 registers 1,2,&3 determine which 1 of 8 data patterns will be run during this test pass. Registers should normally be loaded with "000".

Comparator BIST data block

Customer configured block. The RAM's LSB data bit should be first while the RAM's MSB data line should be last. Additional registers may exist after the RAM's MSB to prevent data signature aliasing.

2.13. Description of Comparator BIST data retention test

Memories can fail due to leakage from the data cells. This can be found by loading the memory with known data and checking that it is still present after time. The COMPBISTCNTL modifies the 21N march algorithm to perform the data retention test.

	(CY1,CY2,CY3)	first address	last address	address counter
Phase 1	(--, W0, R0)	lowest "0"	highest	increment
Phase 2	(--, --, --)	lowest "0"	highest	increment
Phase 3	(--, --, --)	lowest "0"	highest	increment
Phase 4	(R0, W1, R1)	lowest "0"	highest	increment
Phase 5	(--, --, --)	highest	lowest "0"	decrement
Phase 6	(--, --, --)	highest	lowest "0"	decrement
Phase 7	(R1, W0, R0)	highest	lowest "0"	decrement

In phase 2, 3, 5, and 6, the memory is not accessed. This is the wait time used for the data retention test. To extend this time, additional high order address bits are used during these phases (note COMPBISTCNTL signal COUNTDR). This function when combined with additional tester control in a characterization environment, can be the start of a rudimentary data retention test.

Notes:

ASIC REGIONAL DESIGN CENTERS – U.S.A.

California, San Jose (408) 749-0510	Georgia, Atlanta (404) 729-7100	Illinois, Chicago (847) 413-2500	Massachusetts, Marlborough (508) 481-8100

ASIC REGIONAL DESIGN CENTERS – International

European Headquarters Germany, Munich (089) 92103-306	England, Aylesbury, Bucks (0)1296) 395252	France, Velizy (01) 3463900	Holland, Best (04998) 61211
Hong Kong, Tai Po (852)2666-8333	Israel, Tel Aviv (09) 590-303	Italy, Milan (02) 82201	Japan, Tokyo (03) 440-3311
Sweden, Stockholm (08) 734-8800			

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary overtime. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and b are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447 or 602-303-5454

MFAX: RMFAX0@email.sps.mot.com -TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center,
3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-81-3521-8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298