

Functional Test for Shifting-type FIFOs

Ad J. van de Goor*

Ivo Schanstra*

Yervant Zorian**

*Delft Univ. of Technology, P.O.Box 5031, 2600 GA Delft, The Netherlands

** AT&T Bell Laboratories, Eng. Res. Center, P.O.Box 900,
Princeton, N.J. 08542-0900, USA

Abstract

FIFO memories impose special test problems because of their built-in addressing restrictions and access limitations. With the increasing use of FIFOs, as a stand-alone chip or as embedded macros in ASICs, generic algorithms are needed to test FIFOs. This paper addresses the problem of testing the widely available shifting-type FIFOs; it introduces specific fault models and a set of generic tests which have a test length of $O(n)$ and can be used for the stand-alone chip as well as for the embedded macro version of the FIFO.

Keywords: Fault models, memory testing, FIFO testing, functional tests, shifting-type FIFO.

1 Introduction

First-In First-Out 'FIFO' memories are widely used as a building block to buffer data between subsystems operating at different data rates. Various telecommunications oriented systems, such as ATMs, require such memories. Today different types of FIFOs are in use, both as stand-alone chips and as embedded macros in ASICs. They can be distinguished, based on the way they are implemented as:

1. RAM-type FIFOs, which use a RAM (usually an SRAM) to contain the data together with a read and a write address register to access the data. Depending on the way the data can be accessed, the following RAM-type FIFOs can be recognized: 1) single-port arbitration RAM-type FIFOs (The read and write addresses are each specified with an N -bit ($N = \log_2 n$, where n is the number of words in the FIFO) counter accessing a single-port memory using an arbiter.); and 2) dual-port RAM-type FIFOs (They allow for simultaneous read and write operations, while the address-

ing mechanisms can be implemented using counters (counter-address) or shift registers (ring-address)).

2. Shifting-type FIFOs, based on a self-clocking shift register which shifts data from the write port into the last unused location of the read port. They allow for simultaneous read and write operations and have the property that a word written via the write port has to bubble through the n -word FIFO to the read port requiring a bubble-through time of $n - 1$ handshake signals.

The well-known functional tests for SRAMs [2,3,5] often cannot be applied to other types of memories because of the need for specific fault models and test algorithms. Recently, several studies were reported on testing Double Buffered Memories [9] and Single-Address Order Memories [7]. FIFOs also cannot be tested like standard RAMs because of the following built-in addressing restrictions: only a single address order can be used, and a march element is restricted to a maximum of one write operation and one read operation. This means that the classical march algorithms for testing RAMs [2,3,5] cannot be used with FIFOs.

In general, the test approach of a FIFO is like any other part, it could consist of the test sequence: parametric tests, functional tests and asynchronous tests. Algorithms and methods for DC and AC parametric tests have been described in [10]. Asynchronous tests, consisting of verifying the correct FIFO operation for different timing relationships (such as partial overlap between the write and the read clock signals) are reported in [8]; they report failure rates as high as 75% for chips which passed the parametric and functional tests.

Fault models and tests for arbitration SRAM-type FIFOs have been described in [4], and for Ring-address SRAM-type FIFOs in [6,11]. However, no test algorithms for shifting-type FIFOs have been reported. Many manufacturers are providing such chips however: Ad-

vanced Micro Devices (Am2812 and Am2813), Philips (74HC/HT40105 and 74HC/HT7030), Texas Instruments (SN74ALS236), and Monolithic Memories (74S225/A, C67401A/B, C67402A/B, and 67413).

The organization of this paper is as follows: Section 2 describes the functional model of shifting-type FIFOs; Section 3 describes a comprehensive set of fault models covering the memory array faults, the addressing faults and the FIFO functionality logic faults; Section 4 introduces a generic test algorithm for shifting-type FIFOs and analyzes its fault coverage; finally, Section 5 concludes the paper.

2 Shifting-type FIFO functional model

Figure 1 shows a functional model of a shifting-type FIFO consisting of n B -bit words. Two blocks, each consisting of n stages, can be recognized: the memory array 'MA' block, which stores the data, and the FIFO control 'FC' block. Each FC stage controls the corresponding MA stage using a data valid latch 'DVL' indicating that the corresponding MA stage contains valid data. Data is presented to the FIFO via the data input 'DI' lines and shifted into MA[0], stage 0 of the memory array, under control of the write clock 'WCK'. When the next stage does not contain valid data, the data is shifted into that stage; this process is repeated until the data is in the last stage from where it can be read on the data output 'DO' lines, this will occur with a bubble-through delay of $n - 1$ internal handshake (clock) signals. Data may be written, under control of the write clock 'WCK', and read under control of the read clock 'RCK', simultaneously with independent data rates; the shifting of the data internal to the FIFO occurs at yet another rate under control of internal handshake signals.

The FIFO control logic consists of n identical stages. Stage i has the following inputs and outputs (see Figure 1):

1. Data valid input from stage $i - 1$ 'DV $_{i-1}$ ', $i \in \{1..n\}$. The signal value of DV $_i$ is the contents of DVL $[i]$; i.e. DV $_i = (DVL[i])$. The signal DV $_0$ has the external name FF 'Full Flag'; the signal DV $_{n-1}$ has the external name EF 'Empty Flag'.
2. Data read output pulse to stage $i - 1$ 'DR $_i$ ', $i \in \{1..n\}$. The DR $_i$ pulse is used to reset the DVL $[i - 1]$ latch when data is shifted from stage $i - 1$ to stage i . The signal DR $_n$ has the external name RCK 'Read Clock'.

The FIFO control block implements the following functions:

- Empty Flag signal 'EF': it signals that MA $[n - 1]$ does not contain valid data.
- Full Flag signal 'FF': it signals that MA[0] contains valid data.

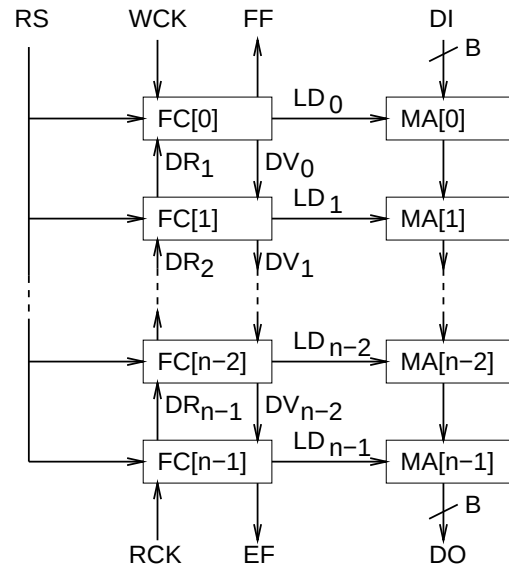


Figure 1: Functional model of a shifting-type FIFO

- Reset operation 'RS': it brings the FIFO in its initial state, regardless of its previous state: DVL $[i] := 0$, $\forall i \in \{0..(n - 1)\}$
- Write operation 'WO': a B -bit word is written into MA[0] under control of WCK, and DVL[0] is set. A WO can only be performed if $FF = 0$. Note: $FF = 0$ means (DVL[0]) = 0. If $FF = 0$ and WCK then Begin MA[0] := DI; DVL[0] := 1; End
- Shift operation 'SO': a B -bit data word is copied from MA $[i]$ into MA $[i + 1]$, under control of the internal handshake, iff (DVL $[i] = 1$ and (DVL $[i + 1] = 0$). If (DVL $[i] = 1$) and (DVL $[i + 1] = 0$) then Begin MA $[i + 1] := (MA[i])$; DVL $[i] := 0$; DVL $[i + 1] := 1$; End, $i \in \{0..(n - 2)\}$.
- Read operation 'RO': a B -bit data word is read from MA $[n - 1]$ under control of RCK, and DVL $[n - 1]$ is reset. A RO can only be performed if $EF = 0$. Note: $EF = 0$ means (DVL $[n + 1] = 1$). If $EF = 0$ and RCK then Begin DVL $[n - 1] := 0$; DO := (MA $[n - 1]$); End

3 Shifting-type FIFO fault models

The FIFO function can be divided three ways: the memory array, the addressing mechanism and the FIFO functionality logic. Each has its specific fault models.

3.1 Memory array fault model

The memory array consists of n B -bit registers organized in such a way that the data from $MA[i]$ can be shifted, word parallel, into $MA[i + 1]$, $i \in \{0..(n - 2)\}$. The faults considered in the cells of the memory array are those relevant to SRAM cells [1]:

1. Stuck-at fault 'SAF' $<\forall/x>$; for any operation denoted by ' $\forall$ ', the cell remains at x ($x \in \{0, 1\}$).
2. Transition fault 'TF' $<\uparrow/0>$, $<\downarrow/1>$; after an up transition ' $\uparrow$ ' the cell remains at '0' and vice versa.
3. Data retention fault 'DRF' [1]; after a time ' Del ' the cell loses its data.

The memory array consists of n B -bit registers; they are implemented as latches and can be read without requiring sense amplifiers. Therefore a stuck-open fault 'SOF' [1] in a cell will behave as a SAF.

Data is transferred to/from a register a word at a time, requiring a word-wide path of B bitlines. A SAF or a SOF on a bitline to a particular cell will appear as a SAF in that cell. The only fault remaining is a bridging fault 'BF' between the bitlines:

4. Bridging fault 'BF': a short between bitlines j and $j + 1$ ($j \in \{0..(B - 2)\}$) of a register. The resulting function of the BF can be the AND (BFand) or the OR (BFor) function.

3.2 Addressing fault model

Loading data into a register $MA[i]$ is done under control of a pulse on the corresponding load data line ' LD_i ' (see Figure 1). LD_i performs the addressing function as well as the control (i.e. clocking) function. A SOF on LD_i will behave as a SAF on LD_i . A SA0 (stuck-at-0) fault on LD_i will prevent $MA[i]$ to accept data from $MA[i - 1]$, $i \in \{0..(n - 1)\}$; note: $MA[-1]$ represents the DI lines. A SA1 fault on LD_i will have the effect that $MA[i]$ accepts data from $MA[i - 1]$ on every internal handshake, regardless of whether the $(DVL[i - 1]) = 1$ and $(DVL[i]) = 0$. Therefore, the addressing faults of interest are:

5. SA0 fault on LD_i
6. SA1 fault on LD_i

3.3 FIFO functionality logic fault model

This fault model comprises the faults related to the control logic of the FIFO control blocks of Figure 1. For control block i they relate to the cases a through c.

- a. The improper operation of $DVL[i]$, $i \in \{0..(n - 1)\}$
A SA0 fault will cause valid data in $MA[i]$ to be ignored such that data transfers from $MA[i]$ to $MA[i + 1]$ will be inhibited; a SA1 fault will inhibit data transfers from $MA[i - 1]$ to $MA[i]$. A SOF will behave as a SAF and a TF will behave as a SAF after one data transfer from $MA[i - 1]$ to $MA[i + 1]$. The effect of a DRF depends on whether the cell will become 0 (a DRF0 fault) or a 1 over time. In case of a DRF0 fault in $DVL[i]$ the data in $MA[i]$ will be lost; in case of a DRF1 fault in $DVL[i]$ the old, invalid data in $MA[i]$ will become valid again. So, effectively only the following faults have to be considered:

7. SA0 fault in $DVL[i]$, $i \in \{0..(n - 1)\}$
8. SA1 fault in $DVL[i]$, $i \in \{0..(n - 1)\}$
9. DRF0 in $DVL[i]$, $i \in \{0..(n - 1)\}$
10. DRF1 in $DVL[i]$, $i \in \{0..(n - 1)\}$

- b. Improper reset operation 'RS', write operation 'WO', shift operation 'SO', and read operation 'RO'.

11. Improper RS 'Reset' operation
($DVL[i]$) $\neq 0$ after an RS operation, $\forall i \in \{0..(n - 1)\}$
12. Improper WO 'Write Operation'. Note: $FF = (DVL[0])$.

- (l) If $(DVL[0]) \neq 0$ and WCK then Begin
 $MA[0] := DI$; $DVL[0] := 1$; End
- (l) If $(DVL[0]) = 0$ and WCK then
 $MA[0] \neq DI$ or $DVL[0] \neq 1$

13. Improper SO 'Shift Operation'

- (a) If $(DVL[i]) = 1$ and $(DVL[i + 1]) \neq 0$ then
Begin $MA[i + 1] := (MA[i])$; $DVL[i] := 0$;
 $DVL[i + 1] := 1$; End, $i \in \{0..(n - 2)\}$
- (b) If $(DVL[i]) \neq 1$ and $(DVL[i + 1]) = 0$ then
Begin $MA[i + 1] := (MA[i])$; $DVL[i] := 0$;
 $DVL[i + 1] := 1$; End, $i \in \{0..(n - 2)\}$
- (c) If $(DVL[i]) = 1$ and $(DVL[i + 1]) = 0$ then
 $MA[i + 1] \neq (MA[i])$ or $DVL[i] \neq 0$ or
 $DVL[i + 1] \neq 1$, $i \in \{0..(n - 2)\}$

14. Improper RO 'Read Operation'. Note: $EF = (DVL[n - 1])$.

- (a) If $(DVL[n - 1]) \neq 1$ and RCK then Begin
 $DVL[n - 1] := 0$; $DO := (MA[n - 1])$; End
- (b) If $(DVL[n - 1]) = 1$ and RCK then Begin
 $DVL[n - 1] \neq 0$ or $DO \neq (MA[n - 1])$

- c. Faults in the DV_i , the RD_i , the EF, FF, WCK and the RCK lines

These faults will be detected by fault models 11 through 14.

4 Test for shifting-type FIFOs

Similar to the fault models of the shifting-type FIFO, the test will be divided three ways: a memory array test, an addressing test and a FIFO functionality logic test. In section 4.4 these three tests will be aggregated into a single composite test.

4.1 Memory array test

A test for the memory array has to cover the faults of Section 3.1: faults 1 through 4, which are SAFs, TFs, DRFs and BFs. Because a B -bit wide memory is used, the test has to use B -bit background patterns [2,4]. In order to detect the BFand and BFor faults the following two background patterns are proposed: 0101..., denoted by the symbol G ; and the inverse pattern 1010..., denoted by the symbol $\overline{G}$. The detection of TFs requires that every cell makes an up and a down transition; while the detection of SAFs requires every cell to contain the 0 and the 1 value. Table 1 shows a test which will detect the proposed faults in the memory array. It consists of 7 steps:

Step 1: the RS 'reset' operation initializes the FIFO

Step 2: the FIFO is filled with the alternating pattern $G|\overline{G}$. The notation is as follows:

$\uparrow_0^{n-1}$ denotes a sequence of operations which is performed from location 0 to location $n - 1$; i.e., it consists of $((n - 1) - 0 + 1)$ operations
 $wG|\overline{G}$ denotes that a write operation takes place at each address; the $G|\overline{G}$ denotes that the alternating pattern pair $G, \overline{G}$ is written (i.e. G to word 0, $\overline{G}$ to word 1, G to word 2, etc.)

Step 3: wait for an amount of time Del for a DRF to develop; a typical value for Del is 100 ms [1].

Step 4: read one word with expected value G . Due to the read operation the contents of words 0 through $n - 2$ will bubble-through to words 1 through $n - 1$ such that each word will now contain its inverse value.

Step 5: write one word with value G (It is assumed that n is even).

Step 6: wait for a DRF to develop; the cells now contain the inverse value of Step 3.

Step 7: read all FIFO words with expected value $\overline{G}|G$.

In the test of Table 1 every cell makes at least $n/2$ up and $n/2$ down transitions such that SAFs and TFs will be detected. DRFs (DRF0 and DRF1) will be detected because each cell has been in respectively a 1 and a 0 state; the rG and wG operations of Steps 4 and 5 respectively guarantee that each cell which contained the value $x, x \in \{0, 1\}$, will

Step #	Operation $O(1) \quad O(n)$	#ops	Comments
1	RS	1	Initialize
2	$\uparrow_0^{n-1} wG \overline{G}$	n	Fill FIFO
3	Del	Del	DRF test
4	rG	1	
5	wG	1	
6	Del	Del	DRF test
7	$\uparrow_0^{n-1} r\overline{G} G$	n	Empty FIFO

Table 1: Memory array test (length: $2n + 3 + 2Del$)

now contain the value $\overline{x}$, after which the second Del operation is applied. BFs will be detected because of the use of the $G|\overline{G}$ background pattern. Note: in case of BFs between any set of bitlines, rather than between logically adjacent bitlines, a walking 1 background pattern may be used. This pattern will be: $G_0, G_1, \dots, G_{G-1}$; where $G_0 = 100 \dots 0$, $G_1 = 010 \dots 0$, and $G_{B-1} = 000 \dots 1$. The proposed tests have to be changed accordingly.

4.2 Addressing test

A test for the addressing mechanism has to cover the faults of Section 3.2: Faults 5 through 6, which are a SA0 and a SA1 fault on the load data lines.

A SA0 fault on LD_i will prevent $MA[i]$ from being loaded with new data; however, the $(MA[i])$ can still be copied to $MA[i + 1]$. This fault will behave as a SAF in all cells of $MA[i]$. A SA1 fault on LD_i will cause $MA[i]$ to be loaded with every internal handshake; a faulty data transfer from $MA[i - 1]$ to $MA[i]$ will take place each handshake following the handshake which first loaded $MA[i]$ correctly.

The test of Table 1 detects LD_i SA0 because it detects SAFs in the memory array. It also detects LD_i SA1 because the faulty data transfers of $(MA[i - 1])$ into $MA[i]$ will take place, while the Del operations allow for sufficient time for these false data transfers from $MA[i - 1]$ to $MA[i]$; through the use of the alternating $G|\overline{G}$ background pattern these faulty data transfers will cause the inverse value to be loaded into $MA[i]$ which will be detected as a SAF in $MA[i]$.

4.3 FIFO functionality logic test

The test for the FIFO functionality logic has to cover Faults 7 through 14 of Section 3.3. The way each of these faults can be detected is described below.

Fault 7: DVL $[i]$ SA0. The result of this fault is that no data transfers can take place from $MA[i]$ which implies that any number of words can be written into the

FIFO, and no words can be read from the FIFO because EF will always be 1.

Fault 8: $DVL[i]$ SA1. The result of this fault is that no data transfers can take place into $MA[i]$ which implies that less than n words can be written into the FIFO; while after the bubble-through time $EF = 0$ and all words $MA[j], \forall j \in \{i..(n-1)\}$, will contain the same data.

Fault 9: DRF0 in $DVL[i]$. This fault causes $(MA[i])$ to be lost after waiting Del sec. A test for this fault consists of filling the FIFO, waiting Del sec. after which the FF will become 0 and less than n words can be read from the FIFO.

Fault 10: DRF1 in $DVL[i]$. The $(MA[i])$ which was invalid, will become valid. A test for this fault consists of waiting Del sec., while the FIFO is empty; after this $EF \neq 1$ such that one or more words can be read from the FIFO, while less than n words can be written into the FIFO.

Fault 11: improper reset operation. $DVL[i] \neq 0$ after an RS operation. The effect of this fault is that after the reset operation $EF \neq 1$, one or more words can be read from the FIFO, and less than n words can be written into the FIFO.

Fault 12: improper write operation

- a. If $(DVL[0]) \neq 0$ and WCK then Begin $MA[0] := DI; DVL[0] := 1$; End. A word can be written into the FIFO while stage 0 is full; i.e. it contains the background pattern G_1 . A test for this fault consists of filling the FIFO, thereafter attempt to write a background pattern G_2 (whereby $G_2 \neq G_1$), thereafter verify the contents of $MA[0]$ (by reading all n words).
- b. If $(DVL[0]) = 0$ and WCK then $MA[0] \neq DI$ or $DVL[0] \neq 1$. The fault $MA[0] \neq DI$ will behave as a SAF in $MA[0]$; the fault $DVL[0] \neq 1$ will behave as Fault 7: $DVL[0]$ SA0.

Fault 13: improper shift operation

- a. A SO into stage $i+1$ takes place while $(DVL[i+1]) \neq 0$. The effect of this fault is similar to Fault 7: any number of words can be written into the FIFO.
- b. A SO into stage $i+1$ takes place while $(DVL[i]) \neq 1$. The effect of this fault is similar to Fault 8: after the bubble-through time $EF = 0$ and all words $MA[j], \forall j \in \{i..(n-1)\}$, will contain the same data.
- c. The incorrect data transfer $MA[i+1] \neq (MA[i])$ will behave as Fault 1: SAF in the memory array. The incorrect operation $DVL[i] \neq 0$ will behave as Fault 8: SA1 fault in $DVL[i]$. The

incorrect operation $DVL[i+1] \neq 1$ will behave as Fault 7: SA0 fault in $DVL[i+1]$.

Fault 14: improper read operation

- a. The effect of this fault is that word $n-1$ in the FIFO can be read an arbitrary number of times, regardless of the fact whether it contains valid data.
- b. The incorrect operation $DVL[n-1] \neq 0$ will have the effect that only the first word written into the FIFO will enter the last stage, where it can be read an arbitrary number of times. The incorrect data transfer $DO := (MA[n-1])$ will behave as Fault 1: a SAF in the memory array.

Step #	Operation	Flag check		#ops	Faults detected
		$O(1)$	$O(n)$		
1	RS	0	1	1	11
2a	$\uparrow_0^{n-2} wG \overline{G}$	0	0	$n-1$	8, 11
2b	$w\overline{G}$	1	0	1	7, 8, 11
3	Del	1	0	Del	9
4	wG	1	0	1	
5a	$\uparrow_0^{n-2} rG \overline{G}$	0	0	$n-1$	
5b	$r\overline{G}$	0	1	1	12a
6	Del	0	1	Del	10

Table 2: FIFO functionality logic test (length: $2n + 2 + 2Del$)

A test for the FIFO functionality faults requires the detection of Fault 7 through 12a; which also cover the remaining Faults 12b, 13 and 14. Table 2 shows this test during which the FF and EF signal have to be monitored. The test consists of the following steps:

- Step 1:** initialize the FIFO. If thereafter $FF \neq 0$ or $EF \neq 1$; Fault 11 (improper RS operation) has been detected.
- Step 2:** fill FIFO. If Step 2 cannot be completed (because n words cannot be written); Fault 8 and 11 will be detected. If $EF \neq 0$, Fault 7 has been detected.
- Step 3:** wait Del sec. If thereafter $FF \neq 1$ then Fault 9 (DRF0 in $DVL[i]$) has been detected.
- Step 4:** attempt to write word $n+1$ in the full FIFO. In case of Fault 12a G will be written into $MA[0]$, which contained the data value $\overline{G}$.
- Step 5:** read n words from FIFO. If data value of last read word $\neq \overline{G}$ then Fault 12a is detected.
- Step 6:** wait for time Del sec. If $EF \neq 1$ then Fault 10 (a DRF1 fault in $DVL[i]$) has been detected.

4.4 Composite test

In this section the two subtests of Table 1 and 2 will be combined into one composite test for shifting-type FIFOs of Table 3. This composite test does not necessarily have to contain all elements of the subtests, it just has to satisfy all test requirements. Faults 1 through 12a are detected as follows.

If after Step 1 $FF \neq 0$ or $EF \neq 1$ then Fault 11 has been detected. If less than n words can be written in Step 2, Fault 8 and 11 are detected; while if $EF \neq 0$ after Step 2, Fault 7 has been detected. If after Step 3 $FF \neq 1$, then Fault 9 has been detected. If Step 8b does not result in reading a data value G then Fault 12a has been detected. If during Step 8 (a and b) a wrong data value is read, Fault 1 through 6 are detected. If after Step 9 $EF \neq 1$, then Fault 10 has been detected.

Step #	Operation		Flag Check		#ops	Faults detected
	$O(1)$	$O(n)$	FF	EF		
1	RS		0	1	1	11
2a	$\uparrow_0^{n-2} wG \overline{G}$		0	0	$n - 1$	8, 11
2b	$w\overline{G}$		1	0	1	7, 8, 11
3	Del		1	0	Del	9
4	wG		1	0	1	
5	rG		0	0	1	
6	wG		1	0	1	
7	Del		1	0	Del	
8a	$\uparrow_0^{n-2} r\overline{G} G$		0	0	$n - 1$	1 - 6
8b	rG		0	1	1	1 - 6, 12a
9	Del		0	1	Del	10

Table 3: Functional test for shifting-type FIFO (length: $2n + 4 + 3Del$)

5 Conclusions

A functional model of a shifting-type FIFO has been given. Analogous to traditional SRAM and DRAM memories, fault models have been derived for the memory array, the addressing mechanism and the FIFO functionality logic. These fault models have been based on classical SRAM fault models; however, they are specific for shifting-type FIFOs. The used methodology may be applied to other types of embedded memories.

Traditional march tests cannot be used for testing FIFOs because of the mandatory use of a single address order and march elements with only one write and/or read operation. In addition to testing the addressing mechanism and the memory array, a FIFO test also has to cover faults in the

FIFO functionality logic (this is the embedded logic which implements the FIFO behaviour).

For each of the faults a (set of) requirement(s) has been derived for a test to detect the faults and an order $O(n)$ test has been given. It has been shown that this test covers all faults. Because of the specific nature of shifting-type FIFOs, this test is unlike any other memory test; while its simplicity makes it well suitable for application as a BIST.

References

1. Dekker, R. *et al.* (1990). A Realistic Fault Model and Test Algorithms for Static Random Access Memories. *IEEE Trans. on Computers*, C-9 (6), pp. 567-572.
2. van de Goor, A.J. and Verruijt, C.A. (1990). An Overview of Deterministic Functional RAM Chip Testing. *ACM Computing Surveys*, 22 (1), pp. 5-33.
3. van de Goor, A.J. (1991). *Testing Semiconductor Memories, Theory and Practice*. John Wiley & Sons: Chichester, UK.
4. van de Goor, A.J. and Zorian, Y. (1992). Functional Tests for Arbitration SRAM-type FIFOs. In *Proc. IEEE First Asian Test Symposium (ATS92)*, pp. 96-101, November 26-27, 1992.
5. van de Goor, Ad J. (1993). Using March Tests to Test SRAMs. *IEEE Design & Test of Computers*, 10(1), pp. 8-14.
6. van de Goor, A.J. Schanstra, I., and zorian, Y. (1994). Fault Models and Tests for Ring Address Type FIFOs. In *Proc. 12th IEEE VLSI Test Symposium*, pp. 300-305, April 1994.
7. van de Goor, A.J. and Zorian, Y. (1994). Effective March Algorithms for Testing Single-Order Addressed Memories. *Journal of Electronic Testing, Theory and Applications (JETTA)*, Vol. 5, No. 4, November 1994.
8. Kanadjian, A. *et al.* (1986). FIFO Test Program Development. In *Proc. IEEE Int. Test Conference*, pp. 819-825.
9. van Sas, J. *et al.* (1993) Test Algorithms for Double Buffered Random Access and Pointer Addressed Memories. *IEEE Design & Test of Computers*, 10(2), pp. 34-44.
10. Stevens, A.K. (1986). *Introduction to Component Testing*. Addison-Wesley: Reading, MA.
11. Zorian, Yervant, van de Goor, Ad J., and Schanstra, Ivo. (1994). An Effective BIST Scheme for Ring-Address Type FIFOs. In *Proc. IEEE Int. Test Conference*, pp. 378-387.