01 Jan 1994

On Fault Modeling and Testing of Content-addressable Memories

Waleed K. Al-Assadi
Missouri University of Science and Technology, waleed@mst.edu

A. P. Jayasumana

Y. K. Malaiya

Recommended Citation

W. K. Al-Assadi et al., "On Fault Modeling and Testing of Content-addressable Memories," *Records of the IEEE International Workshop on Memory Technology, Design and Testing, 1994*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1994.
The definitive version is available at https://doi.org/10.1109/MTDT.1994.397193

On Fault Modeling and Testing of Content-Addressable Memories

W. K. Al-Assadi, A. P. Jayasumana, and Y. K. Malaiya†
Electrical Engineering Department
† Computer Science Department
Colorado State University
Fort Collins, CO 80523

Abstract

Associative or content addressable memories can be used for many computing applications. This paper discusses fault modeling for the content addressable memory (CAM) chips. Detailed examination of a single CAM cell is presented. A functional fault model for a CAM architecture executing exact match derived from the single cell model is presented. Efficient testing strategy can be derived using the proposed fault model.

1 Introduction

Content-addressable memories (CAMs) are a class of memories in which data is accessed on the basis of content instead of data-location address. Such memories are distributed-logic fine-grain parallel processing computational structures capable of supporting a wide range of numeric and non-numeric computational tasks [1, 2]. Their computing power lies in their ability of accessing data based on content independent of the number of words that reside in the memory. Furthermore, their regular and iterable structure makes them highly suitable for VLSI implementations that require intensive data management. CAMs have been used in applications such as memory management, database machines, image processing , and artificial intelligence. With the progress in the neural network technology , CAMs are also being used to solve pattern recognition problems [3, 4].

In a CAM device implementing the exact-match operation, the memory is searched in parallel for words that exactly match an input word. In this type of CAM architecture, stored words are compared independently and simultaneously with the input word.

While testing of static random access memories (SRAM) has received lot of attention, testing CAMs has not been addressed in detail. Testing algorithms for SRAMs have often been based on the fault model proposed by Nair, Thatte and Abraham [5]. In this model the SRAM structure is divided into three blocks; the memory array, the address decoder, and the Read/Write logic. Defects in address decoder and the Read/Write logic are mapped onto functionally equivalent faults in the memory array. This model has the advantage that all faults can be considered to be in the memory array. Fault modeling for a single SRAM cell has been done at the the layout level and extraction to the transistor level using the Inductive Fault Analysis (IFA) [6]

Several testing algorithms for CAM have been reported. A simple testing methodology for to test only for stuck-at faults in the Motorola Memory Management Unit MC68851 was given in [7]. The design for testability approach in [8] can detect complex pattern sensitive faults in addition to the stuck-at faults. Testing can be further enhanced with the aid of good functional fault model of the CAM architecture, similar to that of RAM architecture derived in [5]. In this paper, we investigate the detailed fault model for the CAM cell. All possible shorts and transistor stuck-on/open faults are considered. Using this fault model, the functional fault model for a CAM architecture executing exact match is derived such that all faults in the prephiral registers are mapped onto the CAM array.

2 The CAM architecture

The principal characteristic of the fully parallel CAM architecture is the capability to process data simultaneously. Figure 1 shows the architecture of a CAM that executes the exact match [9]. In this architecture the processing logic hardware is included in each memory bit cell. This allows all the bits in the memory to be interrogated simultaneously. This architecture has four main functions. The write function stores data in unused word locations. The read function returns the data associated with the matched word. This operation is similar to a read operation in the conventional RAM. The match function searches through the stored words in parallel for a match against an input word, indicating whether a match has been found. To address the memory by data content, the comparand data word is placed on the input data register. Before any data interrogation takes place, the input word can be masked in any arbitrary bit positions. The masked bits are excluded from any participation in the memory interrogation process. Input word masking allows the search for stored data that meets only partial information of the input data. Once the memory is interrogated, the stored words that meet the input criteria will have its match-line active and the corresponding response flag bit is set to logic 1. Thus in one cycle the memory is accessed and the result is latched in the response flag register.

0-8186-6245-X/94 $04.00 © 1994 IEEE

The fourth function is the garbage collection function which indicates whether the word location is empty or not from among empty word locations. Thus input data may be automatically written in the cell array without any address handling for garbage word locations.

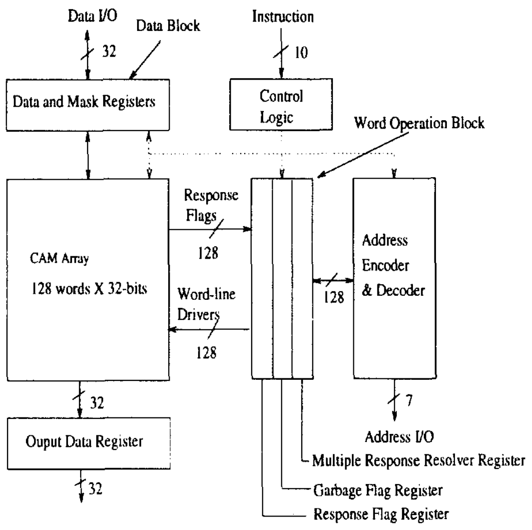

Figure 1: The CAM architecture

3 Detailed examination of the CAM cell

A CAM cell that is widely used is shown in Figure 2. It consists of a storage cell, which is basically a RAM cell and the comparison logic for performing match operation. Transistors T1-T6 constitute the storage cell, while T7 and T8 implement the XOR function. Transistor T9 provides the discharge path for the word match-line in case of a mismatch. The word match-line functions as a wired AND gate with the match lines from each cell in the word as inputs. The functional behavior of the CAM cell is summarized as given in table 1. The word-select line is asserted only when a read/write operation to the RAM is desired. For execution of a match operation, the input data and its complements are placed on the bit and $\overline{bit}$ lines respectively. A comparison between the input bit and the stored bit is performed by the exclusive-OR function. If the input bit does not match the stored bit, the $\overline{BM}$ ($\overline{bit-match}$) line will force the word match-line (initially set high by a pullup) to go low. Otherwise the word match-line will stay high. The word-match line for a given word will be pulled low if any bit in the word does not match the corresponding bit in the input data word. Therefore, the word match-line remains high only when all the bits in the word match perfectly with the bits in the input data word.

Detailed examination of the storage part (RAM cell) indicates that most shorts between single lines cause some type of coupling [10]. It can be either *bit* coupling, $\overline{bit}$ coupling, word coupling, or coupling involving *bit* and word lines or $\overline{bit}$ and word lines [10]. The observations made at the bit and $\overline{bit}$ lines during the read operation are shown in table 2. The $\overline{BM}$ ($\overline{MB}$) line is active (i.e. low) if the match with a specific bit occurs, the word match-line is active if all the participating bits in the word match. This cell is designed such that the two pull-up transistors of the cross-coupled inverters have minimum size to retain charge lost due to leakage current. On the other hand, the pull-down transistors are chosen to be two or three times stronger than the pass transistors in order to avoid charge sharing during read operation. This causes '0' to dominate over '1' when two lines of different logical values are shorted.

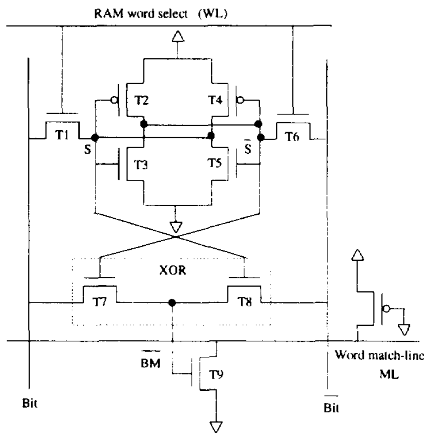

Figure 2: The exact match CAM cell

Table 1: Functional behavior of the CAM cell

Operation	WL	Bit	$\overline{Bit}$	$\overline{BM}$	Comments
Write 1	1	1	0	x	1 stored in the cell
Write 0	1	0	1	x	0 stored in the cell
Match 1	0	1	0	$\overline{S}$	ML discharged if CAM=0
Match 0	0	0	1	S	ML discharged if CAM=1
Read x	1	P	P	x	Stored data on Bit and $\overline{Bit}$
NO operation	0	P	P	x	

W= Word-select line, S=state of the cell

Some faults such as a short between V_{DD} and word line cause logic retention failures. This fault causes the word to be always accessed, therefore after the write operation the cell changes its state and the cell is unable to retain the logic value written into the cell. Such non-retention faults are modeled as NR0/1. NR-1(0) means that the cell changes its value after writing 1(0) and no longer contains 1(0). Stuck-open faults in the pull-up transistors of the cross-coupled inverters do not change the functional behavior of the cell but reduces the capability of retaining stored values in the cell. Most shorts cause abnormal I_{DDQ}.

In examining the CAM cell, we assume that if the storage cell is faulty the comparison part is fault free and vise-versa. This assumption is necessary to avoid considering multiple faults in two different logical parts. The design of the storage cell is assumed to be such that 0 dominates in case of shorts. The observations are made by examining the the state of the $\overline{BM}$ line during the match operation. The M(x) used below refers to matching operation of a logic x with the stored bit in the cell. M(x) is true (matching) if $\overline{BM} = 0$, and false (mismatch) if $\overline{BM} = 1$.

The following definitions are used to characterize the behavior of the faulty CAM cell.

1: The state of the CAM cell is defined as S ($S \in \{0, 1\}$) where S is the logic value of node labeled S (figure 2).

2: The CAM cell is said to be *always matched*, if the match operation results in $\overline{BM} = 0$ irrespective of the state of the CAM.

3: The cell is said to be *always mismatched*, if $\overline{BM} = 1$ irrespective of the match operand and the state of the CAM cell.

4: The cell is said to be *partly matched* if the match operation results in $\overline{BM} = 0$ while the storage cell contains logic x, and in $\overline{BM} = 1$ if the storage cell contains logic $\bar{x}$.

5: The CAM cell is said to be *conditionally matched* if it provides the proper results for the match operation for a logic value, but not for its complementary. CM-1(0) refers to conditionally matched 1(0).

Definition 5 describes a class of faulty behaviors where the storage cell is unable to store the logic value written to the cell. Consider the fault where the storage cell fails to store logic 0 but stores logic 1 properly. The match operation after writing 1 is performed properly, while after writing 0 to the cell, M(1) is true and M(0) is false. Hence M(1) is always matched irrespective of the logic written to the storage cell and M(0) is false when the storage cell fails to store logic 0. An example of such a failure is the short between the V_{dd} and the *bit* line. This fault makes the *bit* line always high and the storage cell always stores 1, i.e. the storage cell is SA1. Then $\overline{BM}$ line is always low when matching 1 and high when matching 0. Table 3 shows all possible shorts in the storage cell and its effects on the CAM match-line.

The effects of faults in the comparison logic are considered assuming fault free storage cell. Possible line shorts and transistor stuck-on/open faults are shown in table 4. Most faults in this part do not affect the writing operations, i.e. writing and storing data in the storage cell is proper. This means that the value at *bit* and $\overline{bit}$ during the read cycle is proper. This implies that cell will show fault-free behavior during the read operation. Some faults cause improper writing of data. An example is the short between V_{ss} and $\overline{BM}$ line. This fault causes the storage cell to always store logic 1, i.e. the cell is SA1. Other shorts involving nodes S, $\overline{S}$ of the storage cell and $\overline{BM}$ line cause improper writing to the storage cell, which is dependent on the logical states of the shorted lines. Hence matching of 1 and 0 can be improper, and such behavior is termed as *complex/indeterminate.* The effects of shorts involving *bit*, $\overline{bit}$ and the $\overline{BM}$ lines can be termed *conditionally matched* operation. With such faults, writing to the storage cell is proper and the storage cell stores the written logic properly. However the comparison is not performed properly. Consider the short between *bit* and $\overline{BM}$ lines. This causes the signal of *bit* to dominate over the signal of the $\overline{BM}$ line. The match operation is properly performed if the storage cell contains 0. However if the storage cell contains 1, M(0) is true and M(1) is false. Shorts involving the storage nodes of the storage cell S, $\overline{S}$ and the $\overline{BM}$ line show that the cell is *partly matched* depending on the state of the storage node. Complex/indeterminate behavior is observed due to the 0 dominance, which as a result could make the state of the match-line very much dependent on previous values stored in the cell.

From tables 3 and 4, functional fault model for the CAM cell can be summarized as given in table 5. In this table, failures in the RAM cell and in the comparison logic can be mapped onto functional description during the match operation by observing the match-line.

Table 4: Behavior of the CAM cell under comparison logic faults

Fault	Write		Match		**Effect on CAM**
	0	1	0	1	
Short:$bit - \overline{BM}$	√	√	?	?	C-M1
Short:$\overline{bit} - \overline{BM}$	√	√	?	?	C-M0
short:$\overline{S} - bit$	?	?	?	?	complex/indeterminate
Short:$S - \overline{bit}$	?	?	?	?	complex/indeterminate
Short:$\overline{S} - \overline{BM}$	√	√	s	s	P-M1
Short:$S - \overline{BM}$	√	√	s	s	P-M0
Short:*bit*-match-line	√	√	?	?	complex/indeterminate
Short:V_{ss}-match line	?	√	?	?	cell is always mismatched
Short:$V_{ss} - \overline{BM}$	?	√	√	?	cell is matched
T7:stuck-open	√	√	s	s	C-M0
T7:stuck-on	√	√	?	?	C-M1
T8:stuck-open	√	√	s	s	C-M1
T8:stuck-on	√	√	?	?	C-M0
T9:stuck-open	√	√	?	?	cell is always matched
T9:stuck-on	√	√	?	?	cell always mismatched

s: CAM state dependent

4 Functional fault model for CAM architecture

The CAM architecture in Figure 1 can be schematically described as an array of CAM cells supported by *bit*-column logic corresponding to the data and mask registers and word row logic corresponding to the word operation block that consists of the response flag registers, encoding/decoding address register, and other registers. Functional fault model for RAM architecture has been successfully applied by mapping defects in the peripheral logic onto functionally equivalent faults in the memory array. The observations are made according to the states of the *bit* and $\overline{bit}$ lines during the read operation. This model has the advantage that all faults can be considered to be in the memory array [5]. For a CAM architecture we can consider a similar approach.

Table 5: Functional fault model for the CAM cell

Functional fault in storage cell	Effect on CAM
cb & c $\bar{b}$	cell always matched
short between V_{ss} and *bit* or $\overline{bit}$	cell always matched
coupling involving word line with *bit* line or with node S	C-M1
coupling involving word line with $\overline{bit}$ line or with node $\overline{S}$	C-M0
word unaccessable	CAM s-a-x
word always accessible	cell always matched
coupling involving *bit* & $\overline{bit}$	cell always mismatched
SA1	C-M1
SA0	C-M0
stuck-on/open in XOR logic	C-M
bit-match transistor stuck-open	cell always matched
bit-match transistor stuck-on or shorted	cell always mismatched
shorts between *bit*, *bit* and $\overline{BM}$	C-M
shorts between nodes S & $\overline{S}$ and $\overline{BM}$	P-M

Examining the CAM cell under all possible faults show that most of the faults are mapped onto improper word match-line state during the match operation. During the match operation comparand data is placed in the data and mask registers. Any faults in any of the latches of the registers or in the data lines feeding the CAM array will result in wrong word matching, i.e. a match is remarked for a wrong word that should not have been targeted. A fault in a CAM cell that is partly matched or conditionally matched can cause such a fault at the array level. A multi match may be observed due to *bit* or $\overline{bit}$ coupling faults that cause more than one word to be remarked as a match. On the other hand, stuck-at faults in the response flag register can be also be shown as equivalent to the faults in the CAM array.

Consider a match-line or a latch in the register to be stuck-at-1. This means that the match-line is pulled-up permanently, i.e. the word corresponding to this match-line is always masked, which is equivalent to a fault in the CAM array. During the write operation, any fault in the flag response register or in any word operation block registers will cause writing to a wrong word. This is equivalent to having a cell mismatch, i.e. the match-line always pulled-down if the the word that should have been targeted is to be compared. Therefore the effect at the CAM architecture level can be derived from the fault model of a single cell. Such effects can be summarized as a mismatch occurs for a word where a match would occur in fault free case, or a match occurs if a mismatch would occur in fault free case, or a match with incorrect words also occurs, in addition to the correct word. Hence this fault model allows to test only the CAM array assuming that the peripheral registers are fault-free. The applicability of this model is such that the response flag register is made observable, i.e. it can readable from the outside.

The advantage of the cell fault model is that it provides the ability to generate tests to detect cell level faults. Hence rather than writing an arbitrary word to a location and checking whether a match is observed for the same pattern, we can generate a specific pattern to test this as well as cover some cell level faults.

5 Summary and Conclusion

A detailed fault model for a CAM cell is presented. The model was derived by investigating the functional failures of the storage cell and the comparison logic. A functional fault model for CAM architecture executing exact match operation has been derived. The advantages of this model are similar to the advantages of the Nair, Thate and Abraham model for the SRAM architectures. Many faults in the peripheral logic can be successfully mapped onto equivalent faults in the CAM array. This model can be extended for different CAM architectures that do not execute exact matches. Some of the coupling faults may not map into this model. They need to be studied separately. Developing a general testing methodology for CAMs using the presented model is under investigation.

Acknowledgments

This work was supported partly by a BMDO funded project monitored by ONR.

References

[1] S. M. S. Jalaleddine and L. G. Johnson, " Associative memory integrated circuit based on neural mutual inhibition", *IEE Proceedings,* vol. 139, Pt.G, no. 4, pp. 445-449, August 1992.

[2] T. Kohonen, *Content Addressable Memories*, New York:Springer-Verlag, 1980.

[3] S. Jones, "Design, selection and implementation of a content-addressable memory for a VLSI CMOS chip architecture", *IEE Proceedings*, vol. 135, Pt. G, no. 3, pp. 165-172, May 1988.

[4] B. Prince, *Semiconductor Memories*, 2nd ed., John Wiley & Sons, New York, 1983.

[5] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random Access Memories," *IEEE Trans. on Computers* vol. C-27, no. 6, pp. 572-576, June 1978.

[6] R. Dekker, F. Beenker, and L. Thijssen, "Fault Modeling and Test Algorithm Development for Static Random Access Memories", *Proceedings of the 1988 IEEE International Test Conference*, pp. 343-352, 1988.

[7] G. Giles and C. Hunter, " A Methodology for Testing Content Addressable Memories", *Proceedings of the 1985 IEEE International Test Conference*, pp. 471-474, 1985.

[8] P. Mazumder, J. H. Patel, and W. K. Fucks, "Design and Algorithms for Parallel Testing of Random Access and Content Addressable Memories", *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 688-694, 1987.

[9] T. Ogura, S-I. Yamada, and T. Nikaido, "A 4-Kbit Associative Memory LSI", *IEEE J. of Solid-State Circuits*, vol. Sc-20, no. 6, pp. 1277-1281, December 1985.

[10] W. K. Al-Assadi, Y. K. Malaiya, and A. P. Jayasumana, "Modeling of Intra-Cell Defects in CMOS SRAM," *Records of the 1993 IEEE International Workshop on Memory Testing*, pp. 78-81 , August 1993.

Table 2: Behavior of the RAM cell under possible shorts and transistors stuck-on/open faults

Fault in RAM	bit^*	$\overline{bit}^*$	**Effect on RAM array**		
Short:V_{ss}-word	H	H	SAx	I	word unaccessable
Short:V_{ss}-bit	L	H	SA0	I	bit line low
Short:V_{ss}-$\overline{bit}$	H	L	SA1	I	$\overline{bit}$ line low
Short:V_{dd}-word	–	–	NR 0/1	I	word always accessed
Short:V_{dd}-bit	H	L	SA1	I	Bit line high
Short:V_{dd}-$\overline{bit}$	L	H	SA0	I	$\overline{bit}$ line high
Short:word-bit	H	L	SA1	I	cbw
Short:word-$\overline{bit}$	L	H	SA0	I	c$\bar{b}$w
Short:bit-$\overline{bit}$	–	–	SAx	I	cb$\bar{b}$
Short:$S - V_{dd}$	H	L	SA1		
Short:$S - V_{ss}$	L	H	SA0		
Short:$S - \overline{S}$	L	L	SAx	I	
Short:S-word	H	H	SAx	I	cw
Short:$S - bit$	–	–	NR1		cb
Short:$\overline{S} - V_{dd}$	L	H	SA0		
Short:$\overline{S} - V_{ss}$	H	L	SA1		
Short:$\overline{S}$-word	H	H	SAx	I	cw
Short:$\overline{S} - \overline{bit}$	–	–	NR0		c$\bar{b}$
T1 Stuck-on	--	--	NR1		cb
T1 stuck-open	H	L	SA1		
T2 stuck-on	S	$\overline{S}$	fault-free		degradation of stored '0' with time
T2 stuck-open	S	$\overline{S}$	fault-free		degradation of stored '1' with time
T3 stuck-on	H	L	SA1	I	
T3 stuck-open	L	H	SA0		
T4 stuck-open	S	$\overline{S}$	fault-free		degradation of stored '1' with time
T4 stuck-on	S	$\overline{S}$	fault-free		degradation of stored '0' with time
T5 stuck-on	L	H	SA0	I	
T5 stuck-open	H	L	SA1		
T6 stuck-on	--	--	NR1		cb
T6 stuck-open	L	H	SA0		

*: values during read operation
I: enhanced I_{DDQ}
cb: coupling involving *bit* line
c$\bar{b}$: coupling involving $\overline{bit}$ line
cw: cell coupled with other cells sharing the same word line
cb$\bar{b}$: coupling involving bit and $\overline{bit}$
cbw: coupling involves bit and word lines
c$\bar{b}$w: coupling involves $\overline{bit}$ and word lines

Table 3: Behavior of the CAM cell under storage cell faults

Fault in storage cell	**Write**		**Match**		**Effect on CAM** *	Match-line *
	0	1	0	1		
Short:V_{ss}-word	x	x	x	x	CAM s-a-x	SAx
Short:$V_{ss} - bit$	$\surd$	?	?	?	cell always matched	pull-up
Short:$V_{ss} - \overline{bit}$	?	$\surd$	?	?	cell always matched	pull-up
Short:V_{dd}-word	$\surd$	$\surd$	?	?	cell always matched	pull-up
Short:$V_{dd} - bit$	?	$\surd$	?	$\surd$	C-M1	
Short:$V_{dd} - \overline{bit}$	$\surd$	?	$\surd$	?	C-M0	
Short:word-bit	?	$\surd$	?	$\surd$	C-M1	
Short:word-$\overline{bit}$	$\surd$	?	$\surd$	?	C-M0	
Short:$bit - \overline{bit}$	?	?	?	?	cell always mismatched	pull-down
Short:$S - V_{dd}$	?	$\surd$	?	$\surd$	C-M1	
Short:$S - V_{ss}$	$\surd$	?	$\surd$	?	C-M0	
Short:$S - \overline{S}$	?	?	?	?	cell always matched	pull-up
Short:S-word	$\surd$	?	$\surd$	?	C-M0	
Short:$S - bit$	$\surd$	$\surd$	?	?	cell always matched	pull-up
Short:$\overline{S} - V_{dd}$	$\surd$	?	$\surd$	?	C-M0	
Short:$\overline{S} - V_{ss}$	?	$\surd$	?	$\surd$	C-M1	
Short:$\overline{S}$-word	?	$\surd$	?	$\surd$	C-M1	
Short:$\overline{S} - \overline{bit}$	$\surd$	$\surd$	?	?	cell always matched	pull-up
T1 stuck-on	$\surd$	$\surd$	$\surd$	?	cell always matched	pull-up
T1 stuck-open	?	$\surd$	?	$\surd$	C-M1	
T2 stuck-on	$\surd$	$\surd$	$\surd$	$\surd$	fault-free	
T2 stuck-open	$\surd$	$\surd$	$\surd$	$\surd$	fault-free	
T3 stuck-on	?	$\surd$	?	$\surd$	C-M1	
T3 stuck-open	$\surd$	$\surd$	$\surd$	?	C-M0	
T4 stuck-on	$\surd$	$\surd$	$\surd$	$\surd$	fault-free	
T4 stuck-open	$\surd$	$\surd$	$\surd$	$\surd$	fault-free	
T5 stuck-on	$\surd$	?	$\surd$	?	C-M0	
T5 stuck-open	$\surd$	$\surd$	?	$\surd$	C-M1	
T6 stuck-on	$\surd$	$\surd$	?	?	cell always matched	pull-up
T6 stuck-open	$\surd$	?	$\surd$	?	C-M1	

*: values during match operation
$\surd$: operation is proper
?: operation is improper