ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2008
Informatyka – Zeszyt 3

Bartosz Sokół[1]

TECHNIKI WYKRYWANIA USZKODZEŃ PAMIĘCI Z WYKORZYSTANIEM STOPNI SWOBODY TESTÓW KROKOWYCH

Streszczenie: Publikacja zawiera opis wybranych metod i technik wykrywania uszkodzeń pamięci z wykorzystaniem stopni swobody transparentnych testów krokowych. Główna uwaga została skupiona na uszkodzeniach uwarunkowanych zawartością typu Pattern Sensitive Faults (*PSF*) jako najtrudniejszych do wykrycia. Zaproponowane wykorzystanie stopni swobody testów krokowych przejawia się możliwością efektywnego przeprowadzenia transparentnego testowania i wykrywania uszkodzeń typu *PSF* przez proste testy krokowe i bazuje na możliwości wielokrotnego uruchomienia testu przy zmianach warunków początkowych (porządku adresowania) dla każdego uruchomienia. Wykorzystane i zaproponowane metody umożliwiają generowanie pełnych sekwencji adresowych oraz pozwalają na optymalny wybór adresów startowych przy wielokrotnym uruchomieniu testów krokowych. W pierwszej części pracy przedstawiona została problematyka testowania pamięci oraz stopnie swobody testów krokowych. Druga część pracy zawiera opis zaproponowanych rozwiązań wraz z wynikami wybranych eksperymentów.

Słowa kluczowe: testowanie pamięci, testy krokowe, uszkodzenia *PSF*, stopnie swobody

1. Wstęp

Pamięci półprzewodnikowe (RAM) zawsze stanowiły jedną z ważniejszych części systemów cyfrowych. Znajdują one szerokie zastosowanie w układach obliczeniowych i sterujących, a także w systemach przetwarzania i przechowywania informacji. Współczesne technologie wytwarzania modułów RAM idą w kierunku zwiększenia stopnia integracji, zmniejszania rozmiarów elementów i zwiększenia gęstości ich rozmieszczenia. W tych warunkach wzrasta prawdopodobieństwo wystąpienia przypadkowych defektów w trakcie procesu produkcji i intensywność pojawiania się uszkodzeń podczas procesu eksploatacji układów RAM, co prowadzi do istotnego obniżenia niezawodności pracy całego systemu. Analiza dotychczasowych prac i badań, poświęconych metodom niedestrukcyjnego testowania modułów RAM ([2]-[4],[7,12]), pozwala zauważyć tendencję do wykorzystywania w konkretnym celu

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

metod opartych na wykorzystywaniu testów krokowych, co uzasadnione jest wysoką skutecznością wykrywania tymi metodami uszkodzeń ze złożonością *O(N)*, gdzie *N* jest to rozmiar pamięci, i małymi narzutami sprzętowymi i programowymi na realizację.

Test krokowy składa się ze skończonej liczby faz. Każda faza testu krokowego składa się ze skończonej liczby poleceń odczytu i zapisu, z których wszystkie oddziałują na określoną komórkę przed przejściem do następnej komórki pamięci, określonej przez przyjęty sposób adresowania. Wysokie wymagania co do niezawodności i czasu pracy urządzeń powodują, że coraz szerzej rozpowszechnione są niedestrukcyjne modyfikacje testów krokowych [7], pozwalające na zachowanie zawartości testowanych komórek pamięci podczas przeprowadzania okresowego testowania w przerwach między normalnym funkcjonowaniem systemu pamięci. Podstawowym wymaganiem, stawianym testerom niedestrukcyjnym, jest obowiązkowy powrót obiektu do stanu wyjściowego po fazie testowania. Jako przykład testu krokowego może posłużyć szeroko wykorzystywany test *March C-*, którego transparentna wersja znajduje się poniżej.

$$\Uparrow (ra,wa^*);\Uparrow (ra^*,wa);\Downarrow (ra,wa^*);\Downarrow (ra^*,wa);\Updownarrow (ra)$$

$$\leftarrow P0 \rightarrow \leftarrow P1 \rightarrow \leftarrow P2 \rightarrow \leftarrow P3 \rightarrow \leftarrow P4 \rightarrow$$

gdzie: *P0...P4* – elementy krokowe / fazy
$\Updownarrow, \Uparrow, \Downarrow$ – kierunek adresacji
(ra,wa)...(ra)* - operacje w komórkach pamięci
$a \in \{0,1\}, a^*$ - wartość przeciwna do a
Przedstawiony algorytm składa się z pięciu elementów krokowych - zbiorów operacji odczytu *(r)* lub zapisu *(w)*, stosowanych do wszystkich komórek pamięci w określonym porządku. Argumenty operacji zapisu określone są przez binarne wartości zapisywane w komórkach pamięci. Argumenty operacji odczytu określają wartości, które powinny być odczytane ze sprawnych komórek pamięci.

Łącząc niedestrukcyjne testowanie z wykorzystaniem stopni swobody właściwych testom krokowym, możemy w łatwy sposób efektywnie testować pamięć i wykrywać złożone uszkodzenia pamięci ([9]-[11],[13]-[15]).

2. Funkcjonalne modele uszkodzeń pamięci RAM

Idealny test dla cyfrowych układów scalonych RAM powinien wykrywać wszystkie możliwe uszkodzenia w strukturze testowanego schematu. Opracowujący testy spotykają się jednak z niemożnością sprawdzenia układu na obecność ogromnej

liczby różnorodnych uszkodzeń w rozsądnym okresie czasu. Dlatego też w praktyce przyjęto budowanie testów na podstawie pewnych analitycznych modeli uszkodzeń [2,5], w ten czy inny sposób odzwierciedlających realne uszkodzenia układów RAM. Zaletą funkcjonalnego testowania jest to, że testy funkcjonalne i odpowiadające im modele uszkodzeń są zaprojektowane z wykorzystaniem ustalonych i zautomatyzowanych reguł [2].

Głównym rodzajem uszkodzeń rozważanych w niniejszej publikacji będą uszkodzenia uwarunkowane zawartością (*Pattern Sensitive Fault - PSF*) [1,2], które można rozpatrywać jako uogólnienie uszkodzeń sprzężeniowych. Niech uszkodzenie uwarunkowane zawartością zawiera *k* komórek agresorów (lub komórek sąsiednich). Wtedy, gdy *k-1* komórek agresorów zawiera określony kod, a zmiana stanu w *k*-tej komórce agresorze wprowadza w komórce ofierze (lub komórce bazowej) pewną określoną wartość, to mówi się o obecności aktywnego uszkodzenia *PSF* (*Active Pattern Sensitive Fault - APSF*). W przypadku pasywnego uszkodzenia *PSF* (*Passive Pattern Sensitive Fault - PPSF*) przy istnieniu określonego kodu w komórkach agresorach komórka ofiara nie może zmienić swego stanu. Przy statycznym uszkodzeniu *PSF* (*Static Pattern Sensitive Fault - SPSF*) określony kod, zawarty w komórkach agresorach, wprowadza w komórce ofierze określoną wartość, która nie może być zmieniona bez uprzedniej zmiany kodu w komórkach agresorach [2]. W praktyce wykorzystuje się uszkodzenia typu *PSF* z ograniczoną liczbą komórek agresorów. Z reguły są to: trzy, pięć lub dziewięć komórek sąsiednich (*PSF3, PSF5, PSF9*) [6].

3. Stopnie swobody testów krokowych

Każdy algorytm testów krokowych może zostać wykorzystany na różne sposoby i nadal będzie skuteczny przy wykrywaniu jego uszkodzeń docelowych. Stopnie swobody (*ang. Degrees of Freedom - DOF*) właściwe dla testów krokowych mogą być wykorzystane do skonstruowania testów i metod, które umożliwią zwiększenie ich skuteczności zarówno przy wykrywaniu ich uszkodzeń docelowych, jak i przy innych uszkodzeniach dotychczas niewykrywanych.

Jak wiemy, testy krokowe używają sekwencji adresów nazywanych wzrastającymi i malejącymi, oznaczanych odpowiednio przez: $\Uparrow$ oraz $\Downarrow$, które nie muszą być obowiązkowo sekwencjami licznikowymi. Stąd można zdefiniować następujące stopnie swobody [8]:

DOF I: Każda przypadkowa sekwencja adresów może być definiowana jako sekwencja wzrastająca, pod warunkiem, że wszystkie możliwe adresy wystąpią dokładnie tylko raz.

DOF II: Sekwencja adresów dla fazy inicjalizacji może być dowolnie wybrana, pod warunkiem, że wszystkie adresy wystąpią przynajmniej raz.

Jak to zostało opisane powyżej, testy krokowe składają się z wielu krokowych faz testowych, w których zbiory operacji odczytu (r) i zapisu (w) przeprowadzane są na wszystkich komórkach pamięci podczas pełnej wzrastającej lub malejącej sekwencji adresów. Pozostałe stopnie swobody właściwe testom krokowym zostały opisane w pracy [8].

W kolejnym rozdziale zaproponowane będą metody wykorzystania stopni swobody w transparentnym testowaniu pamięci do wykrywania złożonych uszkodzeń pamięci, w szczególności typu *PSF*. Zmiana porządku adresów to wykorzystanie *DOF I* i *II*, a równoczesna zmiana zawartości i porządku adresów to wykorzystanie pierwszych czterech stopni swobody.

4. Wykrywanie uszkodzeń pamięci z wykorzystaniem stopni swobody

Poniżej zaprezentowane będą nowe metody i techniki wykorzystywane przy transparentnym testowaniu pamięci do wykrywania złożonych uszkodzeń pamięci.

4.1 Porównanie sekwencji adresowych

Główna idea zaproponowanego rozwiązania polega na zastosowaniu tego samego testu krokowego okresowo z różnymi sekwencjami adresów.

Dla najprostszego przypadku niech będzie to dowolny test krokowy uruchomiony dwukrotnie z różną sekwencją adresów. Pojawia się pytanie, które z dwóch sekwencji *A1* i *A2* muszą zostać użyte, aby osiągnąć najwyższe pokrycie uszkodzeń?

Porównajmy dwie sekwencje adresów *A1* i *A2* w celu wyciągnięcia wniosków co do przykładowych najlepszych zbiorów sekwencji adresów, dzięki którym osiągnięte będzie wysokie pokrycie uszkodzeń. Kandydaci do tego rodzaju zbioru muszą być jak najbardziej różni. Oznacza to, że nie powinno być żadnych wspólnych podzbiorów adresów na tych samych pozycjach w obu sekwencjach oraz wszystkie adresy na tych samych pozycjach w obu przypadkach muszą mieć jak największą różnicę pomiędzy dwoma adresami [11].

W wypadku standardowej sekwencji licznikowej, przykładem mogą być dwie sekwencje adresów, pierwsza ze wzrastającym porządkiem adresów, zaczynając od adresu z samymi zerami 000…00, 000…01, …, 111…11, druga z malejącym porządkiem adresów, zaczynając od adresu z samymi jedynkami 111…11, 111…10, …, 000…00. Dla $m = 2$ mamy S_{ap}*#1* = 00,01,10,11 i S_{ap}*#24* = 11,10,01,00.

Równocześnie S_{ap}*#1* i S_{ap}*#7* = 01,00,10,11 nie są całkowicie różne, ponieważ na trzeciej i czwartej pozycji występują te same adresy, mianowicie 10 i 11 [13].

Pozwólmy sobie na zaproponowanie arytmetycznej odległości *AD(A_k,A_j)* dwóch sekwencji A_k i A_j, jako charakterystyki numerycznej do oszacowania jak różne są dwie sekwencje adresów:

$$AD(A_k,A_j) = \sum_{i=0}^{2^m-1} |A_k(i) - A_j(i)| \tag{1}$$

Dla dwóch określonych sekwencji adresów: $A_k = 2^m-1, 2^m-2, 2^m-3, \ldots, 0$ i $A_j = 0,1,2,\ldots,2^m-1$, ostatnie równanie przyjmuje postać:

$$AD(A_k,A_j) = \sum_{i=0}^{2^m-1} |2^m - 2i - 1| = 2^{2m-1} \tag{2}$$

Na podstawie proponowanej odległości arytmetycznej *AD(A_k,A_j)* ostatnia wartość 2^{2m-1} wygląda na maksymalną możliwą wartość $AD_{max}(A_k,A_j) = 2^{2m-1}$, natomiast minimalna wartość wynosi $AD_{min}(A_k,A_j) = 2$. I rzeczywiście dla $m = 1$ $AD_{max}(A_k,A_j) = 2^{22-1} = 8$ (patrz S_{ap}#1 i S_{ap}#24) oraz $AD_{min}(A_k,A_j) = 2$ dla S_{ap}#1 i S_{ap}#7.

Dla realistycznego algorytmu opisanego w [13] przez:

$$A_{md} = a_1^{\lambda_1} a_2^{\lambda_2} a_3^{\lambda_3} \ldots a_m^{\lambda_m} \tag{3}$$

zapiszmy dwa następujące twierdzenia, których dowody można znaleźć w [11]. Aby uprościć poniższe rozumowanie, pozwólmy na zdefiniowanie standardowej sekwencji licznikowej (0,1,2,...,2^m-1) jako $A_{st} = a_1 a_2 a_3 \ldots a_m$.

Twierdzenie 1 *Odległość arytmetyczna AD(A_{st},A) gdzie* $A = a_1 a_2 a_3 \ldots a_{j-1} a_j^* a_{j+1} \ldots a_{m-1} a_m$, *z* $\lambda_k = 0$ *dla* $k \neq j$ *i* $\lambda_j = 1$ *jest obliczana jako:*

$$AD(A_{st},A) = 2^{2m-j} \tag{4}$$

Twierdzenie 2 *Odległość arytmetyczna AD(A_{st},A) gdzie* $A = a_1 a_2 a_3 \ldots a_{j-1} a_j^* a_{j+1}^{\lambda_{j+1}} \ldots a_{m-1}^{\lambda_{m-1}} a_m^{\lambda_m}$, *z* $\lambda_k = 0$ *dla* $k < j$, $\lambda_j = 1$ *i* $\lambda_k \in \{0,1\}$ *dla* $k > j$ *jest obliczana jako:*

$$AD(A_{st},A) = 2^{2m-j} \tag{5}$$

Jako przykład, obliczmy odległość arytmetyczną dla przypadku, gdy m = 4.
Niech A = $a_1^* a_2 a_3^* a_4^*$, wtedy AD(A_{st},A) = $|0-11| + |1-10| + |2-9| + |3-8| + |4-15| + |5-14| + |6-13| + |7-12| + |8-3| + |9-2| + |10-1| + |11-0| + |12-7| +$

$|13-6|+|14-5|+|15-4| = 11+9+7+5+11+9+7+5+5+7+9+11+$
$+5+7+9+11 = 128 = 2^{2*4-1}$.

Pod względem zmodyfikowanych sekwencji adresów A_{st} może być rozważana jako zmodyfikowana sekwencja A_{md0} z $\lambda_1\lambda_2\lambda_3\ldots\lambda_m = 000\ldots0$ wygenerowana zgodnie z równaniem (3). Dla ogólnego przypadku A_{md0} może być dowolną sekwencją licznikową ze wszystkimi binarnymi wartościami $a_1a_2a_3\ldots a_m$, gdzie $a_i \in \{0,1\}$. Ostatnie dwa twierdzenia, wspólnie z powyższym przykładem pozwalają sformułować ogólne metody szacowania arytmetycznych odległości pomiędzy dwiema zmodyfikowanymi sekwencjami adresów, wygenerowanymi zgodnie z algorytmem (3).

Odległość arytmetyczna pomiędzy dwiema sekwencjami adresów $A_{md0} = a_1a_2a_3\ldots a_m$ i $A_{md1} = a_1a_2a_3\ldots a_{j-1}a_j^*a_{j+1}^{\lambda_{j+1}}\ldots a_{m-1}^{\lambda_{m-1}}a_m^{\lambda_m}$, gdzie A_{md0} jest dowolną sekwencją licznikową, jest definiowana jako:

$$AD(A_{md0},A_{md1}) = 2^{2m-j} \tag{6}$$

gdzie j jest to indeks najbardziej znaczącego bitu w ciągu a_j odwróconego w A_{md1} w porównaniu z sekwencją A_{md0}. Przykład umieszczony w poniższej tabeli potwierdza ostatnie stwierdzenie.

Tabela 1. Odległości pomiędzy różnymi sekwencjami adresowymi

i	$A_{md1}(i) = a_1a_2a_3$	$A_{md2}(i) = a_1^*a_2a_3^*$	$\lvert A_{md1}(i) - A_{md2}(i)\rvert$
0	101 (5)	000(0)	$\lvert 5-0\rvert = 5$
1	010 (2)	111(7)	$\lvert 2-7\rvert = 5$
2	000 (0)	101(5)	$\lvert 0-5\rvert = 5$
3	100 (4)	001(1)	$\lvert 4-1\rvert = 3$
4	110 (6)	011(3)	$\lvert 6-3\rvert = 3$
5	001 (1)	100(4)	$\lvert 1-4\rvert = 3$
6	011 (3)	110(6)	$\lvert 3-6\rvert = 3$
7	111 (7)	010(2)	$\lvert 7-2\rvert = 5$

$\mathrm{AD}(A_{md1},A_{md2}) = 32 = 2^{2*3-1}$

Jak można zauważyć, odległość arytmetyczna *AD(A_{md1},A_{md2})* zależy tylko od liczby j najbardziej znaczącego odwróconego bitu w A_{md2} w porównaniu z A_{md1}.

Spróbujmy wykryć uszkodzenie typu Passive Pattern Sensitive Fault z pięcioma komórkami sąsiednimi *(PPSF5)*, używając różnych sekwencji adresowych wygenerowanych zgodnie z równaniem (3). Sprawdzone zostały wszystkie możliwe ułożenia

zarówno komórki bazowej jak też komórek sąsiednich. Do testowania wykorzystany został prosty test krokowy *MATS++*, który został uruchomiony dwukrotnie, za każdym razem z różnym porządkiem adresów. W pierwszej kolejności użyto sekwencji licznikowej, jako drugiej sekwencji użyto sekwencji z odwróconymi bitami na różnych pozycjach (gwiazdka oznacza odwrócony bit). Otrzymane wyniki są zaprezentowane w poniższej tablicy.

Tabela 2. Procent wykrycia uszkodzeń *PPSF5* z różnymi sekwencjami adresów i negacją bitów

i	Sekwencja adresowa	$\lvert A_{md1}(i) - A_{md2}(i)\rvert$	*PPSF5* (%)
1	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md1}(i) = a_1^*a_2a_3$	$32 = 2^{2*3-1}$	12,5
2	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md2}(i) = a_1a_2^*a_3$	$16 = 2^{2*3-2}$	11,61
3	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md3}(i) = a_1a_2a_3^*$	$8 = 2^{2*3-3}$	9,82
4	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md4}(i) = a_1a_2^*a_3^*$	$16 = 2^{2*3-2}$	12,32
5	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md5}(i) = a_1^*a_2^*a_3$	$32 = 2^{2*3-1}$	12,5
6	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md6}(i) = a_1^*a_2a_3^*$	$32 = 2^{2*3-1}$	12,5
7	$A_{md0}(i) = a_1a_2a_3$;		
	$A_{md7}(i) = a_1^*a_2^*a_3^*$	$32 = 2^{2*3-1}$	12,5

Najlepsze wyniki otrzymano w przypadku, gdy odwrócony został najbardziej znaczący bit, bez względu na inne odwrócone bity, i gdy odległość pomiędzy sekwencjami była największa.

Przedstawione powyżej wyniki ugruntowały twierdzenia 1 i 2, które mówiły, że odległość arytmetyczna pomiędzy dwiema kolejnymi sekwencjami adresów musi być duża, by można było wykryć uszkodzenia pamięci z dużym prawdopodobieństwem, i zależy od pozycji najbardziej znaczącego odwróconego bitu w drugiej sekwencji dla drugiego uruchomienia testu (patrz równanie (6)).

4.2 Optymalne adresowanie pamięci, dobór adresów początkowych

W rozdziale tym rozważane będą uszkodzenia *PPSF*. Każda komórka pamięci, spośród N komórek może być zaangażowana w uszkodzenie *PPSFk*. Notacja ta oznacza, że k przypadkowych komórek pamięci jest zaangażowanych w uszkodzenie *PPSF* a jedną z tych komórek jest komórka bazowa (*base cell* - *b*),

pozostałe *k-1* komórek to komórki wiążące (*neighbors - n*). Istnieje k klas różnych uszkodzeń *PPSFk*. Klasyfikacja ta zależy od uporządkowania w przestrzeni adresów oraz od pozycji wszystkich tych komórek w tej przestrzeni. Niech adresy pamięci $i_0, i_1, i_2, \ldots, i_{k-1}$, dla konkretnego uszkodzenia *PPSFk* będą posortowane w porządku wzrastającym w sposób następujący $i_0 < i_1 < i_2 < \ldots < i_{k-1}$. Wtedy każde uszkodzenie *PPSFk* będzie reprezentowane przez elementy $a_{i0}, a_{i1}, a_{i2}, \ldots, a_{i(k-1)}$ uporządkowane w przestrzeni adresów zgodnie ze wzrastającym porządkiem adresów komórek pamięci. Jedna spośród k komórek jest komórką bazową, więc istnieje k różnych uszkodzeń *PPSFk* zależnych od pozycji komórki bazowej. Oznacza to, że istnieje k klas uszkodzeń *PPSFk* zależnych od pozycji komórki bazowej. Na przykład dla $k = 5$ mamy pięć następujących klas: $b_{i0}, n_{i1}, n_{i2}, n_{i3}, n_{i4}; n_{i0}, b_{i1}, n_{i2}, n_{i3}, n_{i4}; n_{i0}, n_{i1}, b_{i2}, n_{i3}, n_{i4}; n_{i0}, n_{i1}, n_{i2}, b_{i3}, n_{i4}; n_{i0}, n_{i1}, n_{i2}, n_{i3}, b_{i4}$. Istnieje 2^{k-1} różnych kombinacji bitów dla zbioru bitów w komórkach wiążących. Dokładna liczba uszkodzeń *PPSFk* określona jest zgodnie z równaniem:

$$L(PPSFk) = k2^{k-1} * \binom{N}{k} \tag{7}$$

Ten sam rezultat otrzymujemy na podstawie następującej obserwacji. Komórka bazowa może przyjąć każdą spośród N możliwych pozycji wśród komórek pamięci, a dla dowolnej kombinacji bitów w komórkach wiążących istnieje 2^{k-1} różnych zbiorów bitów. Mamy więc:

$$L(PPSFk) = N * 2^{k-1} * \binom{N-1}{k-1} = k2^{k-1} * \binom{N}{k} \tag{8}$$

Ostatnie równanie pozwala na sformułowanie wniosku, że wśród wszystkich k klas danego uszkodzenia istnieje taka sama liczba uszkodzeń *PPSFk*. Przypuśćmy, że używamy testu *MATS+* $\{\Uparrow (ra, wa^*); \Downarrow (ra^*, wa);\}$ do testowania ośmio bitowej pamięci, w której zawartość początkowa jest równa samym zerom $A = a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$
$= 00000000$. Wtedy kolejne stany testowanej pamięci przy wykorzystaniu testu *MATS+* będą wyglądały tak jak przedstawione w tablicy 3.

Należy zauważyć, że sekwencja adresów pamięci została wybrana na podstawie sekwencji licznikowej, a adresem startowym był adres $i_0 = 0$. Jak widać w powyższej tabeli, tylko jeden zbiór bitów, spośród wszystkich możliwych dla każdej aktywnej komórki, jest zaznaczony pochyloną czcionką. W rzeczywistości sprawdzamy (odczyt 0 zapis 1 podczas pierwszej fazy i odczyt 1 zapis 0 podczas drugiej fazy) komórkę a_0 w obu fazach dla tej samej zawartości w pozostałych komórkach. Aktywacja uszkodzenia *PPSFk* może wystąpić tylko w momencie sprawdzania

Tabela 3. Zawartość pamięci podczas testowania testem *MATS+* z różnym adresem startowym

Fazy testu *MATS+*	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Fazy testu *MATS+*	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	0	0	0	0	0	0	0	0									
	1	0	0	0	0	0	0	0		1	1	1	1	1	1	1	*0*
	1	*1*	0	0	0	0	0	0		1	1	1	1	1	1	*0*	0
	1	1	*1*	0	0	0	0	0		1	1	1	1	1	*0*	0	0
	1	1	1	*1*	0	0	0	0		1	1	1	1	*0*	0	0	0
$\Uparrow (ra, wa^*)$	1	1	1	1	*1*	0	0	0	$\Downarrow (ra^*, wa)$	1	1	1	*0*	0	0	0	0
	1	1	1	1	1	*1*	0	0		1	1	*0*	0	0	0	0	0
	1	1	1	1	1	1	*1*	0		1	*0*	0	0	0	0	0	0
	1	1	1	1	1	1	1	*1*		*0*	0	0	0	0	0	0	0
Fazy testu *MATS+*	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Fazy testu *MATS+*	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	0	0	0	0	0	0	0	0									
	0	*1*	0	0	0	0	0	0		*0*	1	1	1	1	1	1	1
	0	1	*1*	0	0	0	0	0		0	1	1	1	1	1	1	*0*
	0	1	1	*1*	0	0	0	0		0	1	1	1	1	1	*0*	0
	0	1	1	1	*1*	0	0	0		0	1	1	1	1	*0*	0	0
$\Uparrow (ra, wa^*)$	0	1	1	1	1	*1*	0	0	$\Downarrow (ra^*, wa)$	0	1	1	1	*0*	0	0	0
	0	1	1	1	1	1	*1*	0		0	1	1	*0*	0	0	0	0
	0	1	1	1	1	1	1	*1*		0	1	*0*	0	0	0	0	0
	1	1	1	1	1	1	1	1		0	*0*	0	0	0	0	0	0

komórki bazowej. Podsumowując widzimy, że wykrycie uszkodzenia *PPSFk* jest możliwe tylko dla jednej kombinacji bitów w pozostałych *k-1* komórkach wiążących spośród 2^{k-1} możliwych kombinacji. Dlatego też liczba *Q1(PPSFk(i_0))* uszkodzeń możliwych do wykrycia podczas pierwszego uruchomienia testu *MATS+* wynosi:

$$Q1(PPSFk(i_0)) = k * \binom{N}{k} \tag{9}$$

A pokrycie uszkodzeń (*ang. fault coverage FC*) dla testu *MATS+* wynosi:

$$FC1\_MATS+(PPSFk(i_0)) = \frac{Q1(PPSFk(i_0))}{L(PPSFk)} 100\% = \frac{1}{2^{k-1}} 100\% \tag{10}$$

Na przykład: *FC_MATS+(PPSF5)* = $(1/2^4)100\%$ = 6,25%. Pokrycie uszkodzeń opisane wzorem (10) jest poprawne dla każdego testu pamięci, z kolejnymi fazami ustawionymi tak jak w przypadku testu *MATS+*, mianowicie $\ldots(ra, \ldots, wa^*); (ra^*, \ldots, wa); \ldots$. W celu podniesienia możliwości wykrycia uszkodzeń *PSF* musimy stosować test z czteroma kolejnymi fazami $\ldots(ra^*, \ldots, wa); (ra, \ldots); \ldots (ra^*, \ldots, wa); (ra, \ldots)\ldots$ (na przykład test *March C*). W praktyce wystarczą mniej niż cztery fazy.

Dla testu *March C*, liczba *Q2(PPSFk(i_0))* wykrywanych uszkodzeń podczas pierwszego uruchomienia testu wynosi:

$$Q2(PPSFk(i_0)) = 2k * \binom{N}{k} \tag{11}$$

A pokrycie uszkodzeń *(FC)* dla testu *March C* wynosi:

$$FC2\_MarchC(PPSFk(i_0)) = \frac{Q2(PPSFk(i_0))}{L(PPSFk)}100\% = \frac{1}{2^{k-2}}100\% \tag{12}$$

W przypadku uszkodzenia *PPSF5* i testu *March C* mamy: FC_March C(*PPSF5*) = $(1/2^3)100\%$ = 12,5%. Oczywiście, dla takiej samej zawartości pamięci każdy test pamięci będzie wykrywał takie same uszkodzenia pamięci, a także taką samą ich liczbę. Bazując na dwóch powyższych przykładach, możemy rozważyć dwa różne przypadki. Przypadek pierwszy, gdy komórka bazowa ma stałą pozycję, a komórki wiążące są na przypadkowych pozycjach w ramach *N-1* komórek pamięci RAM. Drugi przypadek jest bardziej realistyczny, gdy zarówno komórka bazowa jak też komórki wiążące znajdują się na losowych pozycjach w ramach *N*-bitowej pamięci. Dla powyższych przypadków możemy sformułować kolejny problem.

Jak zwiększyć pokrycie uszkodzeń, bazując na krokowych testach pamięci i stałej zawartości?

Odpowiadając na postawione powyżej pytanie, wiemy, że dla stałych dwóch elementów: wybranego testu krokowego oraz zawartości pamięci podczas procedury testowania, jedyną możliwością zwiększenia pokrycia uszkodzeń jest wykorzystanie procedury generowania adresów. Zacznijmy od optymalizacji wartości adresu startowego. Rozważmy przypadek procedury testowej z wielokrotnym uruchomieniem testu krokowego. Bazując na stopniach swobody i stwierdzeniu, że liczba uszkodzeń wykrywanych przez test krokowy nie zależy od porządku adresów, przy kolejnym uruchomieniu testu możemy użyć nowego adresu startowego. Jako przykład rozważmy drugie uruchomienie testu *MATS+* z adresem początkowym równym i_1 = 1. Kolejne stany we wszystkich komórkach pamięci są zaprezentowane w drugiej części tablicy 3.

Ponownie jak poprzednio otrzymaliśmy tylko jedną kombinację bitów w ramach całej pamięci dla każdej aktywnej komórki bazowej *b*. Porównując powyższe wyniki z przypadkiem opisanym w pierwszej części tablicy 3, widzimy, że zostały wygenerowane nowe kombinacje bitów. Dla każdej komórki bazowej *b*, z wyjątkiem przypadku, gdy $b = a_0$, istnieją różne wartości w komórce a_0. Dla przypadku, gdy $b = a_0$, mamy *N-1* różnych wartości w zbiorze bitów w komórkach wiążących. Widzimy, że liczba wykrywanych uszkodzeń *PPSFk* jest taka sama w

obu przypadkach, istnieje jednak grupa uszkodzeń wykrywanych podczas pierwszego uruchomienia z adresem i_0 oraz nowe uszkodzenia *PPSFk* wykrywane tylko podczas drugiego uruchomienia. Na przykład, gdy komórka bazowa $b = a_1$, a adres początkowy $i_0 = 0$ (patrz górna część tablicy 3), mamy dwie klasy uszkodzeń *PPSF5*. Jest to 15 uszkodzeń $b_{i0}, n_{i1}, n_{i2}, n_{i3}, n_{i4} = b0000$ oraz 20 uszkodzeń $n_{i0}, b_{i1}, n_{i2}, n_{i3}, n_{i4} = 1b000$. Dla nowego adresu początkowego $i_1 = 1$ (patrz dolna część tablicy 3) mamy 15 uszkodzeń $b_{i0}, n_{i1}, n_{i2}, n_{i3}, n_{i4} = b0000$ oraz 20 nowych uszkodzeń $n_{i0}, b_{i1}, n_{i2}, n_{i3}, n_{i4} = 0b000$ niewykrytych dotychczas podczas pierwszego uruchomienia testu *MATS+*. Jeżeli komórka bazowa będzie miała inną pozycję, na przykład $b = a_3$, to dla adresu początkowego $i_0 = 0$ mamy cztery klasy uszkodzeń *PPSF5*, mianowicie jedno uszkodzenie $b_{i0}, n_{i1}, n_{i2}, n_{i3}, n_{i4} = b0000$, 12 uszkodzeń $n_{i0}, b_{i1}, n_{i2}, n_{i3}, n_{i4} = 1b000$, 18 uszkodzeń $n_{i0}, n_{i1}, b_{i2}, n_{i3}, n_{i4} = 11b00$ oraz 4 uszkodzenia $n_{i0}, n_{i1}, n_{i2}, b_{i3}, n_{i4} = 111b0$. Z nowym adresem początkowym $i_1 = 1$ podział uszkodzeń będzie inny.

Całkowicie inna sytuacja występuję w przypadku, gdy komórka bazowa jest równa $b = a_0$, a adres początkowy wynosi $i_1 = 1$. W tym przypadku wszystkie uszkodzenia wykrywane podczas drugiego uruchomienia testu będą inne w porównaniu z uszkodzeniami wykrytymi podczas pierwszego uruchomienia testu *MATS+* dla przypadku $i_0 = 0$. Jak widać, podczas pierwszego uruchomienia wykrywane są uszkodzenia *b0000* a podczas drugiego uruchomienia wykrywane są tylko uszkodzenia *b1111*. Jeśli wziąć pod uwagę wszystkie możliwe pozycje komórki bazowej, całkowita liczba nowych uszkodzeń *PPSF5* z nową adresacją i_1 wynosi 175. Jak łatwo pokazać, dla ogólnego przypadku, liczba dodatkowych uszkodzeń *PPSFk* wykrywanych podczas drugiego uruchomienia testu *MATS+* z adresacją $i_1 = 1$ może być oszacowana jako:

$$(N-1)\binom{N-2}{k-2} + \binom{N-1}{k-1} \tag{13}$$

Dla $i_1 = 2$ łatwo wykazać, że:

$$(N-2) * \left[\binom{2}{1} * \binom{N-3}{k-2} + \binom{2}{2} * \binom{N-3}{k-3}\right] + 2 * \left[\binom{N-2}{k-1} + \binom{N-2}{k-2}\right] \tag{14}$$

Należy zaznaczyć, że nowa adresacja pamięci i_1 dzieli wszystkie możliwe (N) komórki bazowe na dwie grupy. Podział jest wykonywany zgodnie z odległością $s = i_1 - i_0$ pomiędzy dwoma adresami startowymi (należy pamiętać, że $i_1 > i_0$). Zgodnie z tymi obliczeniami, całkowity zbiór wszystkich możliwych stałych pozycji komórki bazowej będzie podzielony na dwie grupy. Pierwsza grupa będzie miała s różnych bitów pośród *N-1* komórek pamięci (z wyjątkiem pozycji komórki bazowej) wygenerowanych podczas drugiego uruchomienia testu *MATS+* w porównaniu

z kombinacjami bitów otrzymanymi podczas pierwszego uruchomienia z adresem początkowym $i_0 = 0$. Liczba takich komórek bazowych jest określana przez *N-s*. Reszta komórek bazowych należy do drugiego zbioru z *N-s* różnymi bitami.

Jako przykład weźmy $i_1 = 3$. Dla tego przypadku mamy $s = i_1 - i_0$ = 3-0 = 3. Więc istnieje *N-s* = 8-3 = 5 komórek bazowych z s = 3 różnymi bitami w ramach wszystkich *N-1* = 8-1 = 7 komórek pamięci, jak to zostało pokazane w tablicy 3.

Nowe zbiory bitów będą generowane w komórkach wiążących dla każdej możliwej komórki bazowej, dzięki czemu powinny być wykrywane nowe uszkodzenia *PPSFk*. Liczba nowych uszkodzeń *PPSFk* może być określona przy pomocy odległości Hamminga pomiędzy zawartościami pamięci dla dwóch różnych adresów początkowych. W naszym przypadku odległość $s = i_1 - i_0$ pomiędzy dwoma adresami startowymi oraz wartość $N - s = (i_1 - i_0)$ równa się odległości Hamminga dla zbiorów bitów w komórkach wiążących dla tej samej komórki bazowej przy dwóch uruchomieniach testu *MATS+*. Na przykład, jeżeli odległość $s = i_1 - i_0$ = 2, oznacza to, że dla i_0 = 0, i_1 = 2, oraz dla każdego i_0 i i_1, gdzie $i_0 - i_1$ = 2, mamy 6 (*N-s* = 8-2 = 6) par zbiorów bitów: *((11b00000, 00b00000), (111b0000, 001b0000), (1111b000, 0011b000), (11111b00, 00111b00), (111111b0, 001111b0), (1111111b, 0011111b))*, dla których odległość Hamminga wynosi s = 2, oraz mamy s = 2 pary zbiorów bitów *((b0000000, b0111111), (1b000000, 1b011111))* z odległością *N*-2 = 6.

Dokładna liczba nowych uszkodzeń *PPSFk* wykrytych podczas drugiego uruchomienia testu *MATS+* dla $s > 1$ będzie wyliczona zgodnie z równaniem:

Q1(PPSFk(i_1))=

$$(N-2) * \sum_{i=1}^{min(s,(k-1))} \binom{s}{i} * \binom{N-s-1}{k-i-1} + s * \sum_{i}^{min(s,(k-1))} \binom{s-1}{i-1} * \binom{N-s}{k-i} \quad (15)$$

Analiza powyższego równania pozwala sformułować następujące wnioski. Aby osiągnąć wysokie pokrycie uszkodzeń podczas drugiego uruchomienia testu *MATS+*, musimy użyć porządku adresów i_1:

1. Pierwszy przypadek (stała pozycja komórki bazowej) - musimy otrzymać maksymalną odległość Hamminga pomiędzy zbiorami bitów w komórkach wiążących dla stałej pozycji komórki bazowej. Aby osiągnąć ten cel, nowy adres początkowy, przy drugim uruchomienia testu pamięci, powinien spełniać następujące równanie $i_1 - i_0 = b + 1$, gdzie b jest to numer komórki bazowej.
2. W drugim przypadku (komórka bazowa może przyjąć dowolną pozycję losową) musimy otrzymać, o ile to możliwe, maksymalną sumę *S_HD* odległości Hamminga

dla wszystkich pozycji komórki bazowej. Liczba ta może być policzona jako

$$S\_HD = s(N-s) + (N-s)s \tag{16}$$

Maksimum dla funkcji S_HD może być wyliczone z równania:

$$\frac{\delta(S\_HD)}{\delta s} = \frac{\delta(s(N-s)+(N-s)s)}{\delta s} = (N - \frac{s}{2}) = 0 \tag{17}$$

gdzie wynikiem jest $s = N/2$.

Wracając do naszego przykładu, dla wersji pierwszej mamy $i_1 - i_0 = s = b$. Oczywiście jest to najlepsze rozwiązanie, pozwalające zwiększy pokrycie uszkodzeń. Na przykład, jeżeli ustaloną pozycją komórki bazowej jest 3, to dla i_0 = 0 mamy *111b0000*. Wtedy z równania $i_1 - i_0 = b + 1$=3+1=4 możemy wziąć i_1=4. Łatwo można wykazać, że zaproponowane techniki mogą być rozszerzone na wielokrotne użycie testu pamięci z wykorzystaniem różnych adresów początkowych: $i_0, i_1, i_2, i_3, \ldots$.

5. Podsumowanie

Jak widać z przedstawionych powyżej informacji, wykorzystanie stopni swobody w transparentnym testowaniu pamięci pozwala zwiększyć efektywność testów krokowych przy wykrywaniu złożonych uszkodzeń pamięci. Odpowiednio dobierając sekwencję adresową i adres początkowy jesteśmy w stanie wykrywać z dużym prawdopodobieństwem uszkodzenia uwarunkowane zawartością (*PSF*) przy wykorzystaniu prostych testów krokowych. Dobór sekwencji adresowych i wybór adresu startowego był przedmiotem badań przedstawionych między innymi w pracach ([13] - [15]).

6. Podstawowe pojęcia

PSF - Uszkodzenie uwarunkowane zawartością (*ang. pattern sensitive fault - PSF*) - wartość (lub możliwość zmiany wartości) komórki *i* (komórki bazowej) zależy od wartości (lub ich zmian) wszystkich pozostałych komórek pamięci (komórek wiążących) składających się na to uszkodzenie.
błąd - (*ang. error*) niepoprawna wartość logiczna spowodowana występującym uszkodzeniem.
uszkodzenie - (*ang. fault*) fizyczny defekt powodujący, że wartości logiczne w układzie przyjmują niepoprawne wartości.

defekt - fizyczna zmiana struktury układu prowadząca do jego nieprawidłowego działania.
test krokowy - (*ang. march test*) test pamięci składający się ze skończonej liczby faz. Każda faza testu krokowego składa się ze skończonej liczby poleceń, z których wszystkie oddziałują na określoną komórkę przed przejściem do następnej komórki pamięci. Komórka następna określona jest poprzez sposób adresowania.
test transparentny - (*ang. transparent test*) test gwarantujący (przy poprawnie działającej pamięci), że zawartość pamięci po zakończeniu procesu testowania będzie identyczna z zawartością pamięci w momencie bezpośrednio poprzedzającym rozpoczęcie testu.

Literatura

[1] Cheng K.L., Wu C.W.: Neighbourhood Pattern Sensitive Fault Testing for Semiconductor Memories, Proc. VLSI Design/CAD, Pingtung, Aug. 2000, pp.401-404.
[2] Goor A.J. van de: Testing Semiconductor Memories: Theory and Practice, Chichester: John Wiley and Sons Ltd., 1991.
[3] Goor A.J. van de, Smit B.: Generating March Tests Automatically, IEEE International Test Conference, IEEE Computer Society, Washington, DC, USA, 1994, pp. 870-878.
[4] Goor A.J. van de, Gaydadjiev G.N., Yarmolik V.N., Mikitjuk V.G.: Memory Tests and their Fault Coverage into a New Perspective, Resulting into a New Test, SEMICON, Seul, Korea, Jan. 1996.
[5] Goor A.J. van de, Al-Ars Z.: Functional Memory Faults: A Formal Notation and a Taxonomy, 18th IEEE VLSI Test Symposium (VTS'00), IEEE Computer Society, Montreal, Canada, 2000, pp. 281-289.
[6] Mrozek I., Yarmolik V.N.: Detection of Pattern Sensitive Faults by Multiple Transparent March Tests, Mixed Design of Integrated Circuits and Systems, proc. 10th International Conference, MIXDES 03, Lodz 26-28 June 2003, pp. 542-545.
[7] Nicolaidis M.: Transparent BIST for RAMs, IEEE International Test Conference, IEEE Computer Society, Baltimore, MD, USA, 1992, pp. 596-607.
[8] Niggemeyer D., Otterstedt J., Redeker M.: Detection of Non classical Memory Faults using Degrees of Freedom in March Testing, Rec. 11th Workshop "Test-methods and Reliability of Circuits and Systems", Potsdam, Feb. 1999.
[9] Sokol B., Mrozek I., Yarmolik V.N.: Transparent March Tests to Effective Pattern Sensitive Faults Detection, Proceedings of the IEEE East-West Design and

Test International Workshop (EWDTW 2004), Crimea, September 23-26, 2004, pp.: 166-171.

[10] Sokol B., Yarmolik V.N.: Memory Faults Detection Techniques with Use of Degrees of Freedom in March Tests, Proceedings of the IEEE East-West Design and Test International Workshop (EWDTW 2005), Odessa, Ukraine, September 15-19, 2005, pp.: 96-101.

[11] Sokol B., Yarmolik S.V.: Address Sequences for March Test to Detect Pattern Sensitive Faults, Proceedings of Third IEEE International Workshop on Electronic Design Test and Applications (DELTA'06), Kuala Lumpur, Malaysia, January 17-19, 2006, pp.:354-357.

[12] Yarmolik V.N., Hellebrand S., Wunderlich H.J.: Symetric transparent BIST for RAMs, Design and Test in Europe DATE'99, Munich 1999.

[13] Yarmolik V.N., Sokol B., Yarmolik S.V.: Counter Sequences for Memory Test Address Generation, Mixed Design of Integrated Circuits and Systems, proc. 12th International Conference MIXDES 2005, Krakow, Poland 22-25 June, 2005, pp.413-418.

[14] Yarmolik S.V., Sokol B.: Optimal Memory Address Seeds for Pattern Sensitive Faults Detection, Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS'2006), Prague, Czech Republic, April 18-21, 2006 - pp.220-221.

[15] Yarmolik S.V., Mrozek I., Sokol B.: Address Sequences Generation for Multiple Run Memory Testing, Proceedings of the 6th Internation Conference Computer Information Systems and Industrial Management Applications (CISIM 07), Elk, Poland, June 28-30, 2007 - pp.341-344.

MEMORY FAULTS DETECTION TECHNIQUES WITH USE OF DEGREES OF FREEDOM IN MARCH TESTS

Abstract: Publication shows the description of selected methods and techniques for memory faults detection with use of degrees of freedom inherent to transparent March tests. This paper deals with Pattern Sensitive Faults (*PSF*) as the most difficult to detect. Proposed techniques of use of degrees of freedom in March testing manifest itself in effective transparent memory testing and *PSF* faults detection with use of simple March tests, based on the possibility of multiple run of test with different initial conditions like address order for each run. Proposed methods allow us to choose in an optimal way starting addresses for multiple March tests run. In the first part of this publication, memory testing problems, testing

principles and degrees of freedom were presented. Second part of this publication shows the description of proposed solutions with selected results.

Keywords: memory testing, march tests, *PSF* faults, degrees of freedom

Artykuł zrealizowano w ramach pracy badawczej W/WI/05/06