MROZEK Ireneusz[1]

Wykorzystanie testów symetrycznych w transparentnym testowaniu pamięci RAM

Słowa kluczowe (po polsku),
testowanie pamięci,
transparentne testy pamięci, testy krokowe,
symetryczne testy pamięci,
BIST

Streszczenie

Artykuł przedstawia strategię testowania pamięci wykorzystywaną w technice BIST opartą o symetryczne testy pamięci. Technika symetrycznych testów pamięci gwarantuje, iż proces testowania odbywa się w niezwykle efektywny i szybki sposób. Jej wykorzystanie pozwala zmniejszyć złożoność procesu testowania o około 30% w porównaniu ze standardową metodą testowania transparentnego opartą o sygnaturę odniesienia. Fakt ten ma niezwykle istotne znaczenie jeśli weźmie się pod uwagę bieżące i przyszłe rozmiary używanych pamięci

SYMMETRIC TESTS USAGE IN RAM TRANSPARENT TESTS

Abstract

Computer systems play a significant role almost in each area of life. Therefore we have to ensure their correct working. One of the most important components of each computer system is its memory. So, it is very important to identify a memory fault as soon as it occurs. Memory fault can take place at any time. Therefore we have to test the memory while booting system and periodical when the computer system is in use. In the first case (sometimes in the second case too) we can use a non-transparent memory test, in second case we should use transparent memory test. In this paper we would like to present symmetric version of transparent memory tests which allow us to reduce about 30% the time of the process of memory testing in comparing with the standard non symmetric methods.

1. WSTĘP

Ciągły rozwój technologiczny sprawia, iż w bardzo dużym tempie zwiększa się rynek systemów jednoukładowych i systemów wbudowanych (99% wszystkich obecnie produkowanych procesorów stanowią procesory przeznaczone do systemów wbudowanych i systemów czasu rzeczywistego [9,10, 13]). Rozwój ten powoduje, iż wzrasta zapotrzebowanie na coraz bardziej niezawodne i odporne na awarie (ang. fault tolerant) pamięci półprzewodnikowe. Należy zauważyć, iż w obecnych systemach jednoukładowych, pamięć zajmuje ponad 50% całej powierzchni krzemowej układu SoC, a prognozuje się, iż w niedalekiej przyszłości wielkość ta wzrośnie do 90% [2]. Dodatkowo systemy jednoukładowe i systemy wbudowane pracują jako systemy o znaczeniu krytycznym. Zastosowanie to sprawia, iż bardzo istotny staje się proces testowania wszystkich podzespołów wchodzących w skład tych systemów w tym pamięci, która jest ważnym elementem składowym ich konstrukcji. Aby zapewnić prawidłowe działanie systemów krytycznych testowanie powinno odbywać nie tylko w fazie produkcji, ale również w trakcie ich eksploatacji. Często jednak systemy te wykorzystywane są w miejscach, do których bezpośredni dostęp jest utrudniony lub wręcz niemożliwy. Czytelnym przykładem mogą być sondy kosmiczne (np. system sterujący pojazdem Pathfinder badający powierzchnię Marsa, oparty był o procesor 80C85 i wyposażony w 0.5 MB pamięci RAM). Dlatego coraz częściej układy pamięci półprzewodnikowej produkowane są z wbudowanym mechanizmem testowania BIST (ang. built-in-self test) i samonaprawy BISR (ang. built-in-self repair). Mechanizmy te pozwalają na „ciągłe" samotestowanie się układu, a w przypadku stwierdzenia uszkodzenia – na samonaprawę, bez konieczności interwencji z zewnątrz.

Należy również zauważyć, iż następuje ciągły wzrost pojemności produkowanych układów pamięci. Wzrost ten powoduje, iż algorytmy używane w procesie testowania pamięci, muszą być jak najbardziej efektywne.

W artykule przedstawiona zostanie strategia testowania pamięci wykorzystywana w technice BIST oparta o symetryczne testy pamięci. Technika symetrycznych testów pamięci gwarantuje, iż proces testowania odbywa się w niezwykle efektywny i szybki sposób. Jej wykorzystanie pozwala o około 30% zmniejszyć złożoność procesu testowania w porównaniu ze standardową metodą testowania transparentnego opartą o sygnaturę odniesienia. Fakt ten ma niezwykle istotne znaczenie jeśli weźmie się pod uwagę bieżące i przyszłe rozmiary używanych pamięci.

Symptomem niepoprawnie działającej pamięci najczęściej są błędne wartości odczytywane z poszczególnych jej komórek. Przyczyn fizycznych powodujących takie działanie pamięci jest bardzo dużo. Z uwagi na naturalny fakt, iż pamięci nie można testować poprzez „rozebranie" układu i analizę jego wewnętrznej struktury, zdefiniowane zostały modele uszkodzeń, które pokrywają funkcjonalne efekty zaistniałych defektów fizycznych [4]. Podstawowe funkcjonalne modele uszkodzeń pamięci przedstawia Tabela 1

[1] Instytut Informatyki i Automatyki, Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży, imrozek@pwsip.edu.pl

Tab. 1. Podstawowe modele uszkodzeń funkcjonalnych

Symbol	Nazwa uszkodzenia
SAF	Uszkodzenie sklejeniowe (*ang. stuck-at fault - SAF)*. Uszkodzenie to cechuje stała logiczna wartość w danej komórce pamięci równa zawsze *0* lub równa zawsze *1*.
TF	Uszkodzenie przejścia (*ang. transition fault - TF*). Uszkodzenie tego typu jest szczególnym przypadkiem uszkodzenia *SAF*. Występuje, gdy w komórce pamięci nie jest możliwa zmiana wartości z *0* na *1* lub z *1* na *0*.
CF	Uszkodzenie sprzężeniowe (*ang. coupling fault - CF)*. Uszkodzeniu *CF* przy zmianie wartości komórki pamięci *j* (agresora), pociąga za sobą zmianę wartości komórki *i* (ofiary).
PSF	Uszkodzenie uwarunkowane zawartością (*ang. pattern sensitive fault – PSF)*. W uszkodzeniu tym wartość (lub możliwość zmiany wartości) komórki *i* (komórki bazowej) jest zależna od wartości (lub ich zmian) wszystkich pozostałych komórek pamięci (komórek wiążących) składających się na to uszkodzenie.

Kluczową rolę w technikach testowania pamięci odgrywają testy krokowe (ang. march tests). Test krokowy (test typu March) składa 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, który może być rosnący (ang. increasing address order) lub malejący (ang. decreasing address order). Standardowo przyjmuje się, iż rosnący sposób adresowania to taki w którym adresy wzrastają od 0 do *N*-1, zaś malejący - w którym adresy maleją od N-1 do 0 (N – rozmiar testowanej pamięci). Rosnący sposób adresowania oznaczany jest przez $\Uparrow$, zaś malejący przez $\Downarrow$. W rzeczywistości jednak przyjęte założenia dotyczące porządku adresów w sposobie adresowania nie mają znaczenia. Najważniejszy jest jedynie fakt, aby zostały uwzględnione wszystkie adresy pamięci i uporządkowanie adresów w malejącym sposobie adresowania było odwrotne w stosunku do porządku adresów w rosnącym sposobie adresowania [4]. Cały test typu March ograniczony jest poprzez parę nawiasów klamrowych { ... }, podczas gdy poszczególne jego fazy ograniczone są parą nawiasów okrągłych (...). Polecenie odczytu w teście typu March oznaczane jest poprzez r*x*, gdzie *x* oznacza wartość spodziewaną. Różnica pomiędzy wartością spodziewaną a wartością rzeczywiście odczytaną z pamięci świadczy o występującym uszkodzeniu. Polecenie zapisu oznaczane jest poprzez w*x*, gdzie *x* jest zapisywaną do pamięci wartością (w przypadku pamięci o organizacji bitowej, *x* przybiera wartość 0 lub 1). Jako przykład rozpatrzony zostanie test Mats+[4]:

$$\{\Updownarrow w(0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}.$$
$$P0 \qquad P1 \qquad P2$$

Test ten składa się z trzech faz: *P0*, *P1*, *P2*. Faza *P0* odpowiedzialna jest za ustalenie wartości początkowej pamięci. Adresacja użyta do określenia wartości początkowej może zostać przyjęta dowolnie, stąd jej oznaczenie jako $\Updownarrow$. W powyższym teście stan początkowy pamięci ustalany jest poprzez wpisanie do każdej komórki wartości 0. Faza *P1* realizowana jest zgodnie z rosnącym sposobem adresowania. Składa się z dwóch poleceń: polecenia odczytu ze spodziewaną wartością *0* (*r0*), oraz polecenia zapisu wartości *1* (*w1*). Zakładając standardowy porządek adresów, w fazie tej w kolejności adresów od 0 do *N*-1 (*N*- rozmiar testowanej pamięci), odczytywana jest zawartość pamięci z wartością spodziewana równą *0*, po czym do tej samej komórki pamięci (przed przejściem do komórki następnej) następuje zapis wartości *1*. Faza *P2* testu *MATS+*, w porządku adresów malejących, odczytuje ponownie zawartość każdej komórki pamięci ze spodziewaną wartością równą *1*, po czym wpisuje tam wartości *0*. W przypadku, gdy w którejkolwiek komórce odczytana wartość z pamięci będzie inna od wartości spodziewanej, będzie to sygnał informujący o istniejącym uszkodzeniu. Mimo swej prostoty test ten gwarantuje wykrycie wszystkich uszkodzeń typu *SAF*. Osiągnięte jest to dzięki wpisaniu i odczytaniu ze wszystkich komórek pamięci zarówno wartości *0* jak i wartości *1*. Dodatkowo możliwe jest wykrycie przez powyższy test niektórych uszkodzeń *TF* i *CF*. Wynika to z faktu, iż dzięki fazie *P1*, można stwierdzić, czy polecenie zapisu realizowane w danej komórce nie wpływa na wartość komórki pod wyższym adresem, i analogicznie faza *P2* pozwala stwierdzić, czy zapis do danej komórki nie wpływa na wartość komórki pod niższym adresem. Jednak bardziej złożone testy krokowe pozwalają osiągnąć wyższy współczynnik pokrycia uszkodzeń. Wiele testów pozwala wykryć zarówno wszystkie uszkodzenia *SAF*, *TF* jak również *CF* [4].

Przedstawiony powyżej schemat testowania pamięci oparty o standardowe testy krokowe zakłada, iż zawartość testowanej pamięci może zostać „zamazana". Tak więc w bardzo wielu przypadkach użycie powyższego schematu możliwe jest jedynie podczas fazy inicjacji urządzenia. Należy jednak zauważyć, że wiele urządzeń elektronicznych pracuje w ruchu „ciągłym". Nie są wyłączane wcale, lub są wyłączane bardzo rzadko. Urządzenia te często podejmują decyzje o znaczeniu krytycznym. Należy mieć wtedy pewność, iż na podjętą decyzję nie będzie miało wpływu powstałe w międzyczasie uszkodzenie. Dlatego też istnieją techniki pozwalające sprawdzać poprawność działania poszczególnych elementów w trakcie normalnej pracy systemu. W przypadku pamięci używa się do tego celu transparentnych testów pamięci. Główne założenie działania tych testów mówi, iż zawartość pamięci w momencie bezpośrednio poprzedzającym rozpoczęcie testu

jest taka sama jak w momencie bezpośrednio następującym po zakończeniu testu. Tak więc w przypadku poprawnie działającej pamięci, test transparentny musi pozostawić jej zawartość w stanie „nienaruszonym", zaś w przypadku występującego w pamięci defektu - musi to wykryć.

2. TESTY TRANSPARENTNE

Jedną z pierwszych prac dotyczących transparentnego testowania pamięci był referat B. Koeneman'a wygłoszony w 1986 roku na seminarium „Design For Testability" [5]. Przedstawiona przez niego technika testów transparentnych opierała się na liniowości liczonej sygnatury (sumy kontrolnej) pamięci i realizowana była w następujących krokach:

1. Wyliczenie wartości sygnatury pamięci *s(CONTENTS)*, gdzie *CONTENTS* oznacza zawartość pamięci (bloku pamięci) z momentu bezpośrednio poprzedzającego rozpoczęcie testu.
2. Zmiana zawartości pamięci na *NEW_CONTENTS* zgodnie z następującą formułą:

$$NEW\_CONTENTS = CONTENTS \oplus TP,$$

gdzie *TP*, określa wartość wzorca testowego użytego do testu.

3. Wyliczenie wartości sygnatury nowej zawartości pamięci *s(NEW_CONTENTS)*.

4. Odtworzenie pierwotnej zawartości pamięci zgodnie z formułą:

$$CONTENTS = NEW\_CONTENTS \oplus TP.$$

5. Stwierdzenie faktu poprawnego (lub niepoprawnego) działania pamięci na podstawie prawdziwości (lub nieprawdziwości) następującej formuły:

$$s(CONTENTS) \oplus s(NEW\_CONTENTS) = s(TP).$$

Jak zauważono w [6] podejście to, oprócz niewątpliwych zalet, posiada również pewne niedoskonałości (np. stosunkowo słaba jakość). Dlatego prace nad rozwojem powyższej techniki testowania pamięci trwały i w roku 1992 zaowocowały powstaniem metody pozwalającej przekształcić dowolny test krokowy w test transparentny przy zachowaniu jego jakości[6]. Przykłady podstawowych testów krokowych oraz ich postaci transparentnych przedstawia **Błąd! Nie można odnaleźć źródła odwołania.**.:

Tab. 2. Transparentne postacie podstawowych testów krokowych

Nazwa testu	Postać nietransparentna	Postać transparentna
MATS	⇕ (w0); ⇑(r0,w1); ⇕ (r1)	⇑(ra, wā); ⇑ (rā)
MATS+	⇕ (w0); ⇑(r0,w1); ⇓(r1,w0)	⇑(ra,wā); ⇓(rā,wa).
MATS++	⇕ (w0); ⇑(r0,w1); ⇓(r1,w0,r0)	⇑(ra,wā); ⇓(rā,wa,ra)
March X	⇕ (w0); ⇑(r0,w1); ⇓(r1,w0); ⇕ (r0)	⇑(ra,wā); ⇓(rā,wa); ⇑ (ra)
March Y	⇕ (w0); ⇑(r0,w1,r1); ⇓(r1,w0,r0); ⇕ (r0)	⇑(ra,wā,rā); ⇓(rā,wa,ra); ⇑ (ra)
March C-	⇕ (w0); ⇑(r0,w1); ⇑(r1,w0); ⇓(r0,w1); ⇓(r1,w0); ⇕ (r0)	⇑(ra,wā); ⇑(rā,wa); ⇓(ra,wā); ⇓(rā,wa); ⇕ (ra)

W oparciu o powyższe testy proces transparentnego testowania pamięci realizowany jest w trzech etapach.
ETAP I: Wygenerowanie i zapamiętanie wzorcowej odpowiedzi układu pamięci na zadany test.
Generowana odpowiedź wzorcowa, zależna jest od użytego testu i bieżącej zawartości pamięci. W procesie jej generowania używane są wyłącznie polecenia odczytu transparentnej postaci testu. Każde polecenie odczytu (ra i r$\bar{a}$) powoduje odczytanie zawartości pamięci z tą różnicą, iż przy poleceniu ra odczytana wartość w postaci niezmiennej przekazywana jest do analizatora sygnatury, zaś przy poleceniu r$\bar{a}$ do analizatora sygnatury przekazywana jest wartość odwrotna w stosunku do wartości odczytanej. Etap ten jest więc transparentny dla zawartości pamięci.

ETAP II: Wygenerowanie i zapamiętanie rzeczywistej odpowiedzi układu pamięci na zadany test.
Podobnie jak w Etapie I odpowiedź układu zależna jest od użytego testu i zawartości pamięci. W odróżnieniu jednak od Etapu I przy generowaniu rzeczywistej odpowiedzi układu używane są zarówno polecenia odczytu jak i zapisu. W fazie tej następuje więc modyfikacja zawartości testowanego układu. Jednak w przypadku poprawnie działającej pamięci, realizowany w teście ciąg poleceń zapewnia, że zawartość pamięci, po zakończonym procesie testowania, będzie taka sama jak w chwili bezpośrednio poprzedzającej jego rozpoczęcie. Przykładem takiego ciągu poleceń może być: $(ra, w\bar{a}, r\bar{a}, wa)$. W ciągu tym występują w kolejności następujące działania: odczytanie zawartości komórki pamięci

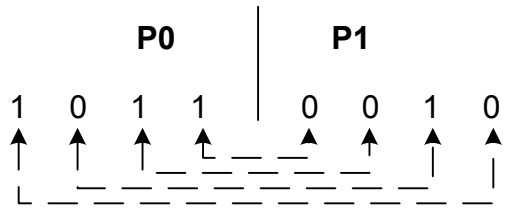
do rejestru, zapisanie zanegowanej zawartości rejestru do komórki pamięci, odczytanie zawartości komórki pamięci do rejestru (będzie to wartość zanegowana w stosunku do wartości pierwotnej), zapisanie zanegowanej zawartości rejestru do rozpatrywanej komórki pamięci. Widać więc, iż podwójne zanegowanie sprawia, że zawartość komórki pamięci po wykonaniu rozpatrywanego ciągu poleceń jest identyczna jak wartość pierwotna. Tak więc w przypadku poprawnie działającej pamięci, etap ten jest również transparentny dla jej zawartości.

ETAP III: Porównanie odpowiedzi rzeczywistej z odpowiedzią wzorcową.
W etapie tym następuje porównanie danych otrzymanych w Etapie I z danymi otrzymanymi w Etapie II. Ewentualna różnica ich wartości sygnalizuje niepoprawne działanie pamięci.

Podejście zaprezentowane powyżej pozwala użyć testów o wysokiej jakości (o dużej liczbie typów wykrywanych uszkodzeń) do transparentnego testowania pamięci. Jednak w porównaniu do tradycyjnego podejścia, podejście transparentne narzuciło dodatkowy koszt w postaci wyliczenia sygnatury. Łatwo zauważyć, iż wyliczenie sygnatury odniesienia realizowane w Etapie I pochłania około 30% czasu trwania całego testu. Przy obecnych, wciąż rosnących rozmiarach pamięci, wielkość ta jest wielkością znaczącą. Dlatego zostało zaproponowane rozwiązanie powyższego problemu. Została zaproponowana technika symetrycznych testów pamięci[7]. Testowanie realizowane zgodnie z powyższą techniką nie wymaga realizacji Etapu I (liczenia sygnatury odniesienia). Tym samym skraca czas realizacji testu transparentnego o około 30%.

3. TESTY SYMETRYCZNE

W większości niedestrukcyjnych algorytmów testowania pamięci generowany wyjściowy ciąg danych cechuje się wysokim stopniem symetrii. Jako przykład rozpatrzmy niedestrukcyjną odmianę algorytmu *MATS+ - {⇑(ra, wā); ⇓(rā,wa)}* - działającą na pamięci o pojemności 4 bitów i zawartości {1,0,1,1}. Zgodnie z pierwszą fazą testu dane wyjściowe przesyłane do rejestru (licznika) sygnatury będą miały następującą postać: d=(1,0,1,1). Podczas drugiej fazy *⇓(rā,wa)* dane wyjściowe pojawią się w kolejności odwrotnej oraz ich wartości będą negacją logiczną wartości występujących w strumieniu danych wyjściowych pierwszej fazy. I tak dane te mają postać: d*=(0,0,1,0) (patrz **Błąd! Nie można odnaleźć źródła odwołania.**).



Rys.1 Symetria w testach krokowych

Połączenie własności symetrii ciągu wyjściowego z odpowiednim sposobem liczenia wartości sygnatury pozwala na całkowite pominięcie fazy wstępnej testu (liczenie sygnatury odniesienia) - niezbędnej w dotychczasowych realizacjach testów transparentnych. Symetryczny ciąg danych wyjściowych i zmiana, w drugiej części testu wielomianu używanego w celu otrzymania sygnatury, na odpowiedni wielomian odwrotny (ang. reciprocal polynomial), pozwala precyzyjnie określić wartość sygnatury odniesienia bez potrzeby realizacji fazy wstępnej testu, która była niezbędna dotychczas [7]. Należy również dodać, iż wartość ta jest niezależna od zawartości i rozmiaru testowanej pamięci. Analizując istniejące testy krokowe można stwierdzić, iż w testach tych pojawiają się dwa typy symetrii. Zakładając, iż pierwszą część ciągu generowanego przez test można zapisać jako $d = (d_0, d_1, \dots, d_{n-1}) \in \{0,1\}^n$, wtedy mówimy, iż:

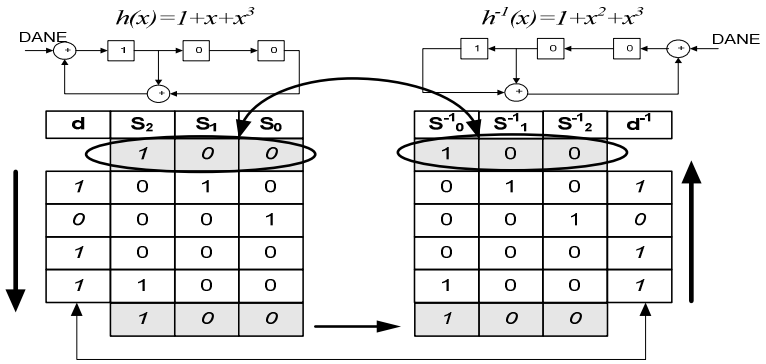
- ciąg danych wyjściowych charakteryzuje się symetrią TYPU 1 jeżeli można go zapisać jako dd^{-1};
- ciąg danych wyjściowych charakteryzuje się symetrią TYPU 2 jeżeli można go zapisać jako dd^{-1c},

gdzie, $d^{-1}:=(d_{n-1}, \dots, d_0) \in \{0,1\}^n$, $d^{-1c}=(d^{n-1c},\dots,d^{0c}) \in \{0,1\}^n$, a wartość d^{kc} jest wartością komplementarną do wartości d^k.

W [7] zostało udowodnione twierdzenie mówiące, że $S(d^{-1} S(d,s,h)\text{-}1,h^{-1})=s^{-1}$, gdzie d i d^{-1} jest odpowiednio pierwszą i drugą częścią danych kierowanych do analizatora sygnatury, a $b=S(d,s,h)$ oznacza wartość sygnatury ciągu d, wyliczoną z użyciem wielomianu charakterystycznego h przy początkowym stanie rejestru sygnatury równym s. Powyższe twierdzenie oznacza, że dla odwróconego ciągu danych d^{-1} analizowanego przez analizator sygnatury ze stanem początkowym b^{-1} (odwrócona sygnatura ciągu d) i wielomianem charakterystycznym h^{-1} (wielomian odwrotny w stosunku do wielomianu h) otrzymuje się wartość sygnatury s^{-1} (odwróconą wartość początkową rejestru sygnatury). Tak więc dla testu, którego ciąg danych wyjściowych charakteryzuje się symetrią *TYPU 1 –* dd^{-1}- nie ma konieczności liczenia sygnatury odniesienia, gdyż dla danych nie zawierających błędu jest ona znana na podstawie wartości początkowej rejestru sygnatury. Podobne twierdzenie zostało sformułowane i udowodnione dla symetrii *TYPU2*. Również zostało pokazane, iż wartość końcową rejestru sygnatury (w przypadku analizy danych postaci dd^{-1c}), można określić na podstawie wartości początkowej tego rejestru.
Celem praktycznego zademonstrowania powyższych rozważań przedstawiona zostanie analiza działania testu MATS+ *{⇑(ra, wā); ⇓(rā,wa)}* w odniesieniu do pamięci o rozmiarze 4 bitów i zawartości {1,0,1,1}. Wartość początkowa rejestru sygnatury niech równa będzie $s=(s_2,s_1,s_0)=(0,0,1)$, zaś wielomian charakterystyczny analizatora *SA* niech ma postać

$h(x)=1+x+x^3$. Wielomian charakterystyczny analizatora SA^{-1} jest wielomianem odwrotnym do wielomianu $h(x)$ czyli będzie to wielomian postaci $h^{-1}(x) = 1+x^2+x^3$. Dla powyższych danych kolejne wartości przyjmowane przez rejestry sygnatury przedstawia rys 2. Należy zauważyć fakt, iż w przypadku, gdy początkowa wartość sygnatury jest równa *0*, to zgodnie z udowodnionymi w [7] twierdzeniami, wartość końcowa dla ciągu danych postaci dd^{-1} również będzie równa *0*. Stwierdzono również, iż dla takiego ciągu, wartość sygnatury nie jest zależna:

- od wartości danych przekazywanych do analizatora sygnatury,
- od długości ciągu danych,
- od używanego testu,
- od wielomianu charakterystycznego analizatora sygnatur.



Rys.2 Przykład realizacji testu symetrycznego

Jednak okazuje się, iż nie wszystkie używane testy krokowe, cechują się pożądaną symetrią. Przedstawia to **Błąd! Nie można odnaleźć źródła odwołania.**.

Tab.3. Porównanie testów krokowych pod kątem symetrii generowanych danych wyjściowych

Nazwa testu	***Typ symetrii***
MATS	*TYP 2*
MATS+	*TYP 2*
MATS++	*BRAK*
MARCH X	*BRAK*
MARCH Y	*BRAK*
MARCH C-	*BRAK*
MARCH C	*TYP 1*
MARCH A	*TYP 1*
MARCH B	*TYP 1*

Z uwagi na powyższe, powstały techniki umożliwiające transformację dowolnego testu krokowego w test symetryczny. Szczegółowy opis powyższych technik proponowanych przez autora można znaleźć w [8]. Należy tu nadmienić, iż proponowane transformacje testu z postaci niesymetrycznej na symetryczną nie powoduje zmniejszenia jakości testu.

4. WNIOSKI

W artykule przedstawiona została idea testowania pamięci RAM. Szczególny nacisk położony został na zademonstrowanie idei testów transparentnych – testów które podczas swojego działania nie niszczą bieżącej zawartości pamięci[5][6]. Wadą standardowego podejścia do transparentnego testowania pamięci jest fakt, iż wymusza ono dodatkowy nakład obliczeniowy (w stosunku do testowania tradycyjnego) niezbędny do wyliczenia sygnatury odniesienia. Zajmuje to około 30% czasu trwania całego testu. Przy obecnych wciąż rosnących rozmiarach pamięci czas ten nabiera coraz większego znaczenia. Dlatego powstało rozwiązanie niwelujące ten problem. Powstała technika symetrycznych testów pamięci pozwalająca pominąć konieczność wyliczania sygnatury odniesienia, a tym samym skrócić w sposób znaczny, czas trwania całego testu[7]. Niestety nie wszystkie testy krokowe w sposób bezpośredni można realizować w oparciu o technikę testów symetrycznych. Zostały zatem zaproponowane algorytmy umożliwiające transformację dowolnego testu krokowego w test symetryczny[8]. Różne techniki implementacji omawianych tutaj testów, oraz rezultaty ich działania w odniesieniu do najbardziej skomplikowanych typów uszkodzeń pamięci – uszkodzeń PSF czytelnik może znaleźć w wielu publikacjach autora w tym w [1][3][8][11].

5. BIBLIOGRAFIA

[1] Sokol B., Mrozek I., Yarmolik V.N.: *Impact of the Address Changing on the Detection of Pattern Sensitive Faults*, Information Processing and Security Systems, Springer 2005, pp.: 217-225

[2] Marinissen E. J., Prince B., Keitel-Schulz D., Zorian Y.: *Challenges in Embedded Memory Design and Test*, Proceedings of the conference on Design, Automation and Test in Europe - Volume 2 (DATE '05), Vol. 2 DATE 2005, pp 722-727

[3] Sokół B., Mrozek I., Yarmolik V.N.: *Transparent March Tests to effective Pattern Sensitive Faults Detection*, EAST-WEST DESIGN & TEST WORKSHOP - EWDTW'04, Jałta, Ukraina 22-26 września 2004, str.: 166-171

[4] van de Goor A. J.: *Testing Semiconductor Memories, Theory and Practice*, Chichester, John Wiley & Sons, (1991).

[5] Koeneman B., oral presentation, Design For Testability Workshop, Vail Colo, Apr. 1986

[6] Nicolaidis M.: *Transparent BIST for RAMs*, Proceedings IEEE International Test Conference 1992,

[7] Yarmolik V.N., Hellebrand. S., Wunderlich H.J.: *Symmetric Transparent BIST for RAMs*, Proceedings of Design and Test in Europe (DATE'99), Munich, March, 1999, pp. 702-707

[8] Mrozek I., *Wykrywanie i lokalizacja uszkodzeń pamięci RAM oparte o transparentne testy pamięci*, rozprawa doktorska, Wydział Informatyki, Politechnika Białostocka, 2004.

[9] Burns A, Wellings A.: *Real time systems and programming languages*, Addision Wesley Longmain (2001).

[10] Lewis W. D.: *Fundamentals of Embedded Software*, Printice Hall. (2002)

[11] Mrozek I., Yarmolik V.N.: *Detection of Pattern Sensitive Faults based on transparent march tests*, Proceedings Mixed Design of Integrated Circuits and Systems – MIXDES'03, pp.542-545,Lodz 26-28 June 2003

[12] Mrozek I, Yarmolik, V.N.: *Transparentne testowanie pamięci RAM oparte na charakterystyce adresowej*, Elektronika, R.51, nr 9, s. 161-163, 2010

[13] Chin-Lung S.,Rei-Fu H.,Cheng-Wen W.,Kun-Lun L., Wen Ching W.: *Built-in Self-Diagnosis and Repair Design With Fail Pattern Identification for Memories* IEEE Trans. VLSI Syst., 12, Vol. 19, 2011, pp. 2184-2194