TRANSCOMP – XIV INTERNATIONAL CONFERENCE
COMPUTER SYSTEMS AIDED SCIENCE, INDUSTRY AND TRANSPORT

Testowanie,
Efektywność,
Pamięć

Eugenia BUSŁOWSKA [1]

ANALIZA PORÓWNAWCZA METOD KOMPAKCJI ODPOWIEDZI UKŁADU PAMIĘCI NA TEST

Artykuł przedstawia wyniki badań symulacyjnych porównujących efektywność wykrywania usterek pamięci RAM, metodą analizy sygnatury oraz zaproponowanymi metodami kompakcji odpowiedzi układu na test.

COMPARATIVE ANALYSIS OF METHODS COMPACTION ANSWERS TO THE TEST OF MEMORY

The article presents the results of simulation studies comparing the effectiveness of fault detection of RAM, a signature method of analysis and proposed methods of compaction system response to the test.

1. WSTĘP

Obecnie złożoność cyfrowych układów scalonych podwaja się średnio, co półtora roku. Wzrost złożoności powoduje również podwyższenie oczekiwań dotyczących szerokiej funkcjonalności, szybkości działania oraz krótkiego czasu wdrażania nowych rozwiązań. Sprostanie wysokim wymaganiom jakościowym, w tym zagwarantowanie adekwatnego poziomu uzysku i niezawodności, wiąże się z koniecznością przeprowadzania sprawdzania poprawności, zwanego testowaniem. Staje się ono nieodzownym elementem nie tylko fazy produkcji, lecz również eksploatacji. Układy pamięci będące ważną częścią systemów cyfrowych, jednym z głównych elementów coraz większej liczby podzespołów komputerowych, muszą być niezawodne. Z roku na rok przybywa zastosowań układów pamięci, których uszkodzenie może mieć poważne skutki, w zależności od roli systemu, od niezadowolenia potencjalnego użytkownika, poprzez straty ekonomiczne, do zagrożenia bezpieczeństwa lub życia ludzkiego. Zapotrzebowanie na coraz bardziej niezawodne układy pamięci, powoduje dążenie do stworzenia procesu testowania, nie wpływającego na pracę urządzenia, który zapewniłby poprawność działania prawie w stu procentach [1, 2].

Najważniejszym celem testu jest, aby po wyczerpującym sprawdzaniu uzyskać możliwie największy poziom ufności do otrzymanych rezultatów testowania i możliwie najwyższy poziom dokładności w jak najkrótszym czasie [1]. Należy również uwzględnić wymóg łatwej testowalności układów cyfrowych już na etapie ich projektowania. Od dawna podkreślana jest konieczność stałego opracowywania nowych metod testowania

[1]Politechnika Białostocka, Wydział Informatyki, 15-351 Białystok, ul. Wiejska 45A, tel. 85-746-90-50, Fax : 085-746-90-57, e-mail: ebuslowska@pb.edu.pl

układów cyfrowych, w celu obniżania ogólnego kosztu ich wytwarzania. Koszt testowania stanowi często blisko połowę nakładów, przez co staje się największym pojedynczym składnikiem ostatecznej ceny układu scalonego. Nakłady na testowanie pojedynczego tranzystora przewyższają koszty jego wyprodukowania.

Nowe testy są opracowywane w taki sposób, aby przy jak najmniejszej długości odpowiedzi układu pamięci na test, wykryć jak największą liczbę błędów. W chwili obecnej najpopularniejsze są testy maszerujące przekształcone do postaci testów transparentnych (ang. transparent test), czyli testów, które nie zmieniają zawartości pamięci po zakończeniu testowania [3]. W miarę poznawania skutków działania poszczególnych usterek pojawiało się coraz więcej testów o dużej skuteczności pokrywania poszczególnych usterek [4, 5].

Proces transparentnego testowania pamięci można podzielić na trzy fazy. W fazie pierwszej jest generowana i zapamiętywana wzorcowa odpowiedź układu pamięci na zadany test, następnie przeprowadza się test i generuje faktyczną odpowiedź układu pamięci na test. W trzeciej fazie następuje porównanie obliczonych, w dwóch poprzednich fazach, odpowiedzi. Jeżeli wystąpi jakakolwiek różnica w wartościach oznacza to nieprawidłową pracę układu pamięci.

2. ALGORYTMY KOMPAKCJI ODPOWIEDZI

Wzrost efektywności współczesnych testów spowodował rozrastanie się samych algorytmów i w konsekwencji długości generowanego ciągu wynikowego. W wewnątrzukładowym testowaniu w analizatorze nie ma możliwości porównania ciągu wyjściowego bit po bicie, w związku, z czym stosuje się rejestry LFSR do kompakcji ciągu wyjściowego. Analiza sygnatury, ze względu na topologię umiejscowienia błędów nie zawsze jest skuteczna. W swoich pracach badawczych zaproponowałam kilka algorytmów kompaktujących, których efektywność może być wyższa, niż przy zastosowaniu rejestrów przesuwających [6, 7, 8, 9, 10]. W celu porównania efektywności zaproponowanych metod kompakcji ze standardowo stosowanymi metodami oraz w celu poprawności komparacji, wielomian charakterystyczny używany do kompakcji w analizie sygnatury zawsze jest takiego samego stopnia, jak długość charakterystyki wykorzystywanej w danej metodzie.
W opracowanych algorytmach charakterystyki miały następujące długości względem wielkości macierzy – $r \times r$:

- Sumy parzystości przy organizacji macierzowej (SP) – $2\log_2(r+1)$,
- Liczba jednakowych sąsiadów (LJS) – $2\log_2(r+1) + 4$,
- Tranzytywność (TR) – $4\log_2(r+1)$,
- Liczność jedynek w kolumnie (LJK) – $3\log_2(r+1)$.

Wykonano wielokrokowe symulacje badające wykrywalność usterek dla kilku warunków pracy. Najpierw zostały przeprowadzone badania na losowej zawartości macierzy o zmiennych wielkościach i losowo generowanych błędach poszczególnych krotności na zasadzie zamiany wartości na przeciwną. Sprawdzono efektywność zaproponowanych metod z tradycyjną metodą analizy sygnatury. Kolejne badania były przeprowadzone dla losowo generowanej zawartości pamięci o zmiennych wielkościach i dla losowo generowanych usterkach poszczególnych krotności i podtypów SAF. Zastosowano transparentny test krokowy Mats++ $\Uparrow$(ra,wā); $\Downarrow$(rā,wa,ra).

3. PORÓWNANIE EFEKTYWNOŚCI OPRACOWANYCH METOD KOMPAKCJI Z ANALIZĄ SYGNATURY

W badaniach symulowano losową zawartość pamięci, dla której liczono charakterystykę wzorcową, jedną z zaproponowanych metod, stosując rejestr przesuwny. Obie charakterystyki były liczone dla tej samej zawartości pamięci i miały taką samą liczbę bitów. W tabeli 1 zostały przedstawione długości charakterystyk dla określonej liczby bitów w adresie – *DlAdr*, rozmiaru macierzy – *RM* i długości ciągu wyjściowego *RW* testu krokowego dla testowanej wielkości pamięci *RP* [10].

Tab. 1. Długości charakterystyk

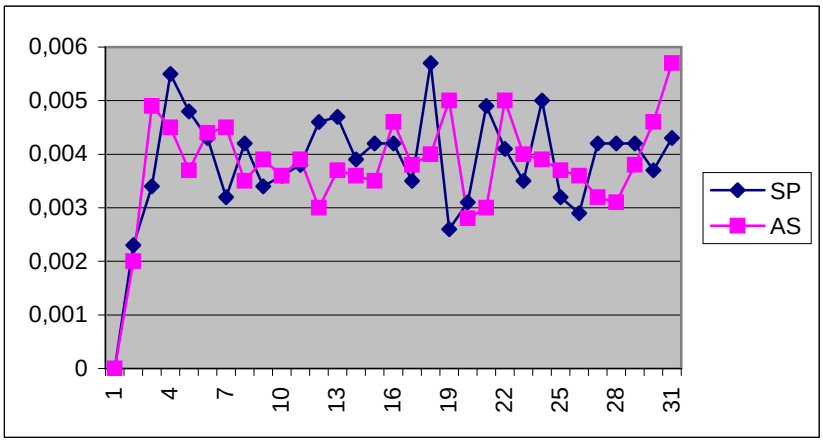
RP	***RW***	***RM***	***DlAdr***	**SP**	**LJS**	**TR**	**LJK**
1023	3069	56	6	12	16	24	18
2047	6141	79	7	14	18	28	21
4095	12285	111	7	14	18	28	21
8191	24573	157	8	16	20	32	24
16383	49149	222	8	16	20	32	24
32767	98301	314	9	18	22	36	27
65535	196605	444	9	18	22	36	27
131071	393213	628	10	20	24	40	30
262143	786429	887	10	20	24	40	30
524287	1572861	1255	11	22	26	44	33
1048575	3145725	1774	11	22	26	44	33

Przykładowo: dla testowanej pamięci o wielkości 1023 bitów długość charakterystyki w algorytmie SP i wielomian charakterystyczny rejestru wynosi dwanaście bitów, dla algorytmu TR – dwadzieścia cztery bity.

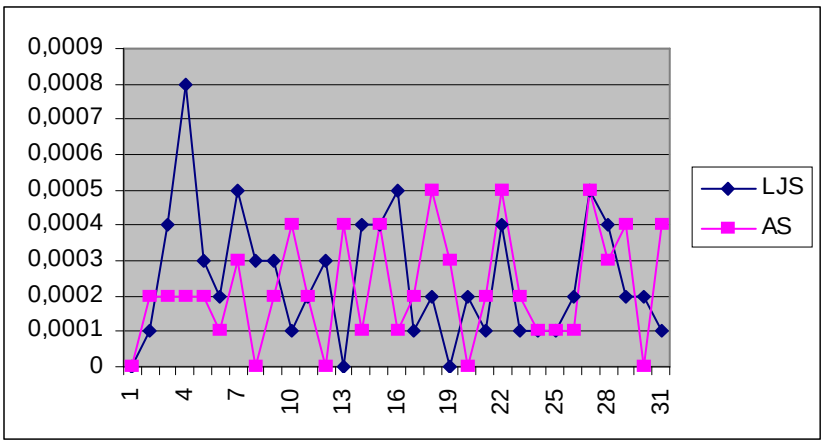
Po obliczeniu charakterystyk, losowo określano dla poszczególnych krotności rozmieszczenie oraz podtyp usterek SAF. Usterki SAF są w transparentnym teście krokowym Mats++ ⇑(ra,wā); ⇓(rā,wa,ra) uwidaczniane w postaci błędów jedno- lub dwukrotnych, w zależności od podtypu usterki i pierwotnej wartości.

W przypadku usterki SA0 występującej w komórce - *w*, jeśli pierwotnie w tej komórce była wartość 0, to błąd pojawi się na jednej pozycji - *3*RP-2*w-2* w ciągu wyjściowym. Jeśli w komórce *w* była wartość 1, to usterka uaktywni się w postaci dwóch błędów w komórce *w* oraz w *3*RP-2*w-1*. Dla usterki SA1, jeśli w komórce *w* była wartość 0, to błąd pojawi się w dwóch komórkach *w* oraz w *3*RP-2*w-1* ciągu wyjściowego testu. Jeśli była wartość 1, to błąd pojawi się tylko w komórce *3*RP-2*w-2*.

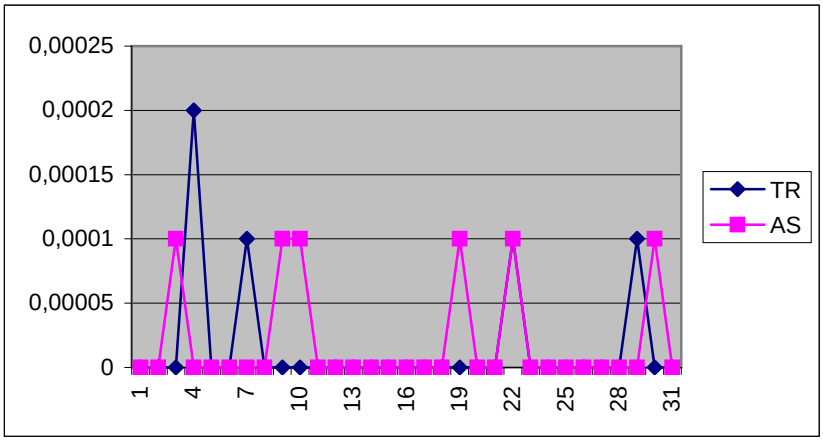
Mając określone pozycje błędów, ciąg wynikowy testu dla opracowanych metod przedstawiono w postaci macierzy i wyliczono charakterystykę, którą porównywano ze wstępnie określoną charakterystyką wzorcową. Jeśli suma modulo-2 obu charakterystyk była różna od 0 to usterki były wykrywane, w przeciwnym razie algorytmy nie wykrywały usterek. Rysunki 1-4 przedstawiają prawdopodobieństwa niewykrycia usterki przez opracowane algorytmy kompaktujące i analizę sygnatury dla testowanej pamięci sześćdziesiąt trzy bitowej.



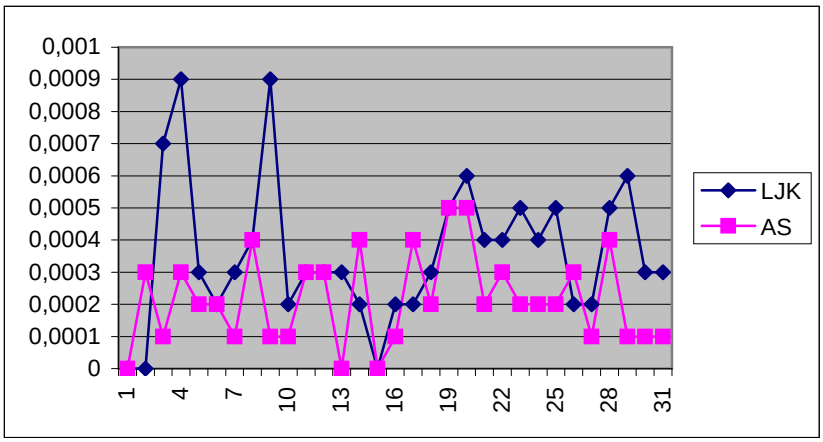
Rys. 1. Prawdopodobieństwa niewykrycia poszczególnych usterek dla 10 000 prób przez algorytm SP i AS



Rys. 2. Prawdopodobieństwa niewykrycia poszczególnych usterek dla 10 000 prób przez algorytm LJS i AS



Rys. 3. Prawdopodobieństwa niewykrycia poszczególnych usterek dla 10 000 prób przez algorytm TR i AS



Rys. 4. Prawdopodobieństwa niewykrycia poszczególnych usterek dla 10 000 prób przez algorytm LJK i AS

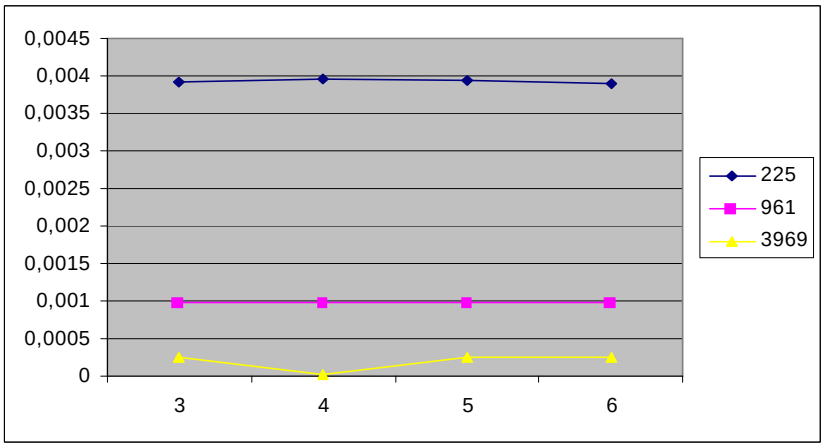
Z powyższych wykresów wynika, że najgorsze wyniki uzyskano dla algorytmu SP. Wynika to z tego, że w tej metodzie charakterystyka jest obliczana tylko na podstawie parzystości wierszy i kolumn i stanowi sumę modulo-2 adresów, dla których parzystość wynosi 1.

W przypadku pozostałych algorytmów większa ich skuteczność, niż w analizie sygnatury, występuje dla wybranych krotności usterek. Jest ona ściśle związana z rozmieszczeniem usterek w testowanej macierzy.

4. SZACOWANA EFEKTYWNOŚCI KOMPAKCJI

Efektywność wykrywania błędów proponowanych algorytmów kompaktujących zależy od ich krotności, ale przede wszystkim od ich rozmieszczenia w macierzy. Jest to związane

z zastosowaniem parzystości kolumnowej i wierszowej w sygnaturze charakteryzującej kompaktowany ciąg bitów.
Ze względu na możliwości obliczeniowe sprzętu przeprowadzono badania symulacyjne tylko dla kilku wielkości pamięci. Można postawić pytanie: jak otrzymane wyniki mają się do rzeczywistych wielkości macierzy? Aby to wyjaśnić, zostaną przedstawione rozkłady prawdopodobieństw niewykrycia błędów krotności od trzech do sześciu [10]. Wszystkie algorytmy wykrywają błędy pojedyncze i podwójne. Na podstawie przeprowadzonych badań prawdopodobieństwa niewykrycia błędów od potrójnych do sześciokrotnych dla pierwszego algorytmu SP są przedstawione na rysunku 5.

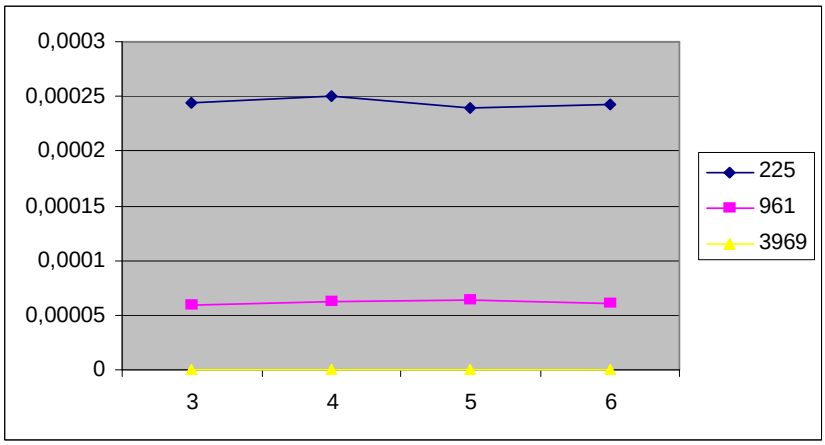


Rys. 5. Prawdopodobieństwo niewykrycia błędów przez algorytm SP

Można zauważyć, że prawdopodobieństwo niewykrycia błędów przebadanych krotności maleje wraz ze wzrostem wielkości macierzy. Wynika to ze zmniejszania się prawdopodobieństwa wystąpienia kombinacji niewykrywalnych błędów.
Na podstawie przeprowadzonych badań stwierdzono:

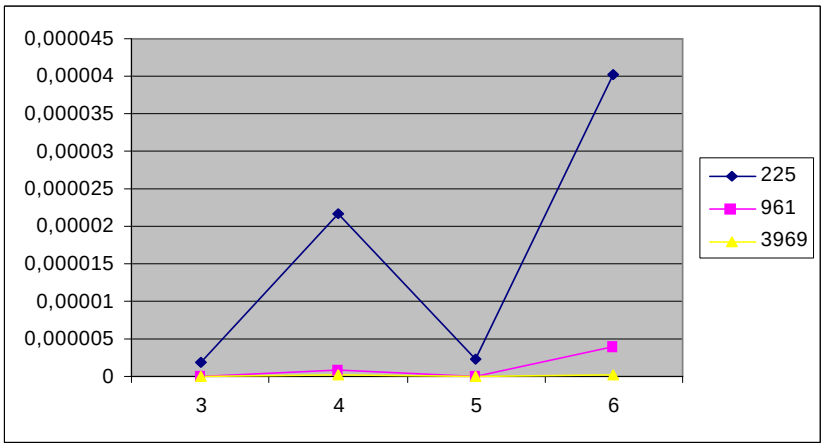
- spadek udziału liczby błędów niewykrywalnych w stosunku do liczby wszystkich błędów w funkcji liczby wierszy i kolumn,
- przy wzroście krotności błędów, liczba niewykrywalnych błędów nie ulega zwiększeniu.

Przedstawione obliczenia dotyczą najprostszego algorytmu, zawierającego w swojej charakterystyce jedynie informację o parzystościach. Pozostałe algorytmy oprócz parzystości, posiadają dodatkowe składowe podnoszące ich efektywność. Efektywność w tym wypadku pojmowana jest jako możliwość wykrycia wystąpień błędów wcześniej nie wykrywanych. Bardzo zbliżoną efektywnością, wykrywalności błędów, do algorytmu SP charakteryzuje się algorytm LJS (rys. 6.).



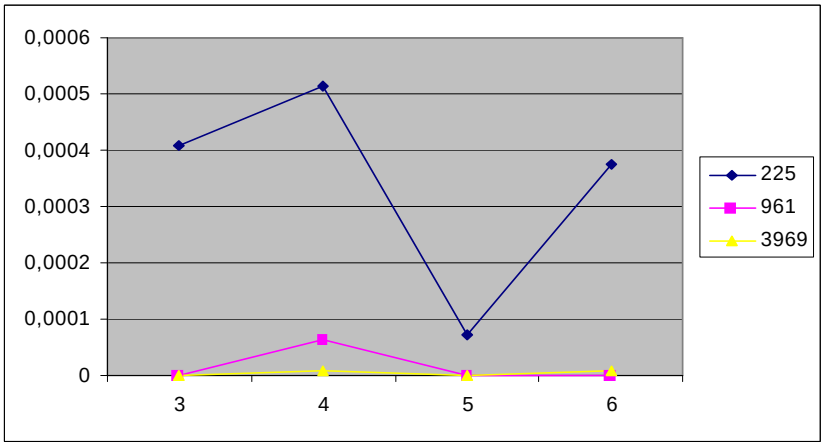
Rys. 6. Prawdopodobieństwo niewykrycia błędów przez algorytm LJS

Z rysunku 6 wynika, że wartości prawdopodobieństw zmniejszają się wraz ze wzrostem wielkości macierzy. W przypadku tego algorytmu wykresy są prawie liniowe. Po dodaniu innych cech do charakterystyki liniowość zanika.
Przykładowo: dla algorytmu TR wykres rozkładów prawdopodobieństw niewykrycia błędów cechuje się dużą różnicą wartości, względem krotności, dla mniejszych rozmiarów macierzy (rys. 7.).



Rys. 7. Prawdopodobieństwo niewykrycia błędów przez algorytm TR

Algorytm charakteryzuje się większą skutecznością wykrywania błędów nieparzystych krotności. Im jest większa macierz pamięci, tym są mniejsze różnice prawdopodobieństw niewykrycia poszczególnych krotności błędów.



Rys. 8. Prawdopodobieństwo niewykrycia błędów przez algorytm LJK

Analogiczna sytuacja występuje dla algorytmu LJK, charakteryzującego się największym prawdopodobieństwem wykrywania błędów pięciokrotnych (rys. 8.). Wraz ze wzrostem wielkości pamięci zmniejszają się różnice prawdopodobieństw, jak również samo prawdopodobieństwo niewykrycia błędów.

5. WNIOSKI

Z powyższych rozważań wynika, że wraz ze wzrostem wielkości kompaktowanego ciągu, efektywność zaproponowanych algorytmów nie będzie malała, ponieważ będzie malała liczba rozmieszczeń niewykrywalnych błędów. Efektywność metod można zwiększyć ingerując w stosowany test marszowy. Jeżeli jest wiadomo, w jakie krotności błędów są transformowane usterki, można tak skonstruować test, żeby usterki były transformowane w krotności wykrywanych błędów [11]. Można również sterować sposobem przedstawienia wynikowego testu w postaci macierzy tak, żeby nie powstawały topologie błędów trudne do wykrycia.

6. BIBLIOGRAFIA

[1] Gizopoulos D.: *Advances in Electronic Testing Challenges and Methodologies*, Springer, 2006.
[2] Hamdioui S.: *Testing Static Random Access Memories: Defects, Fault Models and Test Patterns*, Springer, 2004.
[3] Karpovsky M. G., Yarmolik V. N.: *Transparent random access memory testing for pattern sensitive faults*, Journal of Electronic Testing Volume 9, Number 3, pp. 251-266, DOI: 10.1007/BF00134690.
[4] Al-Harbi S.M., Noor F., Al-Turjman F.M: *March DSS: A New Diagnostic March Test for All Memory Simple Static Faults*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, pp.1713 – 1720, 2007.

[5] Hamdioui S., van de Goor A.J.: *Efficient Test for Realistic Faults in Dual-Port SRAMS*. IEEE Transactions on Computers, VOL. 51. NO. 5, 2002

[6] Busłowska E.: *Memory Testing in Two-Dimensional Space*, 14th International Conference on Systems Science, Vol. 3. Information Systems Technical Applications Non-Technical Applications, pp. 275-279, 2001.

[7] Busłowska E.: *Efektywność algorytmów kompresji danych używanych w procesie testowania*, Prace Naukowe Politechniki Radomskiej. Elektryka, Z. 2, s. 23-28, 2002.

[8] Busłowska E.: *Wykrywanie usterek pamięci RAM metodami kompakcji danych*, XI Warsztaty Naukowe PTSK, Symulacja w Badaniach i Rozwoju (2004), s. 549-554.

[9] Busłowska E., Yarmolik V.N., Fiedorowicz M.: *Kompakcja odpowiedzi układu na test*, 11th International Conference, Computer systems aided science, industry and transport, Vol.1, 2007.

[10] Busłowska E.: *Opracowanie metod i algorytmów kompakcji danych pozwalających na wykrywanie usterek pamięci RAM*, rozprawa doktorska, Wydział Informatyki, 2005.

[11] Al-Harbi S.M., Noor F., Al-Turjman F.M.: *March DSS: A New Diagnostic March Test for All Memory Simple Static Faults*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Volume: 26 , Issue:9, pp. 1713 - 1720, 2007.