Tomasz RUDNICKI
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI

Generator par testowych dla uszkodzeń opóźnieniowych

Dr inż. Tomasz RUDNICKI

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 2006 roku. Jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe obejmują zagadnienia związane z testowaniem uszkodzeń opóźnieniowych w układach cyfrowych, układami logiki programowalnej oraz systemami mikroprocesorowymi.

e-mail: tomasz.rudnicki@polsl.pl

Streszczenie

Rejestr MISR pobudzany słowami odczytywanymi z pamięci ROM jest jednym z ostatnio oferowanych rozwiązań problemu generacji par testowych dla uszkodzeń opóźnieniowych. W niniejszej pracy przedstawiono koncepcję zmniejszania liczby słów programujących oraz takiej modyfikacji grafu pracy ww. generatora par testowych, która pozwala na uzyskanie akceptowalnego czasu testowania przy stosunkowo wysokim współczynniku pokrycia uszkodzeń opóźnieniowych. W pracy przedstawiono rezultaty eksperymentów, w których wygenerowano opracowaną metodą pary testów dla benchmarków ISCAS'89.

Słowa kluczowe: MISR, pary testowe, pokrycie uszkodzeń, CUT, ROM.

Test Pattern Generator for Delay Faults

Abstract

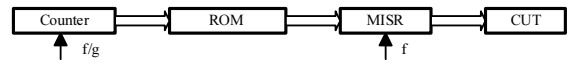
One of the recently proposed solutions to the problem generation of test pairs' patterns to target delay faults is a Multiple Input Signature Register (MISR). The paper proposes a method to minimize control words and to modify the operation diagram of the Test Pattern Generator (TPG) aiming at achieving acceptable test times while ensuring a very high coverage of delay faults. Experimental results are presented, in which the method of test pairs for benchmarks of the ISCAS'89 has been employed. These results confirm a high effectiveness of this method compared to other solutions.

Keywords: MISR, Test Pairs, Cover of Delay Faults, CUT, ROM.

1. Wprowadzenie

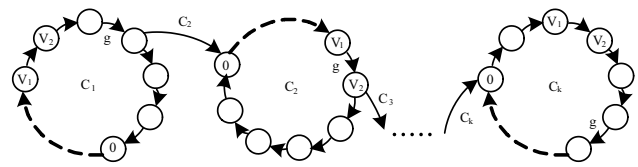
Wraz ze stałym wzrostem liczby tranzystorów zawartych w jednej obudowie pojawiają się większe niedokładności technologii wytwarzania takich układów scalonych. Niedokładności te są przyczyną fizycznych defektów, jakie pojawiają się podczas procesu produkcyjnego. Defekty takie jak: upływność pomiędzy wyprowadzeniami, zanieczyszczenia powierzchniowe, wilgoć mogą prowadzić do powstania uszkodzeń opóźnieniowych (ang. delay faults). Defekty tego typu mogą także powstawać podczas normalnej pracy układu na skutek jego starzenia się wymuszanego warunkami pracy układu takimi jak napięcie zasilania, temperatura, wilgotność czy też ciśnienie.

Do wykrywania uszkodzeń opóźnieniowych stosuje się wymuszenia testowe w postaci par złożonych z dwóch wektorów $P=\{V_1,V_2\}$. Wektor V_1 to wektor inicjalizujący testowany układ cyfrowy CUT, natomiast wektor V_2 to wektor generujący pożądany rodzaj zmiany stanu wybranego wyjścia układu testowanego (01 lub 10). Pożądany rodzaj zbocza pobudza ewentualne uszkodzenia opóźnieniowe i tworzy warunki do propagacji tych pobudzonych uszkodzeń do wyjścia CUT. Taka metoda testowania jest powszechnie znana jako testowanie dwu wektorowe (ang. Two-Pattern Testing - TPT) [1]. Na rys. 1 przedstawiono generator par testowych zbudowany w oparciu o rejestr MISR, pamięć ROM oraz licznik.



Rys. 1. Generator par testowych z pamięcią ROM
Fig. 1. Test Pattern Generator with a ROM

Bezpośrednim źródłem par testowych dla układu testowanego CUT jest rejestr MISR. Do przechowywania L_s n-bitowych słów programujących pracę rejestru MISR zastosowano pamięć ROM. Licznik służy do adresowania pamięci. Zaletą tej struktury jest skalowalność i niezależność od funkcji użytkowych układu CUT. W praktyce oznacza to, że wraz ze zmianą funkcji CUT struktura przedstawionego generatora par testowych nie zmienia się. Zmianie ulegają jedynie słowa programujące zapisane w pamięci ROM. Natomiast kolejność występowania słów w pamięci ROM nie ma znaczenia. Graf pracy takiego generatora par testowych przedstawiono na rys. 2.



Rys. 2. Graf pracy generatora par testowych
Fig. 2. The operation diagram of the TPG

Możliwości stosowania tej struktury do generacji par testowych były badane w [2, 3]. W pracach tych założono, że sprzężenie liniowe n-bitowego rejestru MISR zbudowane jest w oparciu o wielomian pierwotny gwarantujący 2^n –1 taktowy cykl pracy tego rejestru. Jednakże, dzięki licznikowi taktowanemu g razy wolniej niż rejestr MISR, ten ostatni pracuje w przypadku każdego słowa programującego C_i ze skróconym cyklem pracy, zawierającym tylko $g < 2^n$ –1 taktów. Zasadniczą zaletą takiego rozwiązania jest redukcja liczby taktów zegarowych koniecznych do wygenerowania pożądanych par testowych. Układ zaczyna pracę od wyzerowania rejestru MISR po czym przez g taktów zegarowych rejestr MISR pracuje ze słowem programującym C_i. Po g taktach zegarowych następuje zmiana stanu licznika adresującego pamięć ROM oraz następuje dostarczenie kolejnego słowa programującego do rejestru MISR i jego ponowne wyzerowanie. Cały proces powtarza się aż do wykorzystania wszystkich L_s niezbędnych słów programujących. Czas testowania można więc z grubsza określić jako iloczyn $g \cdot L_s$. Oczywiście słowa programujące muszą być dobrane w taki sposób, aby podczas g taktów sygnału zegarowego począwszy od stanu zerowego zostały wygenerowane przez rejestr MISR wszystkie pary testowe związane z danym słowem programującym.

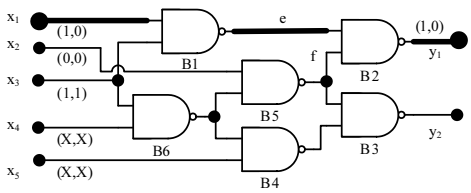
Wadą rozwiązań oferowanych w [2, 3] jest duża liczba L_s słów programujących przechowywanych w pamięci ROM oraz wynikająca z tego zbyt duża liczba taktów zegarowych $g \cdot L_s$ potrzebnych do realizacji testowania. Inną wadą jest mało elastyczny graf pracy generatora par testowych uniemożliwiający poszukiwanie lepszych rozwiązań.

Podstawowym celem pracy jest opracowanie sposobów redukcji liczby L_s słów programujących co pozwoli znacząco skrócić czas testowania oraz umożliwi zmniejszenie zajętości pamięci ROM. Do opracowania tych technik zostanie wykorzystana obecność stanów nieokreślonych w wyznaczonych przez systemy ATPG parach testowych oraz w związanych z nimi słowach programujących. Zaproponowano techniki sklejania par testowych. Sklejanie par testowych powoduje zmniejszenie liczby par testowych, a w związku z tym pośrednio wpływa na redukcję liczby słów programujących.

Innym ważnym celem naszej pracy jest modyfikacja podanego grafu pracy generatora par testowych, której efektem będzie dodatkowe skrócenie czasu testowania. Z przedstawionego na rys. 2 grafu pracy generatora wynika, że istnieją możliwości modyfikacji tego grafu pracy. Można wprowadzić założenie, że w każdym nowym g-taktowym cyklu stanem początkowym zamiast zera może być dowolny inny wybrany stan np. stan, w którym kończy się poprzedni cykl pracy rejestru MISR. Oznaczać to będzie w praktyce, że kolejność słów programujących będzie miała wpływ na liczbę wygenerowanych par testowych, a tym samym na liczbę pokrytych uszkodzeń opóźnieniowych. Tak więc stan początkowy S rejestru MISR, kolejność słów programujących oraz długość cyklu g będą miały bezpośredni wpływ na graf pracy rejestru MISR a pośrednio będą miały wpływ na całkowity czas testowania oraz współczynnik pokrycia uszkodzeń opóźnieniowych przez wygenerowane pary testowe. Sposób określenia tych trzech parametrów, a w szczególności podanie metody wyznaczania kolejki słów programujących będzie kolejnym celem pracy.

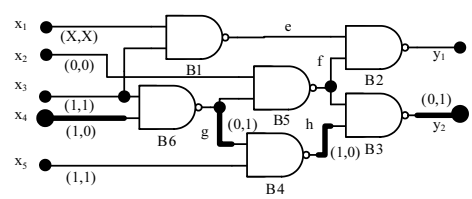
2. Sklejanie par testowych

Stany nieokreślone występujące w parach testowych pozwalają na zmniejszenie liczby słów programujących przechowywanych w pamięci ROM. W celu wyjaśnienia techniki sklejania par testowych wykorzystano przykład prostego benchmarku c17. Na rys. 3 przedstawiono sposób wyznaczania pary testowej P_1 powodującej pobudzenie ścieżki x_1, e, y_1 ze względu na opóźnienie opadającego zbocza na wyjściu y_1.



Rys. 3. Wyznaczanie pary testowej P_1
Fig. 3. Calculation of test pair P_1

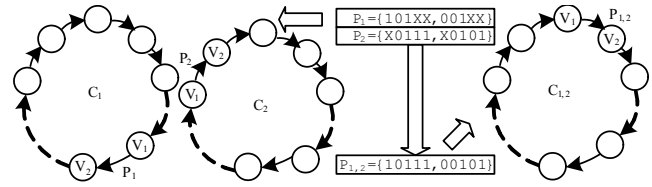
W celu uzyskania narastającego zbocza na wyjściu y_1 w ścieżce e musi występować narastające zbocze natomiast na wejściu x_1 zbocze opadające. Oczywiście wejście x_3 bramki B1 oraz ścieżka f powinny być stale w stanie wysokim. Zatem para testowa $P_1=\{V_1,V_2\}=\{101XX,001XX\}$ gdzie $V_1= V_2=\{x_1,x_2,x_3,x_4,x_5\}$.Na rys. 4 przedstawiono sposób wyznaczania pary testowej P_2 powodującej testowanie ścieżki x_4, g, h, y_2 ze względu na opóźnienie narastającego zbocza na wyjściu y_2.



Rys. 4. Wyznaczanie pary testowej P_2
Fig. 4. Calculation of test pair P_2

W celu uzyskania narastającego zbocza na wyjściu y_2 w ścieżce g musi również występować narastające zbocze natomiast na ścieżce h zbocze opadające. Oczywiście wejście x_5, x_3 oraz ścieżka f powinny być stale w stanie wysokim. Para testowa wynosi $P_2=\{X0111,X0101\}$.

Zauważmy jednak, że pary testowe $P_1=\{101XX,001XX\}$ i $P_2=\{X0111,X0101\}$ są wzajemnie zgodne, co umożliwia ich sklejenie w jedną parę testową $P_{1,2}=\{10111,00101\}$. Wyznaczanie sklejonej pary testowej $P_{1,2}$ przedstawiono na rys. 5.



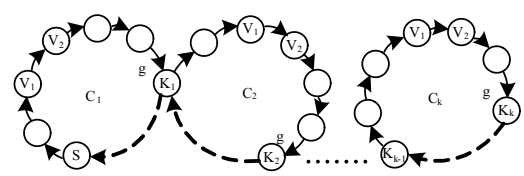
Rys. 5. Wyznaczanie sklejonej pary testowej $P_{1,2}$
Fig. 5. Calculation of the merged test pair $P_{1,2}$

Sklejenie par testowych P_1 oraz P_2 w jedną parę testową $P_{1,2}$ powoduje, że w ramach tylko jednego g-taktowego cyklu w dwóch kolejnych taktach zegarowych zostanie jednocześnie wygenerowana zarówno para testowa P_1 jak i para testowa P_2. To z kolei powoduje, że w tym samym czasie zostanie przetestowana ścieżka x_1, e, y_1 jak i ścieżka x_4, g, h, y_2. Tak więc zamiast 2·g taktów zegarowych oraz dwóch słów programujących potrzebny jest tylko g taktów zegarowych i tylko jedno słowo programujące $C_{1,2}$.

Do sklejania par testowych opracowano specjalny algorytm, który w pierwszej iteracji poszukuje par testowych, które można skleić z parą testową P_1. Jeśli istnieją takie pary wówczas skleja się je z parą testową P_1. Jednocześnie zapamiętana jest informacja o sklejonych parach testowych. W drugiej iteracji poszukuje się pierwszej wolnej niesklejonej do tej pory pary testowej, po czym sprawdza się czy ta para testowa daje się skleić z innymi niesklejonymi do tej pory parami testowymi. Jeśli znajdą się takie pary wówczas następuje proces sklejania z jednoczesnym zapamiętaniem informacji o sklejonych parach. W kolejnych iteracjach cały proces się powtarza aż do wykorzystania wszystkich par testowych.

3. Modyfikacje wprowadzone do grafu pracy generatora par testowych i wynikająca z tego metoda określania kolejki słów programujących

Umówmy się na wstępie, że rejestr MISR-C_i oznaczać będzie rejestr LFSR z negacjami wtrąconymi na wejściach tych stopni rejestru, które są wskazywane przez jedynki słowa programującego C_i. Jednym z celów pracy zasygnalizowanym we wstępie jest modyfikacja grafu pracy generatora par testowych wykorzystywanego w pracach [2, 3]. Jej spodziewanym efektem powinno być dodatkowe skrócenie czasu testowania. Istnieje kilka możliwości modyfikacji grafu pracy przedstawionego na rys. 2. Jedną z nich jest założenie, że rejestr MISR-C_i zaczyna swój pierwszy cykl pracy od stanu początkowego S niekoniecznie zerowego. Zakładamy także, że w każdym następnym g-taktowym cyklu pracy związanym z nowym słowem programującym, stanem początkowym rejestru MISR-C_i oprócz zera może być dowolny inny wybrany stan. Umówmy się, że będzie to stan, w którym kończy się poprzedni cykl pracy rejestru MISR. Oznaczać to będzie w praktyce, że kolejność słów programujących będzie miała wpływ na liczbę wygenerowanych par testowych, a tym samym na liczbę pokrytych uszkodzeń opóźnieniowych. Nowy wynikający z tych założeń graf pracy rejestru MISR przedstawiono na rys. 6.



Rys. 6. Graf pracy generatora par testowych
Fig. 6. Operation diagram of the TPG

W pierwszej kolejności rejestr MISR-C_1 ustawiany jest w stan początkowy S, po czym przez kolejnych g taktów zegarowych pracuje ze słowem programującym C_1. Po g taktach zegarowych

rejestr MISR-C_1 kończy pracę w stanie K_1. Następnie zaczyna pracę rejestr MISR-C_2, który pracuje przez g taktów zegarowych począwszy od stanu K_1, w którym zakończył pracę rejestr MISR-C_1. Po g taktach zegarowych rejestr MISR-C_2 kończy pracę w stanie K_2. Proces powtarza się aż do wykorzystania wszystkich słów programujących. W ogólnym przypadku można powiedzieć, że rejestr MISR-C_1 zaczyna pracę od określonego stanu S dla słowa programującego C_1, natomiast w przypadku pozostałych słów programujących C_i rejestr MISR-C_i zaczyna pracę od stanu początkowego K_{i-1}, w którym zakończył pracę rejestr MISR-C_{i-1}. Tak więc stan początkowy S rejestru MISR-C_1, kolejność słów programujących oraz długość cyklu g mają bezpośredni wpływ na graf pracy rejestru MISR a pośrednio mają wpływ na współczynnik pokrycia uszkodzeń opóźnieniowych przez wygenerowane pary testowe. Jak więc określić te trzy parametry, a w szczególności jaką metodę wyznaczania kolejki słów programujących zastosować, aby uzyskać akceptowalny czas testowania i akceptowalny współczynnik pokrycia uszkodzeń opóźnieniowych?

Najpierw zakładany jest wielomian pierwotny, opisujący liniowe sprzężenie zwrotne rejestru MISR. W kolejnym kroku redukuje się liczbę słów programujących. Jak już wcześniej wspomniano odbywa się to poprzez zastosowanie techniki sklejania par testowych, słów programujących lub obu technik zastosowanych jednocześnie. Po tym etapie przechodzi się do określenia, które ze słów programujących C_i będzie pierwszym słowem programującym (C_1). Następnie wyznacza się stan początkowy S, od którego będzie zaczynał pracę rejestr MISR-C_1 dla uprzednio wyznaczonego pierwszego słowa programującego C_1. Do tego celu zastosowano algorytm, który w sposób losowy dobiera słowo programujące C_1 oraz stan początkowy S. Na początku algorytmu losuje się spośród wszystkich słów programujących C_i to słowo, które będzie pierwszym słowem programującym (C_1). Następnie losowo wypełnia się stany nieokreślone w wektorze V_1 pary testowej związanej z tym słowem programującym. Jeśli tych par testowych jest więcej niż jedna, wówczas należy wybrać tylko jedną z nich. W ten sposób uzyskuje się stan początkowy $S = V_1$, od którego będzie zaczynał pracę rejestr MISR-C_1. To zapewnia wygenerowanie wybranej pary testowej związanej z tym słowem programującym. Po czym losuje się sposób wypełnienia stanów nieokreślonych występujących w uprzednio wylosowanym słowie programującym (C_1). Po wyznaczeniu słowa programującego C_1 oraz stanu początkowego S rejestru MISR-C1 określa się kolejkę pozostałych słów programujących. Dla L_s słów programujących liczba możliwych kombinacji kolejności słów programujących wynosi $L_s!$. Już dla małej liczby L_s liczba możliwych kombinacji kolejności słów programujących jest bardzo duża, co uniemożliwia sprawdzenie wszystkich wariantów. Dodatkowo dochodzi problem wypełniania stanów nieokreślonych występujących w słowach programujących. W związku z tym opracowano specjalny algorytm, który wyznacza odpowiednią kolejkę L_s słów programujących.

4. Wyniki eksperymentów dla benchmarków ISCAS'89

Zaprezentowaną w poprzednim rozdziale metodę poszukiwania właściwej kolejki słów programujących ze zredukowanego zbioru L_s tych słów zastosowano dla kilku układów testowych ISCAS'89. Do symulacji użyto komputer PC z procesorem PIV 3,0GHz oraz 512 RAM. Do opisu metody użyto język C++. W tabeli 1 przedstawiono wyniki (TR) uzyskane dla następującego zbioru benchmarków ISCAS'89 (c17, s27, s386, s1488, s1494, s298, s208). W tabeli 1 w celach porównawczych przedstawiono także wyniki uzyskane w pracy [2] dla tych samych zbiorów benchmarków i dokładnie tych samych par testowych. Pierwsza kolumna tabeli zawiera nazwę układu (Na), liczbę wejść (#In.), liczbę generowanych par testowych (#Pa) oraz liczbę wybranych taktów zegarowych (#g). Następne kolumny zawierają odpowiednio liczbę słów programujących (#Words), liczbę taktów zegarowych (#Time), stan początkowy S oraz liczbę nie wygenerowanych par testowych (#Left).

Tab. 1. Porównanie uzyskanych wyników z wynikami otrzymanymi w [2]
Tab. 1. The comparison between the obtained results and those reported in [2]

Na	#In	#Pa	#g	#Words	
				[2]	TR
c17	5	22	8		7
s27	7	32	16		7
			64	6	
			128	6	
s386	13	232	1024		74
			2048	49	62
			4096	48	58
s1488	14	738	1024	93	178
			2048		153
			4096		130
			16384	56	
s1494	14	725	512		185
			1024	100	178
			4096		138
			16384	58	
s298	17	177	64	19	42
			128	18	30
			256	17	25
			131072	15	
s208	18	209	256	50	44
			262144	32	

#Words		#Time		S		#Left	
[2]	TR	[2]	TR	[2]	TR	[2]	TR
	7		56		3		0
	7		112		34		1
6		384		0		15	
6		768		0		12	
	74		75776		1423		15
49	62	100352	126976	0	5153	133	5
48	58	196608	237568	0	3225	104	0
93	178	95232	182272	0	11251	562	39
	153		313344		7618		46
	130		532480		13804		11
56		917504		0		185	
	185		94720		15616		158
100	178	102400	182272	0	479	548	54
	138		565248		9325		13
58		950272		0		198	
19	42	1216	2688	0	59492	171	5
18	30	2304	3840	0	3442	171	7
17	25	4352	6400	0	16502	154	2
15		1966080		0		25	
50	44	12800	11264	0	71795	128	102
32		8388608		0		77	

Wyniki zawarte w tabeli 1 sygnalizują, że symulacja oparta o początkowy zbiór słów programujących uzyskanych techniką sklejania par testowych (TR) pozwala w porównaniu z wynikami przedstawionymi w [2] uzyskać, dla całego zbioru układów testowych, mniejsze liczby niewygenerowanych par testowych przy porównywalnych zbiorach słów programujących (#Words) do zbiorów uzyskanych w pracy [2]. Uzyskano to dzięki opracowaniu i wykorzystaniu algorytmu sklejającego pary testowe. Mniejsza liczba par testowych oznacza mniejszą liczbę L_s słów programujących, a co za tym idzie mniejszą zajętość reprogramowalnej pamięci ROM.

5. Literatura

[1] K. Furuya, E.J. McCluskey: Two-Pattern Test Capabilities of Autonomous TPG Circuits. IEEE International Test Conference, 1991.
[2] M. Keim, I. Polian, H. Hengster, B. Becker: A scalable BIST architecture for delay faults, European Test Workshop, 1999.
[3] I. Polian, B. Becker: Configuring MISR-Based Two-Pattern BIST Using Boolean Satisfiability, DDECS, 2003.

Artykuł recenzowany