Krzysztof FIRLĄG

WYKORZYSTANIE GENERATORÓW PSEUDOLOSOWYCH W SAMOTESTOWANIU URZĄDZEŃ STEROWANIA RUCHEM DROGOWYM

Streszczenie

Realizacja sterowników ruchu drogowego w formie urządzeń specjalizowanych z wykorzystaniem układów programowalnych wymaga stosowania nowoczesnych metod testowania tych urządzeń. W artykule do testowania sterowników zaproponowano wbudowane samotestowanie oraz odpowiednią architekturę BIST. Po przedstawieniu metod generacji sekwencji testowych i analizy odpowiedzi testowanego układu, przeanalizowano metody generowania pseudolosowych sekwencji testujących. Przedstawiono zastosowanie rejestrów LFSR oraz automatów komórkowych CA do generacji sekwencji pseudolosowych.

W pracy przeanalizowano zastosowanie rejestrów LFSR zarówno w budowie generatorów testów TPG jak i analizatorów odpowiedzi testowanego układu ORA, do generacji sygnatury testowanego układu. Proponowana architektura BIST została zaimplementowana w specjalizowanym sterowniku ruchu drogowego. Przeanalizowano wpływ struktury BIST na parametry sterownika.

WSTĘP

W ostatnich latach dostrzegalny jest znaczny rozwój technologiczny we wszystkich gałęziach życia, wynikający głównie z dynamicznego rozwoju technologii elektronicznych i informatycznych. Nieuniknionym następstwem powszechnego rozwoju technologicznego jest ciągłe doskonalenie urządzeń sterowania ruchem. Reprezentatywnym przykładałem urządzenia sterowania ruchem, podlegającemu takiej ewolucji, jest sterownik lokalny ruchu drogowego. Ewolucja tych urządzeń była typowa, od rozwiązań mechanicznych, poprzez rozwiązania cyfrowe na układach SSI i MSI, aż do chwili obecnej, gdzie praktycznie wszystkie współczesne rozwiązania bazują na układach mikroprocesorowych.

Kolejnym etapem rozwoju elektroniki było opracowanie układów programowalnych PLD (*Programmable Logic Device*). Umożliwiło to realizację urządzeń specjalizowanych, których działanie wynikało bezpośrednio ze sprzętowej struktury układu. Proces konfiguracji układów programowalnych porównywalny jest z programowaniem układów mikroprocesorowych. Proces projektowy, wraz z prototypowaniem urządzenia, można przeprowadzić bezpośrednio u projektanta urządzenia.

Producenci sterowników ruchu drogowego w ograniczony sposób wykorzystali układy programowalne, głównie do sprzętowej realizację wyodrębnionych algorytmów, jak np.: obsługa detektorów indukcyjnych; moduły video-detekcji. Wykorzystanie układów programowalnych do realizacji sterowników ruchu drogowego pozwoliłoby na budowę nowej klasy sterowników specjalizowanych. Metodę realizacji specjalizowanych sterowników ruchu drogowego w układach FPGA (*Field-Programmable Gate Array*) i wynikające z niej korzyści przedstawiono w pracach [4,9].

Wykorzystanie układów programowalnych do budowy urządzeń sterowania ruchem wymaga zagwarantowania odpowiedniego poziomu bezpieczeństwa urządzeń [8] i wiąże się z koniecznością ciągłego testowania poprawności ich pracy. Metody testowania specjalizowanych urządzeń sterowania ruchem realizowanych w układach FPGA przedstawiono w pracy [5]. W pracy [3] wskazano wbudowane samotestowanie jako najlepszą metodę wykrywania uszkodzeń powstałych w czasie użytkowania urządzenia w systemie sterowania. W artykule kontynuowana będzie analiza zagadnień samotestowania ze szczególnym uwzględnieniem metod wykorzystujących generatory wektorów pseudolosowych.

1. TESTOWANIE SPECJALIZOWANYCH URZĄDZEŃ STEROWANIA RUCHEM DROGOWYM

Urządzenia cyfrowe pomimo wieloetapowego procesu projektowania i realizacji, na końcu procesu produkcyjnego mogą nie spełniać postawionych im wymagań. Przyczyny tego zwykle przedstawia się jako [1,7]:

- błędna specyfikacja – rozbieżności pomiędzy działaniem wyspecyfikowanego urządzenia a założeniami, według których miało działać;
- błędna realizacja układu – wynika ze złej dokumentacji, błędnej implementacji;
- uszkodzenia realizacji – niewłaściwe prowadzenie procesu technologicznego, wadliwe podzespoły.

Uszkodzenia mogą też powstać w czasie użytkowania układu na skutek naturalnego procesu starzenia lub niekorzystnej zmiany warunków zewnętrznych. W celu wyeliminowania wymienionych błędów konieczna jest ciągła weryfikacja i testowanie urządzeń cyfrowych na każdym etapie ich wytwarzania, oraz diagnostyka w czasie użytkowania urządzeń.

Błędy na etapie specyfikacji mogą być weryfikowane bezpośrednio na poziomie systemu CAD, w którym nastąpiła specyfikacja. W poprzednich pracach przedstawiono kompletną metodę weryfikacji błędów specyfikacji urządzeń sterowania ruchem [5], wykorzystującą metody testowania oprogramowania w odniesieniu do języka VHDL, będącego platformą opisu urządzeń realizowanych w FPGA. Kod (VHDL) układu weryfikuje się zarówno w oparciu o funkcje urządzenia (testowania funkcjonalne) jak i strukturę programu (testowania strukturalne). Częściowa automatyzacja procesu testowania możliwa jest poprzez wykorzystanie automatycznych środowisk testowych (*Test Bench*). Jako ocenę jakości procesu weryfikacji wykorzystuje się analizę pokrycia strukturalnego.

Błędy oraz uszkodzenia realizacji, powstające podczas wytwarzania układu oraz uszkodzenia powstałe w czasie użytkowania układu można wykryć poprzez testowanie prototypu urządzenia. Testowanie na tym etapie może być przeprowadzone na poziomie pakietu oprogramowania integrującego w sobie narzędzia specyfikacji, implementacji i współpracującego z prototypem urządzenia, lub za pomocą systemów automatycznego testowania ATE (*Automated Test Equipment*).

Zależnie od typów układów programowalnych testowanie ich można podzielić na dwa rodzaje:
- testowanie wszystkich komórek układu i wszystkich połączeń (niezależne od aplikacji);
- testowanie logiki użytej do realizacji funkcji układu (zależne od aplikacji).

Pierwsze testowanie odnosi się do układów ASIC (*Application Specific Integrated Circuit*) i nie zaprogramowanych układów CPLD (*Complex PLD*) i FPGA. Testowanie to ma na celu wykrycie uszkodzeń struktury układu (test produkcyjny). Drugi rodzaj testowania odnosi się do zaprogramowanego układu FPGA. Polega on na testowaniu zaprogramowanej logiki działania układu (funkcji użytkownika). W tej metodzie głównie stosuje się wewnętrzne techniki testowania. Najczęściej stosowaną metodą jest wbudowane samotestowanie, wykorzystujące wbudowane elementy testowe BIST (*Built-In Self-Test*).

2. WBUDOWANE SAMOTESTOWANIE BIST

Wady i zalety samotestowania układów omówiono w [3]. Architektura prostego układu BIST przedstawiona została na rysunku nr 1.

Rys. 1. *Podstawowa struktura BIST*

W skład podstawowej struktury BIST wchodzą:
- testowany układ CUT (*Circuit Under Test*);
- generator sekwencji testowych TPG (*Test Patern Generator*);
- analizator odpowiedzi ORA (*Output Response Analizer*).

Zwykle BIST zawiera również multipleksery, pozwalające na odcięcie układu z systemu i poddaniu procesowi testowania oraz kontroler, nadzorujący proces testowania.

2.1. Metody generacji sekwencji testowych

Sekwencje testowe w TPG generowane są wg kilku metod, stosownie do własności układów poddawanym testowaniu. Generatory sekwencji testowych wykorzystują następujące koncepcje generacji wymuszeń [10]:
- deterministyczne – przeznaczone do wykrywania specyficznych błędów struktur, wykorzystuje się wzorce zapisane w pamięci ROM oraz liczniki do ich adresowania;
- algorytmiczne – podobne od deterministycznych, wykorzystuje się automaty FSM generujące sekwencję deterministyczne wg określonych algorytmów;
- wyczerpujące – wyczerpujące testowanie dla *n*–wejściowych układów kombinacyjnych z wykorzystaniem liczników, niepraktyczne przy większej ilości wejść;
- pseudowyczerpujące – dekompozycja układów kombinacyjnych na mniejsze, które są testowane wyczerpująco. Sprzętowo wykorzystuje się liczniki, rejestry LFSR lub automaty komórkowe CA;
- pseudolosowe – najczęściej stosowane w TPG, sekwencje są cykliczne i powtarzalne. Sprzętowo wykorzystuje się LFSR, CA;
- ważone (*weighted*) pseudolosowe – oparte na LFSR lub CA, wykorzystując bramki AND lub OR modyfikuje się prawdopodobieństwo wystąpienia jedynki lub zera na wyjściach TPG.

2.2. Metody analizy odpowiedzi układu

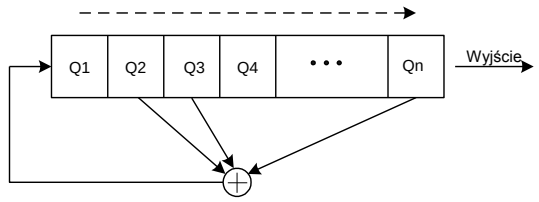
Analiza odpowiedzi układu testowanego na zasadzie komparacji całej odpowiedzi z wzorcem jest możliwa dla małych układów, gdyż wymaga bardzo dużo zasobów pamięci do przechowywania danych. W ORA stosuje się głównie metody polegające na poddawaniu kolejnych odpowiedzi kompakcji prowadzącej do otrzymania sygnatury – pojedynczego słowa binarnego, które ma jednoznacznie określoną wartość w przypadku układu sprawnego, natomiast przyjmuje inne wartości, jeśli w testowanym układzie jest uszkodzenie. Techniki redukcji informacji (kompakcji) w analizatorach odpowiedzi [10]:
- koncentracja odpowiedzi – przekształca sygnały z wielu wyjść na pojedynczy ciąg bitów. Sprzętowo wykorzystuje się kaskady bramek EX–OR;
- komparacja odpowiedzi – wykorzystuje komparatory porównujące odpowiedzi testowanego modułu z odpowiedziami wzorcowego układu lub z odpowiedziami zapisanymi w pamięci ROM. Sprzętowo oprócz komparatorów stosuje się przerzutniki do zapamiętania błędu komparacji;
- techniki licznikowe – zlicza się liczbę jedynek, zer lub zboczy w sygnale wyjściowym z testowanego układu. Stosuje się liczniki na wyjściach układu;
- analiza sygnatury – najpowszechniej stosowana technika, sygnatury tworzy się wykorzystując rejestry LFSR;
- akumulatory – wykorzystywane do wyznaczania sum kontrolnych;
- kontrola parzystości – wyznacza parzystość sumy bitów.

3. SPRZĘTOWE TECHNIKI GENEROWANIA SEKWENCJI PSEUDOLOSOWYCH

Analiza wykazała, że najczęściej stosowaną metodą testowania w BIST jest testowanie pseudolosowe, natomiast analiza odpowiedzi bazuje na analizie sygnatury. Do generacji wektorów pseudolosowych w procesie testowania można wykorzystać liniowe generatory kongruentne. Generatory te są deterministyczne, tzn. zainicjowane tym samym stanem początkowym dają zawsze te same sekwencje liczb pseudolosowych. Generatory pseudolosowe stosowane w BIST wykorzystują sprzętowo rejestry LFSR lub automaty komórkowe CA.

3.1. Podstawy działania rejestru LFSR

LFSR (*Linear Feedback Shift Register*) – rejestr przesuwający z liniowym sprzężeniem zwrotnym, jest rejestrem szeregowym z ciągiem odczepów (rys. 2). Najmniej znaczący bit jest kierowany na wyjście. Wszystkie bity są przesuwane o jeden w prawo. Najstarszy bit jest sumą modulo 2 bitów z odczepów (stanów kilku komórek tworzących rejestr).



Rys. 2. *Budowa rejestru LFSR*

Każdy LFSR jest związany z określonym wielomianem *P(x)* z pierścienia wielomianów nad ciałem skończonym $_p$. Dwuelementowym ciałem skończony $_2$ nazywany jest zbiór elementów *{0,1}* wraz z sumą modulo 2 (⊕) oraz iloczynem modulo 2 (⊙) zdefiniowanymi następująco [6]:

Tab. 1. *Suma i iloczyn modulo 2*

⊕	**0**	**1**
0	0	1
1	1	0

⊙	**0**	**1**
0	1	0
1	0	1

Wielomianem nad ciałem $_2$ nazywa się wyrażenie [6]:

$$P(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \qquad (1)$$

gdzie:

$a_{0\ldots}a_n \in \{0,1\}$ – elementy ciała $_2$;

x – zmienna;

$a \in N$.

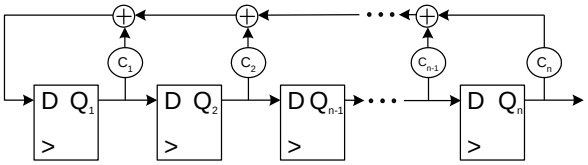
Jeżeli $a_n \neq 0$ wtedy liczba n jest stopniem wielomianu $deg(p_x)$. Wielomianem nierozkładalnym nazywa się taki wielomian $P(x)$ nad ciałem $_2$, który jest podzielny tylko przez wielomian $P(x)$ lub 1 będące elementem ciała $_2$. Okresem wielomianu nierozkładalnego $P(x)$ nazywa się najmniejszą liczbę całkowitą k taką, że dwumian x^k+1 jest podzielny przez $P(x)$. Wielomianem pierwotnym nazywa się wielomian nierozkładalny $P(x)$ stopnia n, którego okres $k=2^n-1$.

Dla potrzeb testowania wykorzystuje się rejestry LFSR o maksymalnej długości cyklu (maksymalny okres), posiadające wielomiany pierwotne nad $_2$. Sekwencja wektorów generowana przez LFSR jest okresowa, okres zależy od:

– stanu początkowego – niedozwolony jest stan początkowy $00\ldots0$, gdzie okres $k=1$; dla każdego innego okres $k=2^n-1$;
– struktury sprzężenia zwrotnego.

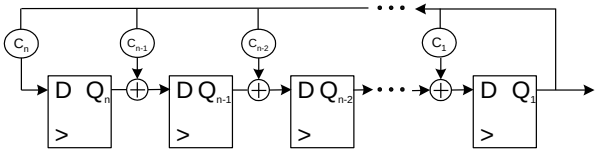
3.2. Typy rejestrów przesuwających LFSR

Wyróżnia się dwa główne typy rejestrów przesuwających różniące się sposobem realizacji sprzężenia zwrotnego. Pierwszym z nich jest rejestr przesuwający z zewnętrznym liniowym sprzężeniem zwrotnym, gdzie sygnały z wybranych odczepów sumowane są modulo 2 i podawane na wejście rejestru (rys. 3).



Rys. 3. *LFSR z zewnętrznym sprzężeniem zwrotnym [1]*

W drugim typie rejestru z wewnętrznym liniowym sprzężeniem zwrotnym sygnał z wyjścia rejestru poprzez sumę modulo 2 wprowadzany jest na wejścia wybranych przerzutników (rys. 4). Czasami wykorzystuje się rejestry o mieszanym sprzężeniu (zewnętrzno-wewnętrznym) [6].



Rys. 4. *LFSR z wewnętrznym sprzężeniem zwrotnym [1]*

Sekwencje binarne generowane przez LFSR opisuje wielomian charakterystyczny:

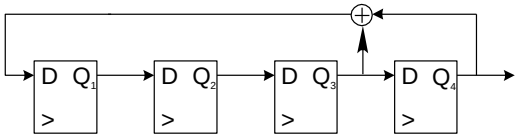
$$P(x) = 1 + C_1 x + C_2 x^2 + \cdots + C_n x^n \qquad (2)$$

gdzie:

$C=0$ – gdy nie ma sprzężenia z przerzutnika Q_i;

$C=1$ – gdy jest sprzężenie z przerzutnika Q_i.

Przykładowy rejestr z zewnętrznym sprzężeniem zwrotnym czwartego stopnia $deg(p_x)=4$ przedstawiony jest na rysunku 5.



Rys. 5. *LFSR z zewnętrznym sprzężeniem zwrotnym czwartego stopnia*

Rejestr LFSR z rysunku 5 opisany jest wielomianem charakterystycznym:

$$P(x) = 1 + x^3 + x^4 \qquad (3)$$

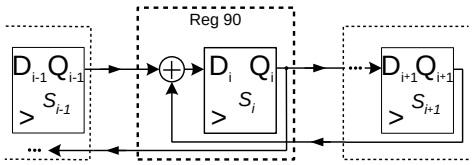
Wielomian ten jest pierwotny, w związku z tym jest to rejestr o maksymalnym okresie wynoszącym $2^n-1=15$. Dąży się, żeby ilość sprzężeń tworzących rejestry o maksymalnym okresie była jak najmniejsza, Zmniejsza to ilość logiki kombinacyjnej niezbędnej do budowy rejestru, czyli zasoby układu. Do budowy takich LFSR wykorzystuje się tablice charakterystycznych wielomianów pierwotnych o minimalnej ilości sprzężeń [11].

Cechy rejestrów LFSR w zastosowaniu do testowania pseudolosowego, to:

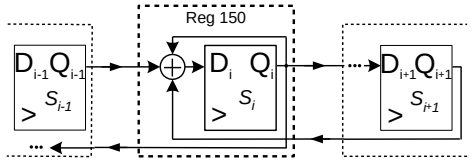
– prosta implementacja sprzętowa;
– mało zasobów kombinacyjnych;
– szybkość działania;
– powtarzalność sekwencji.

3.3. Automaty komórkowe CA

Automaty komórkowe CA (*Cellular Automata*), podobnie jak LFSR, generują sekwencję pseudolosowe. Ze względu na swoją strukturę nazywane są również LHCA (*Linear Hybrid Cellular Automata*). Komórki automatu komunikują się tylko z sąsiednimi komórkami, bez długich linii jak w LFSR. Relacje poszczególnych komórek opisują „reguły". Najbardziej popularne są automat komórkowe zbudowane z dwór typów komórek, o regule 90 i regule 150 przedstawione odpowiednio na rysunku 5 i rysunku 6.



Rys. 5. *Automat komórkowy – komórka o regule 90 [10]*



Rys. 6. *Automat komórkowy – komórka o regule 150 [10]*

Zasadę działania reguł 90 i 150 opisują równania:

$$\text{Reguła 90:} \quad S'_i = S_{i-1} \oplus S_{i+1} \qquad (4)$$

$$\text{Reguła 150:} \quad S'_i = S_{i-1} \oplus S_i \oplus S_{i+1} \qquad (5)$$

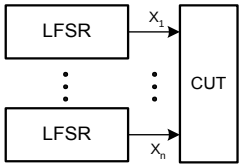
Z komórek o regule 90 i 150 można zbudować automat wytwarzające sekwencję o maksymalnej długości, posiadające okres o długości $k=2^n-1$ dla każdego innego stanu początkowego niż $00\ldots0$. Tablicę automatów komórkowych o minimalnym koszcie, zbudowanych z jednej lub dwóch komórek 150 a pozostałych o regule 90 przedstawiono w [2].

Cechy automatów komórkowych w zastosowaniu do testowania pseudolosowego, to:
- lepsza pseudolosowość niż LFSR;
- brak efektu przesuwania bitów na wyjściach;
- powtarzalność sekwencji;
- większa ilość zasobów kombinacyjnych (EX-OR) niż w LFSR.

4. WYKORZYSTANIE REJESTRÓW LFSR W BIST

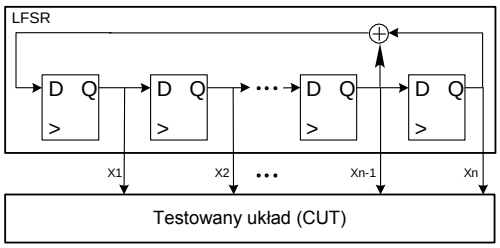
4.1. Generator sekwencji testowej

Najprostszy wariant wykorzystania rejestru LFSR do podania na wejścia testowanego układu sekwencji pseudolosowej przedstawiono na rysunku 7. Rozwiązanie to nie jest efektywne, gdyż przy wielu wejściach układu wymaga wielu rejestrów LFSR, co wiąże się z dużą liczbą dodatkowych przerzutników.



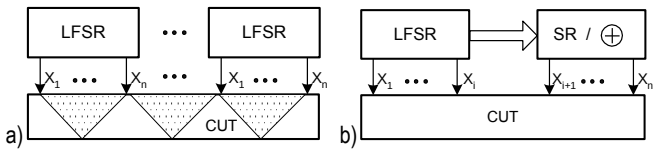
Rys. 7. *Szeregowe podłączenie rejestrów LFSR do wejść układu*

Znacznie lepszym rozwiązaniem jest wykorzystanie jednego rejestru LFSR, liczbą przerzutników odpowiadającego ilości wejść układu. W rozwiązaniu tym z wyjścia każdego przerzutnika wprowadzony jest sygnał na wejście testowanego układu (rys. 8).



Rys. 8. *Równoległe podłączenie rejestru LFSR do wejść układu*

W przypadku gdy liczba wejść testowanego układu jest duża (powyżej 25) wykorzystanie jednego rejestru jest niekorzystne ze względu na duży okres rejestru. Stosowane wówczas rozwiązania obejmują dekompozycję układu na części i testowaniu współbieżne każdej z nich oddzielnym rejestrem (rys. 9a). W celu zmniejszenia wykorzystania zasobów kombinacyjnych można zastosować rozwiązanie przedstawione na rysunku 9b, łączące rejestr LFSR o krótszym okresie niż liczba wejść układu z rejestrem przesuwającym SR. Z kolei chcąc zmniejszyć ilość wykorzystywanych zasobów pamięciowych stosuje się rozwiązania łączące rejestr z układem kombinacyjnym zawierającym bramki EX-OR.

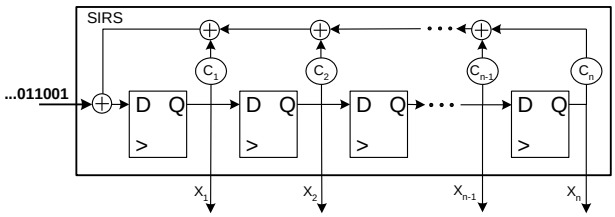


Rys. 9. *Testowania układów o dużej liczbie wejść: a) zastosowanie kilku LFSR; b) LFSR i SR lub EX-OR [10]*

Rejestr LFSR charakteryzuje się tym, że na każdym jego wyjściu prawdopodobieństwo wystąpienia jedynki w cyklu jest stałe $p(x_i=1)=0,5$. Czasami model błędów układu wskazuje, że większe pokrycie błędów dadzą wektory testowe z inną częstością występowania jedynek w cyklu. W takim przypadku zastosowanie znajdują *weighted* LFSR, gdzie poprzez dodatkowe bramki OR lub NAND uzyskuje się różne prawdopodobieństwa wystąpienia jedynki w cyklu.

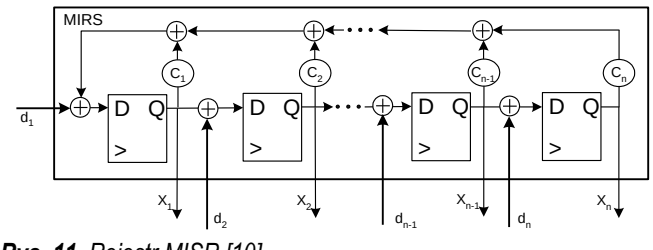
4.2. Analizator odpowiedzi układu

Wykorzystanie rejestrów LFSR w analizatorach odpowiedzi sprowadza się do tworzenia za ich pomocą *n*-bitowej sygnatury, kompaktującej odpowiedzi z wyjść testowanego układu. Najprostsze zastosowanie rejestru LFSR do tworzenia sygnatury przedstawiono na rysunku 10. Rejestr z jednym wejściem danych nazywamy SIRS (*Single Input Signature Register*). W SISR dopuszczalny jest stan początkowy wszystkich przerzutników równy 0. Sygnał z testowanego układu wpuszczany jest w pętle sprzężenia rejestru LFSR i wpływa na zawartość rejestru umożliwiając uzyskania unikalnej *n*-bitowej sygnatury testowanego układu. Pojedyncze błędy w strumieniu danych wejściowych są zawsze wykrywane (wpływają na zmianę sygnatury). Właściwość ta nie zależy od długości sekwencji wejściowej oraz od momentu powstania błędu. Błędy podwójne będą niewykryte, jeżeli pierwszy błąd jest oddzielony od drugiego dokładnie o okres rejestru LFSR. Prawdopodobieństwo maskowania błędów przez *n* stopniowe rejestry szeregowe LFSR dla długich sekwencji jest niezależne od rodzaju wielomianów charakterystycznych i wynosi $p(mask) \approx 2^{-n}$.



Rys. 10. *Rejestr SIRS [10]*

Sygnał na wejściu rejestru może być kompaktowany z wielu wyjść testowanego układu poprzez bramki EX-OR lub można zastosować SIRS dla każdego wyjścia układu. Rozwiązanie to wymaga jednak dużo sprzętu. Lepszym rozwiązaniem jest zastosowanie rejestru MISR (*Multiple Input Signature Register*) przedstawionego na rysunku 11. Rejestr powinien długością odpowiadać ilości wyjść układu testowanego lub też można kompaktować kilka wyjść układu na jedno wejście MISR. W MISR dopuszczalny jest stan początkowy wszystkich przerzutników równy 0. Sygnał z wyjść testowanego układu wpuszczany jest w pętle sprzężenia rejestru LFSR i wpływa na zawartość rejestru umożliwiając uzyskania unikalnej *n*-bitowej sygnatury testowanego układu.

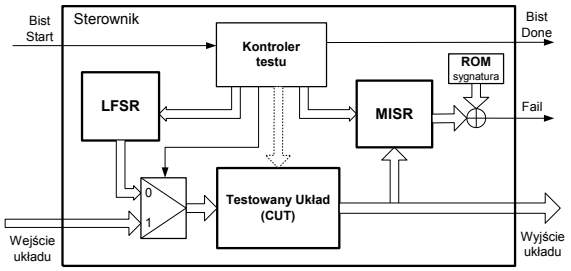


Rys. 11. *Rejestr MISR [10]*

5. IMPLEMENTACJA ARCHITEKTURY BIST

Po analizie metod wykorzystania generatorów pseudolosowych w architekturze BIST, zaproponowano implementację architektury przedstawionej na rysunku 12. Jest to implementacja techniki testowana w czasie bezczynności układu (*off-line*). Zaproponowana architektura zawiera układ multipleksera umożliwiający odseparowanie badanego układu od reszty systemu, LFSR pełniący funkcję

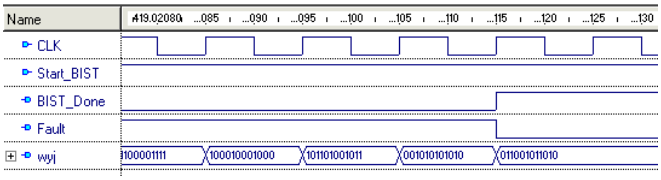
generatora sekwencji testowych, MISR umożliwiający otrzymanie sygnatury układu, układ porównujący otrzymaną sygnaturę z poprawną oraz układ kontrolujący przebieg testu.



Rys. 12. *Zastosowana architektura BIST*

Do badań wykorzystano model specjalizowanego sterownika ruchu drogowego wyspecyfikowanego w języku VHDL. Sterownik posiada 10 detektorów ruchu pojazdów i pieszych oraz obsługuje 6 grup sygnałowych (trzy kołowe, dwie piesze i tramwajową). Specyfikację architektury BIST dla sterownika wykonano w programie Active-HDL (rys. 13). Testowanie uruchamia sygnał *Bist Start*.

Do generacji wektorów testowych zastosowano rejestr LFSR o sprzężeniu zewnętrznym, zbudowany z 11 przerzutników. Rejestr opisany jest wielomianem charakterystycznym $P(x)=1+x^9+x^{11}$. Jest to wielomianem pierwotny – cykl rejestru wynosi 2047. Rejestr MISR zbudowany jest z 12 przerzutników, opisany wielomianem charakterystycznym $P(x)=1+x^6+x^8+x^{11}+x^{12}$, który również jest wielomianem pierwotnym.



Rys. 14. *Fragment weryfikacji funkcjonalnej sterownika*

Testowanie przeprowadzono generując ~4,2 milionów wektorów pseudolosowych (2047 cykli rejestru LFSR). Po ukończeniu testu wystawiany jest sygnał *Bist Done* (*"1"*). Otrzymana w wyniku testowania sygnatura porównywana jest z prawidłową („*011001011010*") a wynik porównania wystawiany na wyjście *Fail*, gdzie sygnał „*0*" oznacza zgodność otrzymanej sygnatury z wzorcową (rys. 14). Czas trwania testu, przy zegarze 10 [MHz] wyniósł 419 [ms].

Implementacji sterownika dokonano w układ FPGA xc3s50vq100. W tabeli 2 przedstawiono wykorzystanie zasobów wykorzystanego układu przez zaimplementowaną architekturę sterownika. Tabela przedstawia porównanie zasobów samego sterownika oraz sterownika z logiką BIST. Implementacja BIST zwiększyła ilość zasobów wykorzystanych przez sterownik w granicach 5 – 15%, jednocześnie bez znaczącego spadku prędkości działania układu.

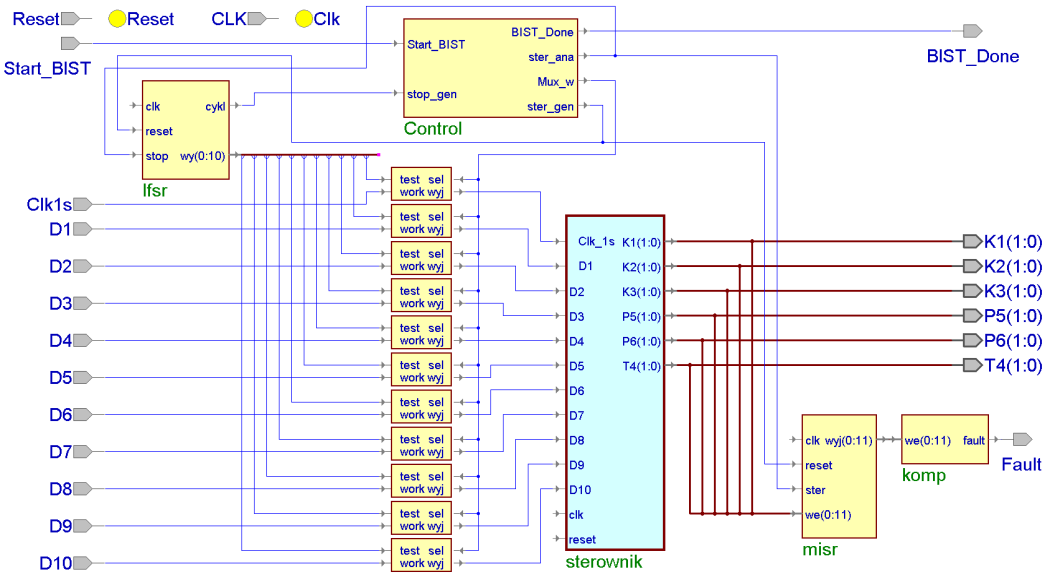
Tab. 2. *Wykorzystanie zasobów układu FPGA*

Zasoby układów	Sterownik FPGA: xc3s50vq100	Sterownik z BIST FPGA: xc3s50vq100
Wyprowadzeń WE/WY	23/63 (36%)	26/63 (40%)
Bloków Slice	306/768 (39%)	328/768 (42%)
Komórek LUT 4 wejściowych	539/1536 (35%)	580/1536 (37%)
Przerzutników	168/1536 (10%)	218/1536 (14%)
Maksymalna częstotliwość pracy	259,336 MHz	238,550 MHz

PODSUMOWANIE

Przeprowadzona analiza metod generowania sekwencji pseudolosowych stosowanych w BIST doprowadziła do wyboru metod bazujących na rejestrach LFSR. W generatorach sekwencji testowych stosuje się rejestry LFSR o maksymalnej długości cyklu, bazujące na wielomianach pierwotnych, umożliwiające generację sekwencji o dobrych parametrach pseudolosowości. Rejestry LFSR wykorzystuje się również w analizatorach odpowiedzi testowanego układu, do generowania unikalnej sygnatury z odpowiedzi układu, która po porównaniu z sygnaturą wzorcową informuje o poprawności działania układu.

Zaproponowano architekturę BIST do testowania specjalizowanych układów sterowania ruchem. Wykorzystano technikę testowana w czasie bezczynności układu (*off-line*), technika ta pozwala na wykrycie uszkodzeń, których wykrycie podczas normalnej pracy układu byłoby niezwykle trudne. Architektura BIST zawiera rejestr LFSR generujący sekwencję losową, rejestr MISR do tworzenia sygnatury testowanego sterownika ruchu oraz kontroler testu. Całość wyspecyfikowano w programie Active-HDL oraz zaimplementowano i uruchomiono w zestawie uruchomieniowym z układem



Rys. 13. *Specyfikacja architektury BIST w programie Active-HDL*

rodziny Spartan3. Testowanie modelu jak i prototypu sterownika z wykorzystaniem BIST przebiegło pomyślnie.

Rozwiązania oparte na LFSR charakteryzują się małym wykorzystaniem zasobów sprzętowych i dużą prędkością działania, przy małym prawdopodobieństwie maskowania błędów.

BIBLIOGRAFIA

1. Bushnell M. L., Agrawal V. D., Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Kluwer Academic Publishers, New York, 2002.
2. Cattell K., Zhang S., Minimal Cost One-Dimensional Linear Hybrid Cellular Automata of Degree Through 500, Journal of Electronic Testing: Theory and Applications, 6, 255-258, Kluwer Academic Publishers, Boston, 1995.
3. Firląg K., Metody samotestowania specjalizowanych urządzeń sterowania ruchem drogowym. Logistyka nr 4/2014, str. 1825-1834, Instytut Logistyki i Magazynowania, Poznań, 2014.
4. Firląg K., Projektowanie i realizacja specjalizowanych sterowników ruchu drogowego w reprogramowalnych strukturach logicznych, Politechnika Warszawska, Prace Naukowe - Transport, z.77, str. 27-44, OWPW, Warszawa, 2011.
5. Firląg K., Testowanie specjalizowanych urządzeń sterowania ruchem drogowym w strukturach FPGA, Politechnika Warszawska, Prace Naukowe - Transport, z.95, str. 115-124, OWPW, Warszawa, 2013.
6. Hławiczka A., Rejestry liniowe – analiza, synteza I zastosowanie w testowaniu układów cyfrowych. Zeszyty Naukowe Politechniki Śląskiej, Wydawnictwo Politechniki Śląskiej, Gliwice, 1997.
7. Jha N. K., Gupta S., Testing of digital systems, Cambridge University Press, 2003.
8. Kawalec P., Firląg K., Reliability analysis of specialized traffic control devices. Archives of transport, volume 19, issue 1-2, str. 75-82, Warszawska Drukarnia Naukowa PAN, Warszawa, 2007.
9. Kawalec P., Firląg K., Synteza specjalizowanych układów sterowania ruchem drogowym w strukturach FPGA. Pomiary Automatyka Kontrola nr 7 bis'2006, str. 8 – 10, Agenda Wydawnicza Stowarzyszenia SIMP, Warszawa, 2006.
10. Stroud C. E., A Designer's Guide to Built-In Self-Test. Kluwer Academic Publishers, 2002.
11. Ward R., Molteno T., Table of Linear Feedback Shift Registers. Electronics technical report No. 2012-1, Electronics Group, University of Otago, 2012.

THE USE OF PSEUDO-RANDOM NUMBER GENERATORS IN SELF-TESTING OF ROAD TRAFFIC CONTROL DEVICES

Abstract

Realization of road traffic controllers in the form of specialized devices with the use of programmable circuits requires application of modern testing methods of these devices. The author of the present paper proposes a built-in self-test (BIST) and a proper BIST architecture for controller testing. After presenting the methods of test sequence generation and the analysis of the tested circuit responses, the methods of pseudo-random test sequences have been analyzed. The application of linear-feedback shift registers (LFSR) and cellular automata (CA) for pseudo-random sequence generation has been presented.

The paper analyzes the application of LFSR registers both in the construction of test pattern generators (TPG) as well as in output response analyzers (ORA) of the tested circuit for generation of the tested circuit signature. The proposed BIST architecture has been implemented in a specialized road traffic controller. The influence has been analyzed of the BIST structure on the parameters of controller operation.

Autorzy:
dr inż. **Krzysztof Firląg** – Wydział Transportu Politechniki Warszawskiej, Zakład Sterowania Ruchem, kfr@wt.pw.edu.pl