

DIAGNOSTYKA SYSTEMÓW KOMPUTEROWYCH

WYKŁADY

OPRACOWALI:

**dr inż. Zbigniew Zieliński
dr inż. Jan Chudzikiewicz**

Literatura podstawowa

Autor	Tytuł	Rok	Sygnatura WAT
Kulesz R.	<i>Podstawy diagnostyki sieci logicznych i komputerowych.</i>	2000	
Holland R.	<i>Testowanie i diagnostyka systemów mikrokomputerowych</i>	1993	II - 80326
Sowiński A.	<i>Automatyczne testowanie w mikroelektronice.</i>	1991	50038
Sapiecha K.	<i>Testowanie i diagnostyka systemów cyfrowych.</i>	1987	48481
Hedtke R.	<i>Systemy mikroprocesorowe.</i>	1987	48640

Literatura uzupełniająca

Hasan Ural, Probert R. L., Gregor von Bochmann	<i>Testing of Communicating Systems Tools and Techniques.</i>	2000	
William R. S., John W.	<i>System Test and Diagnosis.</i>	1994	

POJĘCIA OGÓLNE

Diagnostyka techniczna - dziedzina wiedzy obejmująca całokształt zagadnień teoretycznych i praktycznych związanych z obiektem technicznym, ujmowanym w otoczeniu w jakim on występuje, w celu identyfikacji jego stanu.

Działalność diagnostyczna - działalność obejmująca opracowanie metod diagnostycznych, przygotowanie i realizację diagnozowania, weryfikację metod i opracowanie genezy, diagnozy i prognozy.

System diagnostyczny - zbiór elementów i relacji między nimi, występujących w procesie diagnozowania.

Proces diagnozowania - ciąg działań zawierających badania i wnioskowanie diagnostyczne w celu sformułowania diagnozy.

Diagnoza techniczna - rezultat procesu diagnozowania, zawierający określenie stanu technicznego obiektu diagnozowania.

Symptom diagnostyczny - informacja pozwalająca wnioskować o właściwościach obiektu technicznego.

Sygnał diagnostyczny - sygnał generowany przez badany obiekt techniczny, wykorzystywany w diagnozowaniu.

POJĘCIA DOTYCZĄCE OBIEKTU DIAGNOZOWANIA

Model obiektu diagnozowania - sformalizowany opis obiektu technicznego, niezbędny do diagnozowania.

Stan zdatności obiektu diagnozowania - stan techniczny, w którym obiekt może realizować zadania zgodne z wymaganiami, przy określonym oddziaływaniu otoczenia.

Stan niezdatności obiektu diagnozowania - stan techniczny, w którym obiekt nie może realizować zadań zgodnie z wymaganiami, przy określonym oddziaływaniu otoczenia.

Parametr stanu obiektu - wyróżniona wartość wielkości opisującej stan obiektu technicznego.

Relacja diagnostyczna - relacja przyporządkowująca symptomowi cechę lub cechy stanu (stanów) obiektu technicznego.

Przez pojęcie **system cyfrowy** - rozumieć będziemy złożony układ cyfrowy, przy czym złożoność układu zależna jest od poziomu abstrakcji wymaganej do opisania w sposób kompletny jego operacji.

Poziomy (abstrakcji) przetwarzania informacji w systemie cyfrowym

Sterowanie	Dane	Poziom
Wartości logiczne ("0", "1") lub ich sekwencje		Logiczny
Wartości logiczne	Słowa (bajty)	Rejestrów
Rozkazy	Słowa	Rozkazów
Programy	Struktury danych	Programów
Wiadomości (komunikaty)		Systemowy

- **Diagnozowanie systemu cyfrowego** - jest to proces ustalania stanu niezawodnościowego systemu z ewentualnym wskazaniem miejsca (lokalizacja) i przyczyn występowania niezdatności w przypadku jej ukrycia lub obecności.

Najczęstszym sposobem diagnozowania systemu jest **testowanie**.

- **Testowanie systemu cyfrowego** polega na wymuszeniu na jego wejściu sekwencji stanów, zwanych testami, a następnie sprawdzeniu, czy reakcja układu na tę sekwencję jest zgodna z oczekiwana.

Jeżeli system cyfrowy zachowuje się niezgodnie z oczekiwaniemi, kolejnym krokiem (celem) jest diagnoza przyczyny i lokalizacja niezdatności.

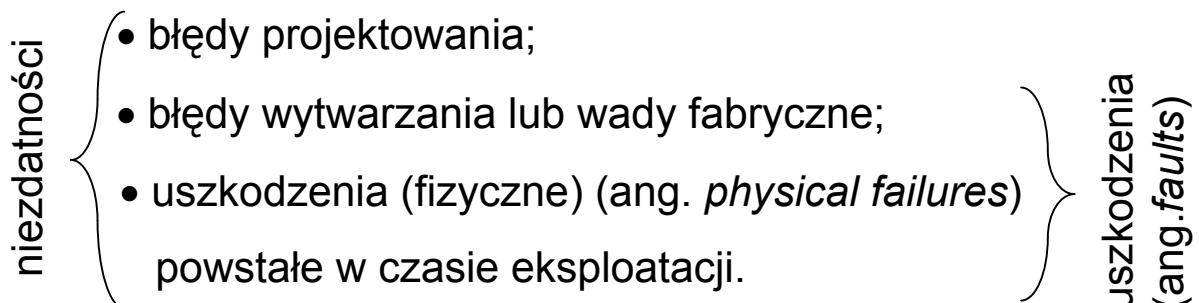
Rodzaje testowania		
Kryteria	Cechy metody testowania	Terminologia
Kiedy jest wykonywane diagnozowanie?	<ul style="list-style-type: none"> • równolegle z normalnym działaniem systemu • jako oddzielną działalność 	<ul style="list-style-type: none"> • współbieżne testowanie • testowanie (off - line)
Gdzie znajduje się źródło wymuszeń?	<ul style="list-style-type: none"> • wewnętrz systemu testowanego • na urządzeniu zewnętrznym (tester) 	<ul style="list-style-type: none"> • samotestowanie • zewnętrzne testowanie
Cel testowania	<ul style="list-style-type: none"> • błędy projektowe • błędy wytwarzania • defekty fabryczne • uszkodzenia fizyczne 	<ul style="list-style-type: none"> • testowanie weryfikacyjne • testowanie akceptacyjne • test zapewnienia jakości • testowanie eksploatacyjne
Rodzaj testowanego obiektu	<ul style="list-style-type: none"> • układ scalony • płyta (pakiet) • system 	<ul style="list-style-type: none"> • testowanie na poziomie komponentu • testowanie pakietu • testowanie na poziomie systemu
Sposób wyznaczania wymuszeń i/lub oczekiwanych odpowiedzi	<ul style="list-style-type: none"> • uzyskiwane z pamięci • generowane w czasie testowania 	<ul style="list-style-type: none"> • testowanie z zapamiętanymi wzorcami • testowanie algorytmiczne • testowanie porównawcze
Porządek zastosowania wymuszeń	<ul style="list-style-type: none"> • ustalony • zależny od rezultatów 	<ul style="list-style-type: none"> • testowanie adaptacyjne

- **Błąd** systemu cyfrowego (ang. *error*) - przypadek niepoprawnej operacji systemu objawiający się zniekształceniem obserwowlanego (wyniku).

Pojęcie błędu ma różne znaczenie na różnych poziomach przetwarzania informacji systemu komputerowego:

- na poziomie programu testowego błąd może objawiać się jako niepoprawny wynik operacji arytmetycznej;
- na poziomie kontroli logicznej układów (sekwencji bitów) - błąd oznacza niepoprawną wartość binarną;

Przyczyną błędów systemu cyfrowego mogą być:



- Uszkodzenie układu - przekroczenie przez pewien parametr tego układu dopuszczalnych dla niego tolerancji lub zniekształcenie struktury (logicznej) układu.
- Uszkodzenia mają charakter fizyczny (defekt obudowy, warstw tlenkowych, złączy, połączeń itp.), jednakże przeważająca większość uszkodzeń układów scalonych uzewnętrznia się w postaci błędów (funkcji logicznej spełnianej przez układ) - uszkodzenia logiczne.

Podział uszkodzeń

➤ Kryterium szkodliwości:

- katastroficzne – uniemożliwiają całkowicie eksploatację systemu. W wyniku ich wystąpienia nie mogą być poprawnie realizowane przez system cyfrowy żadne zadania;
- drugorzędne – umożliwiają wykonywanie przez system zadań, niektórych błędnie, podstawowe mechanizmy systemu cyfrowego znajdują się w stanie zdolności (np. mechanizm pobierania i dekodowania rozkazów, przesyłania danych do/z podzespołów wej/wyj, itp.).

➤ Kryterium czasu trwania:

- trwałe (ang. *permanent*) - zawsze obecne od momentu powstania a skutki ich wystąpienia są niezmienne w czasie. Można je wykryć i zlokalizować;
- przemijające (ang. *intermittent*) – istnieją jedynie w pewnych okresach czasu (przedziałach, interwałach);
- chwilowe (ang. *transient*) – występują jednokrotnie na skutek chwilowych zmian czynników otoczenia (np. promieniowanie). Praktycznie niemożliwe do zlokalizowania.

➤ Kryterium “krotności”:

- pojedyncze;
- wielokrotne.

Przedmiotem naszego zainteresowania będą uszkodzenia **trwałe** pojedyncze. Uszkodzenia **przemijające** i **chwilowe** wymagają ujęcia statystycznego i uwzględnienia prawdopodobieństwa ich występowania.

Ponadto zakładamy, że w układzie występuje tylko uszkodzenie **pojedyncze**. To założenie, może być skompensowane przez strategię odpowiednio częstego testowania, która zakłada, że przy odpowiednio częstym testowaniu systemu prawdopodobieństwo pojawienia się dwóch uszkodzeń pomiędzy kolejnymi testowaniami systemu jest wyjątkowo małe.

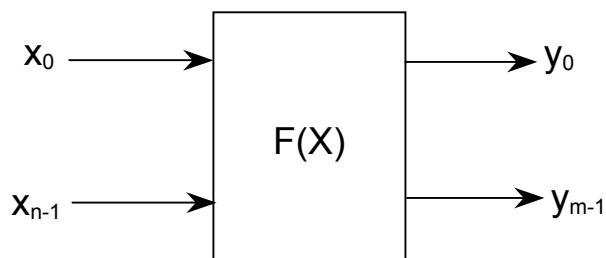
Istnieją sytuacje, w których częste testowanie nie wystarcza dla uniknięcia uszkodzeń wielokrotnych!

Lecz nawet w sytuacji gdy występują uszkodzenia wielokrotne, testy opracowane dla uszkodzeń pojedynczych mogą być stosowane do wykrywania uszkodzeń wielokrotnych ponieważ, **w większości przypadków, uszkodzenie wielokrotne może być wykryte przez testy zaprojektowane dla pojedynczych uszkodzeń składających się na dane uszkodzenie wielokrotne.**

MODELOWANIE

Każdy model dowolnego obiektu fizycznego jest przybliżoną reprezentacją tych jego cech, które są istotne z punktu widzenia, któremu dany model ma służyć.

Na dowolnym poziomie abstrakcji system cyfrowy może być przedstawiony jako „czarna skrzynka”.



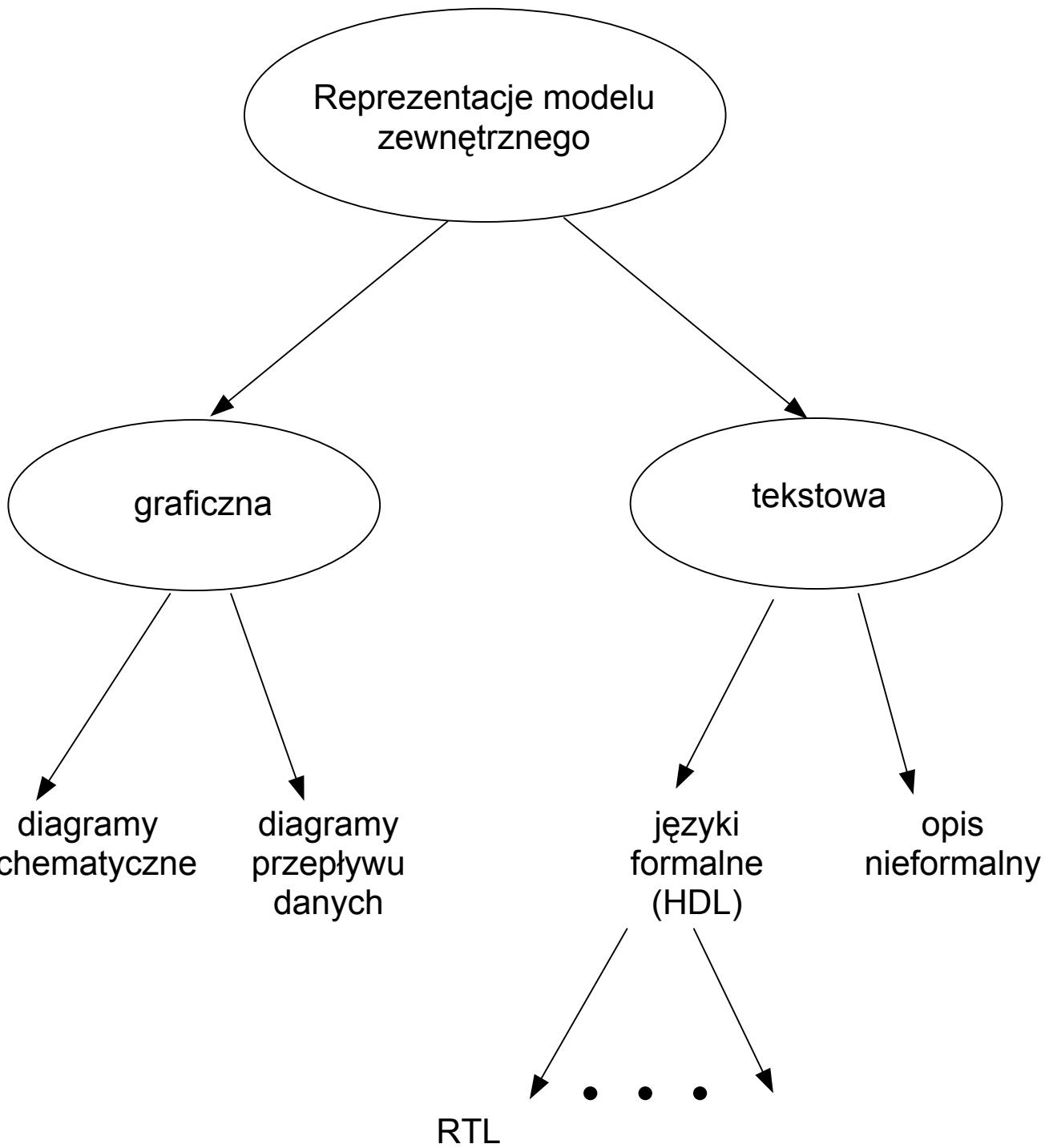
Model funkcjonalny systemu - jest to reprezentacja jego funkcji logicznej.

Model behawiorystyczny (ang. *behavioural model*) - zawiera model funkcjonalny systemu razem z reprezentacją relacji czasowych

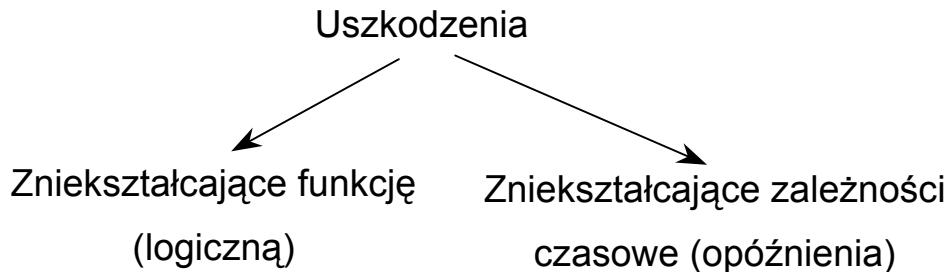
Model strukturalny opisuje system jako zbiór wyróżnionych elementów (komponentów) oraz relacji między nimi. Model strukturalny jest często przedstawiony jako hierarchiczny.

Model zewnętrzny (ang. *external model*) systemu - jest to model „widziany” przez użytkownika.

Model wewnętrzny (ang. *internal model*) przedstawia wewnętrzną strukturę systemu (struktury danych, struktury programów itp.).



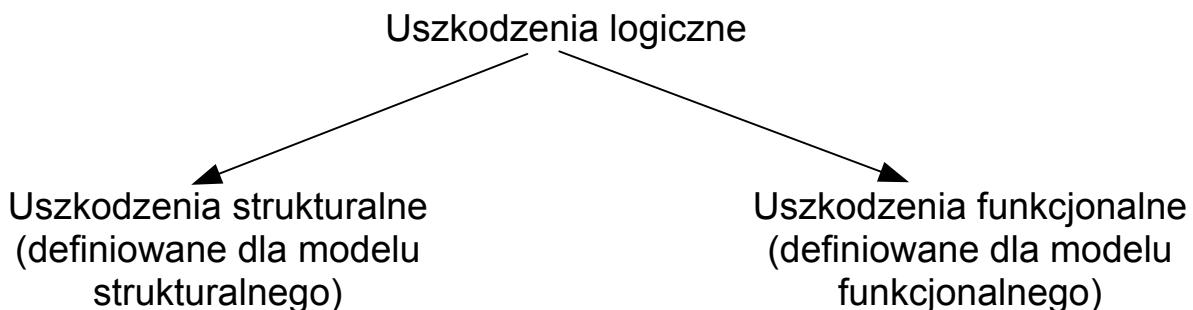
Języki HDL (ang. *Hardware Description Language*) używane do opisu systemu na poziomie rejestrów są określone nazwą RTL (ang. *Register Transfer Language*).



Zaleta produkowanych obecnie systemów cyfrowych – większość ich uszkodzeń uzupełnia się w postaci **błędów funkcji logicznych** spełnianej przez układ. Stąd też zadanie wyznaczenia testu można przenieść na poziom logiczny.

Co uzyskujemy poprzez modelowanie uszkodzeń jako logiczne uszkodzenia?

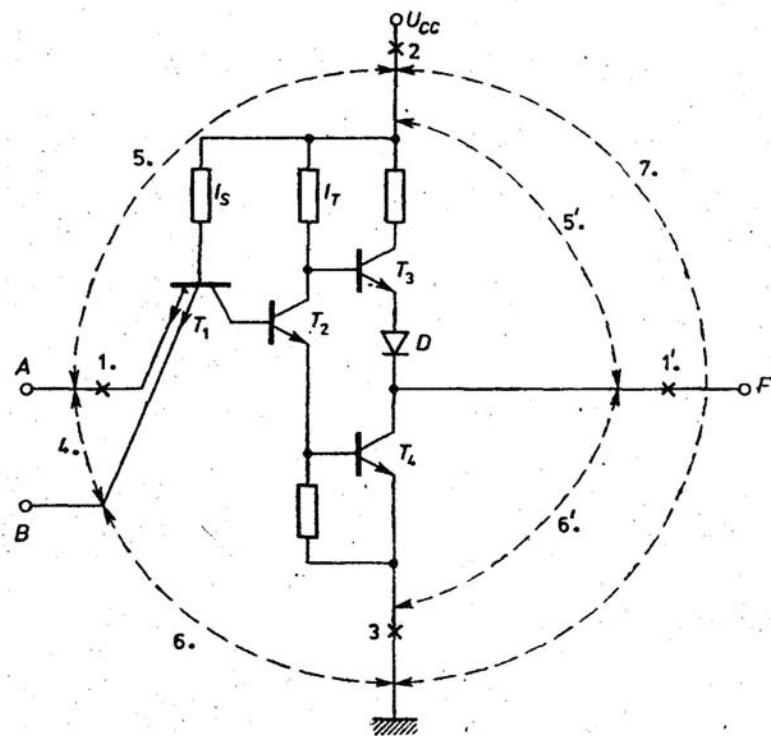
- problem analizy uszkodzeń przenosi się na grunt logiczny, różne uszkodzenia fizyczne mogą być modelowane tymi samymi uszkodzeniami logicznymi;
- uszkodzenia logiczne są niezależne od technologii (ten sam model uszkodzeń jest stosowany do wielu technologii);
- testy ustalone dla logicznych uszkodzeń mogą być użyte dla uszkodzeń fizycznych, których efekt (oddziaływanie) na zachowanie układu jest trudno analizowalne.



Typowe uszkodzenia logiczne

- **uszkodzenia stałosygnałowe** - powodujące stałą wartość $c \in \{0,1\}$ na linii, oznaczenie s-a-c (stuck-at-c) np. $x_i/0$ lub $j/0$;
- **zwarcia (zmostkowania)** linii:
 - typu OR;
 - typu AND.

Przykłady uszkodzeń fizycznych w bramce NAND i odpowiadające im uszkodzenia logiczne



- uszkodzenie 1 odpowiada błędowi logicznemu typu s-a-1;
- uszkodzenie 4 objawia się w momencie gdy zwarte (zmostkowane) linie są w różnych stanach logicznych. Nie można go przedstawić przy pomocy błędów logicznych typu s-a-c;
- uszkodzenie 6 polega na zwarciu linii sygnału do masy i odpowiada błędowi logicznemu typu s-a-0;
- uszkodzenie 7 polega na zwarciu linii zasilania i masy i nie ma interpretacji logicznej.

Detekcja uszkodzeń

Niech $Z(x)$ oznacza funkcję logiczną układu kombinacyjnego N , gdzie x jest wektorem wejściowym układu. Niech $Z(w)$ oznacza odpowiedź układu N na wymuszenie w . Dla układów wielowyściowych $Z(w)$ jest wektorem.

Obecność uszkodzenia f transformuje N do układu N_f . Założymy, że N_f jest układem kombinacyjnym realizującym funkcję $Z_f(x)$. Układ jest testowany poprzez wymuszenie w :

$$T = \langle t_1, t_2, \dots, t_m \rangle$$

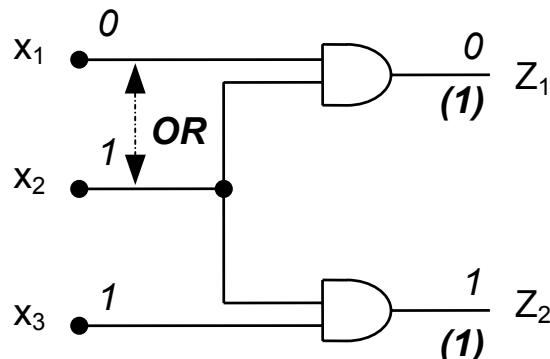
i poprzez porównanie odpowiedzi z (oczekiwana) odpowiedzią wzorcową układu N ,

$$Z(t_1), Z(t_2), \dots, Z(t_m)$$

Definicja

Wymuszenie w jest testem kontrolnym uszkodzenia f , jeżeli $Z_f(w) \neq Z(w)$.

Przykład



Uszkodzenie f (zmostkowanie typu OR linii x_1, x_2) zmienia funkcję układu na:

$$Z_{1f} = x_1 + x_2 \text{ (zamiast } Z_1 = x_1 x_2\text{)}$$

i

$$Z_{2f} = (x_1 + x_2)x_3 \text{ (zamiast } Z_2 = x_2 x_3\text{).}$$

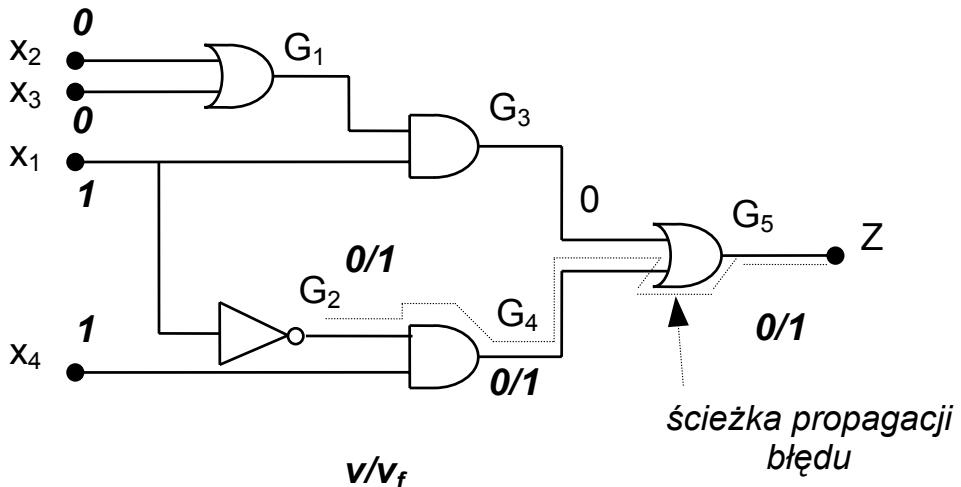
Wymuszenie $w=(011)$ jest testem uszkodzenia f ponieważ $Z(011) = 01$ podczas gdy $Z_f(011) = 11$.

Dla układu z pojedynczym wyjściem:

$$Z(x) \oplus Z_f(x) = 1,$$

gdzie:

\oplus - oznacza operację exclusive – OR.



$$w = 1001$$

Ścieżka wg której wymuszenie w zmienia wartości sygnałów nazywana jest „uczuloną” na błąd f przez wymuszenie w (ścieżka pobudzona).

Uszkodzenie f jest nazywane wykrywalnym, jeśli istnieje wymuszenie w , które jest testem kontrolny dla tego uszkodzenia tzn.

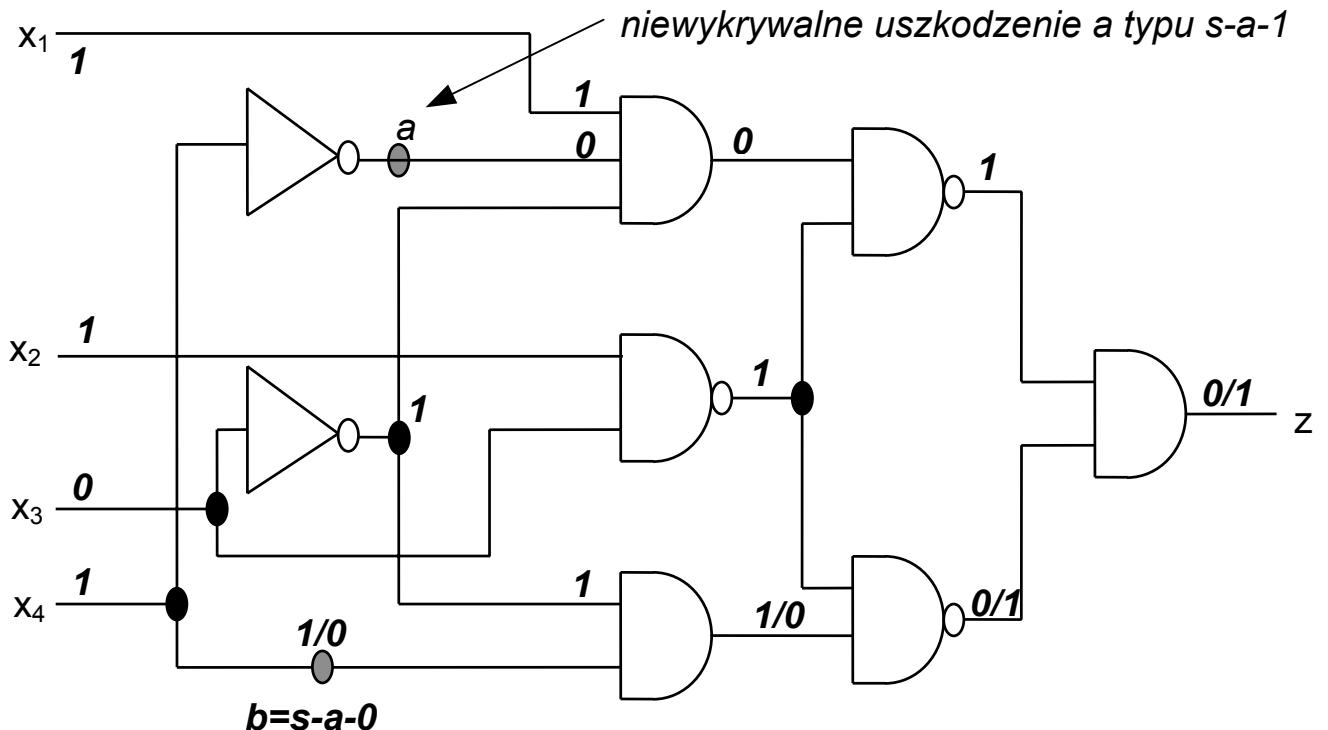
$$Z_f(w) \neq Z(w)$$

$$Y(w, f) \neq Y(w, n_0)$$

Natomiast jeżeli $Z_f(w) = Z(w)$ uszkodzenie f jest niewykrywalne ponieważ nie zmienia funkcji realizowanej przez układ .

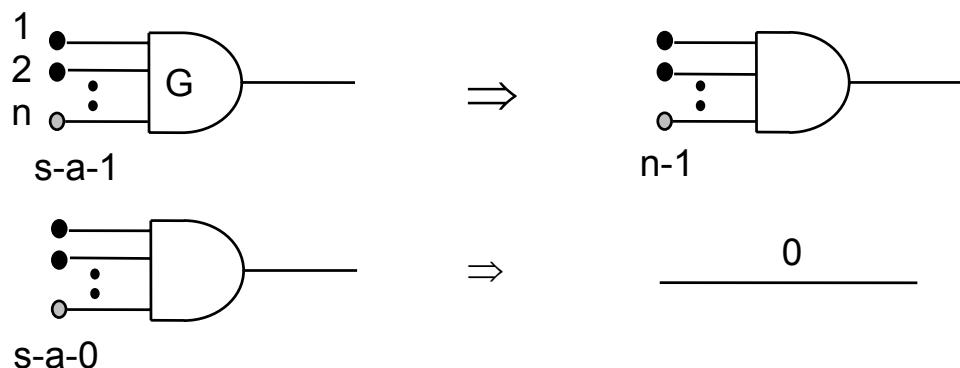
Przykład

Poniższy pokazuje w jaki sposób uszkodzenie $b = s\text{-}a\text{-}0$ jest wykrywalne przez wymuszenie $w = 1101$. Zauważmy, że $a = s\text{-}a\text{-}1$ nie jest wykrywalny przez wymuszenie w jeżeli równocześnie jest obecne uszkodzenie $b = s\text{-}a\text{-}0$.

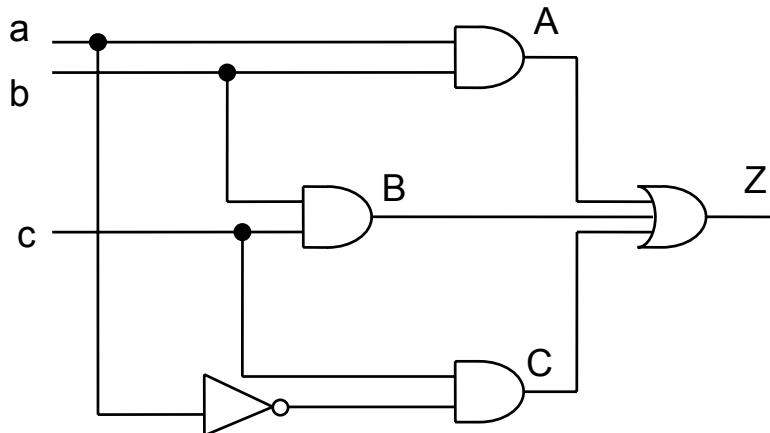


Nadmiarowość (redundancja)

Układ kombinacyjny, który zawiera niewykrywalne uszkodzenia stałosygnalowe nazywamy **redundancyjnym**, ponieważ taki układ zawsze może być uproszczony poprzez usunięcie co najmniej jednej bramki lub jej wejścia:



Redundancja może być wprowadzona do układu, aby zabezpieczyć się przed hazardem. Ilustruje to poniższy rysunek:



$$Z = ab + \bar{a}c + bc = ab + \bar{a}c$$

*B wprowadza element bc, który jest nadmiarowym ale rola **B** - to ochrona przed hazardem bc z **B**. Układ ma statyczny hazard, **B** utrzymuje „1” w czasie zmiany sygnału 111→011.*

Zauważmy, że obecność Y s-a-0 jest niewykrywalna:

1. Jeżeli **f** jest wykrywalnym uszkodzeniem i **g** jest niewykrywalnym uszkodzeniem, to **f** może stać się niewykrywalnym w obecności **g**.
2. Wielokrotne uszkodzenie **{f, g}** może być wykrywalne nawet jeśli po jednym uszkodzeniu **f, g** są niewykrywalne.

Ekwiwalentność uszkodzeń

Definicja

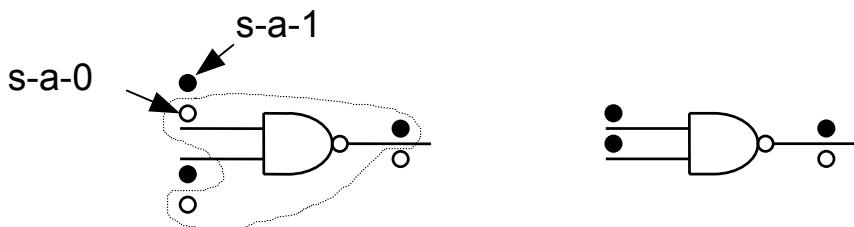
Dwa uszkodzenia **f** i **g** nazywamy funkcjonalnie ekwiwalentnymi względem wymuszenia **w**, jeżeli:

$$Z_f(t) = Z_g(t).$$

Wymuszenie **w** nazywamy rozróżniającym uszkodzenia **f**, **g** jeżeli $Z_f(t) \neq Z_g(t)$. Dla uszkodzeń ekwiwalentnych nie istnieje wymuszenie będące testem rozróżniającym je.

Klasy funkcjonalnej ekwiwalencji - podział możliwych uszkodzeń układu na równoważne podzbiory.

Dla bramki NAND wszystkie wejściowe uszkodzenia s-a-0 i wyjściowe s-a-1 są funkcjonalnie ekwiwalentne.



Dominacja uszkodzeń

Niech wymuszenie **w** będzie testem uszkodzenia **g**. Uszkodzenie **g** dominuje nad uszkodzeniem **f**, jeżeli:

f i **g** są funkcjonalnie ekwiwalentne dla wymuszenia **w**.

Innymi słowy, jeśli **g** dominuje nad **f**, to dowolne wymuszenie **t**, które jest testem **g** tj. $Z_g(t) \neq Z(t)$ jest testem uszkodzenia **f**, ponieważ $Z_f(t) = Z_g(t)$.

Sposoby modelowania układów kombinacyjnych

Do najczęściej spotykanych typów modeli wykorzystywanych do opisywania układów kombinacyjnych należą:

- **model bramkowy** - jest on najprostszą postacią opisu tych układów i pozwala na dowolną implementację przewidzianej przez projektanta funkcji realizowanej przez dany układ. Ważną właściwością modelu bramkowego układu kombinacyjnego, z punktu widzenia diagnostyki takiego układu, jest możliwość analizy wpływu założonej niezdatności w dowolnym punkcie układu na postać wyniku jego działania;
- **sieć informacyjna** - w odróżnieniu od modelu bramkowego stosowana jest zazwyczaj do modelowania pakietów cyfrowych. Do zastosowania sieci informacyjnej opisującej funkcjonowanie układu niezbędna jest znajomość jego struktury wewnętrznej;
- **funkcjonalny model niezawodnościowy** - ma pomóc w diagnozowaniu układu kombinacyjnego, czego efektem powinno być określenie, za pomocą doświadczeń funkcyjonalnych, funkcyjonalnego stanu niezawodnościowego n , ($n \in N$) tego układu, czyli stanu, w którym realizuje on określone, jednoznaczne przekształcenie skończonego zbioru X wymuszeń elementarnych (wektorów binarnych) w skończony zbiór Y wyników testu;
- **binarny diagram decyzyjny** – służy do rozkładu funkcji bulowskiej. Jest on spójnym, acyklicznym w sensie dróg digrafem o jednym węźle pobudzającym i dwóch węzłach spływowych, takim że każdy węzeł przejściowy ma jeden łuk dochodzący oraz, oprócz węzłów spływowych, ma dwa łuki wychodzące.

Binarny diagram decyzyjny

Wykorzystywane są przy wyznaczaniu testów układów kombinacyjnych opisanych funkcją logiczną.

Dowolną funkcję logiczną $f(z), (z = (z_1, \dots, z_n), f(z) \neq \text{const})$ można przedstawić w postaci:

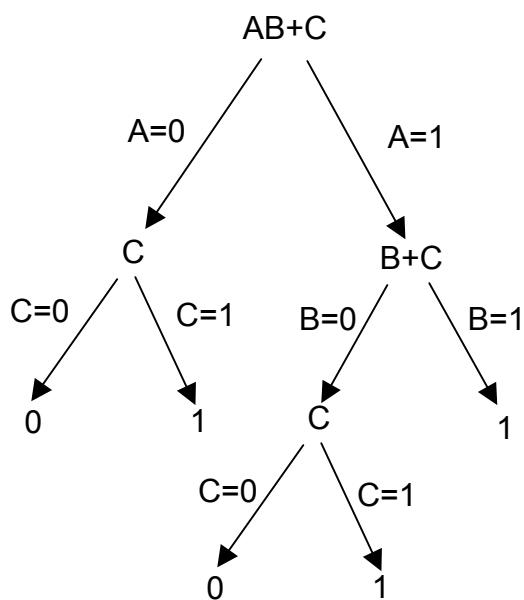
$$f(z) = \overline{z_i} \cdot f(z | z_i = 0) + z_i \cdot f(z | z_i = 1) \quad (i \in \{1, \dots, n\}),$$

gdzie:

$$f(z | z_i = a) = f(z_1, \dots, z_{i-1}, a, z_{i+1}, \dots, z_n) \quad (a \in \{0, 1\}).$$

Funkcję taką oraz jej funkcje składowe można rozkładać w kolejnych krokach względem kolejnych zmiennych. Wyniki takiego rozkładu, aż do uzyskania funkcji przyjmujących wartości stałe 0 lub 1, można przedstawić w postaci binarnego diagramu decyzyjnego.

Przykład binarnego diagramu decyzyjnego dla funkcji logicznej $f = AB + C$. Kolejność rozkładu funkcji względem zmiennych jest następująca: A, B, C.



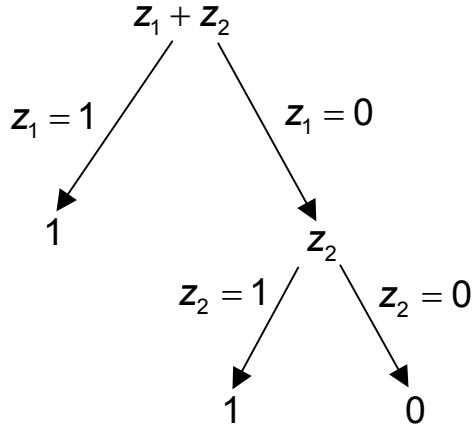
$$F^0 = \{(0x0);(100)\}$$

$$F^1 = \{(0x1);(11x);(101)\}$$

Przykład

Wyznaczyć test kompletny dla układu realizującego funkcję logiczną:
 $F = z_1 + z_2$.

1. Wyznaczenie zbiorów F^0 oraz F^1 przy pomocy binarnego diagramu decyzyjnego.



$$F^0 = \{(00)\}$$

$$F^1 = \{(1x);(01)\}$$

2. Wyznaczenie testów wykrywających niezdatności stałosygnalowe na wejściach układu odpowiadających poszczególnym zmiennym funkcji logicznej F .

Dla zmiennej z_1 : $\{(x0)\} = \{(00);(10)\}$;

Dla zmiennej z_2 : $\{(0x)\} = \{(00);(01)\}$;

3. Wymuszenie w będące testem kompletnym dla danego układu realizującego funkcję logiczną $F = z_1 + z_2$ jest równe: $w = \{(00);(01);(10)\}$.

Funkcjonalny model niezawodnościowy

Definicja

Funkcjonalnym modelem niezawodnościowym obiektu dyskretnego nazywamy parę uporządkowaną:

$$\langle R : S \rightarrow (N \equiv A); P(n = n') \quad n' \in N \rangle,$$

gdzie:

S oraz N oznaczają (odpowiednio) zbiory *stanów fizycznych* oraz *funkcjonalnych stanów niezawodnościowych* obiektu;

A ($A = \{X \rightarrow Y\}$) oznacza zbiór takich jednoznacznych przekształceń zbioru produktów wejściowych X ($1 \leq \text{Card } X < \infty$) w zbiór produktów wyjściowych Y ($1 \leq \text{Card } Y < \infty$), że funkcjonalny stan niezawodnościowy n_0 ($n_0 \in N$), nazywany *funkcjonalnym stanem zdatności* obiektu, utożsamiany jest z przekształceniem A_0 ($A_0 \in A$) opisującym zdolność do poprawnego działania [funkcjonowania] obiektu, a każdy funkcjonalny stan niezawodnościowy n'' ($n'' \in N \setminus n_0$), nazywany (odpowiednim) *funkcjonalnym stanem niezdolności* obiektu – z odpowiednim przekształceniem A'' ($A'' \in A \setminus A_0$) opisującym określone wadliwe działanie obiektu, natomiast $P(n = n')$ ($n' \in N$) oznacza prawdopodobieństwo, że diagnozowany obiekt znajduje się w funkcjonalnym stanie niezawodnościowym n' ($n' \in N$).

Zauważmy, że:

$$|N| = |A| = |Y|^{|X|},$$

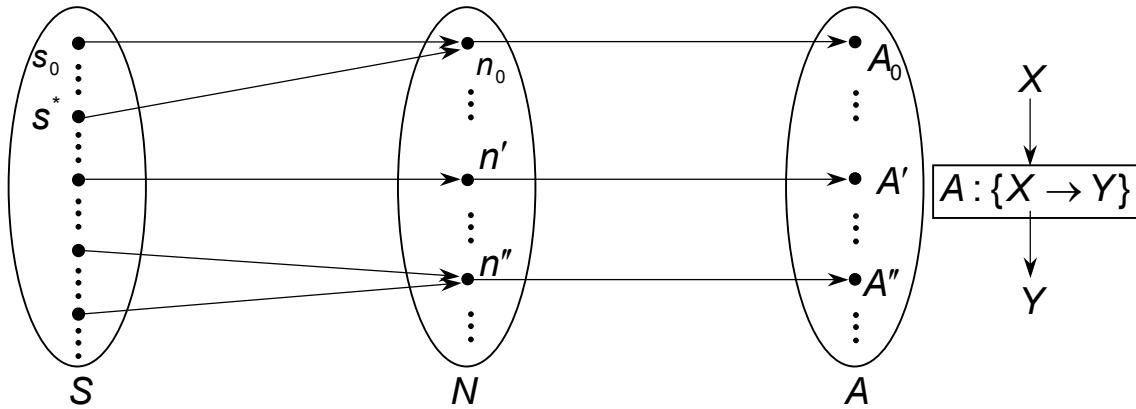
Oznaczmy:

$$N^* = \{n' \in N : P(n = n') > 0\}.$$

Z założenia: $n_0 \in N^*$ oraz $|N^*| \geq 2$.

W wielu, praktycznie spotykanych, przypadkach mamy do czynienia z taką sytuacją, że: $\text{Card } N^* \ll \text{Card } N$.

Funkcjonalny stan niezawodnościowy n ($n \in N$) jest klasą abstrakcji pewnych (nie zawsze w sposób jawny określonych) fizycznych stanów obiektu, które z reguły odpowiadają zaistniałym (lub nie) w obiekcie uszkodzeniom lub wadom (patrz rysunek).



Ilustracja relacji między stanami niezawodnościowymi obiektu dyskretnego a jego własnościami funkcjonalnymi.

Określony stan fizyczny s_0 , w którym obiekt znajduje się w stanie zdatności funkcjonalnej, uważa się za fizyczny stan zdatności obiektu. Zauważmy, że może istnieć stan fizyczny s^* ($s^* \neq s_0$), w którym obiekt również znajduje się w stanie zdatności funkcjonalnej.

Definicja

Element zbioru $W(X)$ ($W(X) = \{X' \subseteq X : X' \neq \emptyset\}$) nazywa się **wymuszeniem** (funkcjonalnym), a wartość $r(w, n)$ ($r(w, n) = \{\langle x, r(x, n) \rangle : x \in w\}$) $r(x, n) \in Y$ - **reakcją** [odpowiedzią] (funkcjonalną) obiektu znajdującego się w stanie niezawodnościowym n ($n \in N$) na wymuszenie w ($w \in W(X)$).

Reakcję [odpowiedź] $r(w, n)$ nazywa się **reakcją** [odpowiedzią] **wzorcową**.

Wymuszenie $w \in X$ nazywa się **wymuszeniem elementarnym**, wymuszenie $w \in W(X) \setminus X$ - **wymuszeniem kompleksowym**, a wymuszenie $w = X$ - **wymuszeniem pełnym**.

Zbiór $\{<x, r(x, n)> : (x \in X) \wedge (n \in N^*)\}$ ($N^* = \{n' \in N : P(n = n') > 0\}$) jest pełnym wzorcem (funkcjonalno-niezawodnościowym) obiektu.

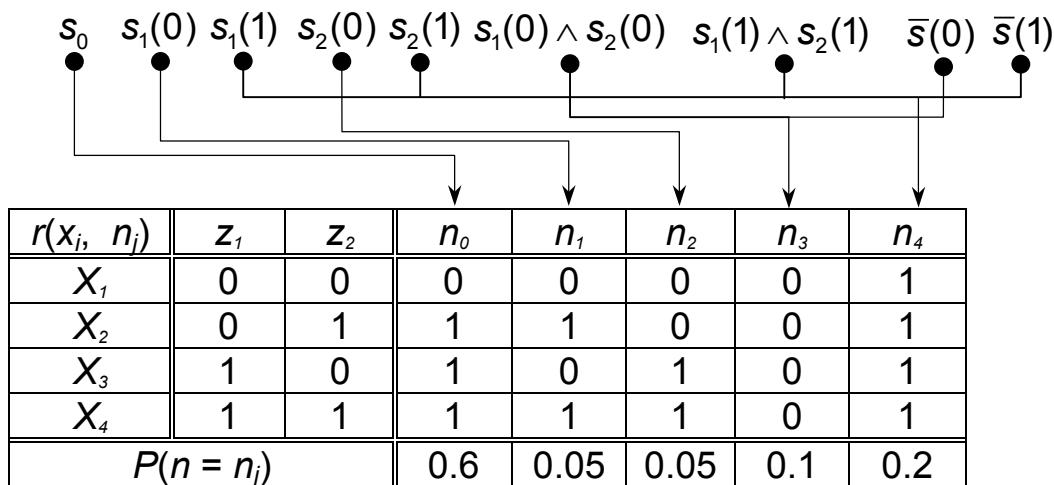
W większości praktycznie spotykanych przypadków pełny wzorzec obiektu nie jest znany, a wiedza eksperymentatora (diagnosty) ogranicza się do znajomości zbioru $\{<x, r(x, n)> : (x \in X') \wedge (n \in N')\}$ ($X' \subseteq X, N' \subseteq N^*$) lub wręcz tylko do znajomości zbioru $\{<x, r(x, n_0)> : x \in X'\}$, nazywanego pełnym (jeśli $X' = X$) lub częściowym (jeśli $X' \neq X$) **wzorcem kontrolnym**.

Przykład

Przedstawić funkcjonalny model niezawodnościowy układu realizującego funkcję logiczną $F = z_1 + z_2$ (OR). Układ ten jest takim obiektem dyskretnym, że $|X| = 4$ (produktem wejściowym jest każda kombinacja wektora binarnego (z_1, z_2)) oraz $|Y| = 2$, stąd też otrzymujemy, że $|N| = 16$.

Niech $s_i(a)$ ($i \in \{1, 2\}, a \in \{0, 1\}$) oraz $\bar{s}(a)$ oznaczają fizyczne stany układu, polegające odpowiednio na tym, że układ funkcjonuje tak, jak gdyby i - te jego wejście oraz (odpowiednio) wyjście układu - miały stale stan logiczny a .

W tabeli przedstawiono funkcjonalny model niezawodnościowy układu, przy czym $|N^*| = 5$ ($N^* = \{n_0, n_1, \dots, n_4\}$).



Dla przykładu fizyczny stan niezawodnościowy $s_2(0)$ powoduje, że układ zamiast funkcji OR realizuje funkcję \bar{z}_1 , która została utożsamiona z funkcjonalnym stanem niezawodnościowym n_2 .

Niech $P(N') = \{N'_1, \dots, N'_K\}$ oznacza K - dzielny ($K \geq 2$) podział zbioru N' ($N' \subseteq N^*$, $\text{Card } N' \geq 2$, $N^* = \{n' \in N : P(n = n') > 0\}$).

Definicja

Wymuszenie w nazywa się **testem** względem podziału $P(N')$ zbioru N' , jeżeli istnieją stany niezawodnościowe n' i n'' , należące do różnych podzbiorów tego podziału, w których reakcje obiektu na to wymuszenie są różne, to jest, jeżeli:

$$\exists [n', n'' \in N' : (n' \in N'_j) \rightarrow (n'' \notin N'_j, 1 \leq j \leq K)] : r(w, n') \neq r(w, n'').$$

Mówimy, że wymuszenie w jest testem względem (dowolnej) pary (n', n'') stanów niezawodnościowych, jeżeli $r(w, n') \neq r(w, n'')$.

Mówimy również, że wymuszenie w jest testem względem zbioru N' , jeżeli istnieje para (n', n'') ($n', n'' \in N'$) względem której jest testem.

Niech $T[X^*, P(N')]$ oznacza zbiór testów względem podziału $P(N')$ zbioru N' istniejących w zbiorze wymuszeń $W(X^*)$ ($X^* \subseteq X$).

Definicja

Wymuszenie w nazywa się **testem kompletnym** względem podziału $P(N')$, jeśli jest testem dla każdej pary stanów niezawodnościowych, należących do różnych podzbiorów tego podziału.

Niech $T^K[X^*, P(N')]$ oznacza zbiór testów kompletnych względem podziału $P(N')$, istniejących w zbiorze $W(X^*)$ ($X^* \subseteq X$).

Definicja

Test kompletny T ($T \in T^K[X^*, P(N')]$) nazywa się **nieredukowalnym testem kompletnym**, jeżeli dowolny podzbiór jego wymuszeń elementarnych nie jest już testem kompletnym, to jest, jeżeli:

$$\forall T' \subset T : T' \notin T^K[X^*, (N')].$$

Niech $T_N^K[X^*, P(N')]$ oznacza zbiór nieredukowalnych testów kompletnych względem podziału $P(N')$, istniejących w zbiorze $W(X^*)$ ($X^* \subseteq X$).

Definicja

Test względem podziału $P(N^*) = \{n_0, N^* \setminus n_0\}$ nazywa się **testem kontrolnym**, natomiast test względem podziału $P(N^* \setminus n_0)$ **testem lokalizacyjnym** względem wymaganej **wnikliwości lokalizacji** stanu niezdatności, określonej przez ten podział.

Wymuszenie w jest testem kontrolnym (względem) niezdatności n' ($n' \in N^* \setminus n_0$), jeżeli $r(w, n') \neq r(w, n_0)$.

Każdy test lokalizacyjny (względem dowolnego podziału $P(N^* \setminus n_0)$) jest jednocześnie testem kontrolnym, bowiem:

$$(w \in T[X^*, P(N^* \setminus n_0)]) \rightarrow \\ \rightarrow (\exists n', n'' \in N^* \setminus n_0 : [r(w, n') \neq r(w, n_0)] \vee [r(w, n'') \neq r(w, n_0)]).$$

Właściwości testu kontrolnego

Definicja

Skutecznością kontrolną $\eta(w)$ wymuszenia w ($w \in W(X)$) nazywamy wartość wyrażenia:

$$\eta(w) = \frac{1}{1 - p_0} P(r(w) \neq r(w, n_0)) \quad (0 \leq \eta(w) \leq 1),$$

gdzie: $p_0 = P(n = n_0)$ ($0 < p_0 < 1$).

Dla równomiernego rozkładu prawdopodobieństwa $P(n = n')$ ($n' \in N^* \setminus n_0$) skuteczność kontrolna wymuszenia w wyraża stosunek liczby niezdatności, względem których wymuszenie w jest testem do ogólnej liczby możliwych niezdatności obiektu, to jest:

$$\eta(w) = \frac{\text{Card } N(w)}{\text{Card } N^* - 1}.$$

gdzie: $N(w) = \{n \in N^* \setminus n_0 : r(w, n) \neq r(w, n_0)\}$ ($N^* = \{n' \in N : P(n = n') > 0\}$) jest zbiorem niezdatności względem których wymuszenie w jest testem kontrolnym.

Definicja

Stopniem pełności $\mu(w)$ wymuszenia w ($w \in W(X)$) nazywamy iloraz liczby składowych wymuszeń elementarnych, tego wymuszenia, do liczby wszystkich możliwych wymuszeń elementarnych, to jest:

$$\mu(w) = \frac{\text{Card } w}{\text{Card } X} \quad (0 < \mu(w) \leq 1).$$

Definicja

Stopniem kompletności $\alpha(w)$ testu kontrolnego w nazywamy wartość wyrażenia:

$$\alpha(w) = \frac{\text{Card } N(w)}{\text{Card } N^* - 1} \quad (0 < \alpha(w) \leq 1).$$

Metoda wyznaczania zbioru neredukowalnych testów kompletnych

Każde wymuszenie w ($w \in W(X^*)$) możemy przedstawić w postaci takiego p -wymiarowego ($p = |X^*|$) wektora binarnego \bar{x} ($\bar{x} = (x_1, \dots, x_p)$, $x_i \in \{0, 1\}$, $1 \leq i \leq p$), w którym składowa x_i przyjmuje wartość 1, wtedy i tylko wtedy, gdy wymuszenie elementarne x_i jest elementem wymuszenia w .

Niech $f(\bar{x} | X^*, P(N'))$ oznacza taką funkcję bulowską, która przyjmuje wartość 1 wtedy i tylko wtedy, gdy zbiór składowych wektora \bar{x} o wartości 1 jest elementem zbioru $T^K[X^*, P(N')]$.

Tak więc, jeżeli $x_{i_1} \cdot x_{i_2} \cdot \dots \cdot x_{i_m}$ ($1 \leq m \leq p$) jest członem koniunkcyjnym (termem) minimalnej normalnej formuły alternatywnej ($m \ n \ f \ a$), czyli minimalnej postaci "sumy iloczynów" funkcji $f(\bar{x} | X^*, P(N'))$, to: $\{x_{i_1} \cdot x_{i_2} \cdot \dots \cdot x_{i_m}\} \in T_N^K[X^*, P(N')]$.

Zbiór $T_N^K[X^*, P(N')]$ wynika więc bezpośrednio z $m \ n \ f \ a$ funkcji $f(\bar{x} | X^*, P(N'))$.

Niech $\Phi[P(N')]$ oznacza zbiór takich par (n', n'') stanów niezawodnościowych należących do zbioru N' , które nie należą do tego samego elementu podziału $P(N')$, a $X'(n', n'')$ - zbiór tych wymuszeń elementarnych, należących do zbioru X' , które są testem względem pary stanów niezawodnościowych (n', n'') .

Zauważmy, że:

$$(T_N^K[X^*, P(N')] \neq \emptyset) \leftrightarrow (\forall (n', n'') \in \Phi[P(N')]: X^*(n', n'') \neq \emptyset).$$

Znajomość $\Phi[P(N')]$ oraz $X^*(n', n'')$ wynika z rodzaju wyznaczanego testu (test kontrolny czy też test lokalizacyjny o wymaganej wnikliwości) oraz ze znajomości reakcji [odpowiedzi] wzorcowych obiektu.

Tak więc, funkcję $f(\bar{x} | X^*, P(N'))$ można określić w postaci normalnej formuły koniunkcyjnej ($n f k$), czyli w postaci "iloczynu sum":

$$f(\bar{x} | X^*, P(N')) = \bigwedge_{(n', n'') \in \Phi[P(N')]} \bigvee_{x' \in X^*(n', n'')} x'.$$

Problem wyznaczenia zbioru niereduwalnych testów kompletnych $T_N^K[X^*, P(N')]$ sprowadza się więc do określenia, zgodnie z powyższą zależnością, $n f k$ funkcji $f(\bar{x} | X^*, P(N'))$ i przekształcenia jej (zgodnie z algorytmem Boole'a) do postaci $m n f a$.

Przykład

Wyznaczmy w zbiorze $W(X)$, zbiór nieredukowalnych, kompletnych testów kontrolnych $T_N^K[X, \{\{n_0\}, \{n_1, \dots, n_4\}\}]$ dla układu cyfrowego OR, rozpatrywanego w poprzednim przykładzie. Z funkcjonalnego modelu niezawodnościowego tego układu, wynika tabela 1a, a z niej tabela 1b.

Tabela 1a

$r(x_i, n_j)$	n_0	n_1	n_2	n_3	n_4
X	x_1	0	0	0	1
	x_2	1	1	0	1
	x_3	1	0	1	0
	x_4	1	1	1	0

Tabela 1b

$\delta(x', (n', n''))$		$\Phi[\{\{n_0\}, \{n_1, n_2, n_3, n_4\}\}]$			
		(n_0, n_1)	(n_0, n_2)	(n_0, n_3)	(n_0, n_4)
X	x_1				1
	x_2		1	1	
	x_3	1		1	
	x_4			1	

$[\delta(x', (n', n'')) = 1] \rightarrow [x' \in X^*(n', n'')]$

Przy czym $\delta(x', (n', n''))$ nazywamy **stopniem pokrycia niezdatności** uzyskanym przy wyznaczaniu testu w .

Z tablicy 1b zgodnie z podaną wcześniej zależnością, wyznaczamy $n f k$ funkcji $f(\bar{x} | X, P(N'))$ i przekształcamy ją do postaci $m n f a$:

$$x_2 \cdot x_3 \cdot (x_2 + x_3 + x_4) \cdot x_1 = x_1 \cdot x_2 \cdot x_3.$$

Tak więc, jedynym nieredukowalnym, kompletnym testem kontrolnym (rozpatrywanego układu cyfrowego OR) jest wymuszenie $w = \{x_1, x_2, x_3\}$.

Możemy łatwo upewnić się (korzystając z tablicy 1b), że każdy podzbiór zbioru $\{x_1, x_2, x_3\}$ nie jest testem kompletnym względem podziału $\{\{n_0\}, \{n_1, \dots, n_4\}\}$ zbioru $\{n_0, \dots, n_4\}$, (np.: wymuszenie $\{x_1, x_2\}$ nie jest testem względem pary (n_0, n_4)).

Przykład

Wyznaczmy w zbiorze $W(X)$, zbiór niereduksowalnych testów kompletnych identyfikujących (lokalizujących z maksymalną wnikliwością) stan niezdarności układu OR, a więc zbiór $T_N^K[X, \{\{n_1\}, \dots, \{n_4\}\}]$.

Tabela 1a

$r(x_i, n_j)$	n_0	n_1	n_2	n_3	n_4
X	x_1	0	0	0	1
	x_2	1	1	0	1
	x_3	1	0	1	0
	x_4	1	1	1	0

Korzystając z tabeli 1a, otrzymujemy tabelę 1-2.

Tabela 1-2

$\delta(x', (n', n''))$		$\Phi[\{\{n_1\}, \dots, \{n_4\}\}]$					
		(n_1, n_2)	(n_1, n_3)	(n_1, n_4)	(n_2, n_3)	(n_2, n_4)	(n_3, n_4)
X	x_1			1		1	1
	x_2	1	1			1	1
	x_3	1		1	1		1
	x_4		1		1		1
$[\delta(x', (n', n'')) = 1] \rightarrow [x' \in X^*(n', n'')]$							

Z tablicy 1-2, zgodnie z podaną wcześniej zależnością, wyznaczamy $n f k$ funkcji $f(\bar{x} | X, \{\{n_1\}, \dots, \{n_4\}\})$ i przekształcamy ją do postaci $m n f a$:

$$(x_2 + x_3) \cdot (x_2 + x_4) \cdot (x_1 + x_3) \cdot (x_3 + x_4) \cdot (x_1 + x_2) \cdot (x_1 + x_2 + x_3 + x_4) = \\ = x_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_4 + x_1 \cdot x_3 \cdot x_4$$

Techniki testowania

Przyjęto ogólnie, że na tzw. *technikę testowania* układu składają się:

- sposób tworzenia sekwencji testów;
- sposób sprawdzania poprawności reakcji układu na przyłożone testy.

Metody tworzenia sekwencji testów:

- **testowanie gruntowne** (ang. *Exhaustive testing*) wymaga sekwencji uwzględniającej wszystkie możliwe przypadki sterowania wejściem układu. I tak dla n -wejściowego układu kombinacyjnego wymuszenie wygenerowane tą metodą składa się z 2^n stanów. Natomiast dla układu sekwencyjnego o n -wejściach oraz m -przerzutnikach wymuszenie składa się z 2^{n+m} stanów.
- **testowanie losowe** (ang. *Random testing*) jest procesem polegającym na losowej generacji wektorów testowych. Długość prawie kompletnej sekwencji losowej zależy od rodzaju układu i może być znaczna. Przy tym testowaniu zakłada się, że błąd nie zmienia charakteru układu, tzn., że układ kombinacyjny nie stanie się układem sekwencyjnym. **Ta metoda generacji testów zależy od modelu uszkodzeń.** Do oceny jakości sekwencji losowych definiuje się miary probabilistyczne.
- **testowanie deterministyczne (systematyczne)** (ang. *Test pattern generation*) wykorzystuje się informację o funkcji lub strukturze testowanego układu (model testowanego układu). Deterministyczny generator testów (GT) może być zorientowany na błędy lub niezależny od błędów. Koszt GT zależy od złożoności układu, dla którego generuje się testy.

D - algebra

D - algebra jest to dwójka uporządkowana $\langle V, \Phi \rangle$,

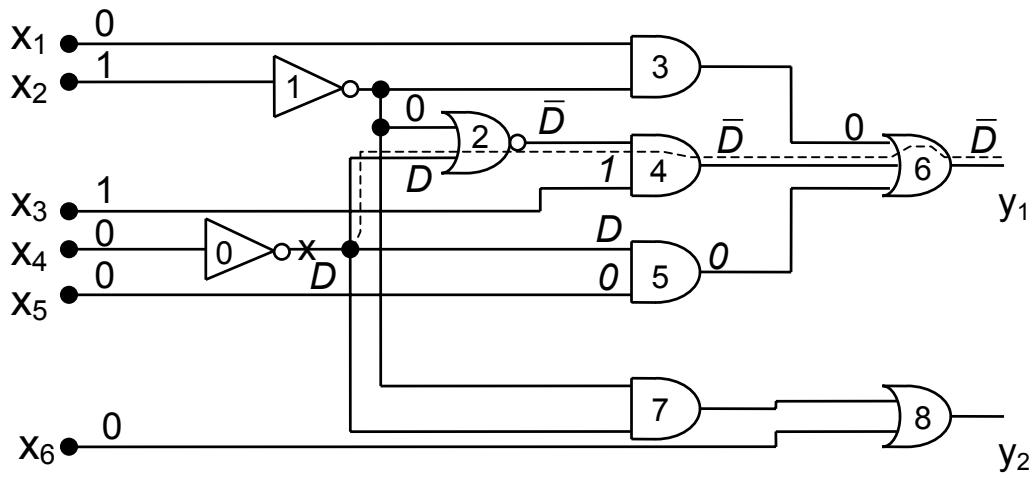
gdzie:

$V = \{0, 1, D, \bar{D}, x\}$ - alfabet służąc do opisu zmian powodowanych uszkodzeniem układu;
 Φ - zbiór operacji nad tym alfabetem.

Znaczenie poszczególnych symboli alfabetu V :

- $0(1)$ - błąd nie zmienia wartości sygnału, jest ona poprawna i równa $0(1)$;
- D - błąd zmienia wartość sygnału cyfrowego z 1 na 0; oznacza to, że w układzie zdatnym sygnał jest równy 1, natomiast w układzie błędny sygnał jest równy 0;
- \bar{D} - błąd zmienia wartość sygnału cyfrowego z 0 na 1; oznacza to, że w układzie zdatnym sygnał jest równy 0, natomiast w układzie błędny sygnał jest równy 1;
- x - wartość sygnału jest nieokreślona i może być równa $0, 1, \bar{D}$ lub D .

Alfabet *D*-algebry umożliwia sformułowanie takiego opisu układu, w który są widoczne wszystkie sygnały błędne i wszystkie sygnały poprawne.



Przykład układu z uszkodzeniem typu s-a-0 opisanego przy pomocy *D*-algebry.

Począwszy od swego źródła, uszkodzenie (utożsamiane z symbolem \bar{D} lub D) rozprzestrzenia się w układzie.

Jeśli propagowane uszkodzenie osiągnie jedno z obserwowlanych wyjść układu, to stan wejścia układu odpowiadający określonym warunkom propagacji (jeżeli nie okażą się one sprzeczne) jest testem rozpatrywanego błędu.

Zbiór Φ operacji *D*-algebry dobiera się tak, aby za jego pomocą można było sporządzić opis układu analogiczny do podanego w przykładzie, przyjmując za punkt wyjścia zaistniałe uszkodzenie.

Na wejściu każdego modułu (bramki) tworzącego układu może pojawić się dowolna z wartości należących do zbioru V . Operacje *D*-algebry określają, w jaki sposób moduły reagują na te wartości.

Definiowanie operacji nad alfabetem V.

Wyznaczenie operacji D -algebry dla dwuwejściowej bramki XOR: $y = f_{XOR}(x_1, x_2)$. Na podstawie opisu jej działania w logice dwuwartościowej (patrz tabela 1) tworzymy dwa zbiory:

$$f_{XOR}^1 = \{(1, 0); (0, 1)\},$$

$$f_{XOR}^0 = \{(0, 0); (1, 1)\}.$$

Tabela 1

x_1	x_2	f_{AND}	f_{OR}	f_{NAND}	f_{NOR}	f_{XOR}
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Błąd obserwowany na wyjściu bramki zmienia stan x_1 , lub/i x_2 z 0 na 1 (z 1 na 0). Jeżeli poprawna kombinacja wejściowa należy do f_{XOR}^α , to błędna kombinacja wejściowa może należeć albo do f_{XOR}^α , albo f_{XOR}^β , gdzie $\alpha \neq \beta$, $\alpha, \beta \in \{0, 1\}$.

W pierwszym przypadku stan wyjścia y bramki pozostaje poprawny, co oznacza, że bramka nie propaguje uszkodzenia (nie zmienia wyjścia).

W drugim przypadku stan wyjścia bramki zmienia się (staje się błędny) co oznacza, że błąd z wejścia przenosi się na wyjście y .

Szczegółowa analiza obu przypadków pozwala utworzyć tabelę definiującą operację \tilde{f}_{XOR} nad alfabetem V . Tabelę tą tworzymy rozpatrując następujące cztery przypadki oddziaływania uszkodzenia na wejście bramki:

(A - oznacza sekwencję poprawną, B - oznacza sekwencję błędnią):

1. $A \in f_{XOR}^0, B \in f_{XOR}^0 : \{(0, 0); (1, 1); (D, D); (\bar{D}, \bar{D})\} \rightarrow 0;$
2. $A \in f_{XOR}^1, B \in f_{XOR}^1 : \{(0, 1); (1, 0); (\bar{D}, D); (D, \bar{D})\} \rightarrow 1;$
3. $A \in f_{XOR}^1, B \in f_{XOR}^0 : \{(0, D); (1, \bar{D}); (\bar{D}, 1); (D, 0)\} \rightarrow D;$
4. $A \in f_{XOR}^0, B \in f_{XOR}^1 : \{(0, \bar{D}); (\bar{D}, 0); (D, 1); (1, D)\} \rightarrow \bar{D};$

Analogicznie możemy określić operacje $\tilde{f}_{AND}, \tilde{f}_{OR}, \tilde{f}_{NAND}, \tilde{f}_{NOR}$, które przedstawiono w tabeli 2.

Tabela 2

x_1	x_2	\tilde{f}_{AND}	\tilde{f}_{OR}	\tilde{f}_{NAND}	\tilde{f}_{NOR}	\tilde{f}_{XOR}
0	0	0	0	1	1	0
0	1	1	0	1	0	1
0	D	D	0	1	\bar{D}	D
0	\bar{D}	\bar{D}	0	1	D	\bar{D}
1	0	1	0	1	0	1
1	1	1	1	0	0	0
1	D	1	D	\bar{D}	0	\bar{D}
1	\bar{D}	1	\bar{D}	D	0	D
D	0	D	0	1	\bar{D}	D
D	1	1	D	\bar{D}	0	\bar{D}
D	D	D	D	\bar{D}	\bar{D}	0
D	\bar{D}	1	0	1	0	1
\bar{D}	0	\bar{D}	0	1	D	\bar{D}
\bar{D}	1	1	\bar{D}	D	0	D
\bar{D}	D	1	0	1	0	1
\bar{D}	\bar{D}	D	\bar{D}	D	D	0

Sposób wyznaczenia poszczególnych wierszy tabeli.

- dla $A \in f_{XOR}^0, B \in f_{XOR}^0 : \{(0, 0); (1, 1); (D, D); (\bar{D}, \bar{D})\} \rightarrow 0;$

A	B	D-algebra	f
0,0	0,0	0,0	0
0,0	1,1	\bar{D}, \bar{D}	
1,1	0,0	D, D	
1,1	1,1	1,1	

- dla $A \in f_{XOR}^1, B \in f_{XOR}^1 : \{(0, 1); (1, 0); (\bar{D}, D); (D, \bar{D})\} \rightarrow 1;$

A	B	D-algebra	f
1,0	1,0	1,0	1
1,0	0,1	D, \bar{D}	
0,1	1,0	\bar{D}, D	
0,1	0,1	0,1	

- dla $A \in f_{XOR}^1, B \in f_{XOR}^0 : \{(0, D); (1, \bar{D}); (\bar{D}, 1); (D, 0)\} \rightarrow D;$

A	B	D-algebra	f
1,0	0,0	$D, 0$	D
1,0	1,1	$1, \bar{D}$	
0,1	0,0	$0, D$	
0,1	1,1	$\bar{D}, 1$	

- dla $A \in f_{XOR}^0, B \in f_{XOR}^1 : \{(0, \bar{D}); (\bar{D}, 0); (D, 1); (1, D)\} \rightarrow \bar{D};$

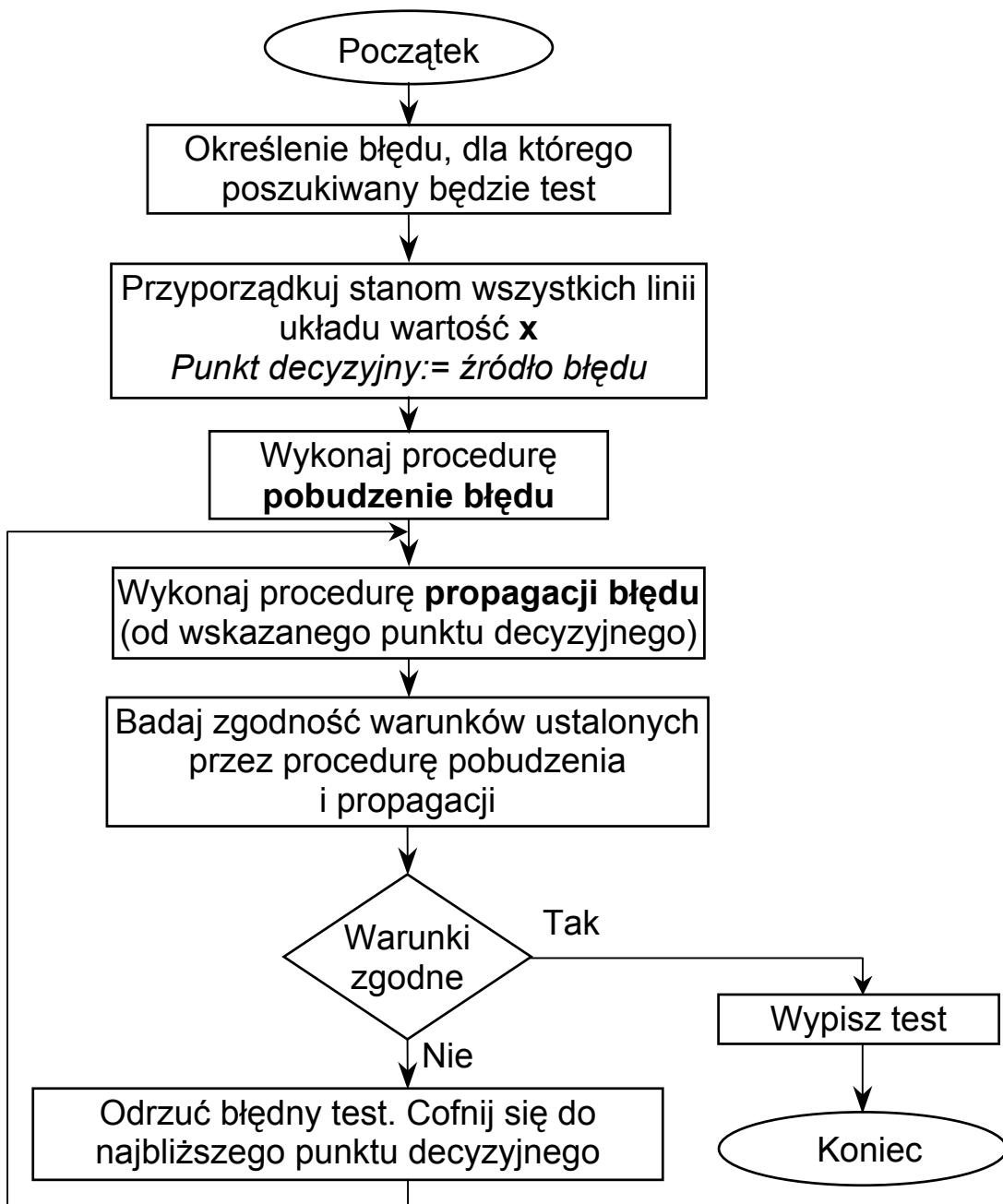
A	B	D-algebra	f
0,0	1,0	$\bar{D}, 0$	\bar{D}
0,0	0,1	$0, \bar{D}$	
1,1	1,0	$1, D$	
1,1	0,1	$D, 1$	

Jeżeli znane są operacje D -algebry niezbędne do opisania wszystkich elementów układu, to można formułować algorytmy systematycznego wyznaczania warunków zapewniających propagację.

D-algorytm

D-algorytm składa się z następujących procedur:

1. pobudzenie źródła błędu;
2. propagacji błędu;
3. badania zgodności warunków pobudzenia i propagacji.



Schemat blokowy D -algorytmu

Procedura pobudzania źródła błędu

Wymuszenie stanu D lub \bar{D} w miejscu występowania błędu. Stan D jest wymuszany w przypadku błędu **s-a-0**, natomiast \bar{D} w przypadku błędu **s-a-1**.

Niech y będzie wyjściem modułu. Jeżeli na linii y wystąpił błąd s-a-0 (s-a-1), to wymuszenie na niej wartości $D(\bar{D})$ polega na ustawieniu wejścia modułu w takim stanie, któremu poprawnie odpowiada wartość 1(0) wyjścia.

Przy realizacji procedury pobudzenia źródła błędu wykorzystuje się tzw. **pierwotne D-sześciiany**. Określają one stany wejść bramek potrzebne do wymuszenia D lub \bar{D} na ich wyjściach. Dla bramek podano je w tabeli 3.

Tabela 3

Bramka	x_1	x_2	f
AND	0	x	\bar{D}
	x	0	\bar{D}
	1	1	D
OR	1	x	D
	x	1	D
	0	0	\bar{D}
NAND	0	x	D
	x	0	D
	1	1	\bar{D}
NOR	1	x	\bar{D}
	x	1	\bar{D}
	0	0	D
XOR	0	0	\bar{D}
	1	1	\bar{D}
	0	1	D
	1	0	D

Procedura propagacja błędu

Procedura propagacji błędu tworzy tzw. **ścieżkę pobudzenia**, pomiędzy źródłem błędu a jedynym z obserwowlanych wyjścia układu.

Ścieżkę nazywamy pobudzoną, jeżeli zmiana stanu linii początkującej ścieżkę powoduje zmianę stanu na jej końcu.

Ścieżki pobudzone tworzy się na podstawie tzw. **przesyłowych D-sześciianów**.

Przesyłowe D-sześciiany modułu logicznego wyznacza się wprost z definicji jego funkcji w D -algebrze. W szczególności tabelę 4 otrzymano z tabeli 2, wybierając z niej wiersze, które odpowiadają propagacji D i \bar{D} przez bramki.

Tabela 4

Bramka	x_1	x_2	f
AND	D	1	D
	1	D	D
	\bar{D}	1	\bar{D}
	1	\bar{D}	\bar{D}
OR	D	0	D
	0	D	D
	\bar{D}	0	\bar{D}
	0	\bar{D}	\bar{D}
NAND	D	1	\bar{D}
	1	D	\bar{D}
	\bar{D}	1	D
	1	\bar{D}	D
NOR	D	0	\bar{D}
	0	D	\bar{D}
	\bar{D}	0	D
	0	\bar{D}	D
XOR	0	D	D
	D	0	D
	\bar{D}	0	\bar{D}
	0	\bar{D}	\bar{D}
	1	D	\bar{D}
	D	1	\bar{D}
	1	\bar{D}	D
	\bar{D}	1	D

Procedura badania zgodności stanów

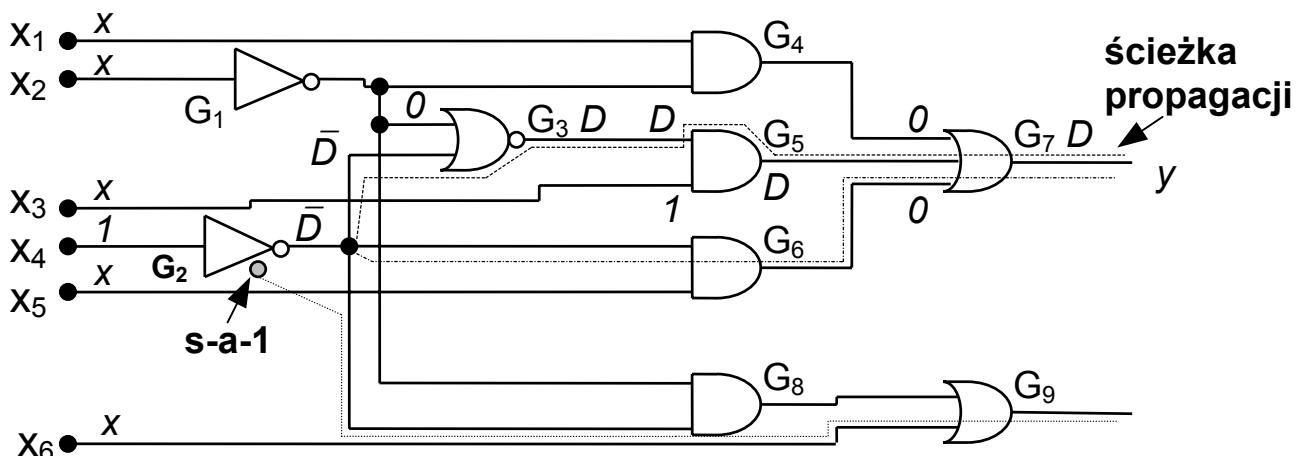
Uzupełnienie i **badanie zgodności stanów** jest treścią ostatniej procedury. Wykorzystuje się w niej tzw. **pierwotne sześciiany** wyjść modułów przedstawione w tabeli 5.

Tabela 5

Bramka	x_1	x_2	f
AND	0	x	0
	x	0	0
	1	1	1
OR	1	x	1
	x	1	1
	0	0	0
NAND	0	x	1
	x	0	1
	1	1	0
NOR	1	x	0
	x	1	0
	0	0	1
XOR	0	0	0
	1	1	0
	1	0	1
	0	1	1

Przykład

Wyznaczyć wymuszenie będące testem błędu s-a-1 na wyjściu bramki G_2 .

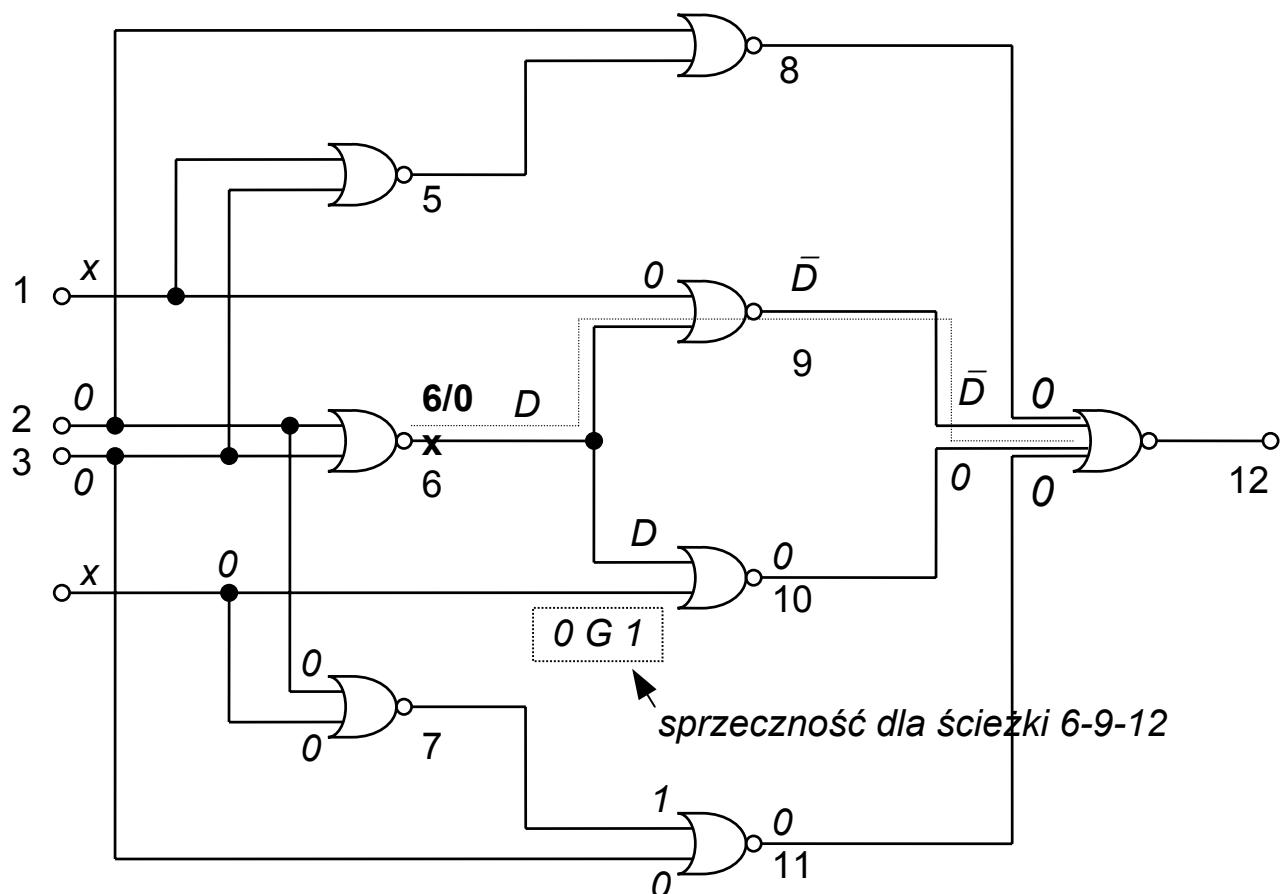


Działanie D-algorytmu

Krok	Stany linii układu														Komentarz
	X_1	X_2	X_3	X_4	X_5	X_6	G_1	G_2	G_3	G_4	G_5	G_6	G_7		
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Stan początkowy
2				1				\bar{D}							Pobudzenie źródła błędu
3															Wybór ścieżki propagacji (G_3, G_5, G_7)
4							0		D						Propagacja przez bramkę NOR (G_3)
5			1							D					Propagacja przez bramkę AND (G_5)
6										0		0	D		Propagacja przez bramkę OR (G_7)
	x	x	1	1	x	x	0	\bar{D}	D	0	D	0	D		Stan końcowy
7	x						0			0					Zgodność stanu na bramce G_4
8		1					0								Zgodność stanu na bramce G_1
9					0							0			Zgodność stanu na bramce G_6
	x	1	1	1	0	x	0	\bar{D}	D	0	D	0	D		Test

Zastosowanie *D*-algorytmu dla układów kombinacyjnych nie stwarza trudności, poza złożonością obliczeniową. Poniższy przykład pokazuje, że próby skutecznego pobudzenia każdej ze ścieżek 6-9-12 i 6-10-12 oddziennie dla błędu 6/0 zawodzą. W toku wykonania *D*-algorytmu dopiero ostatnia, trzecia próba przeprowadzania błędu na wyjście układu kończy się sukcesem.

Jeżeli od źródła błędu do obserwowlnego wyjścia układu prowadzi n ścieżek, to wykonanie *D*-algorytmu może, w najgorszym przypadku wymagać podjęcie $2^n - 1$ prób znalezienia poprawnych propagacji. Liczba $2^n - 1$ może być duża, szczególnie w układach zawierających wiele rozgałęzień.



Błąd 6/0 ma tylko jeden test równy (0, 0, 0, 0)

Funkcje kosztu (typowe)

- **Miary sterowalności** - które wskazują relatywną trudność (koszt) ustalenia pewnej wartości (0 lub 1) na linii.
- **Miary obserwonalności** - które wskazują trudności (koszt) w propagacji błędu od danej linii do linii wyjściowej układu.

Miary sterowalności - mogą być użyte do wyboru zarówno najtrudniejszego ustawienia linii (powiedzmy, ustalenia wartości v na wyjściu bramki G), jak i do wyboru linii wejściowej bramki o stanie x spośród wielu linii, która najłatwiej wysteruje wyjście bramki G do wartości v.

Miary obserwonalności - mogą być wykorzystane do wybierania bramki spośród bramek należących do *D-granicy*, której błąd na wejściu jest najłatwiej obserwowałny.

Rekursywne funkcje kosztu

- Dowolne funkcje kosztu określające sterowalność i obserwonalność powinny być miarami relatywnymi tzn. umożliwiać ranking w ramach danego układu.
- Inne zastosowanie miar sterowalności i obserwonalności to obliczenie miary testowalności układu (systemu).

Miary sterowalności

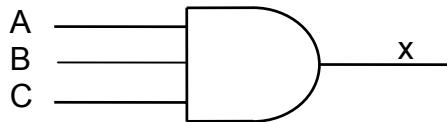
Dla każdej linii L chcemy obliczyć dwie funkcje kosztu:

$$c^0(L) \text{ i } c^1(L),$$

wskazujące relatywnie trudność ustawienia linii L w stanie 0 lub 1 odpowiednio.

Przykład (bez rozgałęzień)

Rozważmy bramkę AND.



Załóżmy, że znamy wartości kosztu c^1 i c^0 dla każdej linii wejściowej.

Aby ustawić $x = 0$, wystarczy podać 0 na dowolną z linii wejściowych bramki, zatem:

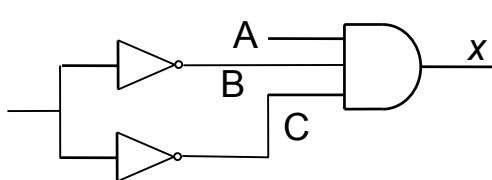
$$c^0(x) = \min\{c^0(A), c^0(B), c^0(C)\}.$$

Aby ustawić $x = 1$, należy podać 1 na wszystkie wejścia układu, zatem:

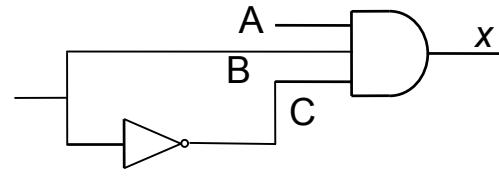
$$c^1(x) = c^1(A) + c^1(B) + c^1(C).$$

Przykład (z rozgałęzieniami)

Jednakże w sytuacjach jak poniżej wejścia mogą być zależne, ze względu na rozgałęzienia:



a)



b)

Dla przypadku z rys. a funkcje kosztów wynoszą odpowiednio:

$$c^0(x) = \min\{c^0(A), c^0(B)\};$$

$$c^1(x) = c^1(A) + c^1(B).$$

Dla przypadku z rys. b, w którym wejścia B i C są komplementarne, funkcje kosztów wynoszą odpowiednio:

$$c^0(x) = \min\{c^0(A), c^0(B)\};$$

$$c^1(x) = 0.$$

Algorytm wyznaczania sterowalności

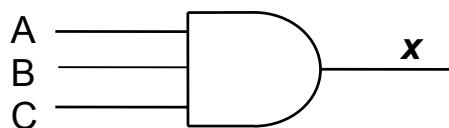
- Nadanie wartości 1 funkcjom kosztów c^1 i c^0 linii wejściowych układu.
- Następnie funkcji kosztów c^1 i c^0 dla każdej bramki poziomu 1, potem poziomu 2 itd.
- Algorytm kończy działanie, gdy zostanie określona funkcja kosztów dla wszystkich linii układu.

Miary obserwonalności

$O(L)$ - oznacza relatywną trudność propagacji błędu od linii L do wyjścia układu (OW).

Przykład (baz rozgałęzień)

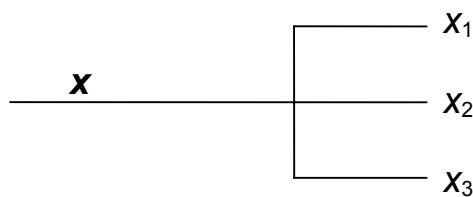
Rozważmy bramkę AND i założymy, że znamy $O(x)$.



Aby przeprowadzić propagację błędu z linii A do x należy ustawić B i C na 1, co pozwoli na przeprowadzenie propagacji błędu z x do OW. Jeżeli te problemy (zadania) są niezależne, otrzymujemy:

$$O(A) = c^1(B) + c^1(C) + O(x)$$

Rozważmy problem obserwonalności rozgałęzienia.



$$O(A) = \min\{O(x_1), O(x_2), O(x_3)\}$$

Algorytm PODEM

(Path Oriented Decision Making)

W algorytmie **PODEM** w porównaniu z **D-algorytmem** eliminuje się czysto heurystyczny wybór ścieżek przez:

1. Zarówno po pobudzeniu źródła błędu, jak też po każdym kroku propagacji błędu wykonuje się badanie zgodności tego kroku z poprzednimi; stosowana procedura “cofania się” wzdłuż ścieżki doprowadza to badanie aż do pierwotnych wejść układu.
2. Jeżeli badanie potwierdziło zgodność, to symuluje się działanie układu dla wyznaczonego stanu jego wejścia, po to aby sprawdzić, czy błąd propaguje się na obserwalne wyjście układu, co oznacza, że utworzono poprawną ścieżkę pobudzoną, a zatem znaleziono test rozpatrywanego błędu.
3. Badanie zgodności decyzji o wyborze pobudzonej ścieżki uzależnia się od wielkości tzw. sterowalności poszczególnych linii układu.

Algorytm wyznaczania sterowalności

1. Nadaj każdej linii i układu wagę początkową równą:

$$WO_i = f_i - 1,$$

gdzie:

i – numer wejścia bramki,
 f_i - liczba rozgałęzień związanych w wejściem i .

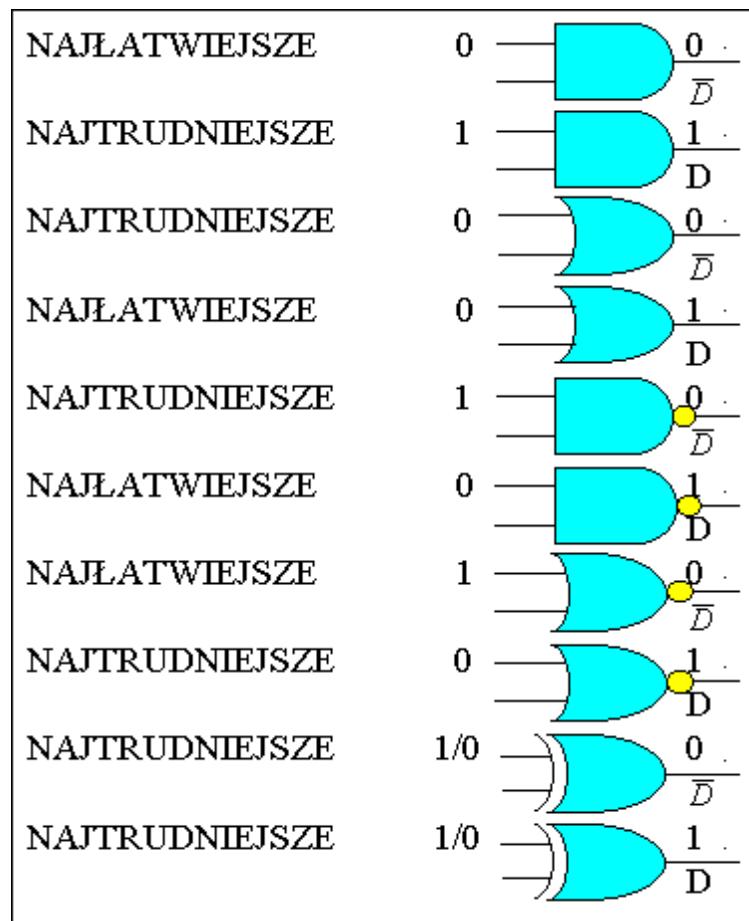
2. Oblicz wagę końcową WK_j wyjścia bramki j równą:

$$WK_j = WO_j + \sum_{i=1}^{m_j} WK_i ,$$

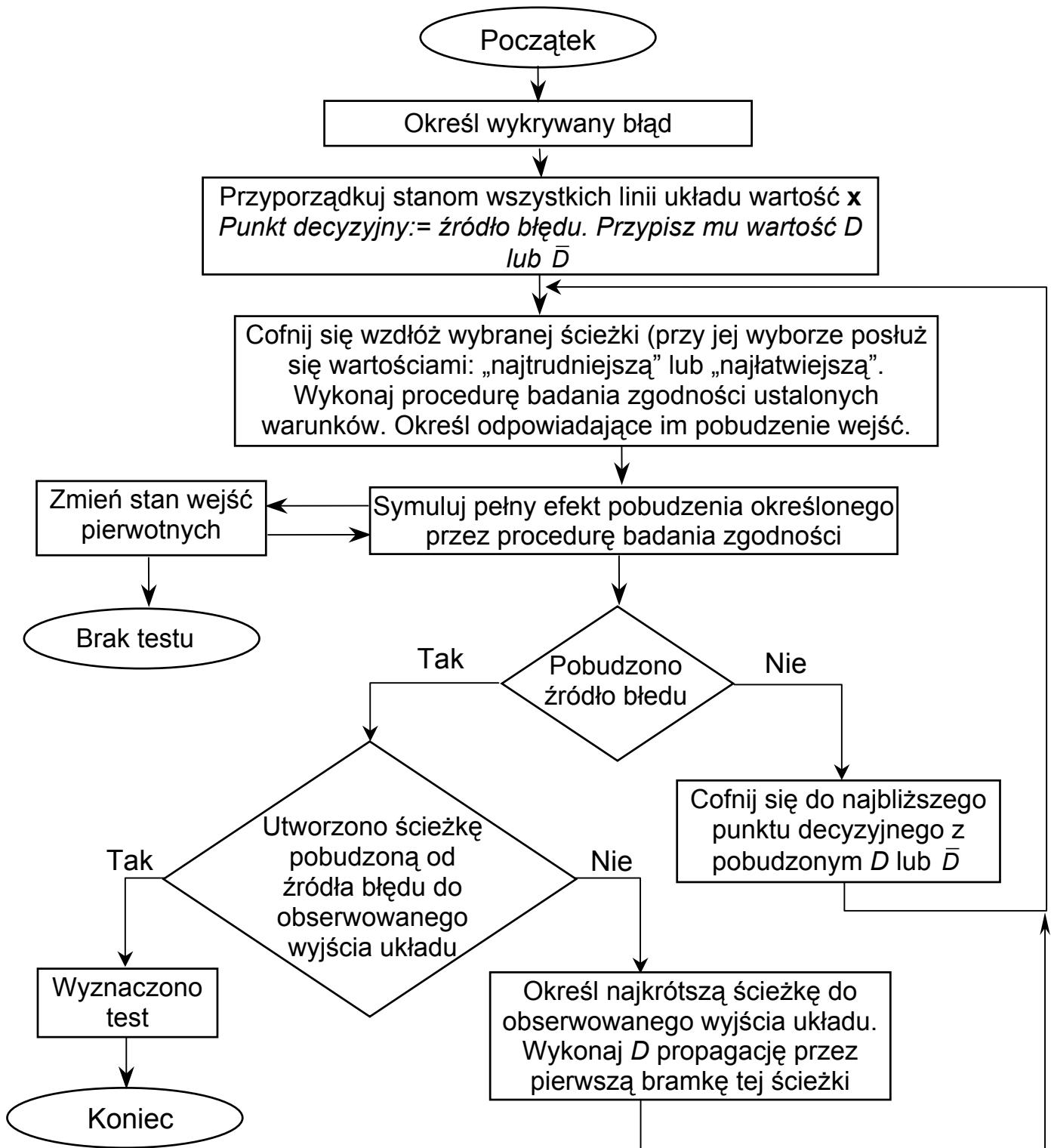
gdzie:

m_j - liczba wejść bramki j .

Zasady wyboru wejść „najłatwiejszych” i „najtrudniejszych”

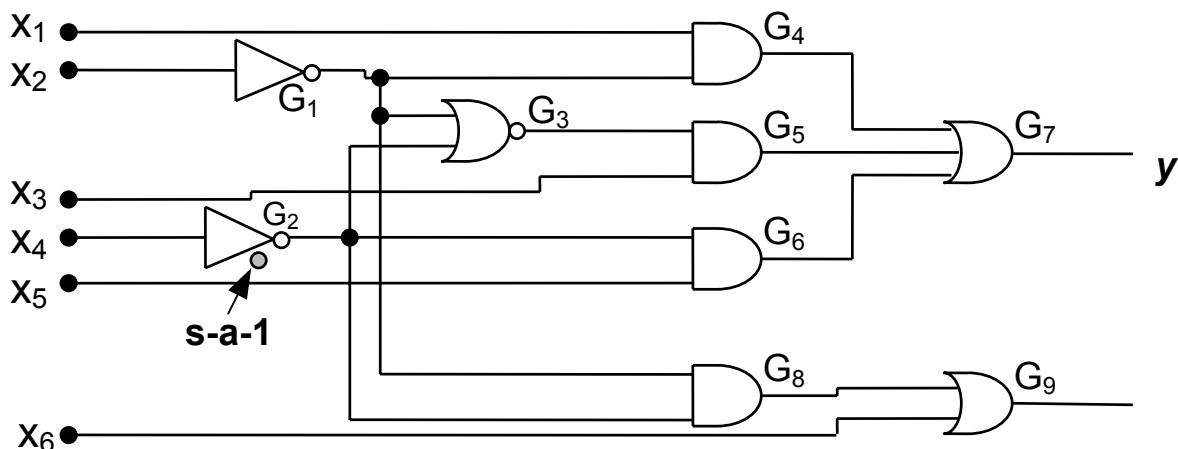


Algorytm



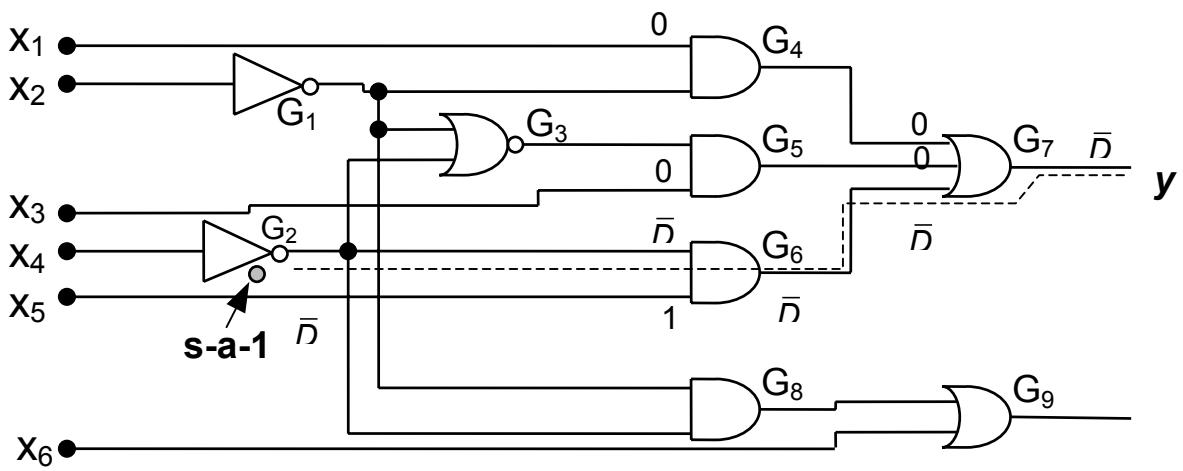
Przykład

Wyznaczyć, wykorzystując **algorytmem PODEM**, test błędu s-a-1 na elemencie G_1 dla układu przedstawionego na poniższym rysunku.



Ocena sterowalności linii

Numer węzła	WO_i	WK_i
X_1	0	0
X_2	0	0
X_3	0	0
X_4	0	0
X_5	0	0
G_1	2	2
G_2	2	2
G_3	0	4
G_4	0	2
G_5	0	4
G_6	0	2
G_7	0	8
G_8	0	4
G_9	0	4

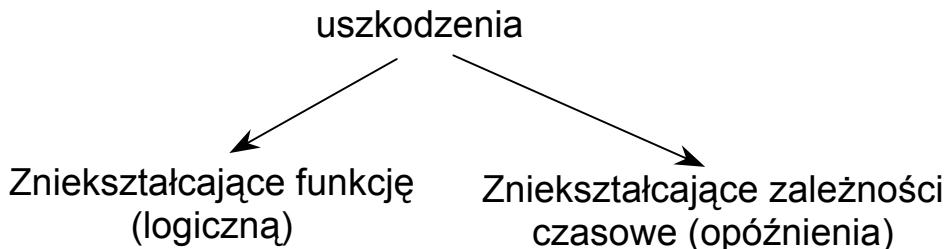


Działanie algorytmu PODEM

Krok	Stany linii układu														Komentarz
	X_1	X_2	X_3	X_4	X_5	X_6	G_1	G_2	G_3	G_4	G_5	G_6	G_7		
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Stan początkowy
2									\bar{D}						Źródło błędu: $G_2 / 1$
3				1											Cofanie się do X_4 (brak warunków „najtrudniejszy” „najłatwiejszy”)
4	x	x	x	1	x	x	x	\bar{D}	x	x	x	x	x	x	Symulacja pobudzenia źródła błędu.
5						1			\bar{D}				\bar{D}		Określenie najkrótszej ścieżki do wyjścia. Propagacja przez pierwszą bramkę tej ścieżki. Sprawdzenie – wejście pierwotne.
6	x	x	x	1	1	x	x	\bar{D}	x	x	x	\bar{D}	x		Symulacja efektu pobudzenia.
7										0	0		\bar{D}		Określenie najkrótszej ścieżki do wyjścia. Propagacja przez pierwszą bramkę tej ścieżki.
8	0														Sprawdz. dla G_4 najłatwiejsze 0, $WK(X_2) = 0$. Wejście pierwotne.
9			0												Sprawdz. dla G_5 najłatwiejsze 0, $WK(X_3) = 0$. Wejście pierwotne.
10	0	x	0	1	1	x	x	\bar{D}	x	0	0	\bar{D}	\bar{D}		Symulacja pobudzenia źródła błędu
11	0	x	0	1	1	x									Test

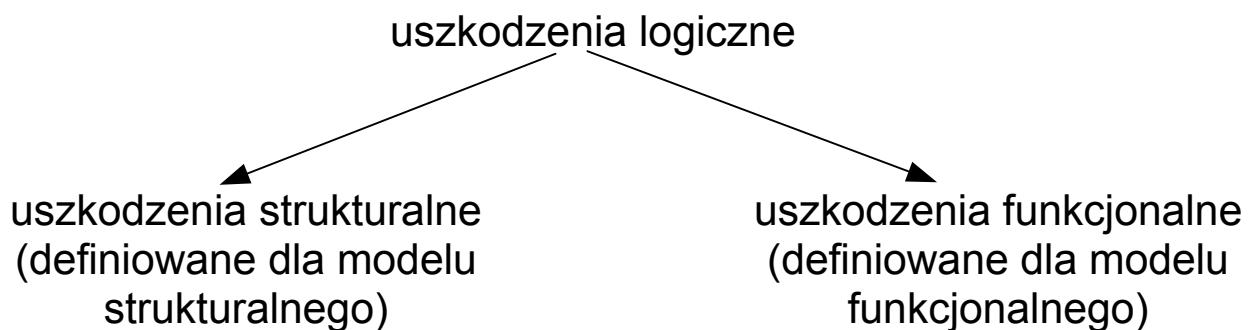
DIAGNOZOWANIE SIECI LOGICZNYCH

Uszkodzenia logiczne reprezentują skutki uszkodzeń fizycznych w funkcjonowaniu systemu

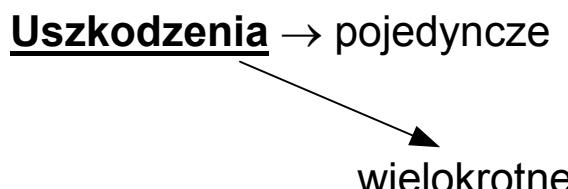


Co uzyskujemy poprzez modelowanie uszkodzeń jako logiczne uszkodzenia?

- problem analizy uszkodzeń przenosi się na grunt logiczny, różne uszkodzenia fizyczne mogą być modelowane tymi samymi uszkodzeniami logicznymi;
- uszkodzenia logiczne są niezależne od technologii (ten sam model uszkodzeń jest stosowany do wielu technologii);
- testy ustalone dla logicznych uszkodzeń mogą być użyte dla uszkodzeń fizycznych, których efekt (oddziaływanie) na zachowanie układu jest trudno analizowalne.



Przedmiotem zainteresowania będą jedynie **uszkodzenia trwające**. Pozostałe uszkodzenia (przemijające i chwilowe wymagają ujęcia statystycznego i uwzględnienia prawdopodobieństwa ich występowania)



Zakładamy, że w układzie występuje tylko **uszkodzenie pojedyncze**. To założenie, może być skompensowane przez strategię odpowiednio częstego testowania, która zakłada, że przy odpowiednio częstym testowaniu systemu prawdopodobieństwo pojawienia się dwóch uszkodzeń pomiędzy kolejnymi testowaniami systemu jest wyjątkowo małe.

Jednakże istnieją sytuacje, w których częste testowanie nie wystarcza dla uniknięcia uszkodzeń wielokrotnych!

Lecz nawet w sytuacji gdy występują uszkodzenia wielokrotnie, testy opracowane dla uszkodzeń pojedynczych mogą być stosowane do wykrywania uszkodzeń wielokrotnych ponieważ, **w większości przypadków, uszkodzenie wielokrotne może być wykryte przez testy zaprojektowane dla pojedynczych uszkodzeń składających się na dane uszkodzenie wielokrotne.**

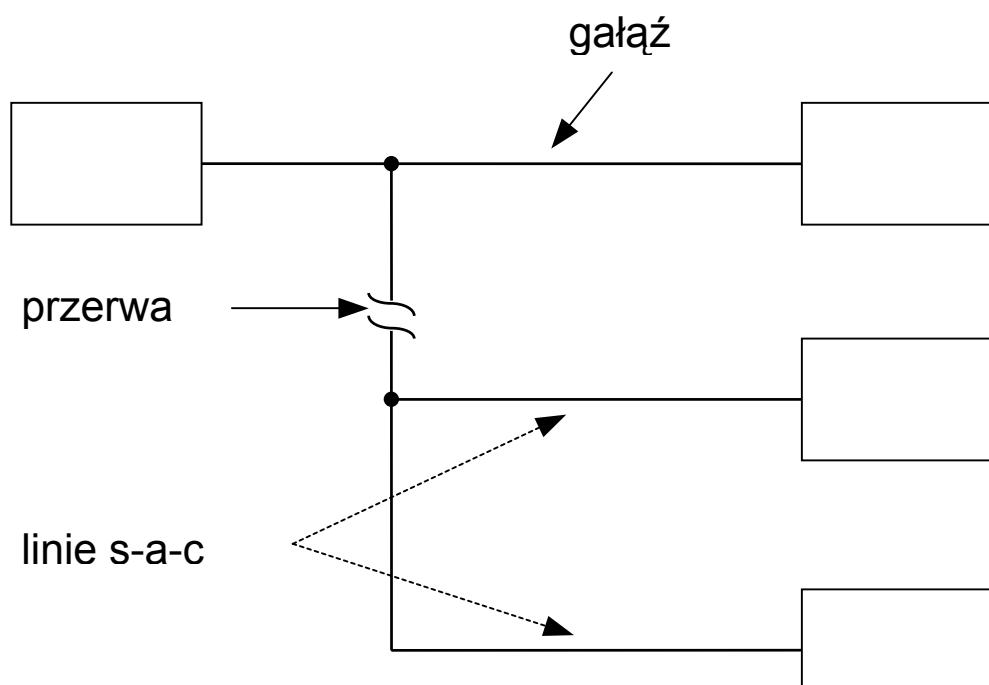
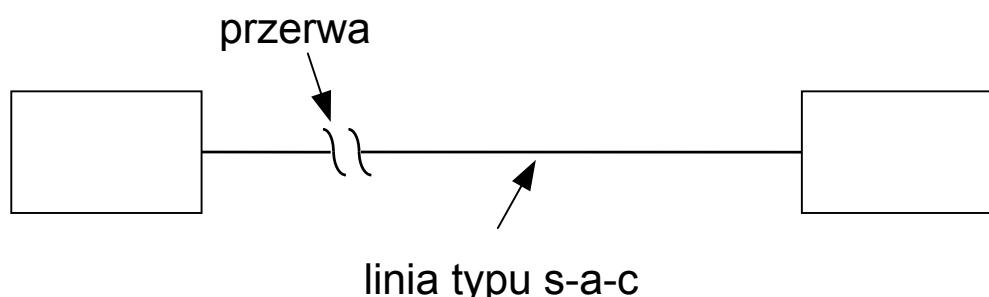
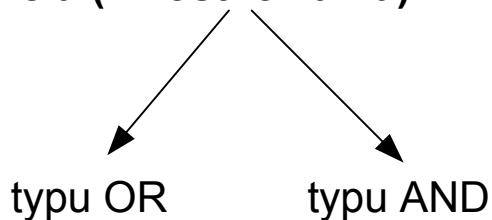
Typowe uszkodzenia logiczne

- **uszkodzenia stałosygnalowe** - powodujące stałą wartość

$c \in \{0,1\}$ na linii, oznaczenie s-a-c (stuck-at-c)

np. $x_j/0$ lub $j/0$

- **zwarcia (zmostkowania) linii**



Detekcja uszkodzeń

1. Układy kombinacyjne

Niech $Z(x)$ oznacza funkcję logiczną układu kombinacyjnego N , gdzie x jest wektorem wejściowym układu. Oznaczmy przez t wektor wejściowy i przez $Z(t)$ odpowiedź układu N na wymuszenie t . Dla układów wielowyjściowych $Z(t)$ jest również wektorem.

Obecność uszkodzenia f transformuje N do układu N_f . Założymy, że N_f jest układem kombinacyjnym z funkcją $Z_f(x)$.
Układ jest testowany poprzez sekwencję testową

$$T = \langle t_1, t_2, \dots, t_m \rangle$$

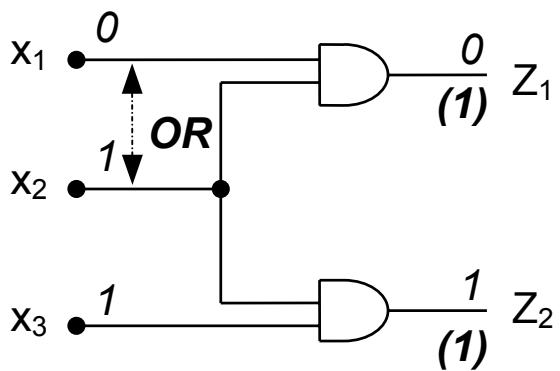
i poprzez porównanie odpowiedzi z (oczekiwana) odpowiedzią wzorcową układu N ,

$$Z(t_1), Z(t_2), \dots, Z(t_m)$$

Definicja. Wymuszenie t wykrywa uszkodzenie f , jeżeli $Z_f(t) \neq Z(t)$.

Wymuszenie t jest więc testem kontrolnym uszkodzenia f .

Przykład:



Uszkodzenie f (zmostkowanie typu OR linii x_1, x_2) zmienia funkcję układu na:

$$Z_{1f} = x_1 + x_2 \text{ (zamiast } Z_1 = x_1 x_2\text{)}$$

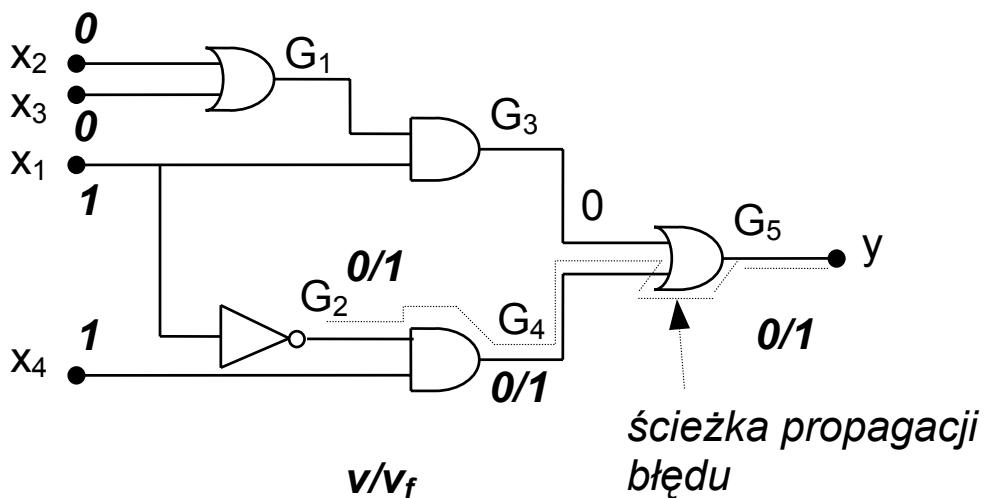
i

$$Z_{2f} = (x_1 + x_2)x_3 \text{ (zamiast } Z_2 = x_2 x_3\text{)} \\ \text{Test } 011 \text{ wykrywa uszkodzenie ponieważ } Z(011) = 01 \text{ podczas gdy } Z_f(011) = 11$$

Dla układu z pojedynczym wejściem

$$Z(x) \oplus Z_f(x) = 1$$

gdzie \oplus - oznacza operację exclusive - OR



$$w = 1001$$

$$f = G_2(s-a-1)$$

“Test w generuje błąd”

Ścieżka wg której test w zmienia wartości sygnałów nazywana jest “uczuloną” na błąd f przez test w (ścieżka pobudzona)

Uszkodzenie f jest nazywane wykrywalnym, jeśli istnieje test kontrolny w dla tego uszkodzenia tzn.

$$Y_f(w) \neq Y(w)$$

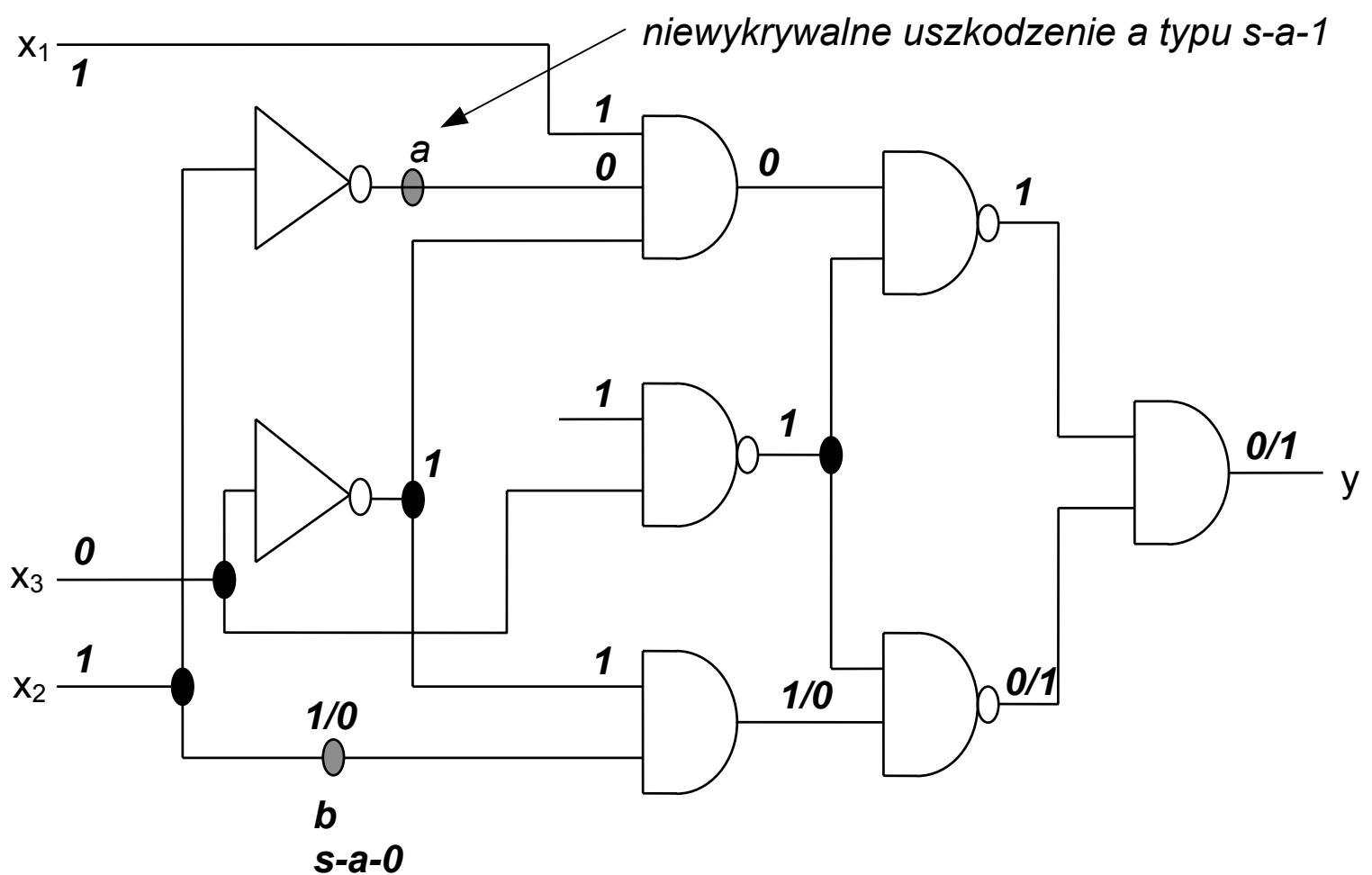
$$Y(w, f) \neq Y(w, n_0)$$

dla $Y_f(w) = Y(w)$ uszkodzenie jest niewykrywalne ponieważ nie zmienia funkcji realizowanej przez układ .

W przykładzie 3(a) błąd a $s-a-c$ jest niewykrywalny.

Przykład poniższy pokazuje w jaki sposób błąd b $s-a-0$ jest wykrywalny przez test $w = 1101$. Zauważmy, że b $s-a-0$ nie jest wykrywalny przez test w jeśli równocześnie jest obecne uszkodzenie a $s-a-1$.

a)

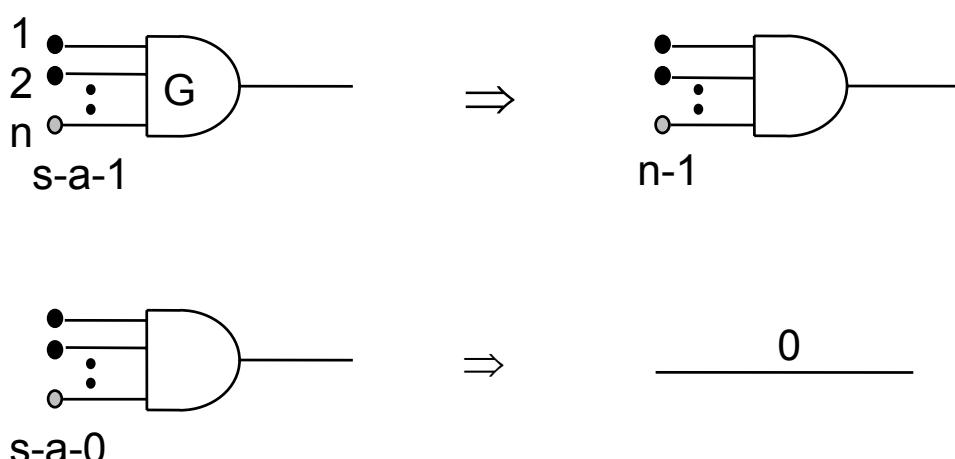


$$w = 1101$$

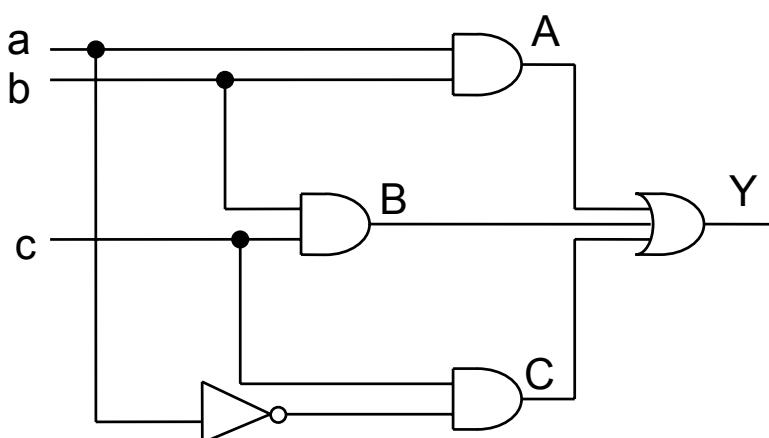
b) istnieją dwa uszkodzenia a i b

Nadmiarowość (redundancja)

Układ kombinacyjny, który zawiera niewykrywalne uszkodzenia stałosygnalowe nazywamy **redundancyjnym**, ponieważ taki układ zawsze może być uproszczony poprzez usunięcie co najmniej jednej bramki lub jej wejścia



Redundancja może być wprowadzona do układu, aby zabezpieczyć się przed hazardem. Ilustruje to poniższy rysunek



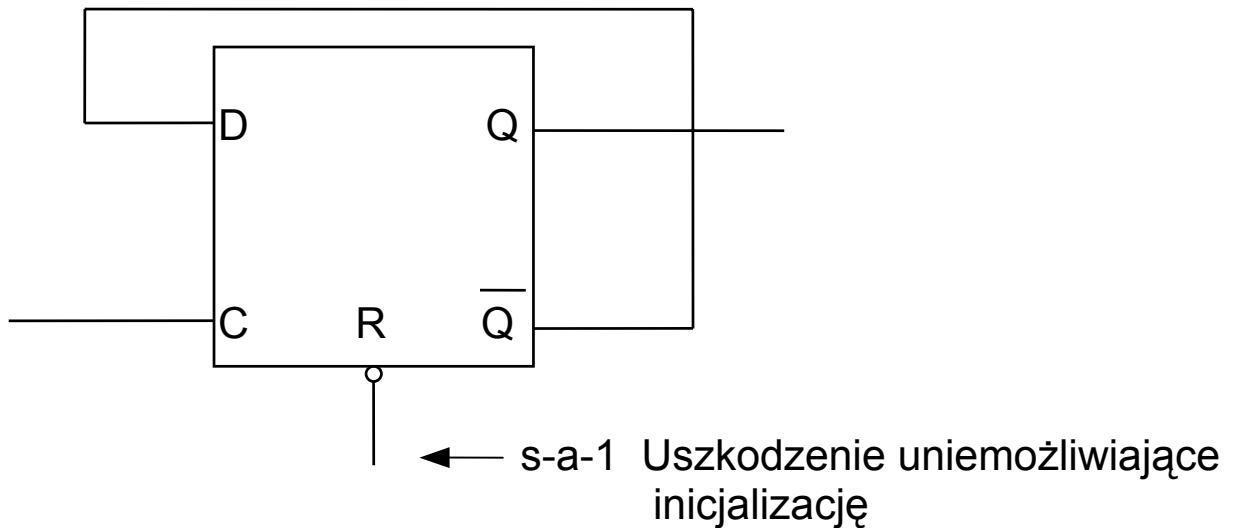
$$Y = ab + \bar{a}c + bc = ab + \bar{a}c$$

B wprowadza element **bc**, który jest nadmiarowym ale rola **B** - to ochrona przed hazardem **bc** z **B** Układ ma statyczny hazard, **B** utrzymuje "1" w czasie zmiany sygnału $111 \rightarrow 011$.

Zauważmy, że obecność **Y s-a-0** jest niewykrywalna

1. Jeżeli **f** jest wykrywalnym uszkodzeniem i **g** jest niewykrywalnym uszkodzeniem, to **f** może stać się niewykrywalnym w obecności **g**
2. Wielokrotne uszkodzenie $\{f, g\}$ może być wykrywalne nawet jeśli po jednym uszkodzeniu **f, g** są niewykrywalne

Układy sekwencyjne



Ekwiwalentność uszkodzeń

1.Układy kombinacyjne

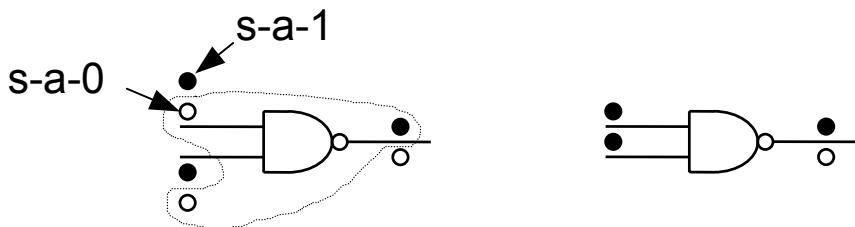
Definicja

Dwa uszkodzenia f i g nazywamy funkcjonalnie ekwiwalentnymi względem zbioru testów T , jeżeli $Y_f(t) = Y_g(t)$ dla każdego testu t należącego do zbioru T .

Test t nazywamy rozróżniającymi uszkodzenia f, g jeżeli $Y_f(t) \neq Y_g(t)$. Dla uszkodzeń ekwiwalentnych nie ma testu rozróżniającego je.

Klasy funkcjonalnej ekwiwalencji - podział możliwych uszkodzeń układu na równoważne podzbiory.

Dla bramki NAND wszystkie wejściowe uszkodzenia s-a-0 i wyjściowe s-a-1 są funkcjonalnie ekwiwalentne



Układy sekwencyjne

Dwa uszkodzenia f i g są ściśle funkcjonalnie ekwiwalentne, jeżeli odpowiednio ich układy U_f , U_g mają ekwiwalentne tablice stanów.

Dwa uszkodzenia **f** i **g** są nazywane **funkcjonalnie ekwiwalentnymi**:

jeżeli $r_f(q_{1f}, T') = r_g(q_{1g}, T')$ dla każdej sekwencji **T'**

Dominacja uszkodzeń

Niech **T_g** będzie zbiorem (wszystkich) testów wykrywających uszkodzenie **g**. Uszkodzenie **g** dominuje nad uszkodzeniem **f**, jeżeli

f i **g** są funkcjonalnie ekwiwalentne dla testu **T_g**.

Innymi słowyami, jeśli **g** dominuje nad **f**, to dowolny test **t**, który wykrywa **g** tj. $Y_g(t) \neq Y(t)$ wykrywa również **f**, ponieważ $Z_f(t)=Z_g(t)$.

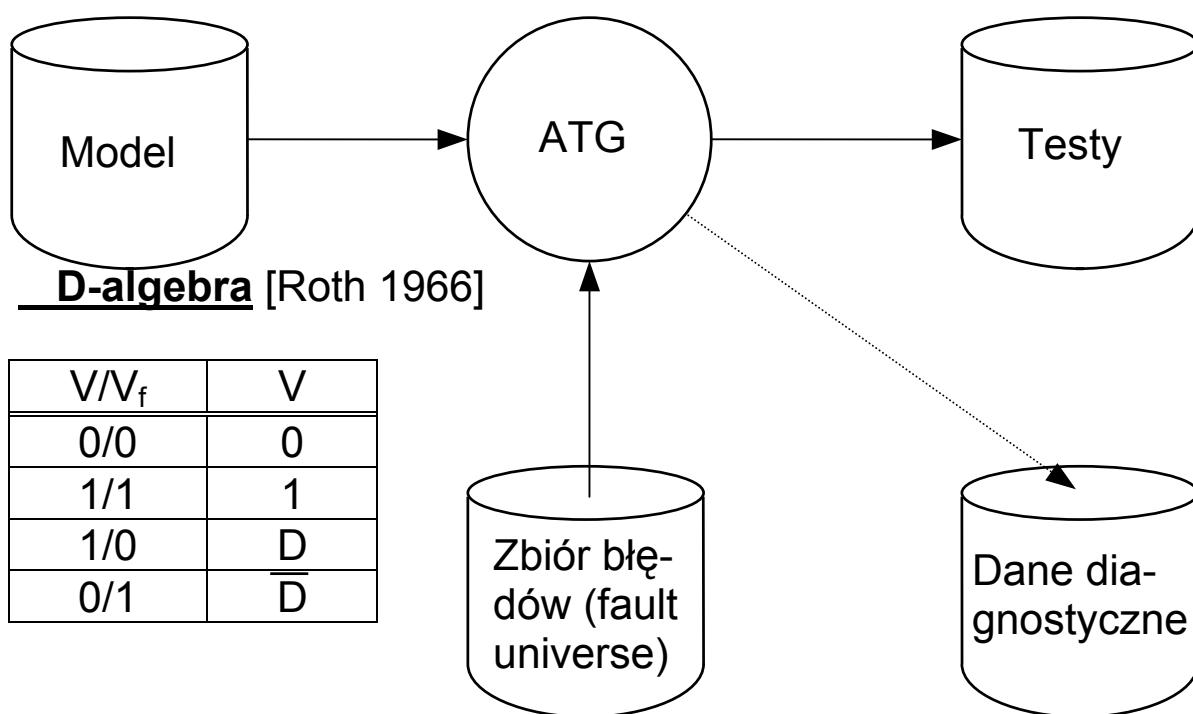
Wyznaczanie testów dla błędów pojedynczych (SSF)

Generacja testów jest złożonym problemem związanym z wieloma aspektami jak :

- ◆ koszt generacji testów,
- ◆ jakość wyznaczanych testów,
- ◆ koszt użycia testów.

Koszt generacji testów (T_G) zależy od złożoności użytej metody:

- **losowe** generowanie testów jest prostym procesem wymagającym generacji losowych wektorów, jednakże dla uzyskania testów o wysokiej jakości (mierzonej, na przykład, jako pokrycie błędów generowanymi testami) - potrzeba bardzo dużego zbioru losowych wektorów. Jeżeli testy są dłuższe - wymagają większego czasu testowania oraz większej pamięci testera.
- przy **deterministycznym** generowaniu testów wykorzystuje się informację o funkcji lub strukturze testowanego układu (model testowanego układu). Deterministyczne generowanie testów jest bardziej kosztowne niż losowe, lecz w zamian dostarcza krótsze testy (wyższej jakości). Deterministyczny generator testów (GT) może być zorientowany na błędy lub niezależny od błędów. Koszt GT zależy od złożoności układu, dla którego generuje się testy.



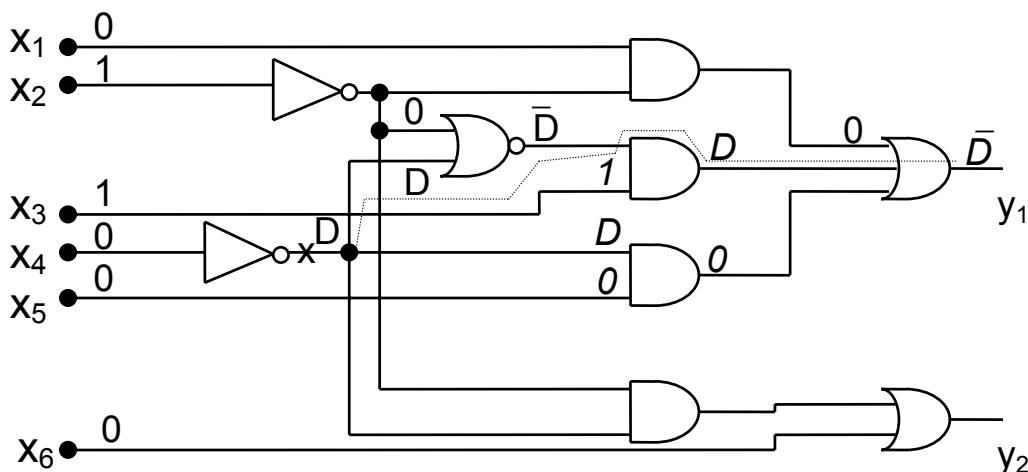
D - algebra jest to dwójka uporządkowana $\langle V, \Phi \rangle$,

gdzie $V = \{ 0, 1, D, \bar{D}, x \}$ - jest alfabetem służącym do opisu zmian powodowanych uszkodzeniem układu, a Φ jest zbiorem operacji nad tym alfabetem.

Symbol D oznacza, że błąd zmienia wartość sygnału cyfrowego z 1 na 0; oznacza to, że w układzie zdarnym sygnał jest równy 1, natomiast w układzie błędny sygnał jest równy 0.

\bar{D} - błąd zmienia wartość sygnału cyfrowego z 0 na 1; oznacza to, że w układzie zdarnym sygnał jest równy 0, natomiast w układzie błędny sygnał jest równy 1.

x - wartość sygnału jest nieokreślona i może być równa 0, 1, D lub \bar{D} .



Przystępując do opisu w/w układu, musimy znać te reakcje (na wartości wejściowe) - zbiór operacji Φ D - algebry, powinien być dobrany tak, aby za jego pomocą można było dokonać opisu układu.

Alfabet D - algebry umożliwia sformułowanie takiego opisu układu, w którym są uwidocznione wszystkie sygnały błędne i wszystkie sygnały poprawne.

Począwszy od swego źródła, błąd (utożsamiany z symbolem D lub \bar{D}) rozprzestrzenia się układzie. Linie ścieżek przenoszących błąd są opatrzone symbolami \bar{D} lub D .

Jeśli propagowany błąd osiągnie jedno z obserwowlanych wyjść układu, to stan wejścia układu odpowiadający określonym warunkom propagacji (jeżeli nie okażą się one sprzeczne) jest testem rozpatrywanego błędu.

Zdefiniujemy operacje nad alfabetem V .

Dla ustalenia uwagi rozpatrzymy dwuczęściową bramkę OR:
 $y=f_{OR}(x_1, x_2)$. Na podstawie opisu jej działania w logice dwuwartościowej

Tabela 1

x_1	x_2	f_{AND}	f_{OR}	f_{NAND}	f_{NOR}
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

$$f^1_{OR} = \{(0, 1), (1, 0), (1, 1)\}$$

$$f^0_{OR} = \{(0, 0)\}$$

Błąd obserwowany na wyjściu bramki zmienia stan x_1 lub/i x_2 z 0 na 1 (z 1 na 0). Jeżeli poprawna kombinacja wejściowa należy do f^{α}_{OR} , to błędna kombinacja wejściowa może należeć albo do f^{α}_{OR} , albo f^{β}_{OR} , gdzie $\alpha \neq \beta$, $\alpha, \beta \in \{0, 1\}$.

W pierwszym przypadku $\beta \in f^{\alpha}_{OR}$ - stan wyjścia pozostaje poprawny (nie zmienia wyjścia).

W drugim przypadku $\beta \in f^{\beta}_{OR}$ - stan wyjścia bramki zmienia się (staje się błędny) co oznacza, że błąd wejścia przenosi się na wyjście.

Szczegółowa analiza obu przypadków pozwala utworzyć tablicę definiującą operację f_{OR} nad alfabetem V. Wiersze tabeli 2 wypełniamy, rozpatrując cztery następujące możliwości oddziaływanego błędu na wejście bramki

(A - oznacza kombinację poprawną,
B - oznacza kombinację błędą):

- 1) $A \in f^0_{OR}, B \in f^0_{OR} : \{(0, 0)\} \rightarrow 0,$
- 2) $A \in f^1_{OR}, B \in f^1_{OR} : \{(\bar{D}, D), (\bar{D}, 1), (D, \bar{D}), (1, \bar{D}), (D, 1), (1, D), (0, 1), (1, 0), (1, 1)\} \rightarrow 1,$
- 3) $A \in f^1_{OR}, B \in f^0_{OR} : \{(0, D), (D, 0), (D, D)\} \rightarrow D,$
- 4) $A \in f^0_{OR}, B \in f^1_{OR} : \{(0, \bar{D}), (\bar{D}, 0), (\bar{D}, \bar{D})\} \rightarrow \bar{D}$

$$\underset{\substack{\parallel \\ A \in f^0_{OR}}}{} \rightarrow A = (0, 0) \quad f_{OR}(A) = 0$$

$$B \in f^0_{OR}, \text{ to } B = (0, 1) \text{ lub } B = (1, 0), \text{ lub } B = (1, 1) \quad f_{OR}(B) = 1$$

Analogicznie możemy określić operacje f_{AND} , f_{NAND} , f_{NOR}

Tabela 2

x_1	x_2	f_{OR}	f_{AND}	f_{NAND}	f_{NOR}
-------	-------	----------	-----------	------------	-----------

0	0	0	0	1	1
0	1	1	0	1	0
0	D	D	0	1	\bar{D}
0	\bar{D}	\bar{D}	0	1	D
1	0	1	0	1	0
1	1	1	1	0	0
1	D	1	D	\bar{D}	0
1	\bar{D}	1	\bar{D}	D	0
D	0	D	0	1	\bar{D}
D	1	1	D	\bar{D}	0
D	D	D	D	\bar{D}	\bar{D}
D	\bar{D}	1	0	1	0
\bar{D}	0	\bar{D}	0	1	D
\bar{D}	1	1	\bar{D}	D	0
\bar{D}	D	1	0	1	0
\bar{D}	\bar{D}	D	\bar{D}	D	D

Podobnie dla synchronicznego przerzutnika D tabela ma postać:

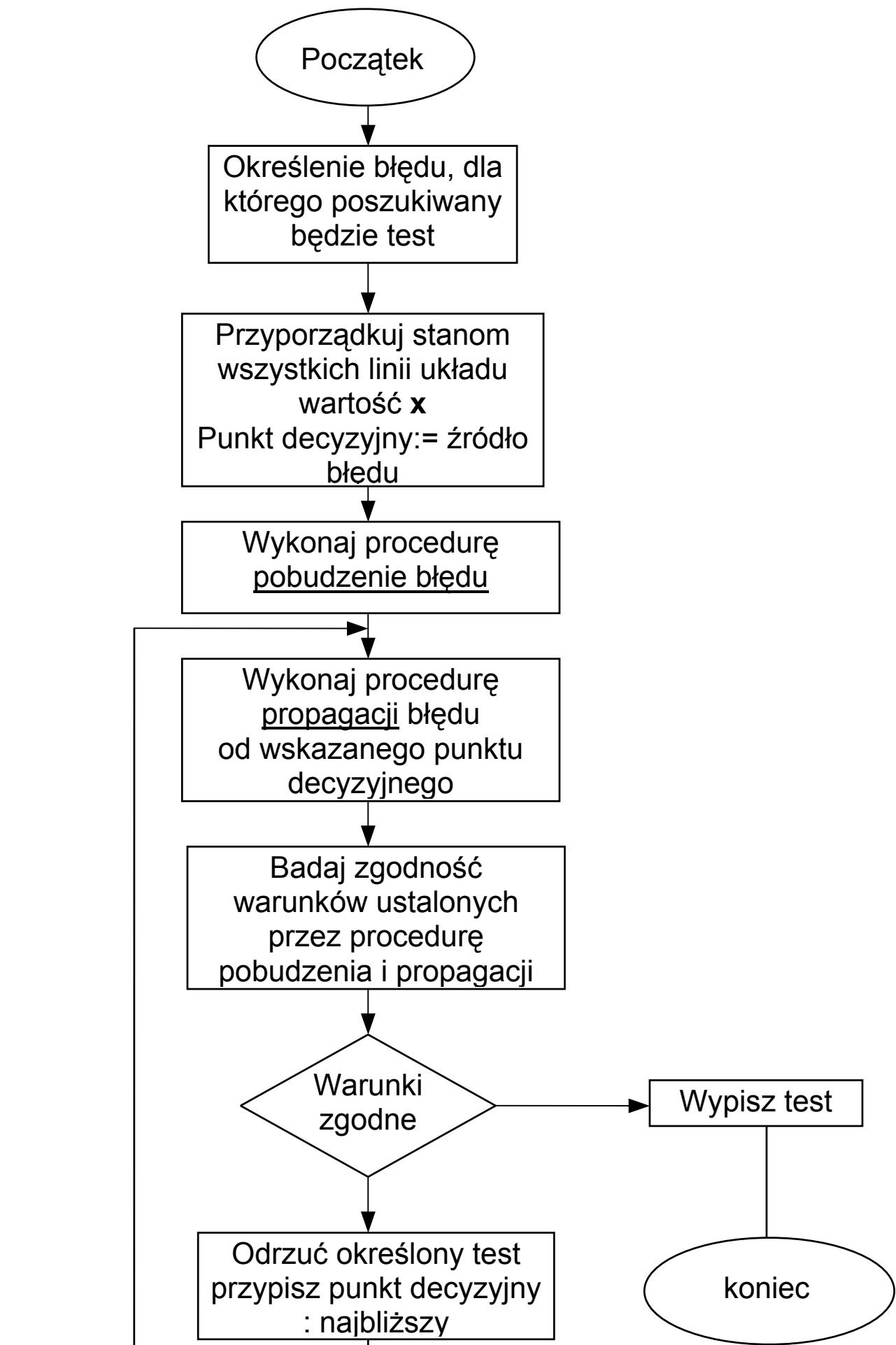
q \ DC	00	0D	0 \bar{D}	10	11	1D	1 \bar{D}	D0	D1	DD	D \bar{D}	$\bar{D}0$	$\bar{D}1$	$\bar{D}D$	$\bar{D} \bar{D}$
	01														
0	00	0	0	0	1	D	\bar{D}	0	D	D	0	0	\bar{D}	0	\bar{D}
1	10	D	\bar{D}	1	1	1	1	1	D	1	D	1	D	\bar{D}	1
D															
\bar{D}															

Jeżeli znane są operacje D-algebry niezbędne do opisania wszystkich elementów układu, to można formułować algorytmy systematycznego wyznaczania warunków zapewniających proporcję.

D-algorytm

D-algorytm składa się z 3 głównych procedur:

- 1) pobudzenie źródła błędu
- 2) propagacja błędu
- 3) badanie zgodności warunków pobudzenia i propagacji



- wymuszanie stanu D lub \bar{D} w miejscu występowania błędu. Stan D jest wymuszany w przypadku błędu s-a-0, natomiast \bar{D} w przypadku błędu s-a-1.

Niech j będzie wyjściem modułu. Jeżeli na linii j wystąpił błąd s-a-0 (s-a-1), to wymuszenie na niej wartości D(\bar{D}) polega na ustawieniu wejścia modułu w takim stanie, któremu poprawne odpowiada wartość 1(0) wyjścia.

Przykład pobudzenia rys.1 . pobudzenie źródła błędu ($x/0$) polega na wymuszeniu $x_4 = 0$.

Przy realizacji procedury pobudzenia źródła błędu wykorzystuje się tzw. **pierwotne D-sześciiany**.

Bramka	x_1	x_2	f
AND	0	x	\bar{D}
	x	0	D
	1	1	D
OR	1	x	D
	x	1	D
	0	0	\bar{D}
NAND	0	x	D
	x	0	D
	1	1	\bar{D}
NOR	1	x	\bar{D}
	x	1	\bar{D}
	0	0	D

Określają one stany wejść bramek potrzebne do wymuszenia D lub \bar{D} na ich wyjściach.

Propagacja błędu

Procedura propagacji błędu tworzy tzw. ścieżkę pobudzenia, pomiędzy źródłem błędu a jedynym z obserwowalnym wyjściem układu

Ścieżkę nazywamy pobudzoną, jeżeli zmiana stanu linii poczynającej ścieżkę powoduje zmianę stanu na jej końcu.

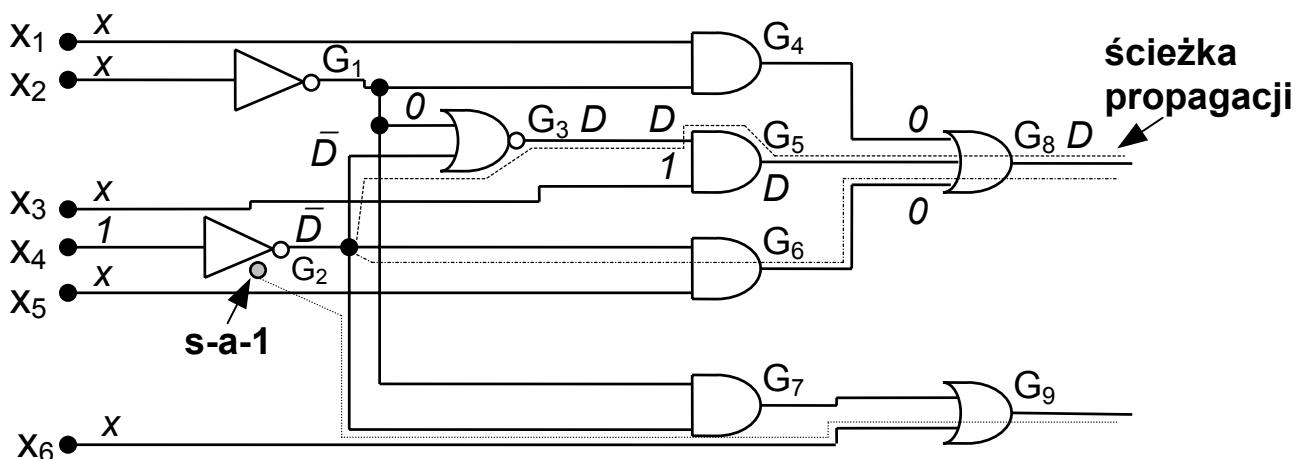
Ścieżki pobudzone tworzy się na podstawie tzw. przesyłowych D-sześciianów.

Dla bramek podaje je tabela:

Bramka	x_1	x_2	f
AND	D	1	D
	1	D	D
	\bar{D}	1	\bar{D}
	1	\bar{D}	\bar{D}
OR	D	0	D
	0	D	D
	\bar{D}	0	\bar{D}
	0	\bar{D}	\bar{D}
NAND	D	1	\bar{D}
	1	D	\bar{D}
	\bar{D}	1	D
	1	\bar{D}	D
NOR	D	0	\bar{D}
	0	D	\bar{D}
	\bar{D}	0	D
	0	\bar{D}	D

Przesyłowe D-sześciiany modułu logicznego wyznacza się wprost z jego definicji funkcji w D-algebrze. W szczególności powyższą tabelę otrzymano z tabeli 2, wybierając z niej wiersze, które odpowiadają proporcji D i \bar{D} przez bramki.

Przykład



Poszukiwany jest test błędu G₂ s-a-1

Pobudzenie : \bar{D} na G₂

Propagacja : propaguje kolejno przez bramki G_3 , G_5 , G_8 . W wyniku wykonania dwóch pierwszych procedur D-algorytmu zostają określone stany niektórych linii układu.

Uzupełnienie i badanie zgodności stanów jest treścią ostatniej procedury. Wykorzystuje się w niej tzw. pierwotne sześciiany wyjść modułów.

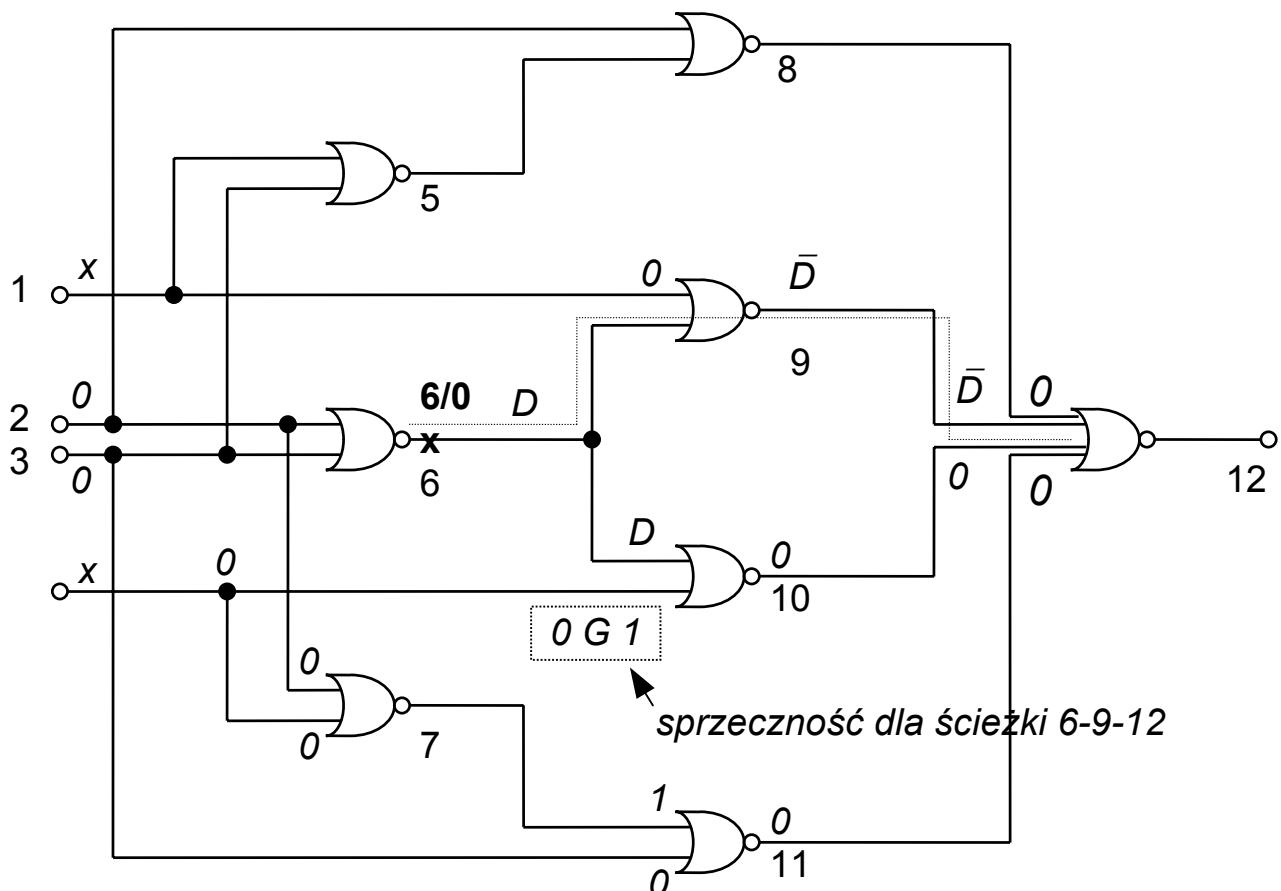
Bramka	x_1	x_2	f
AND	0	x	0
	x	0	0
	1	1	1
OR	1	x	1
	x	1	1
	0	0	0
NAND	0	x	1
	x	0	1
	1	1	0
NOR	1	x	0
	x	1	0
	0	0	1

Zastosowanie D-algorytmu dla układów kombinacyjnych nie stwarza trudności, poza złożonością obliczeniową. Poniższy przykład pokazuje, że próby skutecznego pobudzenia każdej ze ścieżek 6-9-12 i 6-10-12 oddzielnie dla błędu 6/0 zawodzą. W toku wykonania D-algorytmu dopiero ostatnia, trzecia próba

przeprowadzenia błędu na wyjście układu kończy się sukcesem.

Jeżeli od źródła błędu do obserwowalnego wyjścia układu prowadzi n ścieżek, to wykonanie D-algorytmu może, w najgorszym przypadku wymagać podjęcie $2^n - 1$ prób znalezienia poprawnych propagacji. Liczba $2^n - 1$ może być duża, szczególnie w układach zawierających wiele rozgałęzień.

Błąd 6/0 ma tylko jeden test równy (0, 0, 0, 0):



Testowanie układów sekwencyjnych

Metody testowania :

- **badania struktury** (ang. *Circuit-testing approach*) - zakłada znajomość struktury układu oraz klasy prawdopodobnych jego uszkodzeń;
- **identyfikacji** (ang. *Machine-identification approach*) - zakłada znajomość tablicy przejść maszyny sekwencyjnej (automatu skońzonego).

Definicja maszyny sekwencyjnej

$$M = \langle S, X, Z, \delta(X, S), \lambda(X, S) \rangle$$

gdzie:

$S = \{s_1, s_2, \dots, s_n\}$ - skończony zbiór stanów wewnętrznych
o liczności n ;

$X = \{x_1, x_2, \dots, x_q\}$ - skończony zbiór stanów wejściowych
o liczności q ;

$Y = \{y_1, y_2, \dots, y_d\}$ - skończony zbiór stanów wyjściowych
o liczności d ;

$\delta : X \times S \rightarrow S$ - funkcja przejścia;

$\lambda : X \times S \rightarrow Z$ - funkcja wyjścia;

Założenia dla maszyny sekwencyjnej

- Zbiór S jest zbiorem skończonym.
- Maszyna M jest deterministyczną, tzn.

$$|\delta(x, s)| = 1, \quad \forall (x, s) \in X \times S.$$

- Maszyna M jest spójna, tzn.

$$\forall s \in S \exists x \in X : s \in \delta(x, s_0);$$

gdzie: s_0 - stan początkowy (inicjalny);

- Maszyna M jest zredukowaną, tzn. że każda para jego stanów jest rozróżnialna.

Założenia dla uszkodzeń

- Maszyna, w której wystąpiło uszkodzenie jest określona przez funkcje λ_1 i δ_1 .
- Uszkodzenie nie zwiększa liczby stanów.
- W czasie testowania nie powstają nowe uszkodzenia.

Przyjmuje się, że maszyna M znajduje się w stanie niezdolności, jeżeli po zadaniu sekwencji wejściowej x przechodzi do stanu s_i podczas gdy powinna przejść do stanu s_j . Jak również jeżeli po podaniu sekwencji x na wyjściu uzyskujemy sekwencję z_i a powinna pojawić się sekwencja z_j .

Definicje

Sekwencją rozróżniającą nazywamy sekwencję wejściową X_d ($X_d \subset X$), która podana na wejście maszyny M powoduje wygenerowanie n różnych sekwencji wyjściowych, zależnie od stanu początkowego maszyny M .

Sekwencją lokalizującą (synchronizującą) nazywamy sekwencję wejściową X_o ($X_o \subset X$), jeżeli odpowiadająca jej sekwencja wyjściowa określa stan końcowy maszyny M niezależnie od jej stanu początkowego.

Sekwencją sprawdzającą (testującą) X_s ($X_s \subset X$) nazywamy sekwencję wejściową, przeprowadzającą maszynę M , przez wszystkie stany i pozwalającą na stwierdzenie poprawności jej działania poprzez obserwację wyjść maszyny.

Eksperimentem sprawdzającym nazywamy działanie polegające na podaniu sekwencji sprawdzającej na wejście maszyny badanej i porównaniu otrzymanej sekwencji wyjściowej z wynikami poprawnymi określonymi przez funkcje δ i λ .

Stanem początkowym s_0 nazywamy stan, w którym znajduje się maszyna bezpośrednio przed eksperimentem sprawdzającym.

Sekwencją wejściowo-wyjściową nazywamy parę sekwencji: wejściową – podawaną na wejście maszyny i odpowiadającą jej sekwencję wyjściową, zapisywane w postaci:

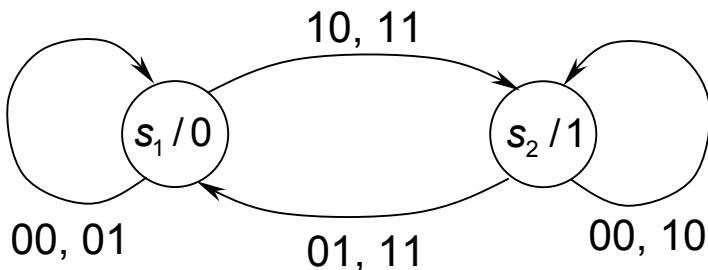
$$\begin{cases} \text{sekwencja wejściowa} \\ \text{sekwencja wyjściowa} \end{cases}$$

Stan s_i nazywamy **rozpoznanym**, gdy funkcje $\lambda(x_j, s_i)$ oraz $\delta(x_j, s_i)$ przyjmują dla każdego x_j wartości zgodne z tablicą przejść M .

Przykład

Wyznaczyć rozróżniającą, lokalizującą oraz sprawdzającą dla przerzutnika **JK**.

Graf przejść przerzutnika



Tablica przejść przerzutnika

Stan \ JK	00	01	11	10	Stan wyjścia
s_1	$s_1 / 0$	$s_1 / 0$	$s_2 / 1$	$s_2 / 1$	0
s_2	$s_2 / 1$	$s_1 / 0$	$s_1 / 0$	$s_2 / 1$	1

Sekwencja rozróżniająca: $X_d = \{(00), (11)\}$.

Sekwencja lokalizująca: $X_o = \{(01), (10)\}$.

Sekwencja sprawdzająca:

$$X_s = \{(01, 00, 11, 00, 11, 00, 10, 00)\}.$$

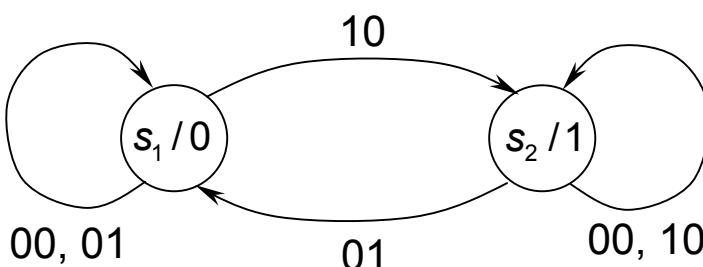
Sekwencja wyjściowa odpowiadająca danej sekwencji wejściowej:

$$0, 0, 1, 1, 0, 0, 1, 1$$

Przykład

Wyznaczyć rozróżniającą, lokalizującą oraz sprawdzającą dla przerzutnika **RS**.

Graf przejść przerzutnika



Wyznaczanie sekwencji rozróżniającej

Niepewnością nazywamy zbiór stanów S' ($S' \subset S$) maszyny, o którym wiadomo, że zawiera stan wyróżniony.

Niepewnością wstępna jest każdy zbiór S'' ($S'' \subset S'$), który zawiera stan początkowy s_0 .

Niepewność wynikająca z zastosowania sekwencji X nazywa się następstwem X .

Drzewo następstw – jest drzewem, w którym każda gałąź jest związana z pewnym wektorem niepewności.

Wektor niepewności, którego składniki są pojedynczymi stanami nazywamy **trywialnym** np. $(s_1)(s_2)(s_3)(s_4)$.

Wektor niepewności, którego składniki są pojedynczymi stanami lub powtórzeniami tych samych stanów nazywamy **homogenicznym** np. $(s_1s_1s_2)(s_3)(s_4)$.

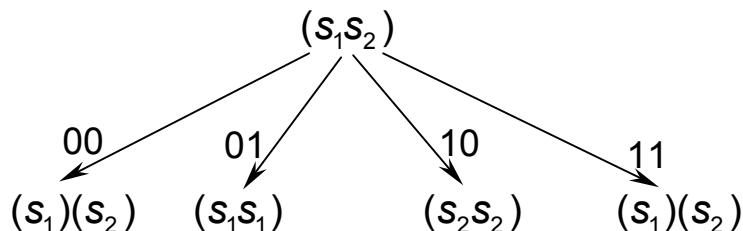
Drzewo następstw nazywamy **drzewem rozróżniającym** jeżeli:

- na poziomach $k, k+1$ pojawi się identyczny wektor niepewności;
- gałąź jest związana z wektorem homogenicznym;
- gałąź jest związana z wektorem trywialnym.

Każda droga w drzewie następstw zaczynająca się od niepewności wstępnej, a kończąca się wektorem trywialnym, odpowiada sekwencji rozróżniającej X_d dla danej maszyny.

Przykład

Określić przy pomocy drzewa rozróżniającego sekwencję rozróżniającą dla przerzutnika JK.

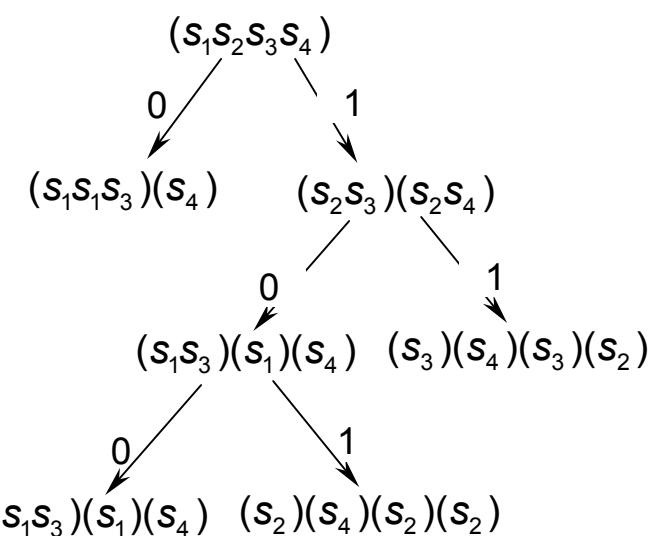


Sekwencja rozróżniająca: $X_d = \{(00), (11)\}$.

Przykład

Określić przy pomocy drzewa rozróżniającego sekwencję rozróżniającą dla maszyny o podanej poniżej tabeli przejść.

Stany \ X	0	1
Stany		
s_1	$s_1 / 0$	$s_2 / 0$
s_2	$s_1 / 0$	$s_3 / 0$
s_3	$s_3 / 0$	$s_4 / 1$
s_4	$s_4 / 1$	$s_2 / 1$

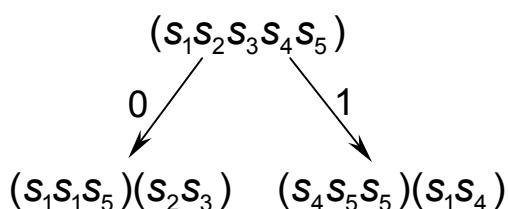


Sekwencja rozróżniająca: $X_d = \{(1,1), (1,0,1)\}$

Przykład

Określić przy pomocy drzewa rozróżniającą sekwencję rozróżniającą dla maszyny o podanej poniżej tabeli przejść.

Stany \ X	0	1
S ₁	s ₁ / 0	s ₄ / 1
S ₂	s ₁ / 0	s ₅ / 1
S ₃	s ₅ / 0	s ₁ / 0
S ₄	s ₃ / 1	s ₄ / 0
S ₅	s ₂ / 1	s ₅ / 1



Brak dla maszyny M sekwencji rozróżniającej.

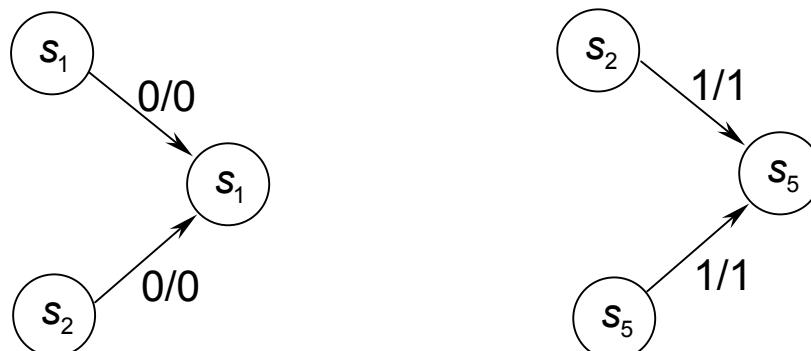
Algorytm przekształcania maszyny M w maszynę M' mającą sekwencję rozróżniającą.

1. Utworzyć na podstawie tablicy przejść tablicę testowania określającą stany, do których przechodzi maszyna z każdej niepewności obejmującej dwa stany.
2. Wyznaczyć z tablicy testowania graf testowania, nie zawierający niepewności obejmujących stany nierożóżnialne.
3. Jeżeli tablica testowania zawiera stany nierożóżnialne, to należy je rozróżnić, dodając dodatkowe wyjście (lub wyjścia).
4. Jeżeli graf testowania zawiera pętle, to należy je rozewrzeć, dodając dodatkowe wyjście (lub wyjścia).

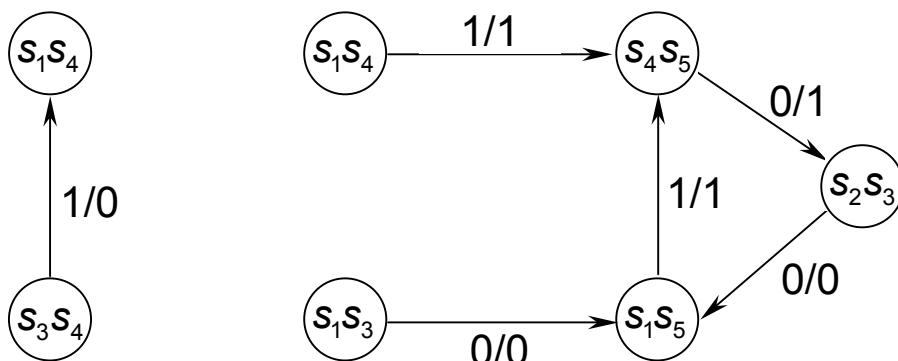
Tablica testowań dla maszyny M z poprzedniego przykładu.

X Stany \	0/0	0/1	1/1	1/1
Stany	S_1	---	S_4	---
S_1	S_1	---	S_4	---
S_2	S_1	---	S_5	---
S_3	S_5	---	---	S_1
S_4	---	S_3	---	S_4
S_5	---	S_2	S_5	---
S_1S_2	S_1S_1	---	S_4S_5	---
S_1S_3	S_1S_5	---	---	---
S_1S_4	---	---	---	---
S_1S_5	---	---	S_4S_5	---
S_2S_3	S_1S_5	---	---	---
S_2S_5	---	---	S_5S_5	---
S_3S_4	---	---	---	S_1S_4
S_4S_5	---	S_2S_3	---	---

Stany nieroóżnialne



Graf testowania



Graf testowania

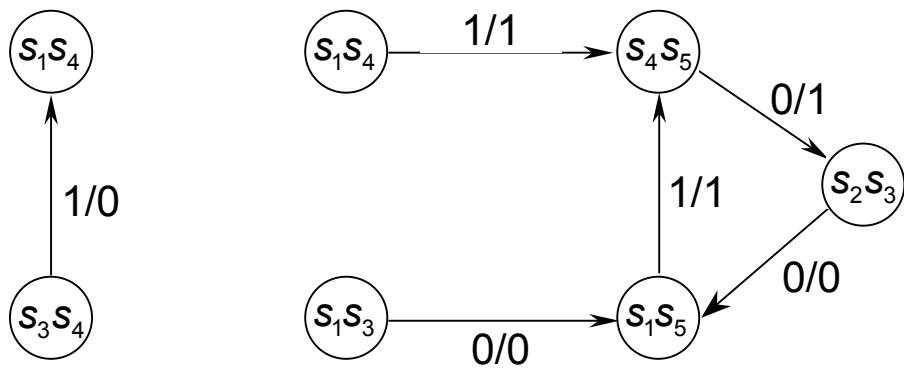


Tabela przejść dla maszyny M' przekształconej z maszyny M

		x	0	1	
		Stan	$S, z_1 z_2$	$S, z_1 z_2$	
1	s_1	$S_1, 00$	$S_4, 10$		
	s_2	$S_1, 01$	$S_5, 10$		
3	s_3	$S_5, 0-$	$S_1, 0-$		
	s_4	$S_3, 10$	$S_4, 0-$		
2	s_5	$S_2, 11$	$S_5, 11$		

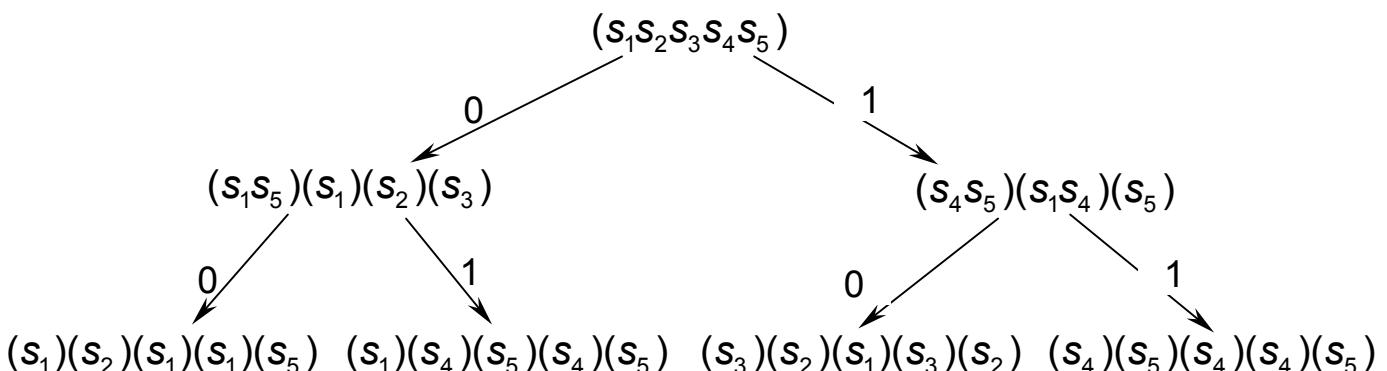
Uwaga. Jako stan dowolny przyjmujemy 0.

Stany nierozróżnialne: $s_1 s_2$ przy $x = 0$; $s_2 s_5$ przy $x = 1$. 1

Cykl: $s_1 s_5 - s_4 s_5 - s_2 s_3 - s_1 s_5$. 2

Najdłuższa droga w grafie: $s_1 s_2 - s_4 s_5 - s_2 s_3 - s_1 s_5$. 3

Drzewo następstw dla maszyny M'



Sekwencja rozróżniająca: $X_d = \{(00), (01), (11), (10)\}$.

Warunki na diagnozowalność maszyny:

- brak cykli w grafie testowania maszyny;
- brak stanów nieroóżnialnych w tablicy testowania maszyny.

Jeżeli najdłuższa droga w acyklicznym grafie testowania zawiera k łuków, a tablica testowania nie ma stanów nieroóżnialnych, to sekwencja rozróżniająca składa się co najwyżej z $k + 1$ symboli.

Każdej mocno spójnej maszynie sekwencyjnej M odpowiada w pełni diagnozowalna maszyna M' , którą otrzymujemy z M poprzez rozszerzenie jej alfabetu wyjściowego.

Metoda sprawdzania poprawności tablicy przejść maszyny sekwencyjnej

Oznaczenia

α - sekwencja kontrolująca poprawność działania sekwencji X_d ;

β - sekwencja sprawdzająca wszystkie przejścia w maszynie sekwencyjnej;

$T(s_i, s_j)$ - sekwencja przeprowadzająca maszynę M ze stanu s_i do stanu s_j ;

$Q_i(s_i, x_d)$ - stan do, którego przechodzi maszyna po zastosowaniu sekwencji x_d , jeżeli stanem początkowym był s_i . Stan s_i nazywamy d -rozpoznanym, natomiast stan Q_i nazywamy q -rozpoznanym.

S^* - zbiór stanów będących źródłami, czyli stanami do których w grafie powstały po zastosowaniu sekwencji x_d dla wszystkich stanów nie dochodzą żadne łuki.

S^R - zbiór stanów rozpoznanych;

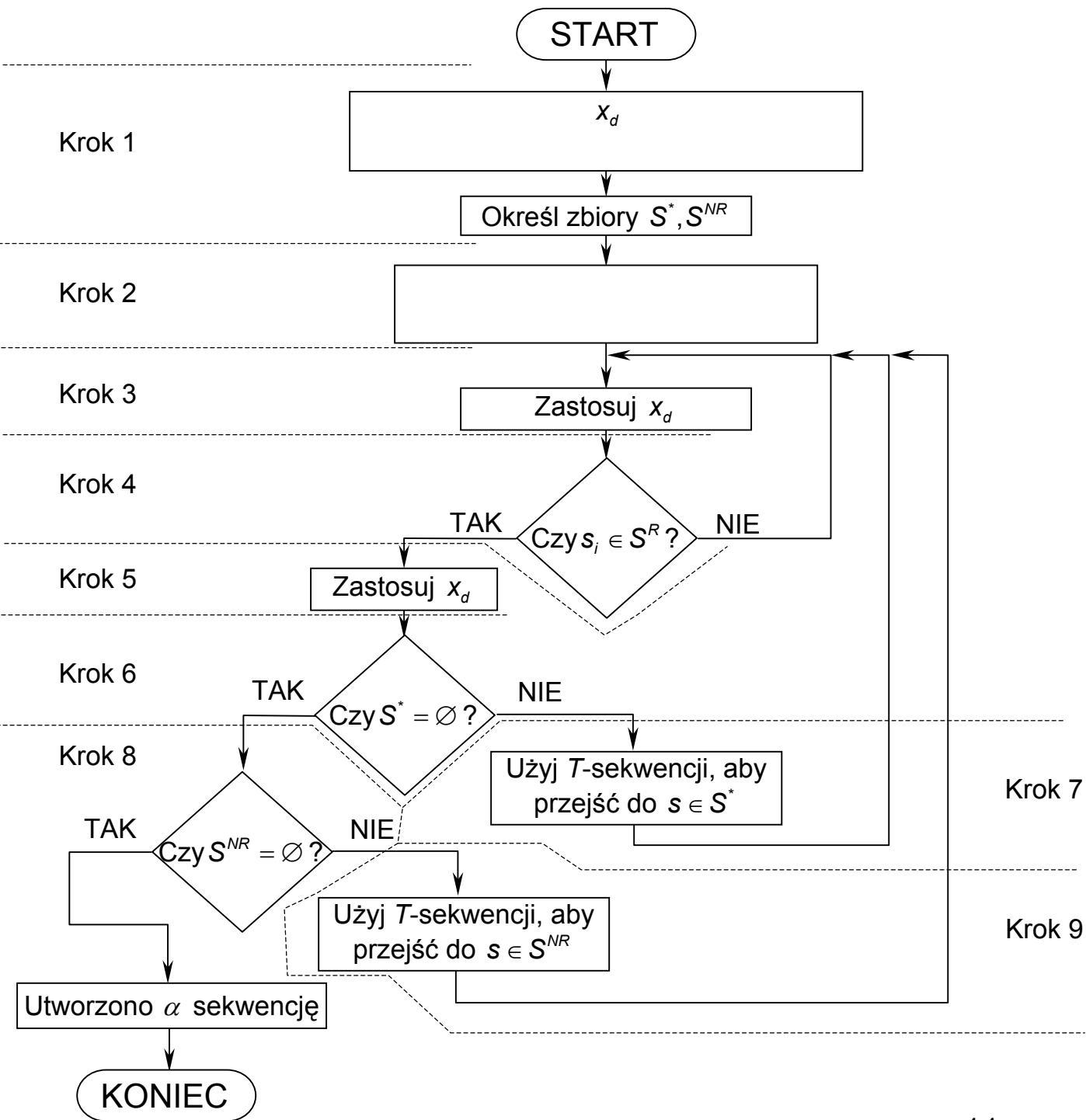
S^{NR} - zbiór stanów nierożpoznanych.

Algorytm wyznaczania α sekwencji

α sekwencja jest wyznaczana poprzez zastosowanie $X_d X_d$ do każdego stanu maszyny. Pierwsza sekwencja X_d rozróżnia stan s_i , a druga wyznacza stan Q_i . Przejścia pomiędzy stanami są uzyskiwane dzięki sukcesywnemu stosowaniu T -sekwencji spełniającej następujące warunki:

- stanem początkowym dla T -sekwencji musi być stan q -rozpoznany;
- stan końcowy musi być elementem zbioru S^* lub S^{NR} ;

ALGORYTM



Przykład

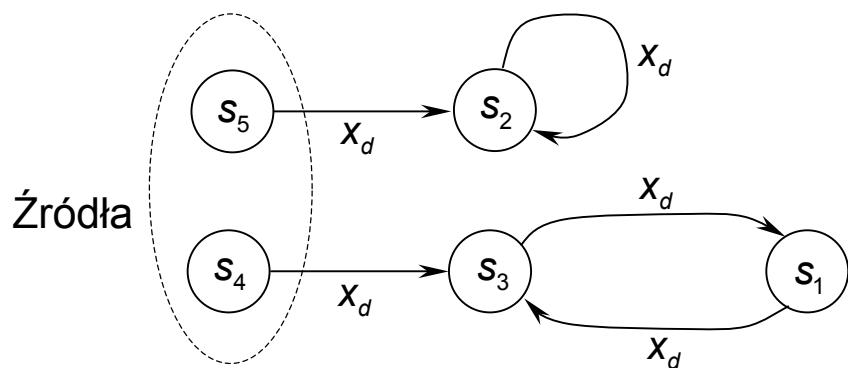
Wyznaczyć α -sekwencję dla maszyny M' , przyjmując: $x_d = 10$.

Tabela x_d

Stan \ x	0	1
Stan	$S, z_1 z_2$	$S, z_1 z_2$
s_1	$S_1, 00$	$S_4, 10$
s_2	$S_1, 01$	$S_5, 10$
s_3	$S_5, 0-$	$S_1, 0-$
s_4	$S_3, 10$	$S_4, 0-$
s_5	$S_2, 11$	$S_5, 11$

s_i	s_1	s_2	s_3	s_4	s_5
Q_j	s_3	s_2	s_1	s_3	s_2
$z_1 z_1$	11	11	00	01	11
$z_2 z_2$	00	01	00	00	11

Graf x_d



Wyznaczona α -sekwencja

Krok	3	3	3	5	7	3	3	5	8
α -sekw.	10	10	10	10	100	10	10	10	10
Stan	s_4^d	s_3^d	s_1^d	s_3^d	s_1^q	s_5^d	s_2^d	s_2^d	s_2^q
$z_1 z_1$	01	00	11	00	110	11	11	11	11
$z_2 z_2$	00	00	00	00	000	11	01	01	

Algorytm wyznaczania β sekwencji

Graf przejść dla wszystkich sekwencji $x_i x_d$, $i = 1, \dots, q$ jest nazywany grafem β .

$$G_\beta = \langle S, U \rangle$$

gdzie:

S – zbiór węzłów reprezentujących stany maszyny sekwencyjnej;

U – zbiór łuków odpowiadających przejściom pomiędzy stanami

$$|U| = n \cdot q ;$$

$\lambda(G_\beta)$ - liczba składowych spójności grafu G_β ;

$\mu^-(s_k)$ - stopień wewnętrzny węzła s_k ;

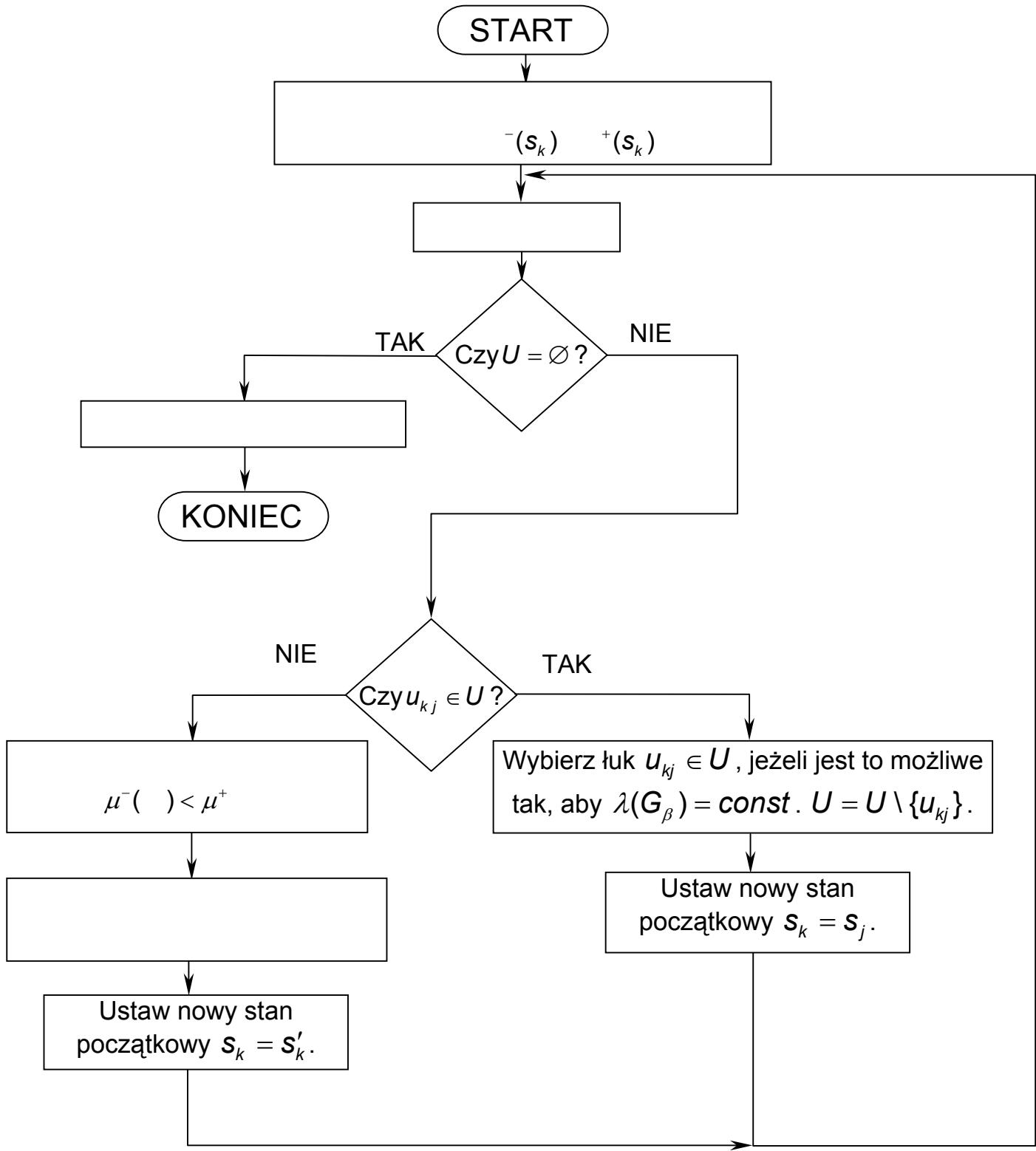
$\mu^+(s_k)$ - stopień zewnętrzny węzła s_k ;

Ścieżką – w grafie G_β nazywamy naprzemienny zbiór węzłów i łuków, w którym każdy z łuków występuje tylko raz.

Pokryciem grafu G_β - nazywamy podział zbioru łuków tego grafu na ścieżki.

Problem wyznaczenia β -sekwencji jest równoważny problemowi poszukiwania minimalnego pokrycia grafu β .

ALGORYTM



Przykład

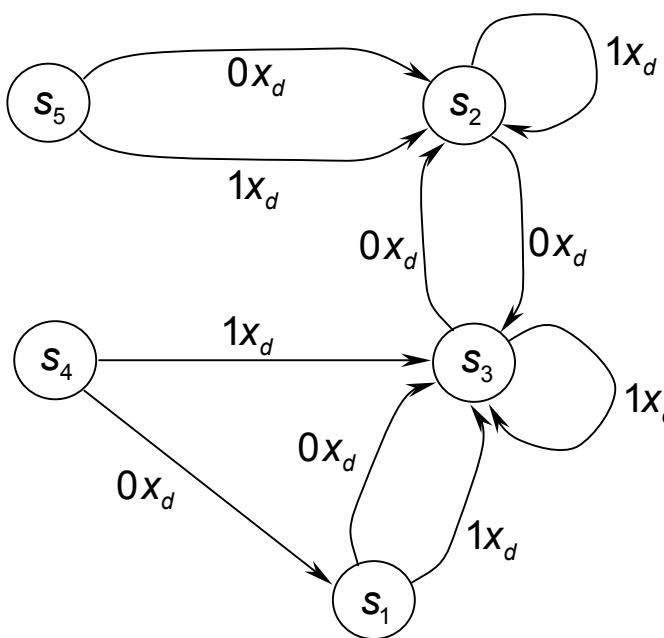
Wyznaczyć β -sekwencję dla maszyny M' , przyjmując: $x_d = 10$.

Tabela X_d

$x \backslash Stan$	0	1
Stan	$S, z_1 z_2$	$S, z_1 z_2$
s_1	$S_1, 00$	$S_4, 10$
s_2	$S_1, 01$	$S_5, 10$
s_3	$S_5, 0-$	$S_1, 0-$
s_4	$S_3, 10$	$S_4, 0-$
s_5	$S_2, 11$	$S_5, 11$

s_i	s_1	s_2	s_3	s_4	s_5
Q_j	s_3	s_2	s_1	s_3	s_2
$z_1 z_1$	11	11	00	01	11
$z_2 z_2$	00	01	00	00	11

Graf β



Pokrycia grafu G_β :

$$S_1^\beta = S_5 S_2 S_2 S_3 S_3 S_2 - S_5 S_2 - S_1 S_3 - S_4 S_3 - S_4 S_1 S_3;$$

$$S_2^\beta = S_4 S_1 S_3 S_3 S_2 S_2 S_3 - S_5 S_2 - S_5 S_2 - S_1 S_3 - S_4 S_3.$$

Przyjmując, że $s_0 = s_2$ dla S_1^β otrzymujemy β :

$$10x_d 1x_d 0x_d 1x_d 0x_d 11x_d 00x_d 111x_d 110x_d 1x_d.$$

Kompletna β oraz sekwencja wyjściowa:

β -sekw.	101011001011001011100010111101101011
0	
$z_1 z_1$	11111110110110111110011010010110010
1	

\mathbf{z}_2	\mathbf{z}_2	<table border="1"><tr><td>01010111000000110111100000000000000000</td></tr><tr><td>0</td></tr></table>	01010111000000110111100000000000000000	0
01010111000000110111100000000000000000				
0				

Testowanie układów LSI oraz VLSI

Metody testowania stosowane dla układów SSI oraz MSI nie spełniają wymagań dla zastosowania ich do testowania układów LSI oraz VLSI ze względu na:

- złożoną i nieregularną strukturę układów SSI oraz VLSI zawierającą bardzo dużą liczbę układów kombinacyjnych i sekwencyjnych;
- liczba wejść i wyjść układu, dostępnych dla celów testowania, jest ograniczona przez liczbę wyprowadzeń układu;
- punkty sieci logicznej układu nie są dostępne dla celów testowania;
- model uszkodzeń przyjęty przy generacji i weryfikacji testów dla układów SSI oraz MSI nie jest adekwatny dla układów VLSI;
- czas realizacji oraz wkład środków niezbędnych do przeprowadzenia testowania metodą klasyczną często przekracza, w praktycznych zastosowaniach, wymagania stawiane przez użytkownika.

Metody generacji testów dla układów SSI oraz MSI zakładają zwykły bramkowy model układu. Czas potrzebny do automatycznego wygenerowania testów można w przybliżeniu określić wzorem:

$$T = KN^3$$

gdzie:

K – współczynnik proporcjonalności,

N – liczba bramek układu.

Przykładowo dla układu LSI, którego model zawiera 10000 bramek, generacja testu trwałaby 10^6 razy dłużej niż dla układu złożonego ze 100 bramek.

Aby opracować metodę testowania układu LSI lub VLSI należy:

- podzielić układ na *obiekty testowania*;
- zdefiniować *rdzeń testowania*;
- określić *kolejność testowania*.

Przykładowe obiekty testowania dla mikroprocesora:

- magistrala danych i adresowa;
- pamięć ROM oraz RAM;
- dekoder adresu z układem sterowania;
- jednostka ALU;
- itp.

Rdzeniem testowania nazywamy minimalny zbiór podzespołów układu niezbędnych do przetestowania pierwszego obiektu. Zakłada się, że podzespoły zaliczone do *rdzenia testowania* mogą znajdować się w stanie niezdatności.

Ustalenie kolejności testowania wymaga uwzględnienia dwóch kryteriów:

- Największego prawdopodobieństwa błędów: *Obiekty testowane z największym prawdopodobieństwem uszkodzenia należy zbadać w pierwszej kolejności.* Przykładem takiego obiektu dla mikroprocesora jest magistrala danych, która jest szczególnie podatna na niezdatności, ponieważ ma dużą liczbę połączeń i źródeł sygnałów oraz często dość długie przewody.
- Najmniejszy rdzeń testowania: *Ze względu na to, że minimalny rdzeń testowania charakteryzuje się najmniejszą liczbą błędów.*

Testy umożliwiające wykrycie błędów w układach LSI oraz VLSI

- testy stałoprądowe, które polegają na wymuszaniu i detekcji stałych napięć lub prądów na określonych wyprowadzeniach;
- testy funkcjonalne, które sprawdzają poprawność logiczną struktur;
- testy dynamiczne, które pozwalają na badanie parametrów dynamicznych, jak czas dostępu, czas cyklu itp.

Metody testowania funkcjonalnego bazują na funkcjonalnym modelu systemu i są połączeniem techniki systematycznego wyznaczania testów, przeniesionej na poziom funkcji realizowanych przez układ, z techniką testowania gruntownego.

Celem testowania funkcjonalnego jest ocena poprawności realizacji przez układ operacji, pod kątem zgodności z jego specyfikacjami funkcjonalnymi.

Istotą testowania funkcjonalnego jest założenie ograniczonego zbioru niezdatności systemu, które wyrażane są na poziomie jego funkcji.

Podstawę testowania gruntownego stanowi założenie o możliwości wystąpienia dowolnego uszkodzenia z przyjętego zbioru niezdatności.

Strategie testowania funkcjonalnego

- **testowania poszczególnych podzespołów układu;**

Zaleta: możliwość przeprowadzenia wyczerpującego testowania poszczególnych podzespołów układu: rejestrów, dekoderów, liczników, multiplekserów.

Wady:

- z faktu poprawnego wykonywania się określonych testów dla danego podzespołu, niekoniecznie wynika poprawność działania pozostałej części struktury logicznej układu;
 - koncentracja uwagi na badaniu poprawności pracy poszczególnych podzespołów, czyli struktury przesyłania danych, powoduje słabe badanie struktury sterującej.
- **testowanie struktury sterującej układu;**

Zaleta: możliwość przeprowadzenia wyczerpującego testowania podzespołów sterujących układu w szczególności wykonywania wszystkich możliwych rozkazów z listy rozkazów.

Wada: koncentracja uwagi na badaniu pracy podzespołów sterujących powoduje słabe badanie struktury przesyłania danych.

- **testowanie mieszane;**

Połączenie obydwu omówionych wcześniej strategii.

- **wykorzystujące pojęcie klasy błędów;**

Zaleta: możliwość przeprowadzenia wyczerpującego testowania pod kątem błędów jakie mogą w układzie wystąpić.

Wada: dokładność określenia klasy błędów powoduje zmianę skali trudności związanych z opracowaniem testów.

Diagnozowanie pamięci półprzewodnikowych

- typu ROM – metoda tworzenia sumy kontrolnej całej zawartości pamięci, która jest zapisywana w komórce pamięci o ostatnim adresie;
- typu RAM – generacja testów dla bloków funkcjonalnych pamięci: *macierzy komórek pamiętających, układów dekodujących oraz układów zapisu/odczytu.*

Modele błędów dla macierzy komórek:

- jedna lub więcej komórek jest stale w stanie 0 lub 1;
- istnieje jedna lub więcej komórek sprzężonych. O dwóch komórkach *i-tej oraz j-tej mówimy, że są sprzężone, jeżeli zmiana stanu jednej z nich pociąga za sobą zmianę stanu drugiej komórki (niekoniecznie identyczną).*

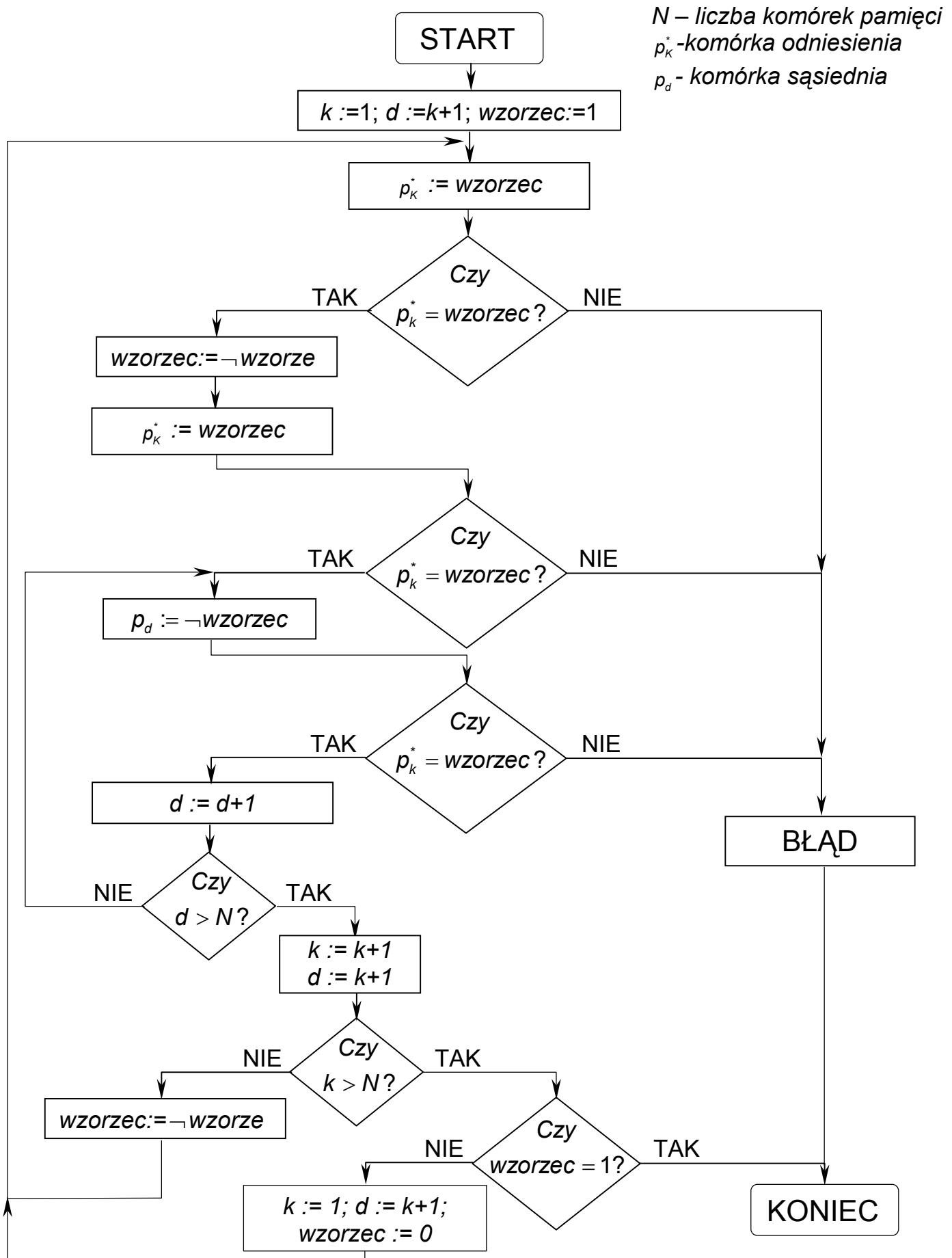
Modele błędów dla układów dekodujących:

- dekoder może wybrać komórkę o adresie innym niż żądana;
- dekoder może wybrać kilka komórek, w tym komórkę żądaną.

Zwarcia wśród linii odczytu/zapisu utożsamiane są ze sprzężeniem komórek pamięci odpowiadającym tym liniom.

Metody testowania pamięci półprzewodnikowych

Metoda ping-pong:



Wykaz metoda testowania pamięci

Nazwa metody	Złożoność	Wykrywane uszkodzenia
Ping-pong	$4N^2$	Pojedynczych bitów, dekoderów adresów, wzorca
Wędrujących jedynek i zer	$2N^2 + 6N$	Pojedynczych bitów, dekoderów adresów, wzorca
Wędrowania	$8N^{3/2} - 6N$	Pojedynczych bitów, dekoderów adresów, wzorca
Przesuwanej przekątnej	$10N^{3/2} - 4N$	Pojedynczych bitów, dekoderów adresów, wzorca
Przesuniętej przekątnej	$4N^{3/2} + 2N$	Pojedynczych bitów, dekoderów adresów
Układu warstwowego w kolumnach i wierszach	$8N$	Pojedynczych bitów, dekoderów adresów
Zakłóceń od najbliższego sąsiada	$N(8b + 6) - 8b - 4$	Pojedynczych bitów, wzorca
Szachownica	$4N$	Pojedynczych bitów
Metodą tła	$4N$	Błędy statyczne
Uzupełnienia adresów	$3N$	Pojedynczych bitów, dekoderów adresów

Diagnozowanie mikroprocesorów

Mikroprocesor reprezentowany jest przez graf skierowany

$$G = \langle W, U \rangle,$$

gdzie:

W - zbiór węzłów grafu G reprezentujący rejesty mikroprocesora

$W = R$ ($R = \{IN, OUT, r_1, r_2, r_3, \dots\}$) przy czym węzły IN i OUT reprezentują kontakt mikroprocesora z otoczeniem (pamięcią, obszarem I/O, urządzeniami);

U - zbiór łuków reprezentujących instrukcje mikroprocesora. Jeżeli w wyniku wykonania instrukcji I_k następuje przesłanie informacji z rejestrów r_i do rejestrów r_j , to w grafie G występuje łuk od węzła r_i do węzła r_j opisany etykietą I_k .

Wyznaczenie sekwencji detekcyjnej rozpoczyna się od utworzenia modelu uszkodzeń wyrażanego za pomocą błędów obserwowanych na poziomie instrukcji mikroprocesora, co uniezależnia ją od szczegółów implementacji.

Podział funkcji spełnianych przez podzespoły mikroprocesora:

- wybieranie rejestrów;
- dekodowanie instrukcji i sterownie;
- pamiętanie informacji;
- przesyłanie informacji;
- przetwarzanie informacji.

Dla funkcji wybierania rejestrów model uszkodzeń dopuszcza wybranie dowolnego, a w szczególności pustego, podzbioru zbioru rejestrów zawierających rejestr właściwy. Funkcję tę oznaczamy przez:

$$f_D : R \rightarrow R \cup \{\Phi\},$$

gdzie Φ oznacza nieistniejący rejestr.

Dla zdatnego układu wybierania rejestrów zachodzi, że:

$$\forall r_i \in R : f_D(r_i) = r_i.$$

Dla układu niezdatnego można wyróżnić cztery następujące uszkodzenia:

- a) $f_D(r_i) = \Phi;$
- b) $f_D(r_i) = r_i, r_j, r_k, \dots;$
- c) $f_D(r_i) = r_i, \quad f_D(r_j) = r_i;$
- d) $f_D(r_i) = r_j, \quad f_D(r_j) = r_i;$

Dla funkcji dekodowania instrukcji i sterowania model uszkodzeń dopuszcza:

- wykonanie zamiast instrukcji I_j innej instrukcji I_k . Błąd ten oznaczamy $f(I_j/I_k);$
- wykonanie instrukcji I_j oraz instrukcji I_k . Błąd ten oznaczamy $f(I_j/I_j + I_k);$
- nie wykonanie żadnej instrukcji. Błąd ten oznaczamy $f(I_j/\Phi).$

Dla uproszczenia przyjmuje się, że:

- jeżeli istnieją błędy $f(I_j/I_k)$ lub $f(I_j/I_j + I_k)$, to instrukcja I_k wykona się poprawnie;
- jeżeli istnieją błędy $f(I_j/I_k), f(I_j/I_j + I_k)$ lub $f(I_j/\Phi)$, to błędy $f(I_q/I_j)$

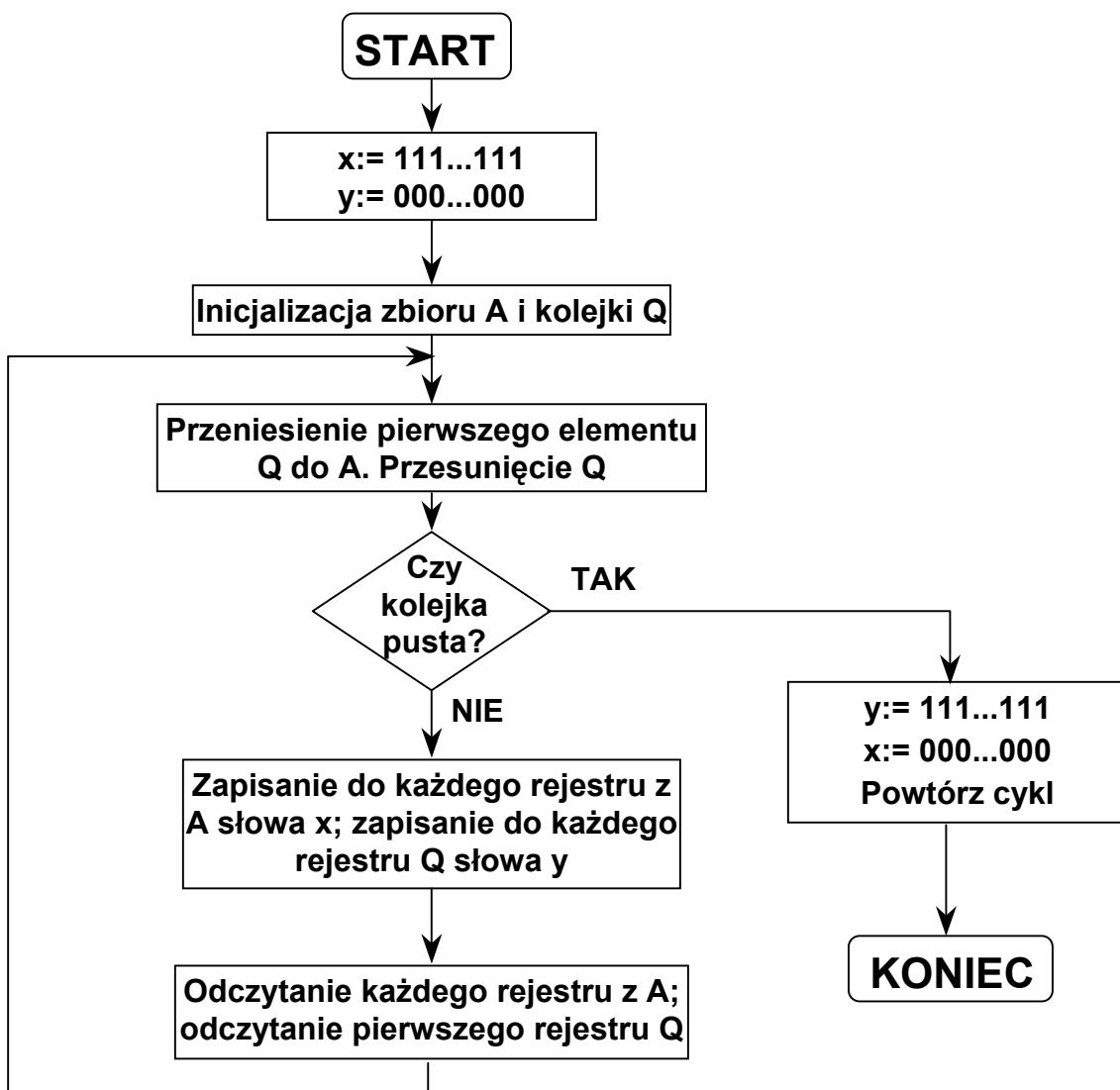
lub $f(I_q/I_q + I_j)$ nie występują.

Dla funkcji pamiętania informacji model uszkodzeń dopuszcza utrzymywanie się stałej wartości 0 lub 1 (czyli s-a-c) na dowolnej liczbie bitów dowolnych rejestrów przeznaczonych do pamiętania informacji.

Dla funkcji przesyłania informacji model uszkodzeń dopuszcza:

- s-a-c każdej linii przesyłowej;
- zwarcie (zmostkowanie) każdej pary linii przesyłowych w przypadku wykonania dowolnej instrukcji mikroprocesora.

Algorytm testowania funkcji wybierania rejestrów



Rozważmy hipotetyczny mikroprocesor.

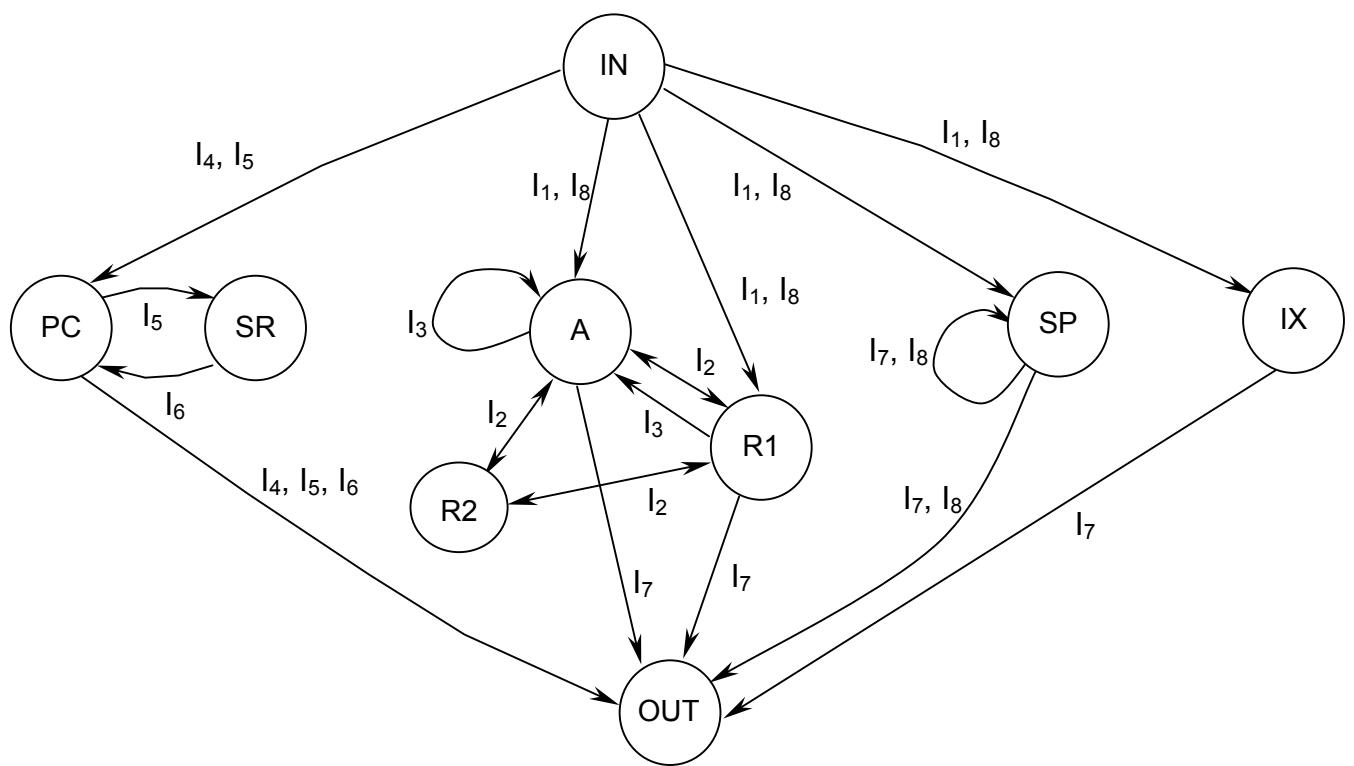
Zbiór rejestrów:

- A – akumulator,
- PC – licznik rozkazów,
- SP – wskaźnik stosu,
- R1 – rejestr ogólnego przeznaczenia,
- R2 – rejestr pomocniczy,
- SR – rejestr procedury (śladu powrotu),
- IX – rejestr indeksowy.

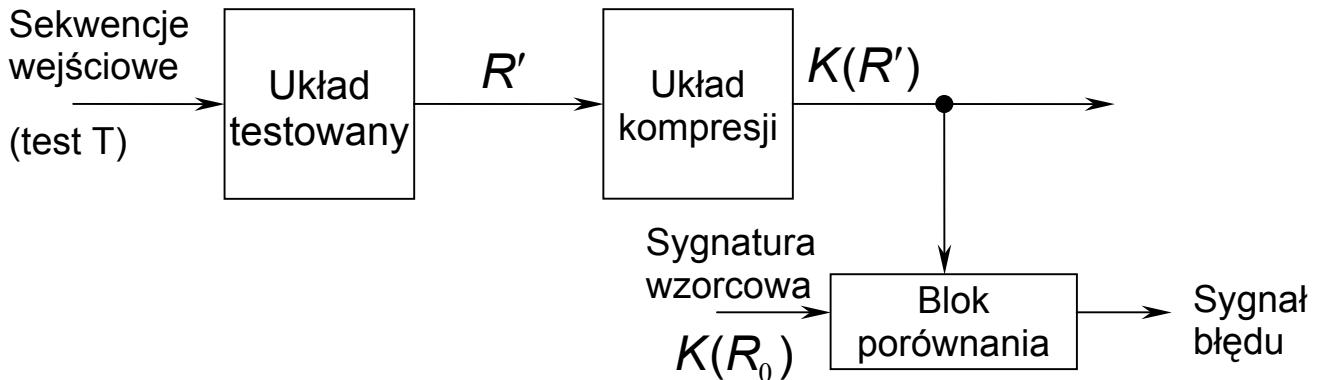
Zbiór instrukcji hipotetycznego mikroprocesora:

Instrukcja	Opis	Operacja	Łuki w grafie
I ₁	MVI R, a $R \in \{A, R1, SP, IX\}$	$R \leftarrow a$	$IN \rightarrow R$
I ₂	MOV R_a, R_b $R_a, R_b \in \{A, R1, R2\}$	$R_a \leftarrow R_b$	$R_a \rightarrow R_b$
I ₃	ADD A, R1	$A \leftarrow A + R1$	$A \rightarrow A$ $R1 \rightarrow A$
I ₄	JMP a	$PC \leftarrow a$	$IN \rightarrow PC$ $PC \rightarrow OUT$
I ₅	CALL a	$SR \leftarrow PC$ $PC \leftarrow a$	$PC \rightarrow SR$ $IN \rightarrow PC$ $PC \rightarrow OUT$
I ₆	RET	$PC \leftarrow SR$	$SR \rightarrow PC$ $PC \rightarrow OUT$
I ₇	PUSH R $R \in \{A, R1\}$	$SP \leftarrow R$ $SP \leftarrow SP + 1$	$SP \rightarrow OUT$ $R \rightarrow OUT$ $SP \rightarrow SP$
I ₈	POP R	$SP \leftarrow SP - 1$ $R \leftarrow (SP)$	$SP \rightarrow SP$ $SP \rightarrow OUT$ $IN \rightarrow R$

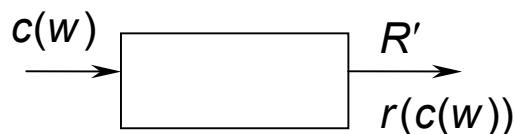
Graf opisujący hipotetyczny mikroprocesor



Testowanie z zastosowaniem techniki kompresji odpowiedzi



Definicja



Kompresją K wyników testowania nazywamy jednoznaczne przekształcenie zbioru $R(c(w))$ możliwych ciągów reakcji na ciąg wymuszeń elementarnych $c(w)$, w zbiór $S^K(c(w)) \rightarrow$ elementów alfabetu abstrakcyjnego

$$K : R(c(w)) \rightarrow S^K(c(w))$$

$$2 \leq |S^K(c(w))| < \text{Card } R(c(w))$$

Metody kompresji

- zliczanie wartości (zero lub jeden),
- zliczanie przejść z $1 \rightarrow 0$ lub $0 \rightarrow 1$,
- kontrola parzystości,
- syndrom,
- analiza sygnatur.

Zliczanie jedynek

Dla układu C z jednym wejściem, odpowiedź $R = r_1, r_2, \dots, r_m$

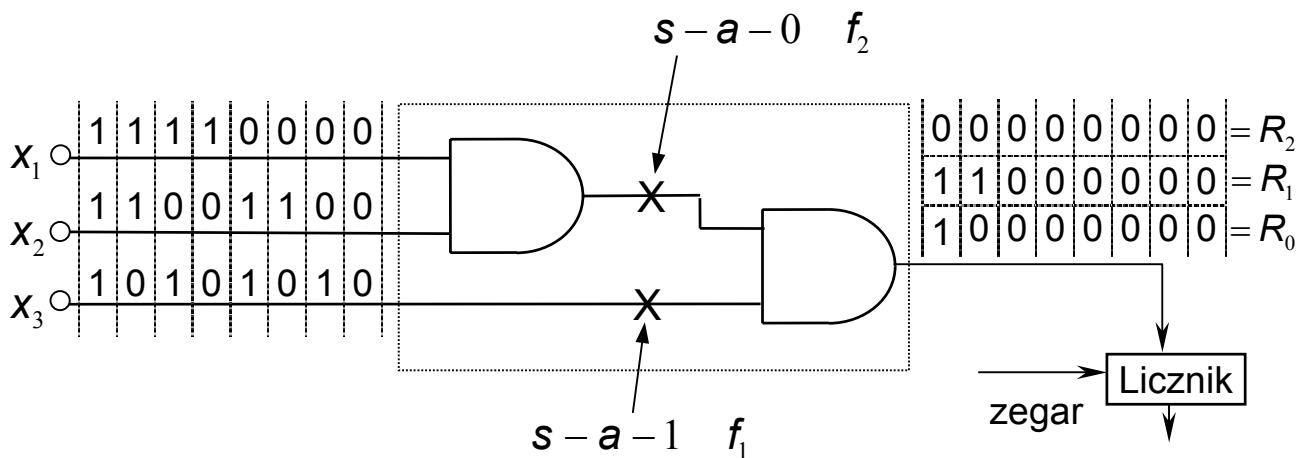
$$K_i^1(R) = \sum_i r_i$$

$$0 \leq K^1(R) \leq m$$

Układ kompresji: licznik

Stopień kompresji: $\lceil \log_2(m+1) \rceil$

Przykład



gdzie:

R_0 - wartość wyjściowa dla układu zdatnego;

R_1 - wartość wyjściowa dla układu z uszkodzeniem s-a-1;

R_2 - wartość wyjściowa dla układu z uszkodzeniem s-a-0;

Rozważmy układ testowany za pomocą m -losowych wektorów.

Niech $K^1(R_0) = r$, $0 \leq r \leq m$:

$$P_{K^1}(M | m, r) = \frac{\binom{m}{r} - 1}{2^m - 1}$$

Jeżeli $2^m - 1$ błędnych sekwencji są jednakowo prawdopodobne to $P(M | m, r)$ - jest prawdopodobieństwem maskowania.

Charakterystyki metody zliczania wartości:

1. prawdopodobieństwo maskowania jest najniższe gdy wartość sygnatury leży na krańcach przedziału i wzrasta do wartości maksymalnej dla $K(R) = \left\lfloor \frac{m}{2} \right\rfloor$;
2. dla $K_1(R_0) = 0$ lub m , nie występuje maskowanie;
3. uszkodzenie generujące nieparzystą liczbę błędów w sekwencji odpowiedzi jest zawsze wykrywane, jeżeli liczba błędów jest parzysta, uszkodzenie może być niewykryte.

TWIERDZENIE

Prawdopodobieństwo maskowania błędu dla układu kombinacyjnego przy zastosowaniu metody zliczania jedynek dąży do wartości $\Pi m^{-\frac{1}{2}}$.

$$P(M) = \sum_{r=0}^m P_{K^1}(M | m, n) \cdot P(r) \quad P(r) = \frac{\binom{m}{n}}{2^m}$$

$$P(M) = \frac{\binom{2m}{n} - 2^m}{2^m(2^m - 1)}$$

$$\binom{2m}{m} = \frac{(2m)!}{m!(2m-m)!} = \frac{(2m)!}{2m!} = \frac{(2\Pi 2m)^{\frac{1}{2}} e^{-2m} (2m)^{2m}}{(2\Pi)^{\frac{1}{2}} e^{-m} m^m}$$

$$P(M) = (\Pi m)^{-\frac{1}{2}}$$

Zliczanie przejść

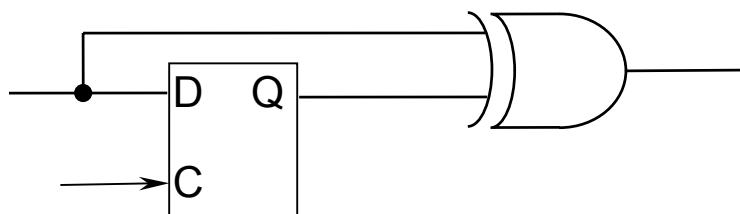
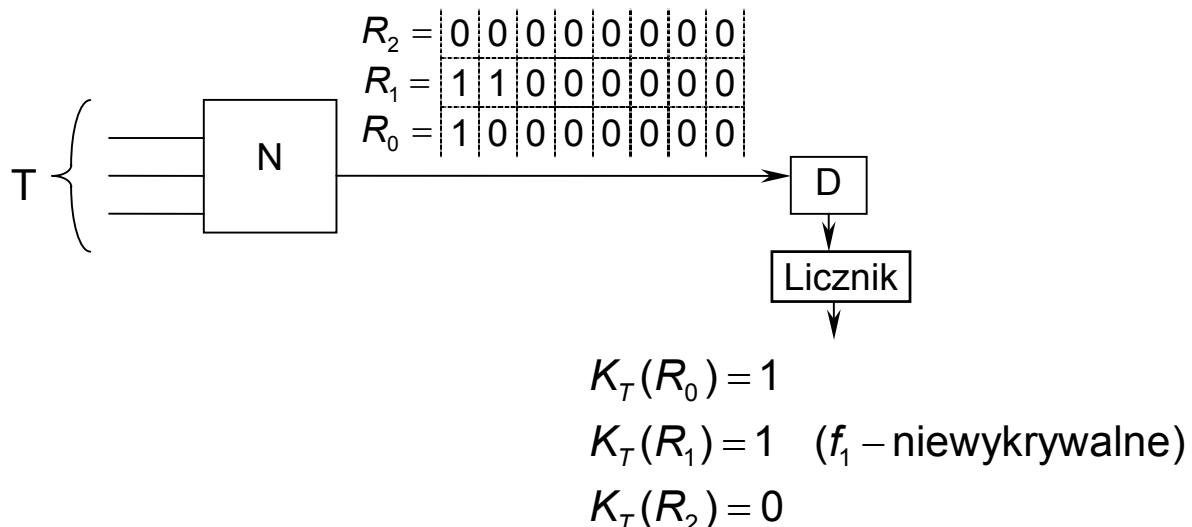
$$K_T(R) = \sum_{i=1}^{m-1} (r_i \oplus r_{i+1})$$

\oplus - oznacza operację dodawania modulo 2

$$0 \leq K_T(R) \leq m - 1$$

Stopień kompresji: $\lceil \log_2 m \rceil$

Przykład



Niech T będzie sekwencją testową o długości m dla układu N i R_0 - odpowiedzią układu zdatnego, gdzie: $K_T = (R_0) = r$.

Niech R' będzie sekwencją o długości m . R' ma $(m-1)$ miejsc (granic) gdzie może nastąpić zamiana ($0 \rightarrow 1$, $1 \rightarrow 0$).

Istnieje $\binom{m-1}{r}$ różnych sposobów zmian takich, że R' będzie miała liczbę przejść r . Zatem istnieje $2 \cdot \binom{m-1}{r}$ różnych ciągów.

Liczba sekwencji błędnych:

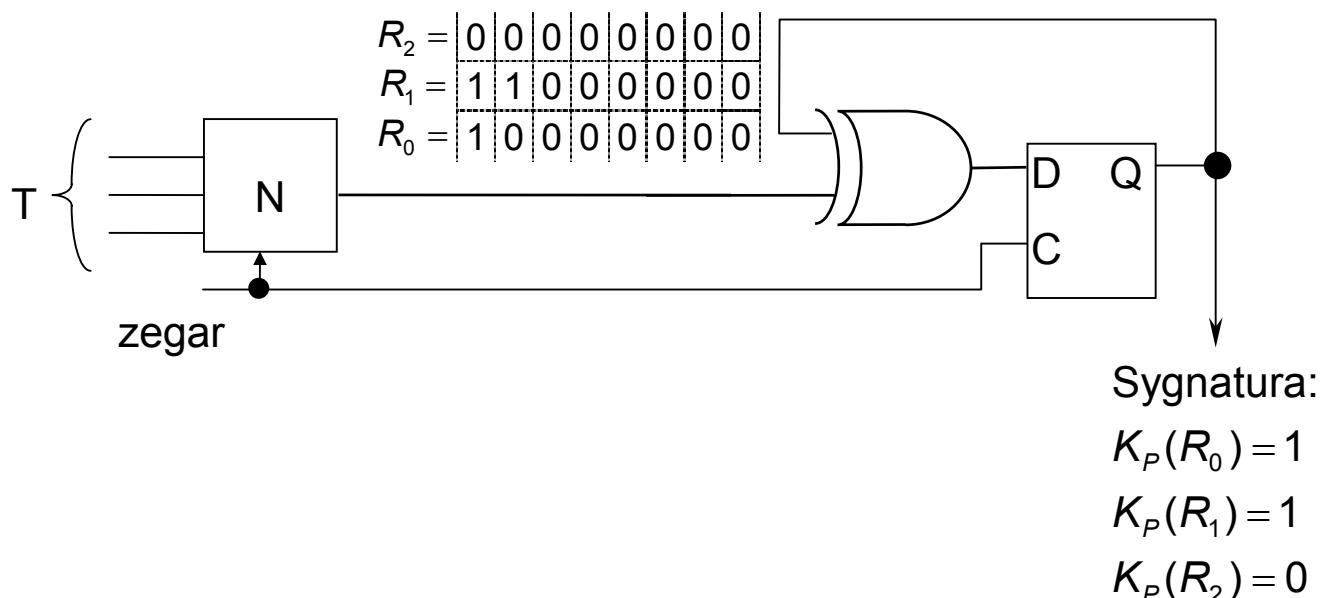
$$2 \cdot \binom{m-1}{r} - 1$$

Prawdopodobieństwo maskowania:

$$P_{K_T}(M | m, r) = \frac{2 \cdot \binom{m-1}{r} - 1}{2^m - 1}$$

Funkcja ta ma podobne właściwości jak w przypadku zliczania jedynek.

Kompresja z kontrolą parzystości



Analiza sygnatur

Ciągi bitów danych pojawiających się w jednym punkcie układu można przedstawić w postaci wielomianu binarnego jednej zmiennej:

$$A(x) = a_n \cdot x_n \oplus \dots \oplus a_1 \cdot x \oplus a_0 = \oplus \sum_{i=0}^n a_i \cdot x_i$$

gdzie:

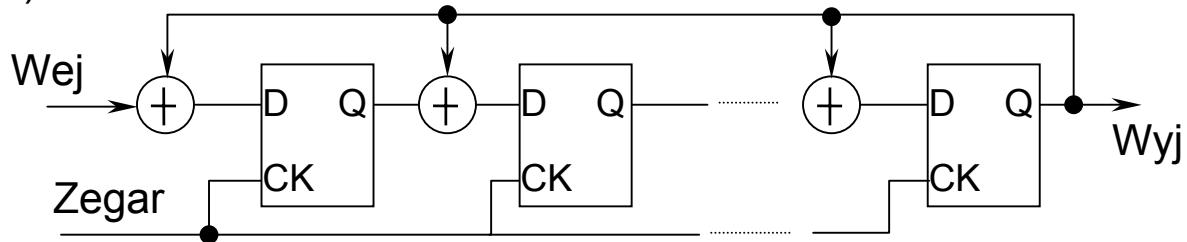
- x - operator przesunięcia o kwant czasu;
- a_i - dana pojawiająca się w chwili i ;
- \bullet - iloczyn logiczny (koniunkcja);
- \oplus - suma modulo 2 (różnica symetryczna);
- $\oplus \sum$ - uogólniona (wieloargumentowa) suma modulo 2.

Cechy charakterystyczne wielomianów binarnych jednej zmiennej:

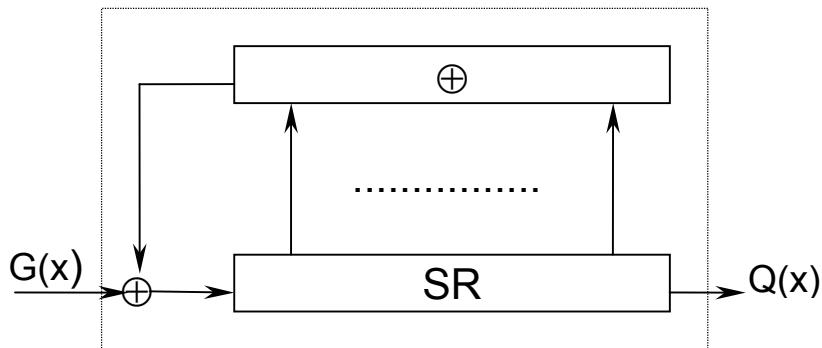
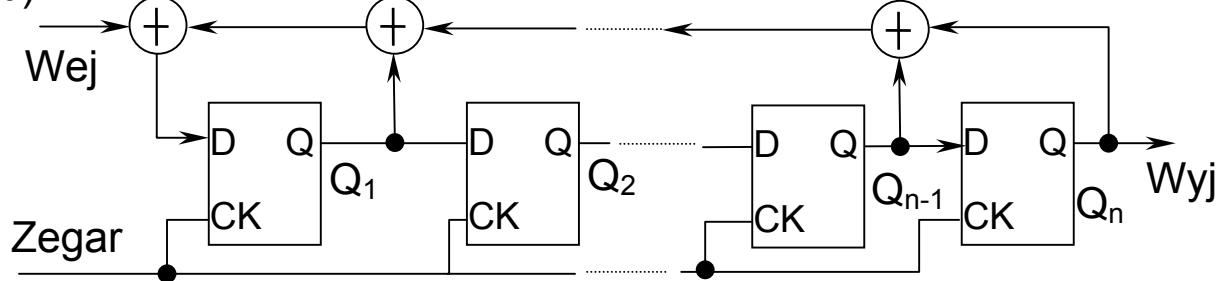
- ◆ współczynniki a_i są zmiennymi binarnymi i mogą przyjmować jedynie wartości 0 lub 1;
- ◆ operacje dodawania i mnożenia są operacjami logicznymi sumy modulo 2 (różnicy symetrycznej) oraz iloczynu logicznego (koniunkcji);
- ◆ operacje dodawania i odejmowania wielomianów binarnych są operacjami równoważnymi ponieważ zachodzą następujące zależności: $w_i = a_i \oplus b_i \Leftrightarrow a_i = w_i \oplus b_i \Leftrightarrow b_i = a_i \oplus w_i$;
- ◆ algorytmy operacji dodawania, odejmowania, dzielenia oraz mnożenia z resztą wielomianów binarnych są analogiczne do znanych algorytmów dla tych operacji na wielomianach, których współczynniki przyjmują wartości ze zbioru liczb rzeczywistych.

Analiza sygnatur wiąże się z wykorzystaniem rejestru przesuwającego z liniowym sprzężeniem zwrotnym LFSR (ang. *Linear Feedback Shift Register*).

a)



b)



Oznaczmy:

$I(x)$ - stan początkowy rejestru SR, który przyjmuje się równy 0;

$P(x)$ - wielomian charakterystyczny rejestru SR;

$R(x)$ - stan końcowy rejestru SR;

$G(x)$ - wielomian danych wejściowych.

Wielomian pojawiający się na wyjściu i -tego przerzutnika można przedstawić w postaci:

$$w_i(x) = x^{-i} \cdot y(x)$$

gdzie:

i – wartość z przedziału 1 do m (m jest liczbą przerzutników w rejestrze);

$w_i(x)$ - wielomian pojawiający się na wyjściu i -tego przerzutnika;

$y(x)$ - wielomian generowany przez układ liniowego sprzężenia zwrotnego.

Stan na wyjściu układu liniowego sprzężenia zwrotnego opisuje wielomian:

$$y(x) = \alpha_0 \cdot G(x) \oplus \sum_{i=1}^m \alpha_i \cdot w_i(x)$$

gdzie:

α_i - tzw. mnożnik binarny, określający stan połączenia wyjścia i -tego przerzutnika rejestru LFSR z funtorem modulo 2, wchodzącym w skład układu liniowego sprzężenia zwrotnego zgodnie z konwencją:

$\alpha_i = 0$ - brak połączenia;

$\alpha_i = 1$ - połączenia istnieje;

Przyjmując, że: $P(x) = \sum_{i=1}^m \alpha_i \cdot x^{m-i}$, otrzymujemy:

$$y(x) = x^m \frac{G(x)}{P(x)} \oplus R(x)$$

gdzie:

$\frac{G(x)}{P(x)}$ - wielomian pojawiający się na wyjściu ostatniego przerzutnika rejestru LFSR;

$R(x)$ - **sygnatura**.

Jeżeli stopień wielomianu $G(x)$ jest równy n , to ma on $n+1$ współczynników przez co $|\{G(x)\}| = 2^{n+1}$.

Natomiast dla LFSR o długości m otrzymujemy, że: $|\{R(x)\}| = 2^m$.

Jeżeli $m < n$, to różne wielomiany $G(x)$ mogą generować taką samą sygnaturę, stąd też odwzorowanie:

$$\text{wielomian } G(x) \rightarrow \text{sygnatura } R(x),$$

$$\text{sygnatura } R(x) \rightarrow \text{wielomian } G(x)$$

nie jest odwzorowaniem wzajemnie jednoznaczny.

Dla wielomianu $G(x)$ stopnia m liczba możliwych wielomianów wejściowych wynosi 2^m , natomiast wielomianu $R(x)$ stopnia n liczba możliwych sygnatur wynosi 2^n . Oznacza to, że sygnaturę poprawną może wygenerować jeden z $N = \frac{2^m}{2^n} = 2^{m-n}$ wielomianów wejściowych. Ponieważ jeden wielomian jest poprawny, stąd też wielomianów z błędami jest $N - 1 = 2^{m-n} - 1$.

Twierdzenie

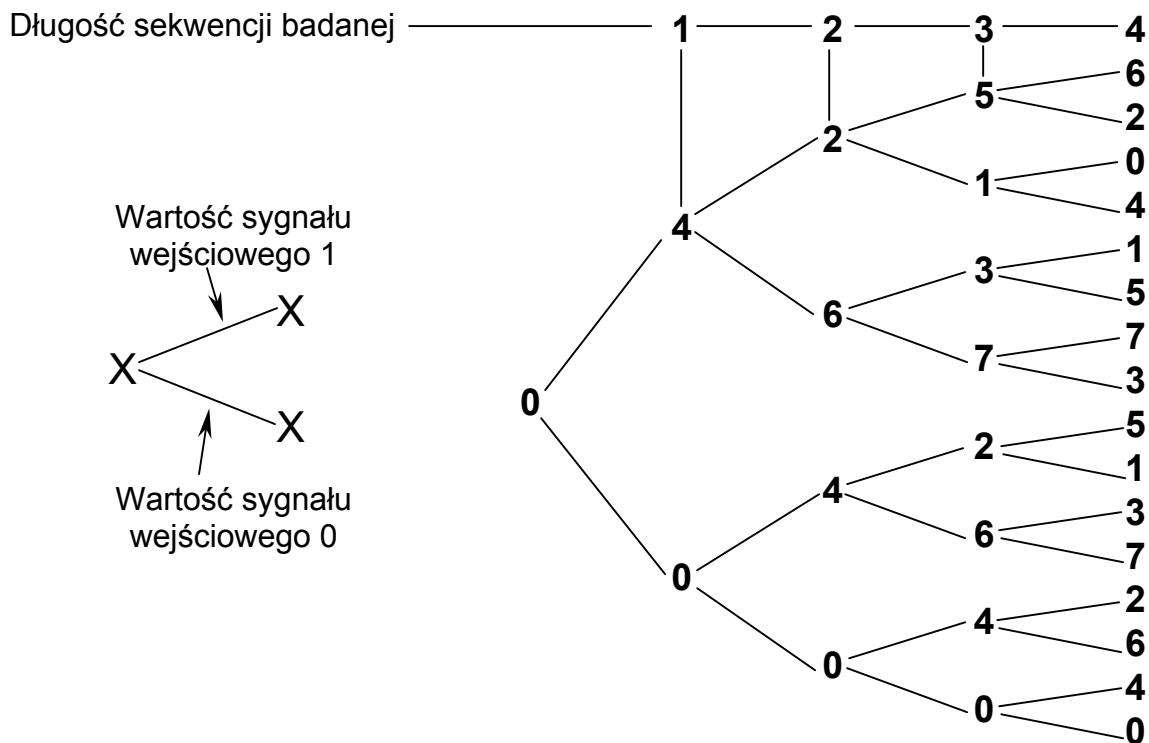
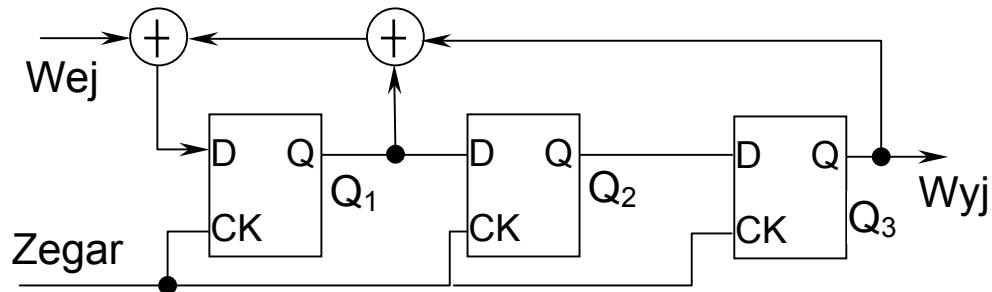
Jeżeli wystąpienie błędnych wielomianów jest jednakowo prawdopodobne, to prawdopodobieństwo tego, że n -bitowy analizator sygnatur nie wykryje błędu wynosi:

$$P(M) = \frac{2^{m-n} - 1}{2^m - 1}.$$

Dla $m \gg n$ $\lim_{m \rightarrow \infty} P(M) = 2^{-n}$.

Długość rejestru LFSR	Prawdopodobieństwo wykrycia błędu
3	87,5%
4	93,75%
8	99,98%
16	99,998%

Przykład 3-bitowego rejestru

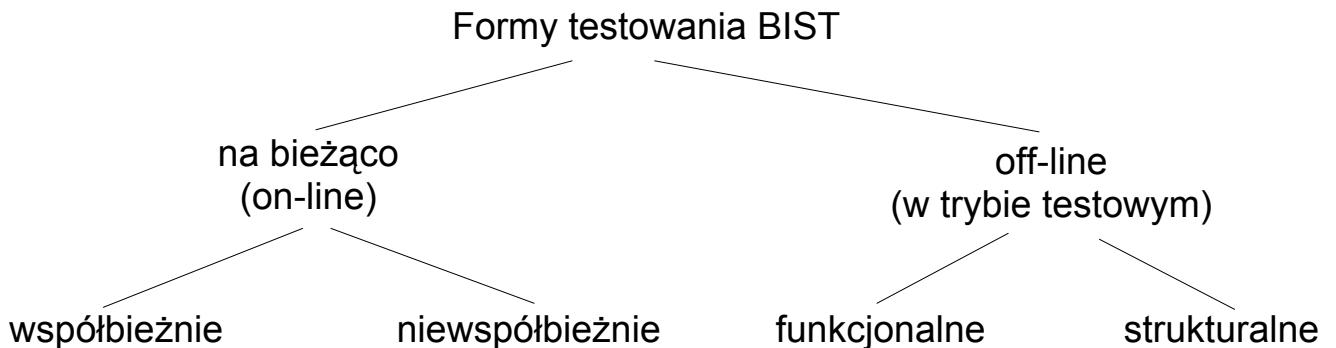


Sygnatura	Stan rejestru przesuwnego	Częstość występowania
0	0 0 0	2
1	0 0 1	2
2	0 1 0	2
3	0 1 1	2
4	1 0 0	2
5	1 0 1	2
6	1 1 0	2
7	1 1 1	2

Techniki BIST

Klasyfikacja

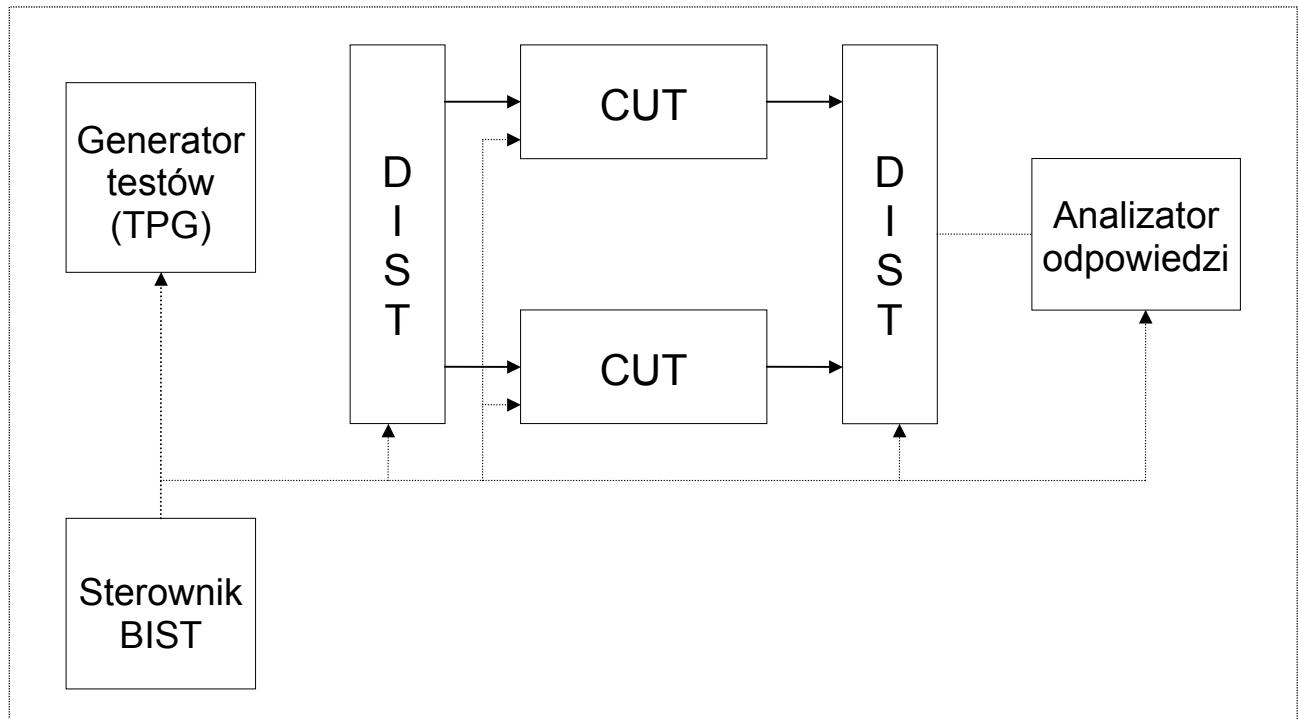
BIST (ang. *Buit-in self-test*) jest to własność układu przejawiająca się zdolnością do autotestowania.



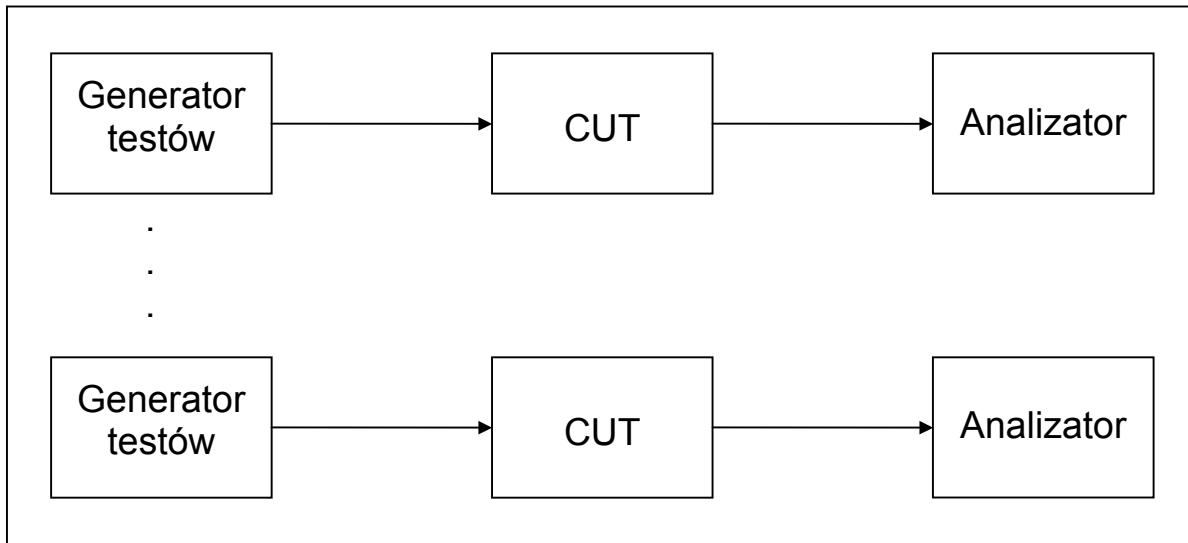
Architektury BIST

- scentralizowane,
- rozproszone.

Architektura scentralizowanego testowania BIST Układ scalony płyta, system



Architektura rozproszona BIST

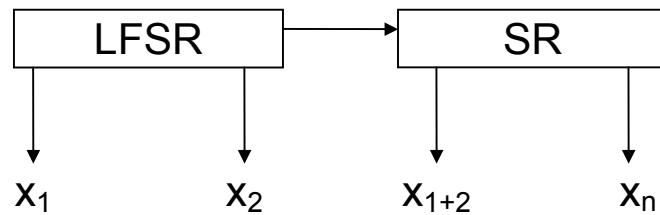


Generacja wzorców testowych dla BIST

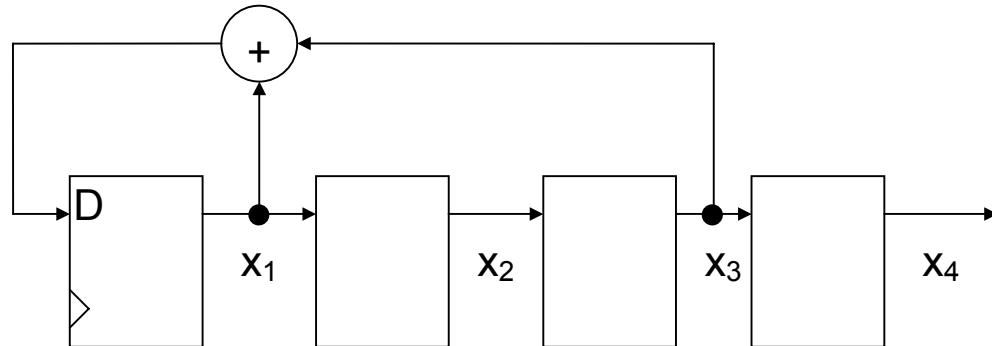
Zakłada się, że układ kombinacyjny ma n -wejścia i m -wyjścia.

- ◆ Testowanie pełne (wyczerpujące) 2^n - testów:
 - pełny zbiór testów (generatory) (implementacja generatora - licznik binarny 2^n zakres stosowalności: $n < 22$)
- ◆ Pseudolosowe testowanie:
 - generatory testów uwzględniające współczynniki zegarowe;
 - adaptacyjne generatory testów.
- ◆ Pseudo-wyczerpujące testowanie:
 - licznik stało-wagowy;
 - licznik syndromu;
 - LFSR i rejestr przesuwny (w kombinacji);
 - LFSR i XOR bramki;
 - cykliczny LFSR.

Połączenie LFSR/SR



np.

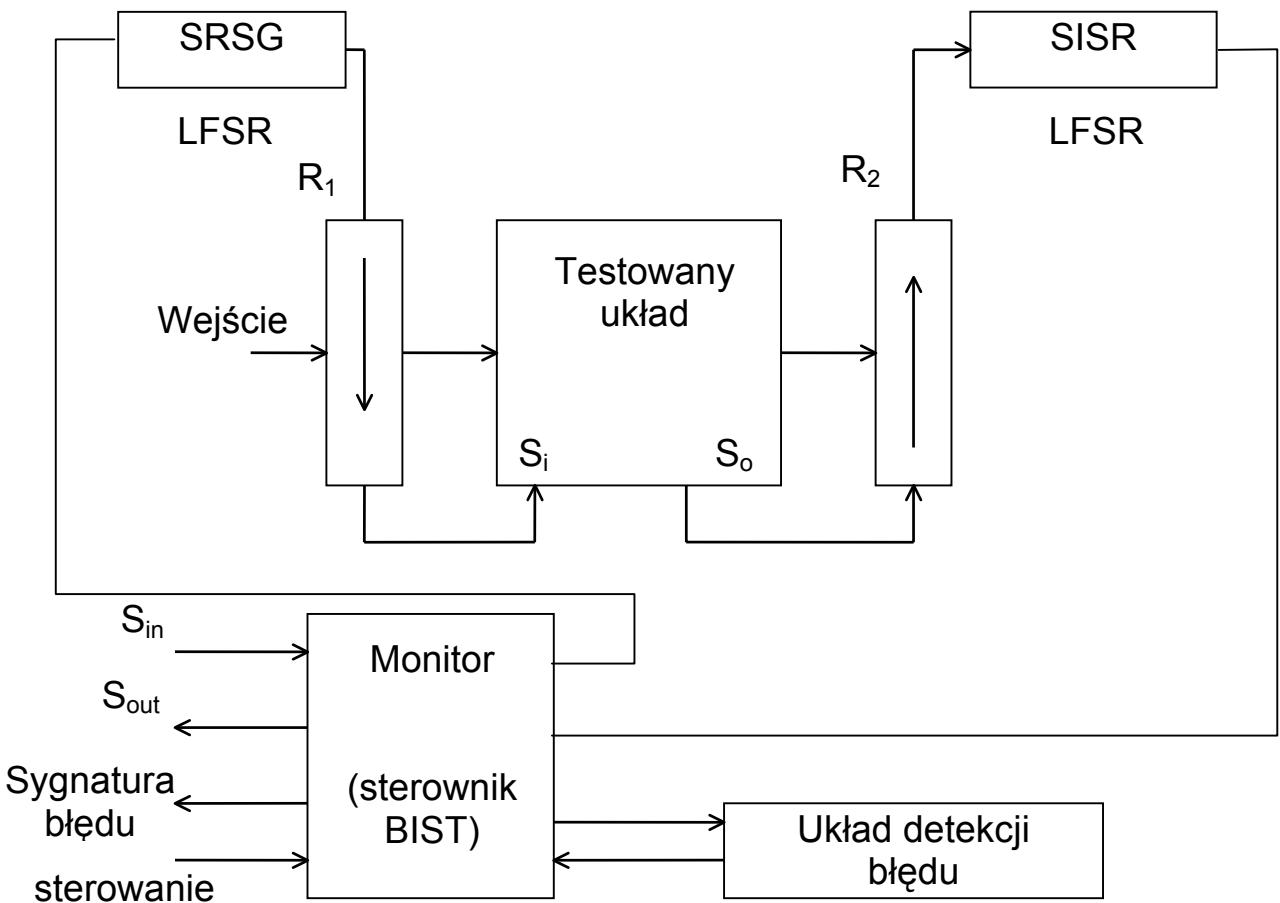


1	1	1	0
0	1	1	1
1	0	1	1
0	1	0	1
0	0	1	0
1	0	0	1
1	1	0	0
<hr/>			
1	1	1	0

Liczba wzorców testowych jest bliska minimalnej, gdy $w \ll n$ np.

$$w < \frac{n}{2}.$$

Specyficzne rozwiązanie BIST



SISR - single input signature register

SRSG - shift-register sequence generator

Przykładowa implementacja:

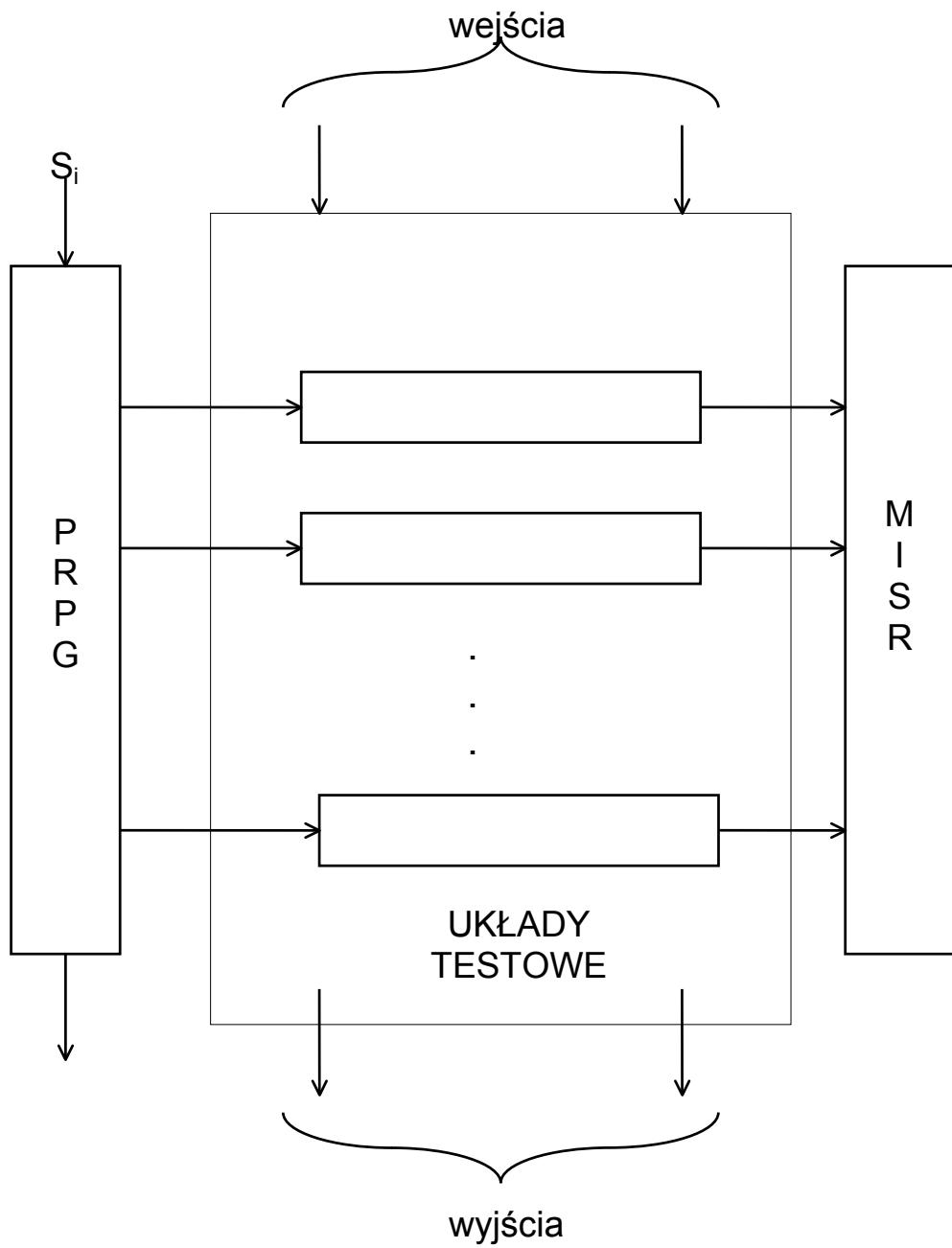
SRSG: wielomian charakterystyczny $x^{20} + x^{17} + 1$

SISR: wielomian charakterystyczny $x^{16} + x^9 + x^7 + x^4 + 1$

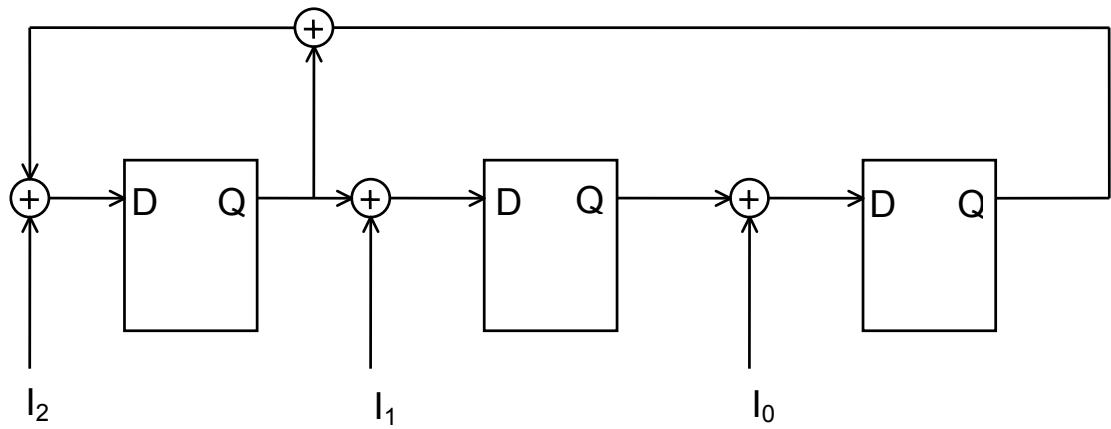
Proces testowania:

1. **Inicjalizacja** : ścieżka testowa jest kodowana danymi początkowymi poprzez linię S_{in} .
2. **Aktywowanie trybu samotestowania** : zakodowane sygnały zegara dla R_1 , R_2 ; włączenie operacji LFSR.
3. **Wykonanie operacji samotestowania** :
 - Załadowanie pseudolosowego wzorca z SRSG. Kompresja danych w SISR (wielokrotne cykle zegara) N -cykli.
 - Aktywacja sygnałów zegarowych systemowych na jeden cykl, załadowanie danych do R_2 i wewnętrznej ścieżki testowej.
 - Powtarzanie a) i b) dopóki odpowiedni poziom pokrycia błędów jest osiągnięty.
4. **Kontrola rezultatów** : sprawdzenie SISR z sygnaturą wzorcową.

Autotestowanie z wykorzystaniem MISR



Multiple input signature register



System tolerujący uszkodzenia to system, który może wykonywać zadane funkcje użytkowe pomimo występujących w nim uszkodzeń (pewnej klasy).

Cechy systemów tolerujących uszkodzenia

- Funkcje tolerowania uszkodzeń są zróżnicowane. Obowiązuje przy tym następująca zasada projektowania: **koszt realizacji mechanizmów zabezpieczeń nie powinien przekraczać kosztów wynikających z usunięcia skutków, jakie spowodowałyby powstałe i niekontrolowane uszkodzenia w systemie.**
- Jednostki systemu (tzn. mikroprocesory, mikrokomputery) posiadają oprócz zadanych możliwości użytkowych także określone zdolności do oceny poprawności wykonania własnych funkcji i/lub funkcji realizowanych przez inne jednostki.
- Systemy takie nie są w pełni bezpieczne, tzn. zawsze można wskazać pewne uszkodzenia, których pojawienie się dezorganizuje pracę systemu. Innymi słowy, są to systemy z niezawodnym jądrem, którego niezawodność powinna znacznie przewyższać niezawodność pozostałych elementów systemu.
- Warunkiem koniecznym tolerowania uszkodzeń jest poprawna ich diagnostyka. Jej jakość ma decydujące znaczenie dla przywrócenia zdatności systemu przez:
 - ◆ wymianę uszkodzonych jednostek,
 - ◆ odłączenie niezdolnych jednostek i rekonfigurację zadań (łagodna degradacja systemu),

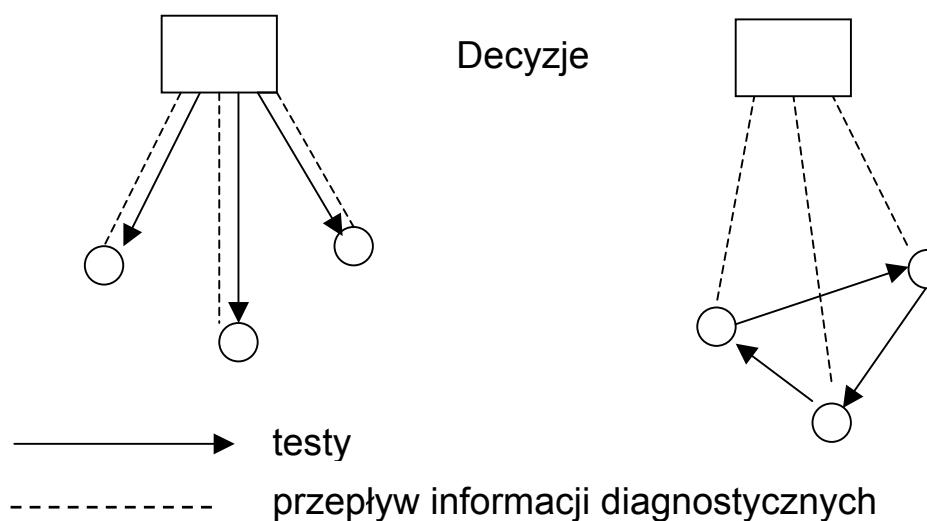
System samodiagnozowy jest to system zdolny do diagnozy własnych uszkodzeń.

Typy systemów samodiagnozowych

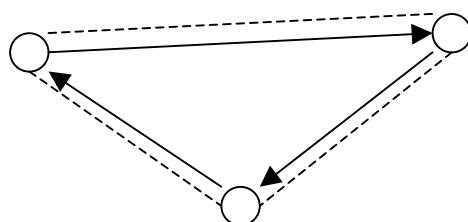
- ✓ Systemy jednoznacznie diagnozowalne;
- ✓ Systemy częściowo diagnozowalne;
- ✓ Systemy nadmiarowo diagnozowalne;
- ✓ Systemy sekwencyjnie diagnozowalne;
- ✓ Systemy przyrostowo diagnozowalne;
- ✓ Systemy diagnozowalne adaptacyjnie.

Strategie diagnostyczne

- strategie skoncentrowane



- strategia rozproszona



- **strategia off-line** jest strategią, w której jednostki biorące udział w diagnozowaniu nie uczestniczą w realizacji zadań użytkowych.
- **strategia on-line** jest strategią, w której stan systemu jest wyznaczany na bieżąco bez zawieszania zadań użytkowych.
- **Strategia jednokrokowa** polega na wykonaniu wszystkich dopuszczalnych testów w systemie i wyznaczeniu wszystkich uszkodzonych jednostek na podstawie otrzymanego syndromu.
- W przypadku **strategii wielokrokowej** proces diagnozy i naprawy przeplatają się nawzajem. Przyjmuje się, że na podstawie syndromu jesteśmy w stanie określić tylko pewien niepusty podzbiór uszkodzonych jednostek. Następnie wymienia się je na zdatne i ponawia się testowanie. Proces powtarzany jest dopóty, dopóki nie stwierdzi się poprawnego funkcjonowania wszystkich jednostek systemu (wymagana liczba iteracji $\leq m$, gdzie m jest liczbą niezdatności).

Miary diagnozowalności

a) dla strategii scentralizowanej

System jest **jednokroko m -diagnozowalny**, jeżeli wszystkie uszkodzone jednostki mogą być zlokalizowane na podstawie jednego syndromu wyników testowania, o ile liczba aktualnie uszkodzonych jednostek nie przekracza m .

System jest **wielokroko*m*-diagnozowa*ny***, jeżeli co najmniej jedna niezdarna jednostka może być zlokalizowana na podstawie jednego syndromu wyników testowania, o ile liczba aktualnie uszkodzonych jednostek nie przekracza *m*.

b) dla strategii rozproszonej

Nie wyróżnia się strategii wielokrokowej, gdyż informacje o wynikach testowania powinny być przekazywane przez jednostki zdąrne.

System z rozproszoną strategią diagnozowania jest *m-diagnozowa*ny*, jeżeli każda zdarna jednostka jest w stanie zlokalizować wszystkie niezdarte jednostki, o ile ich liczba nie przekracza *m*.*

Metody diagnozowania systemów

- opiniowania diagnostycznego:
 - model PMC (*Preparata, Metza, Chien*);
 - model BGM (*Barsi, Grandoni, Maestroni*).
- dialogu diagnostycznego.

Sposób wzajemnego testowania się określonych komputerów sieci komputerowej przedstawiony jest, z reguły, w postaci odpowiedniego grafu nazywanego (w przypadku ogólnym) **grafem diagnostycznym**, który odpowiednio (do metody wzajemnego testowania się komputerów sieci komputerowej) nazywa się **grafem opiniowania diagnostycznego** lub **grafem dialogu diagnostycznego**.

Spójny digraf (unigraf zorientowany) bez pętli G ($G = \langle E, U \rangle$) nazywamy **grafem opiniowania diagnostycznego** (GOD) zbioru E komputerów sieci komputerowej, jeżeli $\langle e', e'' \rangle \in U$ ($e', e'' \in E$, $e' \neq e''$), wtedy i tylko wtedy, gdy komputer e' opiniuje stan niezawodnościowy komputera e'' .

GOD, który zapewnia zlokalizowanie m niezdatnych elementów systemu nazywa się grafem **m -diagnozowalnym**, a graf **m -diagnozowalny** o minimalnej liczbie łuków – grafem **m -optymalnym**.

Stan niezawodnościowy systemu

Niech k -wymiarowy ($k = |E|$) wektor binarny n ($n = (n_1, \dots, n_k)$) oznacza taki stan niezawodnościowy zbioru E komputerów sieci komputerowej, że jeżeli $n_s = 0$ ($1 \leq s \leq k$), to komputer e_s jest zdany oraz jeżeli $n_s = 1$, to komputer e_s jest niezdany, a N - zbiór wszystkich możliwych takich stanów niezawodnościowych.

Niech $d_{st} = 0$ oraz $d_{st} = 1$ oznacza, że komputer e_s , w wyniku testowania kontrolnego komputera e_t , opiniuje [ocenia] komputer e_t (odpowiednio) jako zdany oraz jako niezdany, a $n(e')$ oraz $n_0(e')$ niech oznacza (odpowiednio) stan niezawodnościowy oraz stan zdatności komputera e' .

Dla ustalonego G ($G = \langle E, U \rangle$) oraz określonego n ($n \in N$), po ustalonym uporządkowaniu zmiennych d_{st} , otrzymamy określony podsześciian $d(n) | U |$ - wymiarowego hipersześcianu binarnego, natomiast po identycznym uporządkowaniu opinii (wydanych przez wszystkie komputery, które testują inne komputery) otrzymamy $| U |$ - wymiarowy wektor binarny d nazywany **opinią globalną (syndromem)** komputerów sieci komputerowej.

Gdy jednostka testująca jest uszkodzona, wyróżnić możemy dwa typy unieważniania się testów:

- a) symetryczny (model PMC),
- b) asymetryczny (model BGM).

$$[n(\mathbf{e}_s) = n_0(\mathbf{e}_s)] \rightarrow \left[d_{st} = \begin{cases} 0 & \text{dla } n(\mathbf{e}_t) = n_0(\mathbf{e}_t) \\ 1 & \text{dla } n(\mathbf{e}_t) \neq n_0(\mathbf{e}_t) \end{cases} \right]$$

dla a):

$$[n(\mathbf{e}_s) \neq n_0(\mathbf{e}_s)] \rightarrow [d_{st} = x] \quad (x \in \{0, 1\}),$$

dla b):

$$[n(\mathbf{e}_s) \neq n_0(\mathbf{e}_s)] \rightarrow \left[d_{st} = \begin{cases} x & \text{dla } n(\mathbf{e}_t) = n_0(\mathbf{e}_t) \\ 1 & \text{dla } n(\mathbf{e}_t) \neq n_0(\mathbf{e}_t) \end{cases} \right].$$

Graf opiniowania diagnostycznego G ($G = \langle E, U \rangle$) jest grafem 1-krokowo m -diagnozowalnym, jeżeli (model PMC):

- a) $|E| \geq 2 \cdot m + 1$,
- b) $\mu^-(e) \geq m$, $e \in E$;
- c) $(\forall 0 \leq p \leq m-1 \ \forall E' \subset E : |E'| = |E| - 2 \cdot m + p) : |\Gamma(E')| > p$

lub

$$\forall n', n'' \in N^m \ \exists \langle e_s, e_t \rangle \in U : (d_{st}(n') \neq x) \wedge \\ \wedge (d_{st}(n'') \neq x) \wedge (d_{st}(n') \neq d_{st}(n'')).$$

Graf opiniowania diagnostycznego G ($G = \langle E, U \rangle$) jest grafem 1-krokowo m -diagnozowalnym, jeżeli (model BGM):

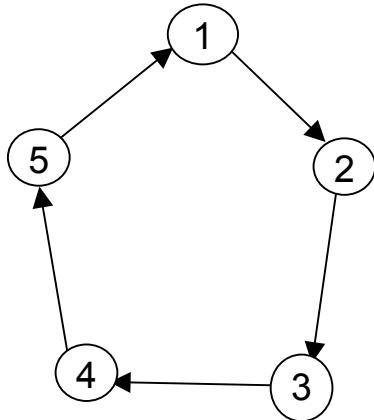
a) $|E| \geq m + 2$,

b) $\mu^-(e) \geq m$,

c)

$$\forall e', e'' \in E : \mu^-(e') = \mu^-(e'') = m \cdot [(\exists e^* \in \Gamma^{-1}(e') \setminus \Gamma^{-1}(e'') \cap \Gamma^{-1}(e'') : \\ : \Gamma^{-1}(e^*) \neq \Gamma^{-1}(e'')) \vee (\exists e^{**} \in \Gamma^{-1}(e'') \setminus \Gamma^{-1}(e') \cap \Gamma^{-1}(e'') : \Gamma^{-1}(e^{**}) \neq \Gamma^{-1}(e'))].$$

Przykład .



	d_{12}	d_{23}	d_{34}	d_{45}	d_{51}	jednostki niezdane
a	x	0	0	0	1	1
b	1	x	0	0	0	2
c	0	1	x	0	0	3
d	0	0	1	x	0	4
e	0	0	0	1	x	5
f	x	x	0	0	1	{1,2}

Niezdatności w węzłach 1 i 2 są rozróżnialne (różne syndromy).

Niezdatności w węzłach 1 oraz {1,2} są nierozróżnialne.

Stan niezawodnościowy systemu n jest jednoznacznie diagnozowalny, jeżeli jego syndrom $d(n)$ jest unikalny dla wszystkich możliwych (prawdopodobnych) stanów niezawodnościowych.

Diagnoza jest kompletna, jeżeli wszystkie niezdane jednostki systemu mogą być zidentyfikowane na podstawie syndromu dla danego stanu niezawodnościowego, w przeciwnym przypadku diagnoza jest niekompletna.

Diagnoza jest poprawna, jeżeli na bazie syndromu systemu jednostki zdane nie są identyfikowane jako niezdane.

Algorytmy diagnozowania sieci wg metody PMC

1. Algorytm NEW_SELF.
2. Algorytm EVENT_SELF.
3. Algorytmy adaptacyjne: ADSD, ADAPT2.

Algorytm NEW_SELF

Założenia:

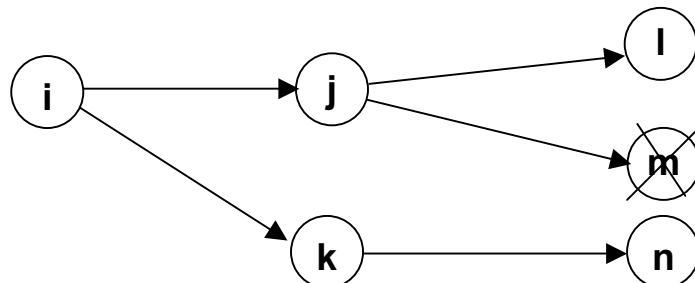
1. Maksymalna liczba niezdanych węzłów jest ograniczona $\leq m$;
2. Dla sieci określony jest stały (niezmienny) przydział testów;
3. Każdy węzeł jest testowany przez co najmniej $m + 1$ innych węzłów;
4. Węzły zdatne przekazują raport z testowania do węzłów sąsiednich, raporty docierają do wszystkich węzłów poprzez węzły pośrednie;
5. Nie przyjmuje się założeń, dotyczących zachowania węzłów niezdanych;
6. Każdy węzeł niezależnie od innych określa diagnozę stanu niezawodnościowego sieci, wykorzystując wyniki testów własnych oraz otrzymane raporty z węzłów sąsiednich.

Sposób działania

Każdy węzeł testuje swoich sąsiadów i generuje raport dla każdego otrzymanego rezultatu testu. **Raport ten jest przechowywany lokalnie i jest stopniowo przesyłany do wszystkich węzłów testujących dany węzeł.**

Algorytm zapewnia poprawność przesyłanych raportów poprzez ograniczenie przesyłania raportów pomiędzy węzłami zdarnymi.

Węzeł akceptuje jedynie informacje z innych węzłów, które są przez niego testowane i ich stan został określony jako zdatny. Potwierdzony raport wyniku jest przesyłany pomiędzy węzłami zdatnymi w odwrotnym kierunku niż testy.



Schemat testowania i walidacji:

- 1) węzeł i testuje węzeł j , wynik testu – *zdatny*;
- 2) węzeł i otrzymuje raport od węzła j ;
- 3) węzeł i testuje węzeł j – wynik – *zdatny*;
- 4) węzeł i zakłada, że informacje diagnostyczne otrzymane w kroku 2 są potwierdzone (wiarygodne).

W celu zapewnienia poprawnej diagnozy algorytm NEW_SELF zakłada, że każdy zdatny węzeł otrzymuje raporty generowane przez inne zdatne węzły. Warunek ten może być spełniony jeżeli każdy węzeł jest testowany przez $m + 1$ węzłów.

Ocena efektywności algorytmu NEW_SELF

Rozważmy sieć składającą się z N węzłów, która ma być m -diagnozowalna.

Liczba testów:

$$\geq N \cdot (m + 1)$$

Liczba komunikatów, potrzebna do przesłania wyników testów:

$$N^2 \cdot (m + 1)^2$$

Dla 2-diagnozowej sieci o $N=8$ węzłach liczba przesyłanych komunikatów: 576.

Algorytm EVENT_SELF

Jest to modyfikacja algorytmu NEW_SELF zmniejszająca wykorzystanie zasobów sieci. Algorytm ten jest „sterowany zdarzeniami”. Mechanizm ten został wprowadzony, aby zmniejszyć liczbę przesyłanych komunikatów.

Raport o wynikach testowania jest przekazywany dalej (do innych węzłów) przez dany węzeł, jeżeli jego dane różnią się od dotychczasowych wyników, przechowywanych w węźle.

Istnieją jedynie 2 sytuacje, kiedy węzeł musi przekazać wynik testu do swoich testerów:

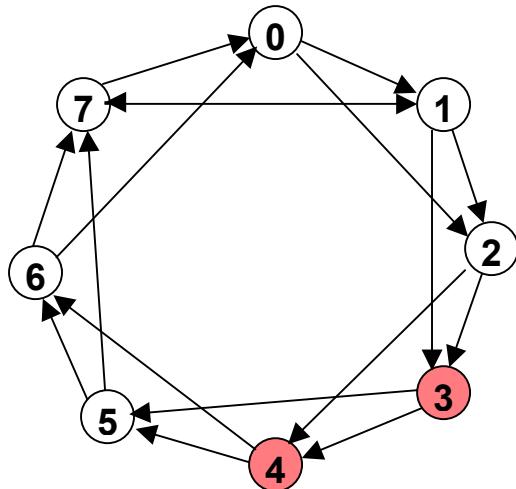
- W dwóch kolejnych testach **wyniki testu są różne** (raport w tym przypadku jest przekazywany do wszystkich testerów danego węzła).
- W przypadku, gdy węzeł diagnozuje jednego z testowanych węzłów jako **niezdarny**, a następnie otrzymuje meldunek, że tester ten jest w stanie zdolności.

W algorytmie EVENT_SELF liczba przesyłanych komunikatów jest znacznie zredukowana.

Istotnym parametrem jest **opóźnienie diagnozy** – jest to czas jaki upływa od momentu wykrycia niezdolności do momentu przekazania tej informacji do wszystkich węzłów.

Podstawowe wady algorytmów: NEW_SELF i EVENT_SELF

1. Ograniczona diagnozowalność.



W przypadku niezdarnych węzłów 3 i 4 rezultaty testów nie będą przekazywane do pozostałych zdarnych węzłów. Przykładowo, węzeł nr 2 nie otrzyma informacji o stanie węzła nr 5.

2. Występuje znaczna redundancja w testowaniu między poszczególnymi węzłami, jak i przy przesyłaniu komunikatów.

Testowanie funkcjonalne

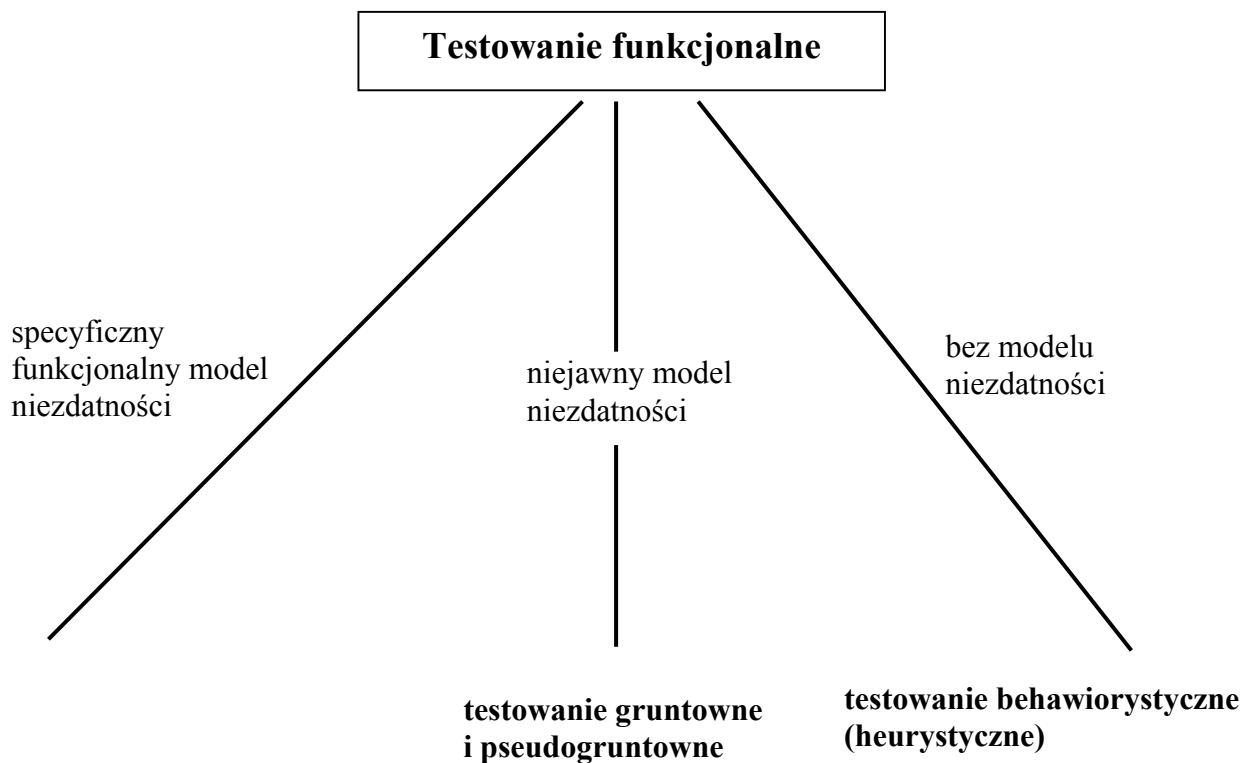
Dotychczas omówione metody generowania testów opierały się na strukturalnym modelu systemu i ich celem było wyznaczenie testów dla wykrywania niezdatności na poziomie struktury systemu, takich jak uszkodzenia stałosygnalowe lub zwarcia zmostkowania.

Wady takiego podejścia:

- trudności w zastosowaniu do współczesnych systemów,
- brak modeli układów VLSI,
- zbyt wielka złożoność współczesnych układów powoduje określone problemy związane ze złożonością obliczeniową.

Metody testowania funkcjonalnego bazują na funkcjonalnym modelu systemu.

Celem testowania funkcjonalnego jest ocena poprawności operacji systemu ze względu na zgodność z jego specyfikacjami funkcjonalnymi.



Istotą **testowania funkcjonalnego** jest założenie ograniczonego zbioru niezdatności systemu, które wyrażane są na poziomie jego funkcji.

Podstawę testowania gruntownego stanowi założenie o możliwości wystąpienia dowolnego uszkodzenia z przyjętego zbioru niezdatności.

1. Testowanie funkcjonalne bez modelu niezdatności

1.1. Metody heurystyczne

Metody heurystyczne wyrażają próby sprawdzenia funkcji systemu sformułowane *ad hoc*.

Przykład 1.

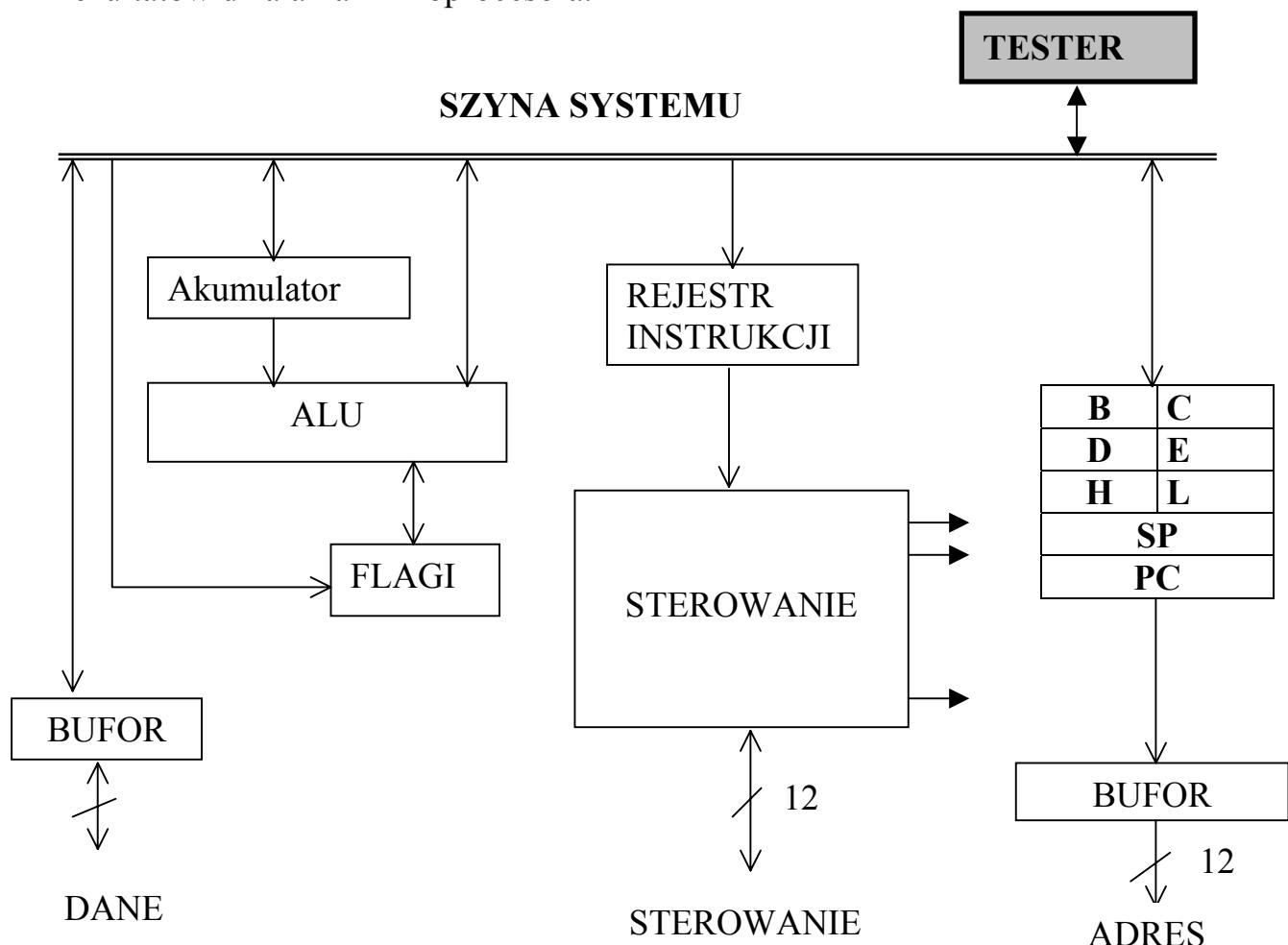
Test funkcjonalny przerzutnika obejmuje:

- test ustawienia wyjścia na wartość „0” i zmiany „ $0 \rightarrow 1$ ”,
- test ustawienia wyjścia na wartość „1” i zmiany „ $1 \rightarrow 0$ ”,
- ocena, czy przerzutnik może utrzymywać ustalony stan.

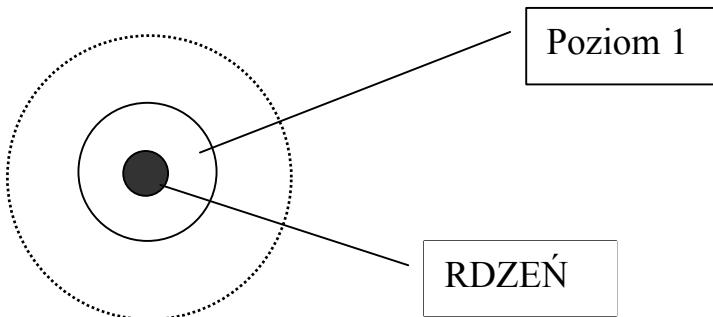
Przykład 2.

Testowanie mikroprocesora INTEL 8080.

Zakłada się, że testowanie realizowane jest przez tester zewnętrzny. Tester zewnętrzny jest podłączony do szyny systemowej (steruje szyną systemową). Tester zawiera program (instrukcje), które są wykonywane dla sprawdzenia rezultatów działania mikroprocesora.



Testy funkcjonalne mikroprocesora:



1. Test licznika rozkazów (PC):
 - a) reset procesora 8080 i ustawienie inicjalne PC,
 - b) umieszczenie operacji NOP na szynie danych i zmuszenie 8080 do cyklicznego wykonywania tej operacji aż PC przejdzie przez 2^{16} stanów (zawartość PC jest dostępna na szynie adresowej).
2. Test rejestrów H i L:
 - a) zapis 8-bitowych wzorców do H i L za pomocą instrukcji MVI (operand bezpośredni),
 - b) przesłanie zawartości H i L do PC (instrukcja PCHL),
 - c) punkty a) i b) powtarzane są dla wszystkich wzorców 8-bitowych.
3. Test rejestrów B, C, D i E:
W podobny sposób tester zapisuje 8-bitowe wzorce danych do rejestru $R \in \{B, C, D, E\}$. Następnie R jest przesyłane do PC poprzez H lub L (R nie może być bezpośrednio przesłane do PC). Wykorzystuje się tu fakt, że PC i H, L zostały przetestowane w pkt. 1 i 2.
4. Test wskaźnika stosu (SP):
SP jest zwiększany i zmniejszany poprzez wszystkie jego stany oraz sprawdzany poprzez PC.
5. Test akumulatora A:
Do A są zapisywane i odczytywane wszystkie możliwe wzorce danych. Może to być wykonane za pośrednictwem uprzednio przetestowanych rejestrów.
6. Test ALU i rejestru FLAG:
Wykonywanie arytmetycznych i logicznych instrukcji. Operandy określane są bezpośrednio lub poprzez już sprawdzone rejesty. FLAGI są sprawdzane poprzez skoki warunkowe, których efekty są obserwowane za pośrednictwem PC.
7. Testowanie pozostałych rozkazów.

Ważnym założeniem w funkcjonalnym testowaniu mikroprocesora jest ustalenie czy **zbiór instrukcji jest ortogonalny**.

Ortogonalność pozwala na użycie każdej operacji w różnych trybach adresowania. Jeżeli zbiór instrukcji nie jest ortogonalny, to każda instrukcja musi być testowana dla jej wszystkich trybów adresacji. Ortogonalność pozwala na zmniejszenie liczby testów.

W funkcjonalnym testowaniu mikroprocesora stosuje się podejście znane pod nazwą *bootstrapingu* - w którym w kolejnych etapach testowania używa się komponentów sprawdzonych w etapach poprzednich.

Główny problem podejścia heurystycznego – to nieokreślona jakość uzyskanych testów funkcjonalnych. Doświadczenia pokazują, że testy opracowane metodami heurystycznymi zapewniają pokrycie 50-70% niezdatności.

Niekiedy stosuje się pewne miary heurystyczne do estymowania „kompletności” testu w odniesieniu do przepływu sterowania systemu. Miary te mogą być oparte na monitorowaniu aktywacji operacji w modelu RTL, np.:

$$\frac{\text{liczba aktywowanych ścieżek } (k)}{\text{liczba możliwych ścieżek } (n)}$$

1.2. Zastosowanie Binarnych Diagramów Decyzyjnych (BDD)

Można wykazać, że zbiór eksperymentów (testów) wyprowadzony za pomocą przebiegu wszystkich ścieżek, odpowiadających funkcjom wyjściowym, tworzą kompletne testy systemu (specyfikacje systemu).

2. Testowanie gruntowne i pseudogruntowne

Dla testowania gruntownego, zakłada się, że testy wykrywają wszystkie niezdatności z założonego, uniwersalnego zbioru niezdatności.

Uniwersalny zbiór niezdatności obejmuje dowolną niezdatność, która nie zmienia liczby stanów systemu cyfrowego (układu). Dla układu kombinacyjnego N realizującego funkcję $Z(X)$, uniwersalny model niezdatności obejmuje dowolną niezdatność f , która przekształca funkcję układu do postaci $Z_f(X)$.

Układy kombinacyjne

Do przetestowania wszystkich niezdatności definiowanych przez uniwersalny model niezdatności w układach kombinacyjnych o n - wejściach niezbędne jest zastosowanie wymuszenia pełnego (2^n wektorów). Ekspotencjany wzrost liczby wektorów ogranicza praktycznie testowanie gruntowe do układów posiadających nie więcej niż 20 linii wejściowych.

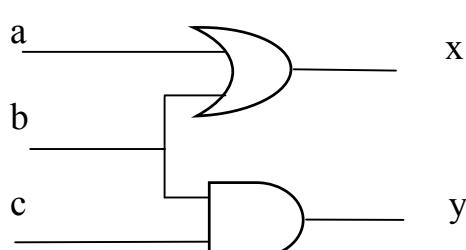
Metody testowania pseudogruntownego pozwalają na testowanie prawie wszystkich niezdatności ze zbioru uniwersalnego za pomocą liczby testów znacznie mniejszej niż 2^n :

- wykorzystanie układów o częściowej zależności,
- techniki segmentacji układów.

2.1. Wykorzystanie układów o częściowej zależności

Niech O_1, O_2, \dots, O_m oznaczają linie wyjściowe układu zawierającego n wejść, a n_i oznacza liczbę wejść wpływających na O_i . Układ, w którym nie występują linie wyjściowe zależne od wszystkich wejść (tj. $n_i < n$ dla wszystkich O_i), jest układem o częściowej zależności. Dla takich układów pseudogruntowne testowanie wymaga 2^{n_i} testów dla każdej linii O_i .

Przykład



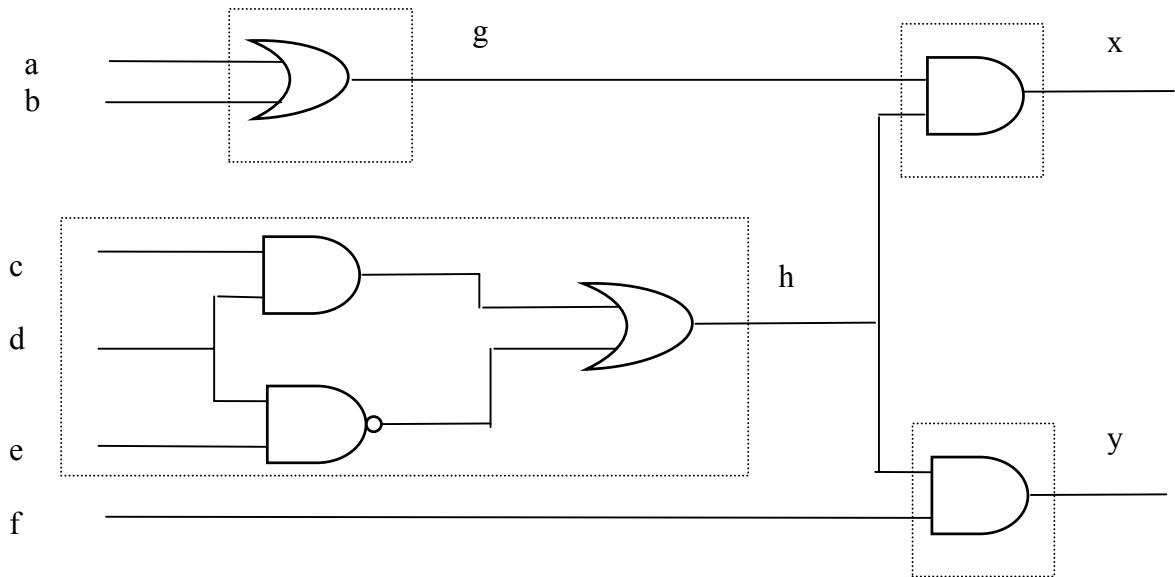
Zbiór testów :

a	b	c
0	0	0
0	1	0
1	0	1
1	1	1

Techniki segmentacji

Zasada podziału układu na segmenty:
 liczba wejść każdego segmentu jest znacznie mniejsza niż liczba linii wejściowych układu.

Przykład.



Lp.	a	b	c	d	e	f	g	h	x	y
1			0	0	0	1		1		
2			0	0	1	1		1		
3			0	1	0	1		1		
4			0	1	1	1	0	0		
5	0	0	1	0	0	1	0	1		
6	0	1	1	0	1	1	1	1		
7	1	0	1	1	0	1	1	1		
8	1	1	1	1	1	1	1	1		
9	0	1	0	1	1	0	1	0		
10		0	0	0	0	0		1		

Układy sekwencyjne

Dla układów sekwencyjnych, uniwersalny zbiór niezdatności obejmuje każdą niezdatność, która zniekształca tablicę stanów układu bez zmiany liczby stanów.

Wejściowa sekwencja wykrywająca każdą niezdatność należącą do tego modelu musi rozróżnić dany n - stanowy automat spośród wszystkich innych automatów skończonych o takiej samej liczbie wejść i wyjść oraz posiadający co najwyżej n – stanów.

Jakiego rzędu jest to problem ?

Twierdzenie [Moore 1956]

Dla każdego zredukowanego, spójnego automatu skońzonego M istnieje para sekwencji wejściowej i wyjściowej, generowanej przez M , która nie może być generowana przez żaden z automatów M' , posiadający co najwyżej n – stanów.

Metoda generowania testów dla mikroprocesorów na podstawie jego modelu

Wady nieformalnego podejścia:

- trudności w przenoszeniu na inne mikroprocesory,
- brak oceny stopnia wykrywalności niezdatności.

Zalożenia:

1. Mikroprocesor reprezentuje się grafem skierowanym

$$G = \langle W, U \rangle$$

gdzie **W** - jest zbiorem węzłów **W**, $W = R \cup \{IN\} \cup \{OUT\}$,

$R = \{R_1, R_2, R_3, \dots\}$ - jest zbiorem rejestrów mikroprocesora,

węzły **IN** i **OUT** reprezentują kontakt mikroprocesora z otoczeniem
(pamięcią, obszarem I/O, urządzeniami);

U - oznacza zbiór łuków.

Jeżeli w wyniku wykonania instrukcji I_k następuje przesłanie informacji z rejestrów R_i do rejestrów R_j , to w grafie **G** występuje łuk od węzła R_i do węzła R_j opisany etykietą I_k .

Rozważmy hipotetyczny mikroprocesor

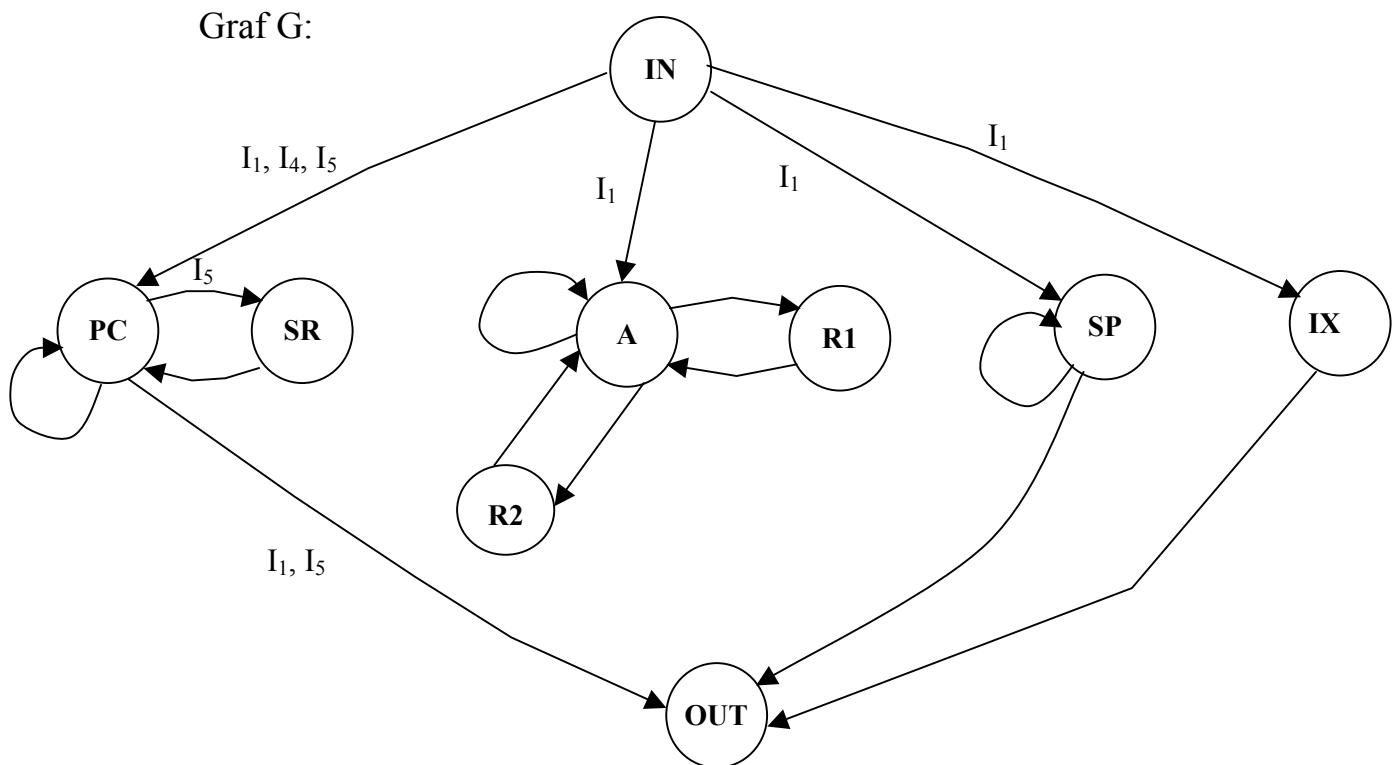
Zbiór rejestrów:

A – akumulator,
PC – licznik rozkazów,
SP – wskaźnik stosu,
R1 – rejestr ogólnego przeznaczenia,
R2 – rejestr pomocniczy,
SR – rejestr procedury (śladu powrotu),
IX – rejestr indeksowy.

Zbiór instrukcji hipotetycznego mikroprocesora:

Instrukcja	Opis	Operacja	Łuki w grafie
I ₁	MVI R, a $R \in \{A, R1, SP, IX\}$	$R \leftarrow a$	$IN \rightarrow R$
I ₂	MOV R_a, R_b $R_a, R_b \in \{A, R1, R2\}$	$R_a \leftarrow R_b$	$R_a \rightarrow R_b$
I ₃	ADD A, R1	$A \leftarrow A + R1$	$A \rightarrow A$ $R1 \rightarrow A$
I ₄	JMP a	$PC \leftarrow a$	$IN \rightarrow PC$ $PC \rightarrow OUT$
I ₅	CALL a	$SR \leftarrow PC$ $PC \leftarrow a$	$PC \rightarrow SR$ $IN \rightarrow PC$ $PC \rightarrow OUT$
I ₆	RET	$PC \leftarrow SR$	$SR \rightarrow PC$ $PC \rightarrow OUT$
I ₇	PUSH R $R \in \{A, R1\}$	$SP \leftarrow R$ $SP \leftarrow SP + 1$	$SP \rightarrow OUT$ $R \rightarrow OUT$ $SP \rightarrow SP$
I ₈	POP R	$SP \leftarrow SP - 1$ $R \leftarrow (SP)$	$SP \rightarrow SP$ $SP \rightarrow OUT$ $IN \rightarrow R$

Graf G:



2. Funkcje spełniane przez układy mikroprocesora dzieli się na pięć klas:

- wybieranie rejestrów,
- wybieranie instrukcji i sterowania,
- pamiętanie informacji,
- przesyłanie informacji,
- przetwarzanie informacji.

3. Model niezdatności formuluje się na poziomie ww. funkcji.

Dopuszcza się równoczesne wystąpienie dowolnej liczby błędów, ale tylko w obrębie jednej klasy funkcji.

Dla funkcji wybierania rejestrów:

funkcja wybierania rejestrów:

$$f_D : \mathfrak{R} \rightarrow \mathfrak{R} \cup \{\Phi\}$$

jeżeli procesor jest zdatny to

$$\bigwedge_{R_i \in \mathfrak{R}} f_D(R_i) = R_i$$

Błędne zachowanie procesora:

$$\begin{aligned} f_D(R_i) &= \Phi \\ f_D(R_i) &= R_i, R_j, R_k, \dots \\ f_D(R_i) &= R_i, \quad f_D(R_j) = R_i. \end{aligned}$$

Podobnie formuluje się model niezdatności dla pozostałych funkcji procesora.

4. Testy programowe generuje się na podstawie grafu G tak, aby zapewnić pokrycie przyjętych klas błędów.