

Implementace software na Test Generation a Fault Diagnosis

Implementation of software for Test Generation and Fault Diagnosis

Bc. Jakub Sedláček



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Bc. Jakub SEDLÁČEK
Osobní číslo: A09395
Studijní program: N 3902 Inženýrská informatika
Studijní obor: Bezpečnostní technologie, systémy a management

Téma práce: Implementace software na Test Generation a Fault Diagnosis

Zásady pro vypracování:

1. Zpracujte literární průzkum z oblasti logických funkcí a jejich diagnostiky.
2. Seznamte se s volně dostupným software, který provádí Test Generation pomocí řady metod, např. fault-oriented, fault independent taktéž Fault Diagnosis, např. pomocí metody adaptivního testování.
3. Vytvořte novou laboratorní úlohu pro předmět Diagnostika číslicových systémů, která bude pomocí těchto metod na konkrétní příkladech ověřovat znalosti studentů.
4. Vypracujte návod pro tuto úlohu, který bude obsahovat nezbytnou teorii.
5. Umístěte na samostatné médium jako přílohu práce.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

1. ANTOŠOVÁ, M.; DAVIDEK, V. Číslicová Technika. Vyd.1. České Budějovice : KOOP, 2003. 288 s. ISBN 80-7232-206-0.
2. UBAR, R., WUTTKE, H.-D. Action Based Learning System for Teaching Digital Electronics and Test. Proc. of 3rd European Workshop on Microelectronics Education, Aix-en-Provence, France, 2000, p.65-66, ISBN 978-0-7923-6456-6.
3. UBAR, R., ORASSON, E., EVARTSON, T. Java Applet for Self-Learning of Digital Test Issues. 13th EAEIE Conference, York, Great Britain, 2002, ISBN 1-85911-008-8.
4. DEVADZE, S., JUTMAN, A., SUDNITSON, A., UBAR, R. Web-based training system for teaching basics of RT-level Digital Design, Test, and Design for Test, in Proc. of 9th International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES 2002), Wroclaw, Poland, June 20-22, 2002, pp. 699-704
5. KORBICZ, J.; KOSCIELNY, J. M.; KOWALCZUK, Z. Fault diagnosis : models, artificial intelligence, applications. 1. vyd. Germany : Springer-Verlag Berlin Heidelberg, 2004. 920 s., ISBN 978-3-540-40767-6.

Vedoucí diplomové práce:

Ing. Karel Perůtka, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

25. února 2011

Termín odevzdání diplomové práce:

27. května 2011

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Hlavním cílem diplomové práce bylo vytvořit laboratorní úkoly na problematiku diagnostiky integrovaných obvodů. Práce je zaměřena na použité metody na odhalování chyb v logických obvodech. V další části je práce zaměřena na software, který bude sloužit k vypracování laboratorních úloh. Dále obsahuje popis tohoto programu a laboratorní úlohy k vypracování.

Klíčová slova: Test Generation, Fault Diagnosis, Diagnostika, Tabulka chyb, Testovací vektor, Elektronický obvod

ABSTRACT

The main objective of this thesis was to create laboratory diagnostic tasks on the issues of integrate circuits. The work is focused on the method used to detect faults in logic circuits. In another part of the work is focused on software that will be used to develop laboratory exercises. Further it contains a description of this program and to develop laboratory tasks.

Keywords: Test Generation, Fault Diagnosis, Diagnosis, Fault Table, Test vectors, Electronic Circuit

Poděkování, motto

Děkuji tímto vedoucímu mé diplomové práce Ing. Karlu Perůtkovi, Ph.D. za odborné vedení, trpělivost při konzultacích, cenné rady a připomínky, které mi pomohly v průběhu jejího řešení.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	10
I TEORETICKÁ ČÁST.....	11
1 LOGIKA.....	12
1.1 ČÍSELNÉ SOUSTAVY	12
1.1.1 Soustava dvojková	12
1.1.2 Soustava osmičková	12
1.1.3 Soustava desítková	13
1.1.4 Soustava šestnáctková	13
1.1.5 Přehled jednotlivých soustav	13
1.2 LOGICKÉ FUNKCE	13
1.2.1 Logické funkce jedné proměnné	14
1.2.2 Logické funkce dvou proměnných	14
2 DIAGNOSTIKA	18
2.1 ZÁKLADNÍ POJMY	18
2.2 METODY KOMBINAČNÍCH PORUCH DIAGNOSTIKY	19
2.2.1 Tabulka chyb	19
2.2.2 Porucha adresáře	20
2.2.3 Minimalizace diagnostických dat	21
2.2.4 Chyba umístění ze strukturálních analýz	21
2.3 METODY SEKVENČNÍ DIAGNOSTIKY PORUCH	21
2.3.1 Místo poruchy v edge-pin testování	22
2.3.2 Generování testů k odlišení závad	23
2.3.3 Guided – probe testování	24
2.3.4 Snížení umístění chyb v UUT	26
2.4 TESTOVÁNÍ ELEKTRONICKÝCH PRVKŮ	26
2.5 DIAGNOSTIKA ČÍSLICOVÝCH OBVODŮ	27
2.5.1 Strukturální testy	27
2.5.1.1 Intuitivní zcitlivění cesty	28
2.5.1.2 Algoritmus D	28
2.5.2 Funkční testy	29
2.5.3 Náhodné a pseudonáhodné test	29
2.5.4 Poruchy modelování a simulace	29
2.5.4.1 Logické chyby modelů	29
2.5.4.2 Strukturální chyby	30
2.5.5 Detekce poruch a redundance	30
2.5.5.1 Detekce poruch v kombinačních obvodech	30
2.5.5.2 Odhalení závad	32
2.5.5.3 Redundance	32
2.5.6 Poruchy rovnocennosti a lokalizace poruchy	33
2.5.6.1 Porucha třídy ekvivalence	33
2.5.6.2 Ekvivalence poruchy zhroucení	34
2.5.6.3 Lokalizace poruchy	34
2.5.7 Dominantní chyba	34
2.5.8 Single stuck-fault model	35
2.5.9 Multiple stuck-fault model	35

2.5.9.1	Počet opakujících se chyb	35
2.5.9.2	Chyba maskování	35
2.5.9.3	Kruhové poruchy maskování	36
3	TEST GENERÁTOR	38
3.1	GATE-LEVEL FAULT-ORIENTED TEST GENERÁTOR	38
3.1.1	Fanout-free obvody	38
3.1.1.1	Porucha aktivace	38
3.1.1.2	Porucha šíření	39
3.1.1.3	Odůvodnění řádku	39
II	PRAKTICKÁ ČÁST	41
4	PROGRAM PRO VÝUKU ZÁKLADŮ DIAGNOSTIKY	42
4.1	ZÁKLADNÍ POPIS	42
4.2	ZKRÁCENÁ TEORIE	42
4.2.1	Test generátor	42
4.2.2	Diagnostika poruch	43
4.2.3	Metoda kombinační diagnostiky poruch	43
4.2.4	Metody sekvenčních diagnostických poruch	44
4.2.5	Příklad lokalizace poruch na edge-pin testování	44
4.2.6	Guided-probe testování	45
4.3	PŘÍRUČKA K PROGRAMU	45
4.3.1	Pracovní okno	45
4.3.2	Panel vložení vektoru	45
4.3.3	Panel pro schéma	46
4.3.4	Schéma zobrazení průběhu dat	47
4.3.5	Panel dat	47
4.3.6	Menu	48
4.3.7	První spuštění	48
4.3.8	Vkládání test vektoru	48
4.3.9	Test generátor	49
4.3.10	Chyba simulace	50
4.3.11	Diagnostika	50
5	CVIČENÍ	52
5.1	POSTUP PŘI PROVÁDĚNÍ DIAGNOSTIKY	52
5.1.1	Pseudonáhodně generovaný test pomocí LFSR	52
5.1.2	Studie tabulky poruch	56
5.1.3	Manuální (deterministický) zkušební vzor generování	56
5.1.4	Guided probing diagnosis	57
5.1.5	Kombinační diagnostika (fault table)	58
5.1.6	Příklad	58
5.1.7	Pseudonáhodné generování testu pomocí LFSR	59
5.1.8	Studie tabulky chyb	60
5.1.9	Manuální (deterministický) zkušební vzor generování – příklad	60
5.1.10	Guided probing diagnosis – příklad	61
5.1.11	Kombinační diagnostika (fault table) – příklad	62
6	ZADÁNÍ PROTOKOLŮ	65

6.1	ZADÁNÍ PROTOKOLU Č. 1.....	65
6.2	ZADÁNÍ PROTOKOLU Č. 2.....	66
6.3	ZADÁNÍ PROTOKOLU Č. 3.....	67
6.4	ZADÁNÍ PROTOKOLU Č. 4.....	68
6.5	ZADÁNÍ PROTOKOLU Č. 5.....	69
6.6	ZADÁNÍ PROTOKOLU Č. 6.....	70
6.7	ZADÁNÍ PROTOKOLU Č. 7.....	71
6.8	ZADÁNÍ PROTOKOLU Č. 8.....	72
6.9	ZADÁNÍ PROTOKOLU Č. 9.....	73
6.10	ZADÁNÍ PROTOKOLU Č. 10.....	74
6.11	ZADÁNÍ PROTOKOLU Č. 11.....	75
6.12	ZADÁNÍ PROTOKOLU Č. 12.....	76
7	VYPRACOVÁNÍ PROTOKOLŮ.....	77
7.1	VYPRACOVÁNÍ PROTOKOLU Č. 1.....	77
7.2	VYPRACOVÁNÍ PROTOKOLU Č. 2.....	80
7.3	VYPRACOVÁNÍ PROTOKOLU Č. 3.....	82
7.4	VYPRACOVÁNÍ PROTOKOLU Č. 4.....	85
7.5	VYPRACOVÁNÍ PROTOKOLU Č. 5.....	87
7.6	VYPRACOVÁNÍ PROTOKOLU Č. 6.....	91
7.7	VYPRACOVÁNÍ PROTOKOLU Č. 7.....	95
7.8	VYPRACOVÁNÍ PROTOKOLU Č. 8.....	100
7.9	VYPRACOVÁNÍ PROTOKOLU Č. 9.....	104
7.10	VYPRACOVÁNÍ PROTOKOLU Č. 10.....	108
7.11	VYPRACOVÁNÍ PROTOKOLU Č. 11.....	112
7.12	VYPRACOVÁNÍ PROTOKOLU Č. 12.....	116
	ZÁVĚR	120
	ZÁVĚR V ANGLIČTINĚ.....	121
	SEZNAM POUŽITÉ LITERATURY.....	122
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	123
	SEZNAM OBRÁZKŮ	124
	SEZNAM TABULEK.....	126
	SEZNAM PŘÍLOH.....	127

ÚVOD

V dnešní době je integrování velice rozšířené. V elektronice je dnešním trendem dosáhnout miniaturizace a integrace více prvků a systému do jednoho. Proto se veškeré elektronické obvody skládají z integrovaných obvodů. Tyto jsou v dnešní době rozsáhlejší a složitější než dříve. Dnešní elektronický obvod obsahují více integrovaných obvodů, protože zapojení je jednodušší. Problém ovšem nastane, pokud se v obvodu vyskytne chyba. Je složitější ji najít a určit kde daná chyba vznikla.

Hlavním cílem diplomové práce je vytvořit vhodný výukový materiál pro studenty, kteří se seznámí s identifikacemi chyb v elektronických obvodech.

Práci lze rozdělit do dvou částí. V první části se nachází prvky z číslicové techniky, protože obvody nevyužívají základní desítkovou soustavu. Popis různých integrovaných obvodů a jejich vlastností. Dále jsou zde popsány druhy poruch, které mohou vzniknout ať již během výroby nebo během odzkoušení integrovaných obvodů. A následně druhy identifikací poruch v elektronických obvodech. Jak se postupuje při vyhledávání závad a co je třeba udělat pro odstranění takových poruch.

Druhá část se zaměřuje na konkrétní výukový program, který bude sloužit pro studenty jako simulační výukový materiál. Je zde popsáno základní používání programu, jeho funkce a vzorový příklad pro lepší pochopení funkce celého programu. Poslední část práce je zaměřena na laboratorní úlohy. Tyto úkoly jsou sestaveny pro studenty tak aby pochopili jednotlivé metody, které program nabízí a určili, která z metod je vhodnější. Na těchto laboratorních úlohách si studenti vyzkouší své znalosti a dovednosti. Protokol by měl obsahovat naměřená data, postup práce, celkové zhodnocení měření a problémy, které při měření nastaly.

I. TEORETICKÁ ČÁST

1 LOGIKA

Logika má více významů. V češtině se běžně používá ve smyslu myšlenkové cesty, který vede k daným závěrům. Logika je také formální věda, zkoumající právě onen způsob vyvozování závěrů. Jako mnoho dalších věd vznikla logika coby součást filosofie a částečně takové zařazení stále platí. Logika se výrazně rozvinula i v matematice, a tak se řazena i do matematiky. Některé části logiky mají blíže k filosofii, některé k matematice proto se někdy rozlišuje matematická logika. Logika je nauka o základech myšlení v procesu vytváření úsudku a důkazů.[2]

1.1 Číselné soustavy

Každé číslicové zařízení pracuje s čísly, která jsou pro nás symbolem určitého množství. Desítkovou soustavu využíváme v běžném životě a při aritmetických operacích. Dostáváme tak jednoznačné a nám srozumitelné výsledky. Desítková soustava je sestavena z 10 čísel od 0 po 9. Tato soustava je vhodná pro nás, ale ne pro počítačové nebo číslicové systémy. Pro vyhodnocení by bylo zapotřebí rozlišovat 10 různých úrovní, což by kladlo velké požadavky na přesnost a kvalitu zařízení. Proto využíváme jiné vhodnější číslicové soustavy a to soustavy dvojkové, která využívá dvou hodnot 0 a 1. [1]

1.1.1 Soustava dvojková

Též nazývána jako soustava binární nebo dyadická. Soustava používá jen dvě číslice 0 a 1. Tyto hodnoty jsou vyhovující pro číslicové obvody, protože si můžeme například definovat tak, že hodnota 1 bude odpovídat vysokým hodnotám napětí 9 až 10 V a hodnota 0 odpovídá nízkým hodnotám napětí 0 až 1 V. [2]

1.1.2 Soustava osmičková

Tato soustava taktéž nazývána oktálová využívá osm číslic násobících koeficientů. Jedná se o čísla 0,1,2,3,4,5,6,7. Tato soustava se dříve společně se šestnáctkovou soustavou používala u starších typů sálových počítačů s logickými obvody. Dnes je jejich použití omezeno. [3]

1.1.3 Soustava desítková

Soustava dekadický je nejpoužívanější metoda v běžném životě a aritmetice. Využívá znaky 0,1,2,3,4,5,6,7,8,9. [4]

1.1.4 Soustava šestnáctková

Též soustava hexadecimální využívá 16 číslic a znaků. Pro vyjádření znaků 10 až 15 využíváme alfabetickými znaky A až F. Šestnáctková soustava se používá v mikropočítačové technice k popisu dat na adresové a datové sběrnici. [5]

1.1.5 Přehled jednotlivých soustav

Dekadická soustava	Dvojková soustava	Osmičková soustava	Šestnáctková soustava
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Tab. 1 Přehled nejčastěji používaných čísel v číselných soustavách [1]

1.2 Logické funkce

Logika je nauka o základech myšlení v procesu vytváření úsudku a důkazu. Logický obvod je takový obvod, u něhož může každá veličina na vstupu i výstupu v ustáleném stavu nabývat s určitou přesností jen jedné ze dvou možných hodnot a který obsahuje takové prvky, jejichž vstupní a výstupní veličiny mohou nabývat také jen jedné ze dvou

možných hodnot. Logický obvod je realizován skupinou logických členů vzájemně spojených tak, aby realizovaly požadované logické funkce. [1]

Logické obvody dělíme dle druhu realizované logiky na:

- Kombinační logické obvody – je takový obvod, který má výstupní signál jednoznačně určený okamžitými hodnotami vstupních signálů
- Sekvenční logické obvody – sekvence je chápána jako časová posloupnost, výstup sekvenčního obvodu závisí na vstupních kombinacích a na jejich předchozím stavu případně i na vnitřním stavu. Jediné kombinaci vstupů odpovídá tedy obecně více různých hodnot výstupů. Sekvenční obvod má tedy paměť pro všechny, nebo jen několik vstupních a výstupních hodnot. Ve schématu můžeme sekvenční obvod rozpoznat podle jeho specifické zpětné vazby. [6]

1.2.1 Logické funkce jedné proměnné

Logickou funkci jedné nezávislé proměnné lze vyjádřit čtyřmi způsoby. Vstupní proměnná je označena x , logické funkce jsou f_0 , f_1 , f_2 a f_3 . [1]

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Tab. 2 Logické funkce jedné nezávislé proměnné [1]

Význam jednotlivých logických funkcí:

- f_0 – triviální identita, nulová funkce
- f_1 – identická funkce, její hodnota se shoduje s hodnotou proměnné x
- f_2 – funkce negace, hodnota funkce se shoduje s hodnotou \bar{x}
- f_3 – triviální identita, jednotková funkce

1.2.2 Logické funkce dvou proměnných

Pro dvě vstupní proměnné můžeme najít šestnáct různých logických funkcí. Vstupní proměnné označíme x , y a logické funkce f_0 až f_{15} . [7]

x	0	0	1	1	Význam jednotlivých funkcí
y	0	1	0	1	
f_0	0	0	0	0	$f_0 = 0$, triviální identita
f_1	0	0	0	1	$f_1 = a \cdot b$, logický součin. AND
f_2	0	0	1	0	$f_2 = a \cdot \bar{b}$, přímá inhibice
f_3	0	0	1	1	$f_3 = a$, funkce identická
f_4	0	1	0	0	$f_4 = \bar{a} \cdot b$, zpětná vazba
f_5	0	1	0	1	$f_5 = b$, funkce identická
f_6	0	1	1	0	$f_6 = a \oplus b$, nonekvivalence, XOR, součet modulo 2
f_7	0	1	1	1	$f_7 = a + b$, logický součet, OR
f_8	1	0	0	0	$f_8 = \bar{a} + \bar{b}$, negovaný logický součet, NOR, Piercova funkce
f_9	1	0	0	1	$f_9 = \bar{a} \oplus \bar{b}$, ekvivalence, XNOR
f_{10}	1	0	1	0	$f_{10} = \bar{b}$, negace
f_{11}	1	0	1	1	$f_{11} = a + \bar{b}$, zpětná vazba
f_{12}	1	1	0	0	$f_{12} = \bar{a}$, negace
f_{13}	1	1	0	1	$f_{13} = \bar{a} + b$, přímá implikace
f_{14}	1	1	1	0	$f_{14} = \bar{a} \cdot \bar{b}$, negovaný logický součin, NAND, Shefferova funkce
f_{15}	1	1	1	1	$f_{15} = 1$, triviální identita

Tab. 3 Logické funkce dvou vstupních proměnných [1]

Předchozí tabulka je složena ze tří základních logických funkcí.

- Logický součin – jedná se o funkci v tabulce označenou jako f_1 . Máme dvě proměnné x a y označené jako AND. Pokud označíme výstup logického obvodu jako z , pak logický součin lze zapsat jako

$$z = x \cdot y$$

- Logický součet- jedná se o funkci v tabulce označenou jako f_7 . Tato funkce je označena jako OR a lze ji zapsat vztahem

$$z = x + y$$

- Negace - jedná se o funkci v tabulce označenou jako f_{10} a f_{12} . Je označena NOT a popisuje ji vztah

$$z = \bar{x} \quad \text{nebo} \quad z = \bar{y}$$

Tyto vztahy se nazývají úplný soubor logických funkcí, protože jejich kombinací můžeme navrhnout jakýkoliv číslicový obvod. [8]

Další významné funkce uvedené v tabulce jsou.

- NAND - negovaný logický součin označen jako funkce f_{14}

$$z = \overline{x \cdot y}$$

- NOR - negovaný logický součet označen jako funkce f_8

$$z = \overline{x + y}$$

Pomocí funkcí NAND a NOR lze vyjádřit a realizovat každou logickou funkci.

Poslední rovnice nacházející se v tabulce jsou.

- Nonekvivalence – funkce f_6

$$z = x \oplus y = \overline{x} \cdot y + x \cdot \overline{y}$$



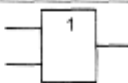

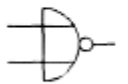

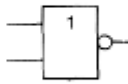
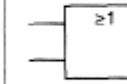



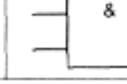




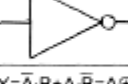
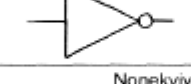
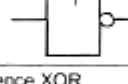





Funkce nonekvivalence nabývá hodnoty 1, jsou-li hodnoty obou proměnných různé. Tato funkce se také označuje jako Exclusive OR a značí se XOR.

- Ekvivalence – funkce f_9 XNOR

$$z = \overline{x \oplus y} = x \cdot y + \overline{x \cdot y}$$

Funkce ekvivalence nabývá hodnoty 1, jsou-li hodnoty obou proměnných stejné. Jedná se o negaci funkce nonekvivalence a označuje se XNOR. Funkce XOR a XNOR jsou významné pro realizaci alfanumerických obvodů a číslicových komparátorů. [9]

Logické funkce realizujeme pomocí základních logických členů, jejichž spojováním získáme logický obvod. Pro kreslení schémat, logických schémat se využívá schematických značek. [10]

$Y=A+B$	Log. součet OR		
			
$Y=\overline{A+B}$	Negovaný log. součet NOR		
			
$Y=A \cdot B$	Log. součin AND		
			
$Y=\overline{A \cdot B}$	Negovaný log. součin NAND		
			
$Y=\overline{A}$	Negace NOT		
			
$Y=\overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$	Nonekvivalence XOR		
			
DIN	US	ČSN	IEC

Tab. 4 Logické funkce dvou vstupních proměnných [1]

Označení DIN je z německých norem, ale v dnešní době je zastaralé a již se v širší míře nepoužívá. Označení US je americké a najdeme je ve většině zahraničních katalogů číslíkových prvků. ČSN je dříve používané označení našimi normami. IEC je označení Mezinárodní elektrotechnické komise. [11]

2 DIAGNOSTIKA

Diagnostika je nauka o zjišťování poruch, popřípadě celkového stavu diagnostického zařízení. Je používána v různých oborech s rozdílem použití logiky, analytických zkušeností k určení příčiny a následku mezi jednotlivými vztahy.[2]

2.1 Základní pojmy

Testované jednotky (UUT) selžou, pokud jejich výsledná reakce je odlišná od reakcí předpokládaných. Diagnostika se skládá z umístění fyzické poruchy do strukturálního modelu testované jednotky UUT. Stupeň přesnosti, do níž mohou být chyby umístěny, se nazývá diagnostické rozlišení. Funkčně ekvivalentní poruchy (FEF) není možné odlišit. Rozdělení všech poruch na různé podskupiny FEF definuje maximální chyba rozlišení. Test, který dosáhne maximálního rozlišení chyb, se nazývá kompletní fault-location test.[4]

Oprava UUT se často skládá z nahrazení jedné ze svých vyměnitelných jednotek (RU) uvedených jako vadná RU. Spíše než přesná identifikace skutečné poruchy uvnitř RU. Charakterizujeme tento proces jako vyřešení RU. Předpokládejme, že výsledný test neumožňuje rozlišovat mezi dvěma podezřelými vyměnitelnými jednotkami RU U1 a U2. Mohli bychom nahradit jednu z těchto RU, například U1 dobrou jednotkou a vrátit se tak do zkušebního stavu. Pokud budou nové výsledky správné, byla vadná jednotka nahrazena jinak je vadná jednotka U2. Tomuto typu postupu se říká *sequential diagnosis procedure*. [4]

Diagnostický proces je často hierarchický, prováděný jako top-down proces (s operačním systémem), nebo bottom-up proces (během výrobního procesu). [4]

V top-down přístupu (systém – deska - IO) první úroveň diagnostiky se může zabývat „velkým“ RU jako takzvané desky se nazývají pole vyměnitelných jednotek. Vadná deska je pak testována v údržbovém centru kde hledají vadné komponenty (IO) na desce. Přesné zjištění poruchy uvnitř vadných IO může být také užitečné pro zlepšení výrobního procesu. [4]

V bottom-up přístupu (IO – deska - systém) vyšší úroveň je sestavena pouze z komponent již testovaných na nižší úrovni. Takto se sníží náklady za diagnostiku a opravy, čímž se zvýší úroveň, při které jsou zjištěny závady. [4]

Ve zpracovatelském průmyslu s výskytem největší pravděpodobnosti vzniku chyb jsou chyby, které ovlivňují výrobu spojené se součástkami, v oblasti největší pravděpodobnosti vzniku chyby jsou fyzické poruchy uvnitř komponent (protože každá UUT byla úspěšně testována). Při určování pravděpodobnosti vzniku chyby pomáhá lokalizace chyb. [4]

2.2 Metody kombinačních poruch diagnostiky

Tento postup dělá většinu práce před zkušebním testováním. Využívá chyby vytvořené simulací k určení možných výsledků na daný test s přítomností poruchy. Databáze postavená v tomto kroku se nazývá tabulka poruch (fault table) nebo porucha adresáře (fault dictionary). Pro lokalizaci chyby se snažíme porovnat skutečné výsledky z experimentálního testu s jedním očekávaným spočteným výsledkem uloženým v databázi. Výsledky experimentálního testu představují kombinaci účinků poruch na každý zkušební model. Proto se tento přístup nazývá metoda kombinační diagnostiky. Pokud je tento proces úspěšně proveden tak tabulka poruch (adresáře) udává odpovídající chyby. [4]

2.2.1 Tabulka chyb

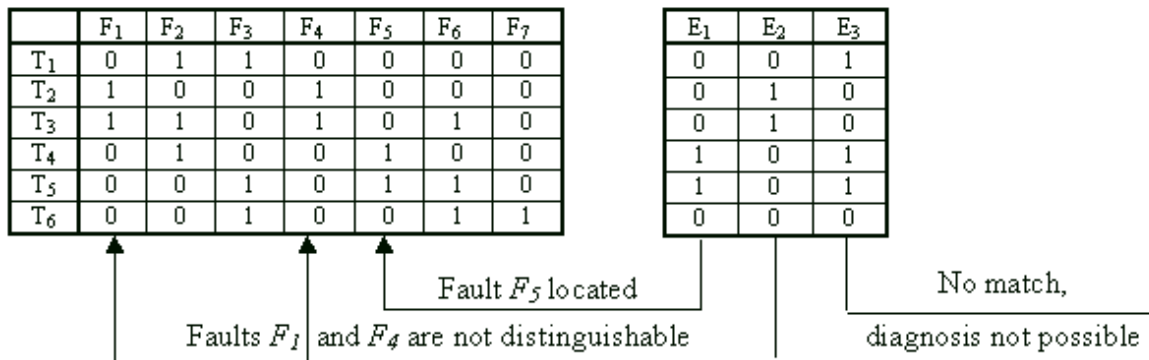
Obecně platí, že tabulka chyb je matice $FT = \|a_{ij}\|$ kde sloupce F_j představují závadu a řádky T_i představují zkušební vzory, $a_{ij} = 1$, pokud zkušební vzorek T_i detekuje poruchu F_j , jinak v případě, že testovací předloha T_i nezjistí závady F_j , $a_{ij} = 0$. [4]

Označme skutečný výsledek daného testovaného vzoru 1, pokud se liší od očekávaného přepočteného výsledku, označíme jej 0. Výsledný experimentální test je reprezentovaný vektorem $E = |e_i|$ kde $e_i = 1$. Pokud se skutečný výsledek testovaného modelu neshoduje s očekávaným výsledkem je $e_i = 0$. Každý sloupec vektoru f_j odpovídá poruše F_j , představuje tak možný výsledek experimentálního testu v případě poruchy F_j . [4]

Mohou nastat tři případy při provádění experimentální zkoušky závislé na kvalitě zkušebních modelů. [4]

- Výsledek testu E odpovídá jednomu sloupci vektoru f_j v matici FT . Tento výsledek odpovídá případu, kdy je zjištěna jedna porucha F_j . Jinými slovy jsme dosáhli maximálního diagnostického rozlišení.

- Výsledek testu E odpovídá podmnožině sloupcí vektorů $\{f_i, f_j, \dots, f_k\}$ v matici FT . Tento výsledek odpovídá případu, kdy podmnožina nerozeznatelných poruch $\{F_i, j \dots F F_k\}$ je již zjištěna.
- Nebyla získána žádná shoda pro E se sloupci a vektory v matici FT

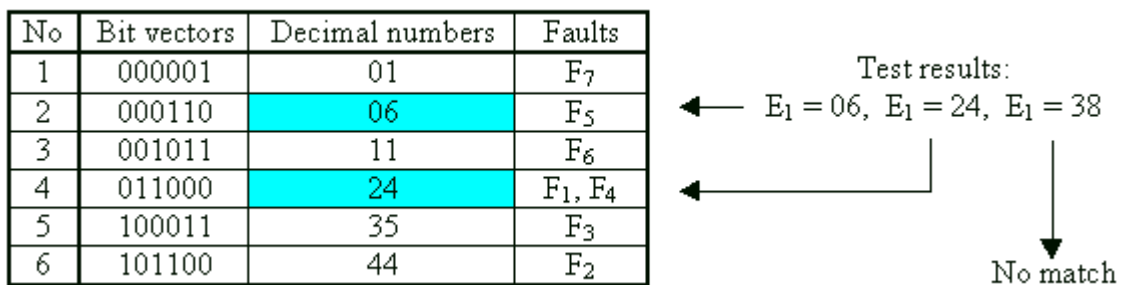


Obr. 1 Příklad tabulkových poruch [4]

Obrázek nám znázorňuje výsledky popsáných bodů. E_1 odpovídá prvnímu bodu, kdy je chyba jen jedna. E_2 odpovídá druhému bodu, kdy se dvě nerozeznatelné chyby nachází ve stejné podmnožině. E_3 představuje závadu, která nemůže být umístěna z důvodu nesouladu se sloupci vektorů v tabulce chyb. [4]

2.2.2 Porucha adresáře

Porucha adresáře (FD) obsahuje stejné údaje jako tabulka chyb s tím rozdílem, že data jsou reorganizována. V FD mapování mezi potenciálními výsledky testovaných experimentů a poruch je zastoupena ve více redukované a přísnější formě. Například sloupce bitů vektorů mohou být reprezentovány jako desetinný kód, nebo jako nějaký druh komprimovaného podpisu. [4]



Obr. 2 Příklad poruchy adresáře [4]

2.2.3 Minimalizace diagnostických dat

Chcete-li snížit množství výpočetního úsilí, zapojte se do vytvoření tabulky chyb, v simulaci poruch se zjištěné závady sníží ze souboru simulovaných poruch. Proto, všechny poruchy zjištěné v témže vektoru budou produkovat stejný sloupcový vektor v tabulce poruch a budou zahrnuty do stejné třídy rovnocennosti poruch. V tomto případě může být zkušební test zastaven po prvním selhání testu, protože informace poskytované těmito testy se nepoužívají. Takovýto test dosáhne nižšího diagnostického rozlišení. Kompromis mezi výpočetním časem a diagnostickým rozlišením lze dosáhnout tím, že vypustíme chyby $k > 1$ detekcí. [4]

Ve vytvořené tabulce chyb s chybnou simulací způsobených chybami pádu, je třeba pouze 19 chyb simulačně srovnat s procesem ze 42 chyb kdy simulace s chybami pádu je odnášena pryč (simulační chyby jsou v tabulce chyb zobrazeny vybarveným políčkem). Nicméně v důsledku poruch pádu tyto chyby zůstanou nerozlišitelné ($\{F_2, F_3\}$, $\{F_1, F_4\}$, $\{F_2, F_6\}$). [4]

	F ₁	F ₂	F ₃	F ₄	F ₅	F ₆	F ₇
T ₁	0	1	1	0	0	0	0
T ₂	1	0	0	1	0	0	0
T ₃	0	0	0	0	0	1	0
T ₄	0	0	0	0	1	0	0
T ₅	0	0	0	0	0	0	0
T ₆	0	0	0	0	0	0	1

Obr. 3 Minimalizace diagnostických dat [4]

2.2.4 Chyba umístění ze strukturálních analýz

Předpokládejme jednu poruchu v obvodu. Měla by existovat cesta z místa poruchy ke každému z výstupů, kde byla zjištěna chyba. Proto by měla chyba polohy patřit k průsečíku všech selhaných výstupů. Jednoduché strukturální analýzy mohou pomoci nalézt chyby, které mohou vysvětlit všechny pozorované chyby. [4]

2.3 Metody sekvenční diagnostiky poruch

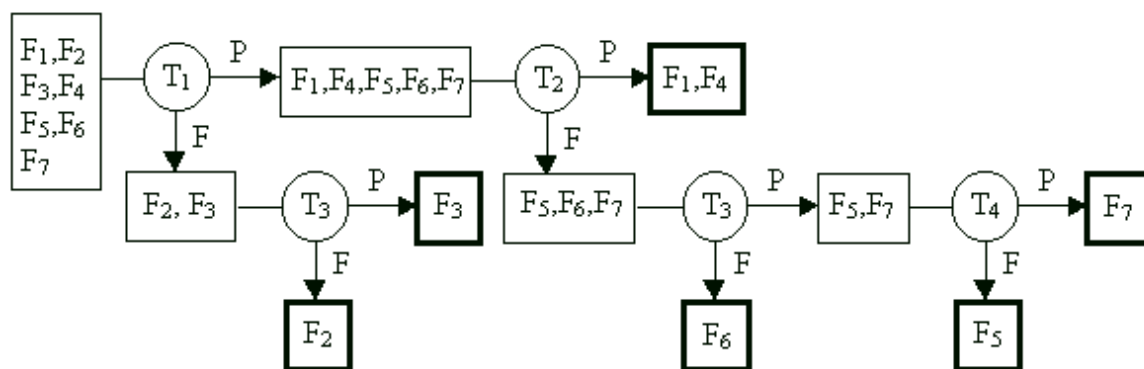
V sekvenčním diagnostickém procesu se lokalizace poruchy provádí krok po kroku, kde každý krok závisí na výsledku diagnostického testu v předchozím kroku. Takový zkušební test se nazývá adaptivní testování. Sekvenční experimenty mohou být prováděny buď tím, že sledujeme pouze výstupní hodnoty zkoušené jednotky nebo zjišťováním

pomocí speciální sondy na vnitřním kontrolním bodu zkoušené jednotky (řízené snímání). Postup sekvenční diagnostiky může být graficky znázorněn jako diagnostický strom. [4]

2.3.1 Místo poruchy v edge-pin testování

Poruchy diagnostického testu jsou aplikovány na UUT krok za krokem. V každém kroku, jsou sledovány pouze výstupní signály z edge-pin z UUT a jejich hodnoty jsou srovnávány s očekávanými hodnotami. Další testovaný model, který se provádí, je adaptivní testování závislé na výsledku předchozího kroku. Diagnostický strom tohoto procesu se skládá z uzlů poruch FN (obdélníků) a zkušebních uzlů TN (kruhů). FN je označen jako soubor dosud nerozlišených vad. Výchozí chyba uzlu je označena jako soubor všech poruch. Ke každému FN k TN je připojen zkušební model T_k a může být použit jako další. Každý testovaný model rozlišuje mezi chybami detekce a ostatními. Úkolem tetu T_k je rozdělit vady FN k do dvou skupin, zjistitelné a nezjistitelné závady T_k . Každý testovaný uzel má dva výstupní stavy odpovídající výsledku experimentu daného testovaného modelu. Výsledky jsou uváděny jako vyhovující (P) nebo nevyhovující (F). Soubor poruch je v aktuálním chybném uzlu (obdélník) jsou ekvivalentní (nerozlišující) na základě současné použité testovací sady. [4]

Na následujícím obrázku můžeme vidět, že většina poruch je jednoznačně identifikována, dvě chyby F_1, F_4 zůstávají nerozeznány. Ne všechny testovací vzory použité v tabulce chyb jsou potřebné. Různé závady potřebují pro identifikaci testovacích sekvencí různé délky. Nejkratší test obsahuje dva modely s nejdelšími čtyřmi modely. [4]



Obr. 4 Místo poruchy v edge-pin testování [4]

Spíše než použití celé testovací sekvence v pevném pořadí, jako v diagnostice kombinačních poruch, adaptivní testování určuje další vektor, který se použije na základě

výsledků získaných v předchozím vektoru. V našem příkladu, jestliže T_1 selže, možné chyby jsou $\{F_2, F_3\}$. V tomto bodě, použití T_2 by bylo nešetrné, protože T_2 nerozlišuje mezi těmito chybami. Použití adaptivního testování může podstatně snížit průměrný počet zkoušek k nalezení chyb. [4]

2.3.2 Generování testů k odlišení závad

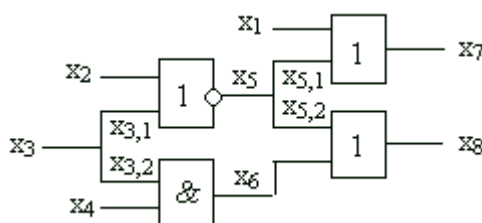
Pro zlepšení chyby řešení daného souboru testů T , je nutné vytvořit testy tak aby bylo možné rozlišovat mezi poruchami ekvivalence pod T . [4]

Zvažme problémy generování testů v rozlišení mezi poruchami F_1 a F_2 . Takový test musí detekovat jednu z těchto chyb, ale ne jinou, nebo naopak. Jsou možné tyto případy. [4]

- F_1 a F_2 nemají vliv na stejnou sadu výstupů. Necht' $OUT(F_k)$ je množina výstupů ovlivněných chyb F_k . Zkouška by měla být generována pro F_1 pouze pomocí obvodu napájení výstupu $OUT(F_1)$, nebo pro F_2 pouze pomocí obvodu napájení výstupu $OUT(F_2)$
- F_1 a F_2 ovlivňují stejnou sadu výstupů. Zkouška by měla být generována pro F_1 bez aktivace F_2 , nebo naopak, za F_2 bez aktivace F_1

Mohou být zmíněny tři možnosti k udržení chyb F_2 : $X_k \equiv e$ není aktivovaný, kde X_k je linka v obvodu a $e \in \{0, 1\}$:

- Hodnota $\neg e$ by měla být přiřazena k lince X_k
- Pokud to není možné, pak se aktivuje, část z F_2 by měla být blokována, tak aby chyba F_2 se nemohla šířit aktivní cestou z F_1
- Pokud není možný ani druhý případ, pak propagované hodnoty z místa F_1 a F_2 sahají ke stejnému hradlu G a měly by být naproti na vstupu hradla G .



Obr. 5 Odlišení závad [4]

- V obvodu jsou dvě chyby – F_1 : $X_{3,1} \equiv 0$ a F_2 : $X_4 \equiv 1$. Chyba F_1 může ovlivnit oba výstupy, chyba F_2 může ovlivnit pouze výstup x_8 . Testovací

vzor 0010 aktivuje F_1 až do obou výstupů a F_2 pouze na X_8 . Jestliže jsou oba výstupy špatné, F_1 je chybný a pokud je pouze výstup X_8 špatný, je chybný F_2 .

- V obvodu jsou dvě chyby – F_1 : $X_{3,2} \equiv 0$ a F_2 : $X_{5,2} \equiv 1$. Obě mají vliv na stejný výstup v obvodu. Testovací vzor 0100 aktivuje chybu F_2 . Chyba F_1 není aktivována, protože linka $X_{3,2}$ má stejnou hodnotu jakou by měla kdyby F_1 byla aktivována.
- V obvodu jsou dvě stejné chyby – F_1 : $X_{3,2} \equiv 0$ a F_2 : $X_{5,2} \equiv 1$. Obě mají vliv na stejný výstup v obvodu. Testovací vzor 0110 aktivuje chybu F_2 . Chyba F_1 je aktivována na svém místě, ale nešíří se přes bránu AND, protože hodnota $X_4 = 0$ na jejím vstupu.
- V obvodu jsou dvě chyby – F_1 : $X_{3,1} \equiv 0$ a F_2 : $X_{3,2} \equiv 1$. Testovací vzor se skládá z 1001 hodnoty $X_1 = 1$, který vytváří stav, kdy obě závady můžou ovlivnit pouze stejný výstup X_8 . Na druhou stranu testovací vzor 1001 aktivuje obě závady na stejné bráně OR (žádný z nich není blokován). Nicméně, chyby produkují různé hodnoty na vstupu brány a proto jsou odlišeny. Pokud výstupní hodnoty na X_8 budou 0, F_1 je aktivován. Jinak, pokud výstupní hodnoty na X_8 budou 1, také F_2 bude aktivován nebo žádná z chyb F_1 a F_2 je aktivována. [4]

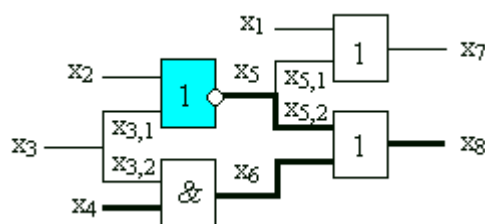
2.3.3 Guided – probe testování

Guided – probe testování rozšiřuje edge –pin testovací proces monitorováním vnitřních signálů v UUT pomocí sondy, jež se pohybuje (obvykle ovládána operátorem) v návaznosti na pokynech testovaného zařízení. Princip guided – probe testování je zpětné trasování chyb z primárního výstupu, který byl pozorován při edge – pin testování na jeho fyzickém umístění v UUT. Snímání se provádí krok za krokem. V každém kroku je vnitřní signál sledován sondou a porovnáván s očekávanou hodnotou. Další snímání závisí na výsledku předchozího kroku. [4]

Diagnostickeý strom může být vytvořen pro daný testovací model pro kontrolu procesu snímání. Strom se skládá z vnitřních uzlů (kruhů), označená vnitřní linka je sondována a koncových uzlů (obdélníků), aby ukázala možný výsledek diagnostiky. Výsledky sondáže jsou označeny jako vyhovující (P) nebo nevyhovující (F). [4]

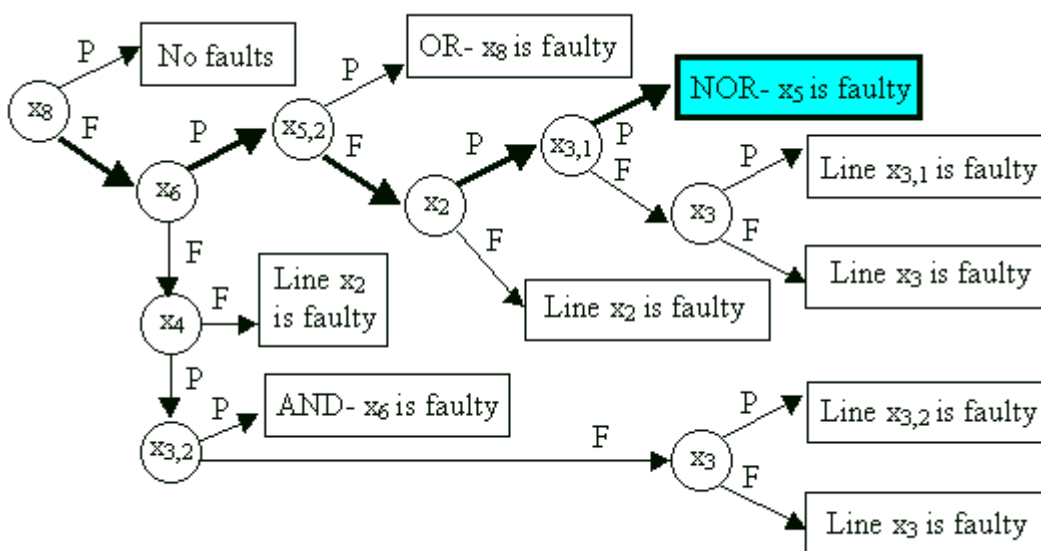
Typické umístění chyb jsou otevřené a vadné součástky. Otevření mezi dvěma body A a B ve spoji je identifikováno nesouladem mezi pozorovanými chybami v B a správné hodnotě měřené na A. Chybné zařízení je identifikováno pomocí detekce chyb na jednom z výstupů, zatímco správné hodnoty jsou měřeny na vstupech. [4]

Nejvíce časově náročné část guided – probe je pohybování sondou. Chceme-li urychlit proces umístění chyb, musíme snížit počet sondovaných linek. Je mnoho metod pro snížení počtu sondování. [4]



Obr. 6 Snížení počtu sondování [4]

Zkušební model má hodnoty 1010 přivedeny na vstup. Diagnostický strom vytvořený pro tento model ukazuje. Na výstupu X_8 místo očekávané hodnoty 0, je detekován chybný signál 1. Podle návrhu (označeného tučně) je vadným komponentem NOR na pozici X_5 . [4]



Obr. 7 Diagnostický strom [4]

Diagnostický strom umožňuje provést optimalizaci umístění postupu poruchy, například generování postupu s minimálním průměrným počtem použití sond. [4]

2.3.4 Snížení umístění chyb v UUT

Nejprve UUT je kompletní obvod a proces začíná, když její test selže. Zatímco selže UUT, může být rozdělena, polovina UUT je vypnuta a druhá polovina je testována. Pokud test projde, chyba musí být ve druhé části UUT. Pokud se tedy test nezdaří, testuje se současná část UUT. [4]

2.4 Testování elektronických prvků

Mámě několik druhů testování elektronických prvků.

- Testování před osazením
- Vnitro-obvodové testování
- Parametrické testování
- Funkční testování
- Testování pasivních prvků
- Testování Aktivních prvků
- Testování analogových prvků
- Testování číslicových prvků

K dosažení správné diagnostiky musí jednotlivé komponenty projít určitými fázemi testování. Při samotném vývoji je vyroben prototyp, u kterého ověřujeme jeho správnost dle doloženého návrhu. Dále je zde fáze ověřitelnosti, která nám pomůže ověřit vyrobiteľnost tohoto prvku. Výrobní fáze nám pomáhá ověřit spolehlivost a stabilitu výrobního procesu. Zahoření výrobku znamená první testy a odzkoušení jednotlivých parametrů. Výstupní kontrola určuje závěrečné ověření technických specifikací. Zákaznická fáze obsahuje servis, údržbu a prevenci správného chodu zařízení. A poslední fází je test životnosti. Kdy je prvek namáhán tak dlouho, dokud přestane splňovat předem určené parametry. [5]

V případě vzniku poruchy máme několik hledisek, která nám pomáhají určit možnou vzniklou poruchu. Prvním hlediskem jsou provozní podmínky, kde je přístroj umístěn a zde se zaměřujeme na příčiny vnější a vnitřní, kdy vnější příčiny představují provozní podmínky a vnitřní příčiny představují nevhodný návrh výrobku. Dále se zaměřujeme na časový interval vzniku poruchy. Zde rozlišujeme poruchy na náhlé, postupné, stálé a nestálé někdy nazývané občasně. Další kvalifikací je porucha dle stupně,

kdy rozlišujeme poruchu úplnou, která znemožní funkci celého přístroje anebo poruchu částečnou, která omezí funkce přístroje. Kombinací stupně poruchy a časového průběhu změn nám mohou vzniknout dva druhy poruch a to katastrofická porucha, která je náhlá a úplná anebo degradační porucha, která se projevuje postupnou a částečnou nefunkčností. [5]

Poruchy mohou vzniknout na různých místech v přístroji. Může se jednat o kabely, konektory, součástky či vodivé cesty plošných spojů. [5]

Mezi nejvíce poruchové části patří polovodičové součástky. Jedná se o velice náchylné součástky, které mohou mít vzniklou vadu způsobenou hned z výroby. Může se jednat o povrchové defekty, chyby montáže spojení, oxidové vrstvy, metalizaci, mechanické poškození a další poruchy. [5]

2.5 Diagnostika číslicových obvodů

Máme několik druhů modelů poruch, kdy můžeme zjistit poruchy v číslicových obvodech. [5]

Jedním z nich je zkrat, který může vzniknout mezi dvěma vodiči na téže citlivé cestě. Tím pádem vznikne zpětná vazba, ze které může dojít k přemostění lichého počtu negací, což vede k oscilaci. Ve druhém případě může dojít k přemostění sudého počtu negací a vznikne sekvenční chování. [5]

Poruchy zpoždění jsou definovány tak, že obvod může projít statickým testem na trvalé poruchy, ale selhává při provozu na systémovém kmitočtu. Příkladem může být čip, který projde testem na kmitočtu 10MHz, ale selhává při provozním kmitočtu okolo 100 MHz. [5]

Dále rozlišujeme Gate-Delay Fault což je porucha zpoždění hradla a Path-Delay Fault jedná se o poruchu zpoždění ve spoji. [5]

2.5.1 Strukturální testy

Většina poruch je vztažena k vodičům. Předpokládá se výskyt pouze jedné chyby v jeden okamžik. [5]

2.5.1.1 *Intuitivní zcitlivění cesty*

Tato diagnostika se týká výhradně jen vodičů. Jedná se o zajištění průchodu logického signálu logickými členy z primárního vstupu na primární výstup. [5]

Při provádění testu postupujeme tak, že si zvolíme typ poruchy, následně přivedeme opačnou logickou hodnotu do místa poruchy. Vytvoříme citlivou cestu z místa poruchy na primární výstup. Odvodíme ostatní proměnné na primárních vstupech a určíme všechny poruchy pokrytých daným krokem testu. [5]

V případě, že máme větší logický obvod, může porucha vzniknout v určitém uzlu, kde se cesty rozbíhají a následně se v dalších hradlech spojují. Takto vzniklou chybou je postižen zbytek obvodu. V takovém případě použijeme paritu inverze. Pokud je parita inverze lichá, obsahuje obvod redundanci. V tomto případě nelze dosáhnout úplného pokrytí testem. [5]

2.5.1.2 *Algoritmus D*

Tento druh algoritmu lze označit jako způsob pro odkrytí a vícenásobné zcitlivění cest. Při sestavování postupujeme označením signálu v místě poruchy. Zcitlivěním cest přivedeme D (\overline{D}) na některých primárních vstupech. Operátor konzistence určí příslušný vstupní vektor. Pro použití Algoritmu D máme jisté předpoklady, kdy jej můžeme použít. Jedná se o poruchy logického obvodu typu t_0 a t_1 a jsou vztaženy k vodičům obvodu. Na primární vstup lze přivést libovolný vektor. Všechny primární výstupy jsou přístupné. Předpokládáme výskyt pouze jedné chyby. [5]

Na základě popisu vstupů a výstupů logických členů se do obvodu vnutí předpokládaná porucha. Postupně vytváříme všechny možné citlivé cesty z místa poruchy na primární výstup. Bereme v potaz i šíření několika cestami současně. Vyloučíme nepoužitelné citlivé cesty. Dále hledáme odpovídající vstupní vektor pro získané citlivé cesty. Problémem u D algoritmu je, že tato metoda nebere v potaz na topologii obvodu. Dochází ke konfliktům mezi původně přidělenými a nově navrženými hodnotami proměnných. [5]

2.5.2 Funkční testy

Funkční testování má za cíl zjištění, s jakou přesností systém vykonává funkce, které se od něj očekávají. Typicky jde o reakce na uživatelské příkazy, různé manipulace s daty, vyhledávání a business procesy, uživatelské obrazovky a integraci s okolními systémy. [5]

2.5.3 Náhodné a pseudonáhodné test

Náhodné děje jsou nereprodukovatelné a vznikají bez ohledu na vůli uživatele. Můžeme si je představit jako hod kostkou, kdy předem nemůžeme uhodnout, jaká hodnota padne. [5]

Pseudonáhodný děj se zdá být náhodný, ale ve skutečnosti se jedná o deterministický algoritmus. Je představován generátorem náhodných čísel a lze jej použít při testování v reálném čase. Předem ovšem nemůžeme určit správnost odezvy. Diagnostické pokrytí nelze ověřit simulací a tak použijeme statistické metody. [5]

Statistické metody jsou pro nás vlastně předpoklady. Známe vlastnosti použitého generátoru. Díky tomu, můžeme určit pravděpodobnost výskytu jednotlivých vektorů. Určíme délku náhodného testu, která zaručí, že pravděpodobnost detekce všech poruch v obvodu bude větší než určitá mezní hodnota, kterou si zvolíme podle výtěžnosti. Oproti diagnostickému pokrytí se u náhodných testů určuje pravděpodobnost, že žádná z poruch nebude zanedbána. [5]

2.5.4 Poruchy modelování a simulace

2.5.4.1 Logické chyby modelů

Logické chyby představují vliv fyzikálních poruch na chování systému. Fyzické vady, můžeme snížit pomocí složitosti simulace. Mnoho fyzických vad může být simulováno jednou logickou chybou. Jedna logická chyba může být použita do mnoha technologií. Testy logických chyb mohou být použity pro fyzické vady, jejichž účinek nám není zcela znám. [6]

Logické chyby modelů mohou být.

- Explicitní – poruchy můžeme pojmenovat a vysvětlit
- Implicitní – poruchy jsou způsobeny charakteristickými vlastnostmi

Logické chyby mohou být.

- Strukturální – poruchy jsou spojeny se strukturálními modely, které upravují propojení mezi komponenty
- Funkční – poruchy jsou spojeny s funkčními modely, které upravují funkce komponent

2.5.4.2 Strukturální chyby

Při strukturálních chybách předpokládáme, že komponenty jsou bezchybné, ovlivněno je pouze jejich propojení. Rozlišujeme dva druhy ovlivnění. [6]

- Krátké – tvoří spojovací body, které nejsou určeny k připojení
- Otevřen – výsledky ze ztráty spojení

Strukturální poruchy modelu máme.

- Linka I je přiřazena na pevné hodnotě, v logice ($v \in \{0,1\}$), označené I/v ;
Také říkáme, linka má chybu přiřazenou na 1(sa-1) nebo přiřazenou na 0 (sa-0)

Příkladem může být:

- Zkrat mezi zemí (sa-0) nebo napájením (sa-1) a na signální lince
- Otevřená jednosměrná linka
- Jakákoliv vnitřní chyba v komponentě, která řídí svůj výkon a udržuje jej na konstantní hodnotě
- Přemostění poruch – (zkrat mezi signálními linkami) máme dva druhy, AND a OR přemostění poruch (závisí na technologii) [6]

2.5.5 Detekce poruch a redundance

Rozlišujeme tři druhy.

- Detekce poruch v kombinačních obvodech
- Zjistitelnost chyby
- Redundace

2.5.5.1 Detekce poruch v kombinačních obvodech

Nechť $Z(x)$ je logickou funkcí obvodu N , kde x představuje libovolný vstupní vektor a $Z(x)$ označuje mapování realizované N . [6]

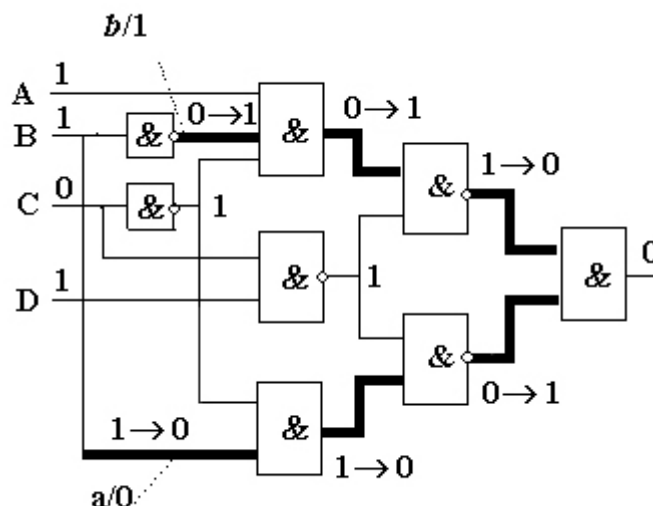
2.5.5.2 Odhalení závad

Chyba f je zjistitelná, pokud existuje test, který zjistí $t f$, jinak, f je nezjistitelná porucha. Pro nezjistitelné poruchy f platí,

$$Z(t) = Z_f(t).$$

Žádný test nemůže současně aktivovat chybu f a vytvořit citlivou cestu k primárnímu výstupu. Přítomnost nezjistitelné poruchy f může zabránit detekci jiného zavinění g , pokud existuje test, který zjistí závadu g . [6]

Na následujícím obrázku (Obr. 9) je chyba $b/1$ nezjistitelná. V předchozím případě test $t = 1101$ detekuje chybu na pozici $a/0$. Nicméně v části $b/1$ zkouška t není schopna odhalit chybu $a/0$. [6]



Obr. 9 Nezjistitelná porucha [6]

2.5.5.3 Redundance

Kombinační obvod, který obsahuje nezjistitelné chyby, se nazývá nadbytečný. Takovýto okruh můžeme zjednodušit tím, že odstraníme alespoň jedno hradlo nebo vstupní hradlo.

Předpokládejme, že na vstupu $sa-1$ hradla AND je nezjistitelná porucha. Funkce hradla se nezmění v přítomnosti poruchy. Můžeme tedy trvale nastavit hodnotu 1 na tomto vstupu. Ale na n -vstupu hradla AND s konstantní hodnotou 1 je logický ekvivalent $(n-1)$ vstupu AND získaný odstraněným vstupem hradla s konstantním signálem.

Podobně pokud vstup AND sa-0 je nezjistitelný, může být hradlo odstraněno a nahrazeno signálem 0. [6]

Zjednodušený popsání proces lze určit následujícími pravidly.

Nezjistitelná závada	Zjednodušení pravidel
AND (NAND) vstup s-a-1	Odebrat vstup
AND (NAND) vstup s-a-0	Odstranit hradlo, nahradí 0 (1)
OR (NOR) vstup s-a-0	Odebrat vstup
OR (NOR) vstup s-a-1	Odstranit hradlo, nahradí 1 (0)

Tab. 5 Přehled pravidel zjednodušení [6]

Kombinační obvod, ve kterém jsou všechny poruchy detekovatelné, se nazývá neredundantní. [6]

Redundance může být zavedena v následujících případech.

- Při navrhování chybové toleranci nebo samo testování obvodů
- V navrhování obvodů, aby se zabránilo nebezpečí

2.5.6 Poruchy rovnocennosti a lokalizace poruchy

Máme tři druhy

- Porucha třídy ekvivalence
- Ekvivalence poruch zhroucení
- Lokalizace poruchy

2.5.6.1 Porucha třídy ekvivalence

Abychom řekli, že dvě chyby f a g jsou funkčně ekvivalentní, musí platit.

$$Z_f(x) = Z_g(x)$$

Testu T , říkáme rozlišovací, mezi dvěma poruchami f a g jestliže,

$$Z_f(t) \neq Z_g(t)$$

Takové chyby se liší. Neexistuje žádný test, který může rozlišovat mezi dvěma funkčně ekvivalentními chybami. Vztah mezi částmi funkční ekvivalence a všemi

možnými poruchami se nazývá funkcí ekvivalence tříd. Pro analýzu chyb je dostačující pouze jednoho zástupce chyby v každé ekvivalenci tříd. [6]

2.5.6.2 Ekvivalence poruchy zhroucení

Na jakémkoliv n -vstupu hradla můžeme přiřadit $2(n+1)$ jeden prvek chyb. Pro hradlo NAND jsou všechny vstupy $sa-0$ chybové a výstupy $sa-1$ jsou funkčně ekvivalentní. Obecně platí, že hradlo s kontrolní hodnotou c a inverzí i má všechny vstupy $sa-c$ chybové a výstupy $sa-(c \oplus i)$ jsou funkčně ekvivalentní. Proto je pro n -vstupů hradla ($n > 1$) musí ovšem vzít v potaz pouze $n+2$ jednoho druhu chyby. Tento druh redukce poruch souboru má být analyzován na základě rovnocennosti vztahu a nazývá se ekvivalence poruchy zhroucení. [6]

2.5.6.3 Lokalizace poruchy

Pokud přidáme detekci poruch, testováním získáme lokalizaci poruch a použitím testu detekujeme nejen zjistitelné vady, ale rozlišuje druhy vad. Kompletní lokalizační test rozlišuje mezi každým párem rozeznatelných poruch v obvodu. Přítomnost nezjistitelné poruchy může způsobit zrušení kompletního lokalizačního testu. Pokud f a g jsou dvě rozlišné chyby, mohou se stát funkčně ekvivalentní v přítomnosti nezjistitelné poruchy. Kompletní test může určit místo poruchy v rámci funkční ekvivalenční třídy. Jedná se o maximální diagnostické rozlišení, které lze dosáhnout. [6]

Dvě chyby f a g jsou funkčně ekvivalentní pod testem T jestliže $Z_f(t) = Z_g(t)$ pro každý testovaný vektor platí $t \in T$. [6]

Funkční ekvivalence znamená rovnocennost podle jakékoliv zkoušky, ale rovnocennost v rámci daného testu neznamena funkční ekvivalenci. [6]

2.5.7 Dominantní chyba

Pokud je objektem testování pouze krátký test na zjišťování poruch, pak se kromě poruchy ekvivalence, dalších poruch může dojít ke snížení počtu chyb, které musí být zváženy. [6]

Nechť T_g je množina všech testovacích vektorů, které detekují poruchy g . Chyba f dominuje nad chybou g , jestliže f a g jsou funkčně ekvivalentní pod T_g . Pokud f dominuje g , potom nějaký test, který zjistí t , detekuje také g a bude také detekovat f . Proto je pro

detekci chyby zbytečné zkoumat dominující poruchy f , neboť odvozením testu g jsme získali automatický test, který odhalí f . [6]

2.5.8 Single stuck-fault model

Single stuck-fault model (SSF) je klasický nebo standardní model chyby. Jeho užitečnost lze popsat následujícími body.

- Představuje mnoho různých fyzikálních poruch
- Je závislý na technologiích
- Zkušenosti ukázaly, že test, který odhalil SSF, odhalil také mnoho neklasických chyb
- V porovnání s ostatními modely poruch je počet SSF v obvodu malý
- SSF mohou být použity k jiným typům modelů poruch

2.5.9 Multiple stuck-fault model

- Počet opakujících se chyb
- Chyba maskování
- Kruhové poruchy maskování

2.5.9.1 Počet opakujících se chyb

Multiple stuck-fault model (MSF) je přímo rozšířen SSF, ve kterém může být několik řádku spojeno. Jestliže je n počet možných lokalit SSF, je tam $2n$ možných SSF, ale jsou tam $3^n - 1$ možné MSF. Budeme-li předpokládat, že množství chyb není větší než k , pak počet možných MSF je. [6]

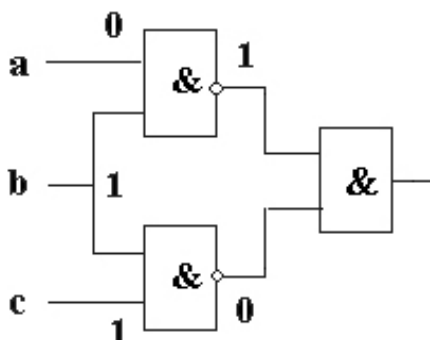
$$\sum_{i=1}^k \{C_i^n\} 2^i$$

Počet opakujících se chyb je velmi velký. Nicméně je potřeba jejich možné zavinění maskovat. [6]

2.5.9.2 Chyba maskování

Nechť T_g je test, který zjistí chybu g . Říkáme, že chyba f je funkčně maskovaná chyba g právě, když více chyb $\{f, g\}$ není detekována žádným testem v T_g . [6]

Následující obrázek (*Obr. 10*) představí test 011. Je to test, který pouze odhalí chybu na pozici $c/0$. Tento samý test nemůže detekovat více chyb $\{c/0, a/1\}$. Tedy $a/1$ je maska $c/0$. [6]



Obr. 10 Porucha maskování [6]

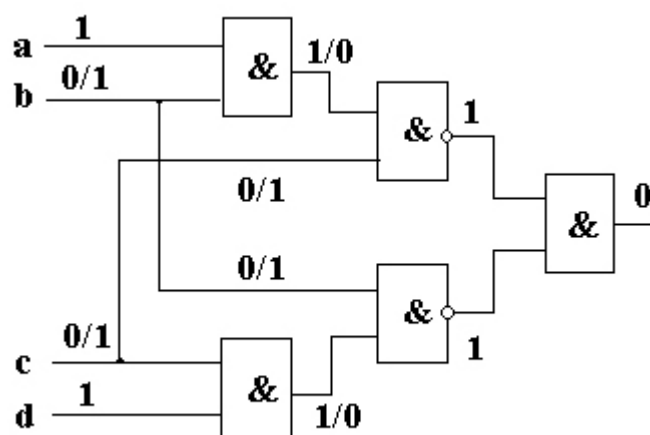
Nechť $T_g' \subseteq T$ je soubor všech testů v T , které detekují poruchy v g . Chyba f masky chyby g podle testu T právě tehdy, když více chyb $\{f, g\}$ nejsou detekovány žádným testem v T_g' . [6]

Funkční maskování znamená maskování ve všech zkušebních testech, ale výsledné prohlášení není vždy pravdivé. [6]

2.5.9.3 Kruhové poruchy maskování

Více poruch F nemusí být detekování kompletním testem T jako jednotlivé chyby, protože se jedná o kruhové maskování vztahů v rámci T mezi složkami z F . [6]

Mějme test $T = \{1111, 0111, 1110, 1001, 1010, 0101\}$ detekuje každou chybu SSF v obvodu na obrázku (*Obr. 11*). Nechť f je $b/1$ a g je $c/1$. Pouze test v T detekuje jedinou závadu f a g je 1001. Avšak více chyb $\{f, g\}$ není rozpoznáno, protože v rámci zkušebního vektoru 1001, f maskuje g a g maskuje f . [6]



Obr. 11 Kruhové poruchy maskování [6]

3 TEST GENERÁTOR

Test generátor (TG) je komplexní problém se mnoha vzájemně se ovlivňujícími aspekty, např. cena TG (složitost metody, zkušební doba) a kvalita vytvořených testů (chyba pokrytí). Generování testů může být vytvořeno na mnoha různých úrovních, např. gate-level, macro-level a functiona-level. Existují také různé metody používané při zkoušce generace, např. deterministické metody (fault-oriented TG, fault independent TG), random TG, kombinované deterministické/náhodné TG. [7]

3.1 Gate-level fault-oriented test generátor

Složitost metod a algoritmů TG závisí na struktuře gate-level obvodů. Zvláště následují struktury:

- Fanout-free obvody
- Kombinační obvody s fanout
- Sekvenční obvody

3.1.1 Fanout-free obvody

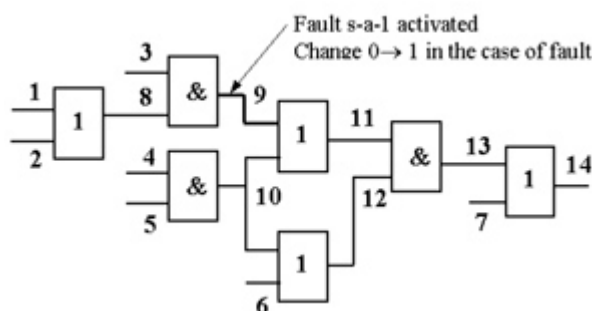
Tři základní kroky jsou používány při generování testu pro chyby-SAV:

- Pro aktivaci poruchy
- Propagovat výsledné chyby tp primárního výstupu (PO)
- Odůvodnění přidělené hodnotami

3.1.1.1 Porucha aktivace

Chcete-li aktivovat poruchy na lince 1 v obvodu, znamená to přiřadit hodnotu lince 1 (některé doplňkové hodnoty mohou být na dalších řádcích), takže hodnota na řádku 1 se změní na opačnou v případě přítomnosti poruchy. [7]

Chcete-li aktivovat chybu přiřazenou na 1 (sa-1) na řádku 9 v obvodu musíme přiřadit k řádku 9 hodnotu 0. V případě poruchy na sa-1 se hodnota 0 na tomto řádku změní na 1. [7]

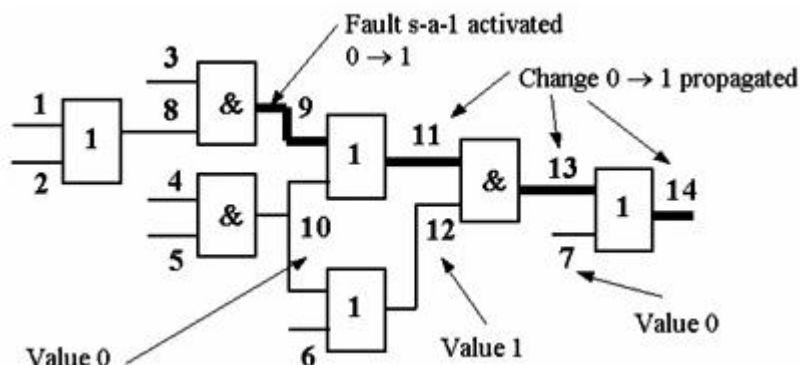


Obr. 12 Chyba aktivace [7]

Na druhou stranu, k přemostění AND typ přemostění poruch b (9, 10) mezi řádky 9 a 10, musíme přiřadit hodnotu 1 na řádek 9 a hodnotu 0 na řádek 10 (nebo jako druhou možnost můžeme hodnoty obrátit 0 na řádek 9 a 1 na řádek 10). V případě této chyby, hodnota 1 na řádku 9 se změní na 0 (nebo hodnota 1 na řádku 10 se změní na 0 v případě druhé možnosti). [7]

3.1.1.2 Porucha šíření

Šířit chyby signálu přidělené na 1 (sa-1) na řádku 9 až do hradla OR 11 musíme přidělit hodnotu 0 na řádku 10. V případě poruchy sa-1 na řádku 9 hodnota 0 se na řádku 11 změní na 1. [7]

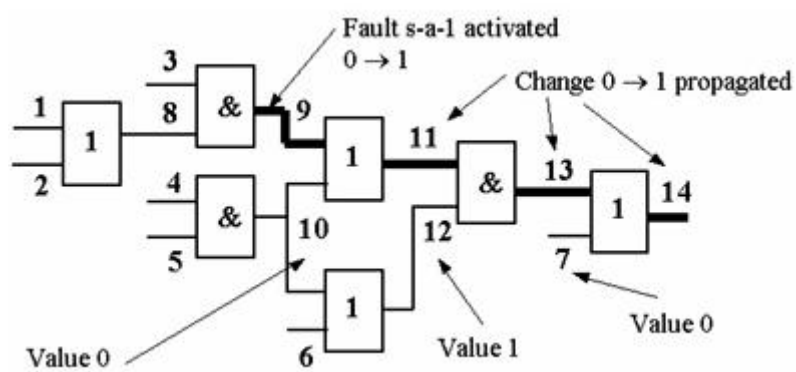


Obr. 13 Porucha množení [7]

Dále se pak chyby signálu rozšiřují z řádku 11 na primární výstup 14, musíme tedy přiřadit hodnoty 1 a 0 na řádek 12 a 7 odpovídajícím způsobem. V případě poruchy sa-1 na řádku 9, hodnota 0 se na řádku 14 změní na 1. [7]

3.1.1.3 Odůvodnění řádku

Abychom odůvodnili hodnotu 0 na řádku 9, musíme přiřadit hodnotu 0 buď na řádek 3 nebo na řádek 8 anebo na oba řádky. [7]



Obr. 14 Odůvodnění řádku [7]

II. PRAKTICKÁ ČÁST

4 PROGRAM PRO VÝUKU ZÁKLADŮ DIAGNOSTIKY

Pro práci na seznámení se s problematikou test generation jsem si zvolil program Diagnostic tester vytvořený v java Applet. Je jednoduchý, dostupný pro všechny a k jeho provozuschopnosti stačí jen připojení k internetu.

4.1 Základní popis

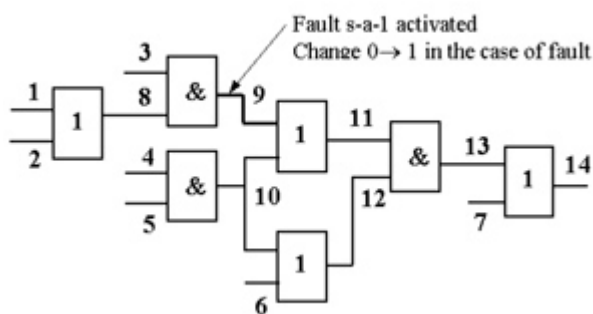
Diagnostic tester podporuje učení základů digitálních testů přes internet. Nabízí sadu nástrojů pro pochopení principů test generation, poruchy simulace, poruchy diagnostiky a závady v digitálních obvodech. Obsahuje velké množství jednoduchých kombinačních obvodů, které jsou navrženy na zkoušení na obrazovce v interaktivním režimu na pochopení důležitých technik a algoritmů. Tento program umožňuje snadné akce a reakce (klikni a sleduj) s možností vzdáleného učení.

4.2 Zkrácená teorie

4.2.1 Test generátor

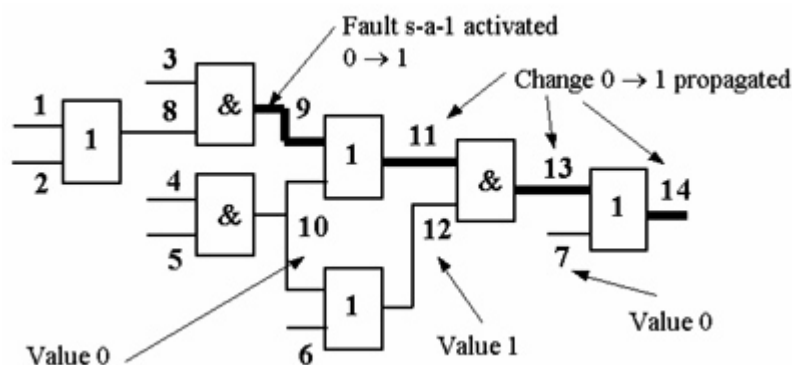
Test generátor (TG) je komplexní problém se mnoha vzájemně se ovlivňujícími aspekty, např. cena TG (složitost metody, zkušební doba) a kvalita vytvořených testů (chyba pokrytí). Existují různé metody používané při zkoušce generování: deterministické metody (fault-oriented TG, fault-independent TG), náhodné TG, kombinované deterministické/náhodné TG a jiné. Složitost metod a algoritmů TG závisí na struktuře gate-level obvodů. Používají se hlavní tři kroky při generování testu poruch. [8]

- Pro aktivaci poruchy – Chcete-li aktivovat chybu přidělenou v sa-1 na řádku 9, musíme přiřadit k řádku 9 hodnotu 0. V případě poruchy sa-1 na hodnotě 0 na tomto řádku se změní na 1.



Obr. 15 Aktivace poruchy [8]

- Šířit výsledné chyby na primární výstup (PO) – Šířit chyby signálu přiřazené v sa-1 na řádku 9 skrze hradlo OR 11 musíme přiřadit hodnotu 0 na řádek 10. V případě poruchy sa-1 na řádku 9 hodnoty 0 se na řádku 11 změní na 1.
- Odůvodnit přidělené hodnoty – Odůvodnit hodnotu 0 na řádku 9, musíme přiřadit 0 buď na řádek 3 nebo na řádek 8 nebo na obě linky.



Obr. 16 Odůvodnění přidělených hodnot [8]

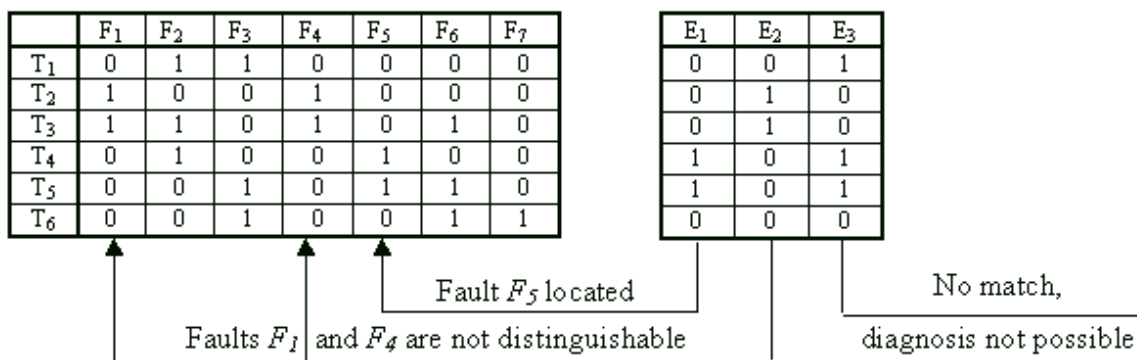
4.2.2 Diagnostika poruch

Testované jednotky (UUT) selžou, pokud jejich výsledná reakce je odlišná od reakcí předpokládaných. Diagnostika se skládá z umístění fyzické poruchy ve strukturální části modelu zkoušené jednotky. Diagnostický proces je často hierarchický, prováděný jako top-down proces (s operačním systémem) nebo bottom-up proces (během výrobního procesu). [8]

4.2.3 Metoda kombinační diagnostiky poruch

Tento postup tvoří většinu práce před testovací zkouškou. Využívá chyby simulace k určení možných odpovědí na daný test v případě výskytu chyby. Databáze sestavené

v tomto kroku se nazývá tabulka chyb nebo porucha adresáře. Chceme-li určit chybu, snažíme se skutečný výsledek testu porovnat s očekávaným výsledkem uloženým v databázi. Výsledek testu představuje kombinaci možných poruch na zkušebním modelu. [8]



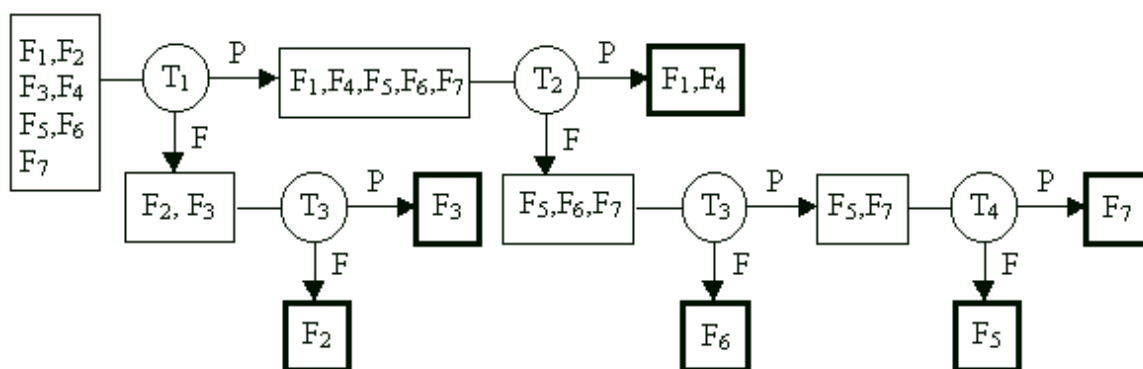
Obr. 17 Tabulka chyb [8]

4.2.4 Metody sekvenčních diagnostických poruch

V sekvenčním diagnostickém procesu se lokalizace poruchy provádí krok po kroku, kde každý krok závisí na výsledku diagnostického testu v předcházejícím kroku. Takový experiment se nazývá adaptivní testování. Sekvenční experimenty mohou být prováděny buď tím, že se sleduje pouze výsledek na výstupu zkoušené jednotky nebo sledujeme pomocí speciální sondy i vnitřní kontrolní body zkoušené jednotky (řízené snímání). Sekvenční diagnostický postup může být graficky znázorněn jako diagnostický strom. [8]

4.2.5 Příklad lokalizace poruch na edge-pin testování

Diagnostický strom na obrázku odpovídá příkladu tabulce chyb. Můžeme vidět, že většinu poruch lze jednoznačně identifikovat, dvě chyby F_1 , F_4 zůstávají neidentifikovány. Ne všechny testovací vzorce použité v tabulce chyb jsou potřebné. Různé závady potřebují pro identifikaci různě dlouhé testovací sekvence. Nejkratší test obsahuje dva modely s nejdelšími čtyřmi vzory. [8]



Obr. 18 Model edge-pin testování [8]

4.2.6 Guided-probe testování

Guided-probe testování rozšiřuje edge-pin testovací proces monitorováním vnitřních signálů v UUT pomocí sondy, jež se pohybuje (obvykle operátorem) v návaznosti na pokynech zkušebního zařízení. Princip guided-probe testování je zpětné hledání chyby z primárního výstupu, který byl sledován během edge-pin testování na jeho fyzickém umístění v UUT. Snímání se provádí krok po kroku. V každém kroku je vnitřní signál sondován a porovnáván s očekávanou hodnotou. Další snímání závisí na výsledku předchozího kroku. [8]

4.3 Příručka k programu

4.3.1 Pracovní okno

Pracovní okno se skládá ze tří částí

- Panel vložení vektoru
- Zobrazovací panel pro zvolené schéma
- Zobrazovací panel pro zobrazení dat a průběhů

4.3.2 Panel vložení vektoru

Tento panel má dva panely.

- Panel „I“ pro vkládání jednotlivých vstupních vektorů
- Panel „S“ k nastavení zpětné vazby konfigurace ze zpětné vazby lineárního posuvného registru (LFSR), které mají být použity pro automatické generování testovacích vektorů

V LFSR režimu se první panel používá pro inicializaci LFSR. LFSR obsahuje základní automatický model zkušebního generátoru (ATPG), používá se pro emulaci různých BIST návrhů. Lze emulovat následující architektury:

- Buildt-In-Logick-Observer (BILBO)
- Circular-Self-Test-Path (CSTP)

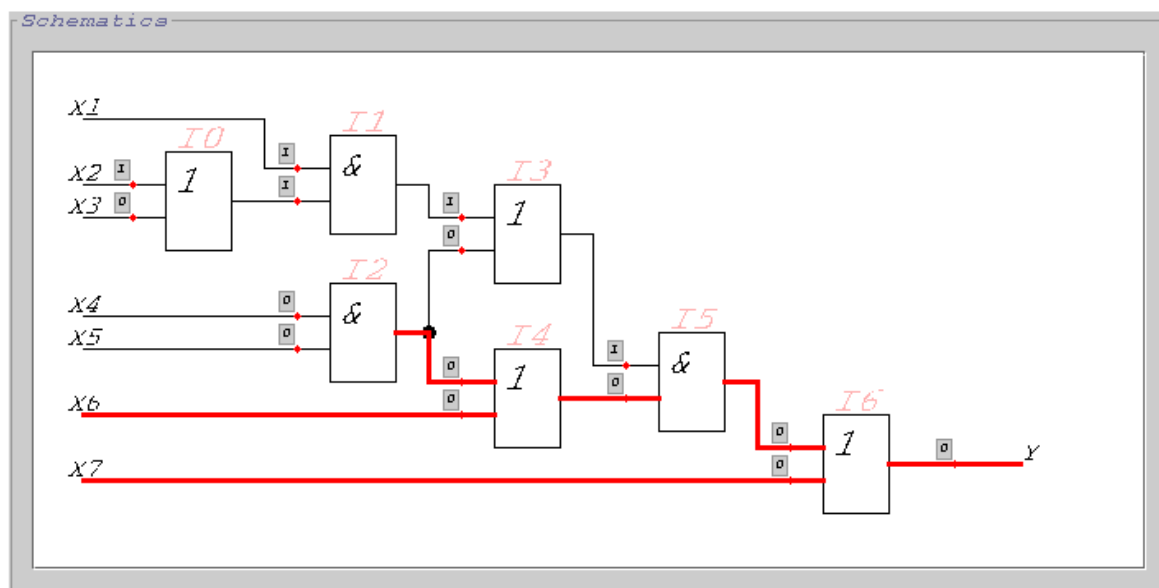
První panel je také používán při vytváření testovacích vektorů pro speciální detekce poruch. V tomto případě jsou hodnoty vloženy jedna po druhé do signálových částí na připojení v návrhu schématu a vstupní vektor bude odvozen od těchto vnitřních hodnot signálů. [9]

Obr. 19 Panel vložení vektoru [9]

4.3.3 Panel pro schéma

Panel pro schéma zobrazuje aktuálně vybraný obvod. Malé kolonky na linkách zobrazují vnitřní hodnoty signálu na připojení. Kolonky jsou manuálně měnitelné pomocí klikání. V módu generace lze potřebnou hodnotu signálu pro aktivaci poruchy nebo poruchy šíření vložit přímo pomocí připojení. V režimu diagnostických poruch, klikáním

na kolonky můžeme zkoumat postup simulování. Kliknutí na kolonku ukáže výsledek měření „reálného“ signálu na odpovídající vadný obvodu. [9]



Obr. 20 Panel pro schéma [9]

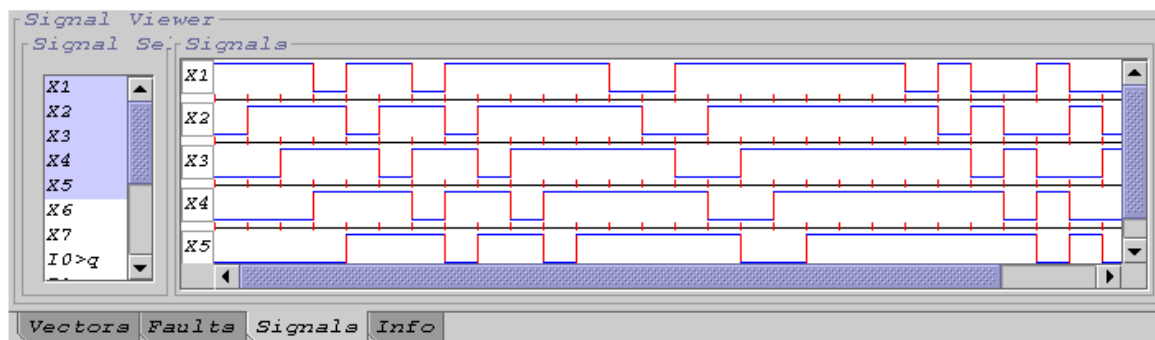
4.3.4 Schéma zobrazení průběhu dat

Zjištěné závady, signálové závady a podobně jsou zobrazeny jako barevné tučné čáry. Barevné kódy jsou následující.

- Červená – sa-1 je zjištěna chyba
- Zelená – sa-0 je zjištěna chyba
- Šedá – nedefinovaný signál
- Modrá – protichůdný signál

4.3.5 Panel dat

Panel dat (Obr. 21) zobrazuje informace o simulovaných testovacích vektorech a zjištěných závad. V režimu simulace poruchy můžete klepnout na řádek daného vektoru a zobrazí se, které chyby jsou detekovány tímto vektorem. V signálu (průběh) režimu můžete volit všechny zajímavé signály a vynechat ty, které zajímavé nejsou.[9]



Obr. 21 Zobrazení průběhu dat [9]

4.3.6 Menu

Menu obsahuje čtyři části

- Schéma
- Režim
- Jazyk
- Pomoc

Část schéma obsahuje seznam předdefinovaných obvodů. Prostřednictvím části jazyk si uživatel vybere jeden z podporovaných jazyků. Menu nápovědy poskytuje užitečné tipy a vysvětlivky. Díky volbě v menu lze určit co se má udělat – test vektor vložení, manuální test vektor generace, poruchy simulace nebo diagnostika (dva možné postupy: sekvenční a kombinační diagnostika). [9]

4.3.7 První spuštění

Při začátku práce s Diagnostic testerem vybereme v menu schematický obvod z předdefinovaných obvodů. Pak můžeme začít provádět různé experimenty s tímto obvodem pomocí výběru správně zvoleného režimu z menu. [9]

4.3.8 Vkládání test vektoru

V režimu vložení vektoru, vytvoříme testovací vektor buď automaticky pomocí LFSR, nebo vložení vektoru ručně. V manuálním režimu se generuje krok za krokem vstupní vzor, který je současně simulován. Kolonky na linkách ve schématu panelu zobrazují

simulovaný výsledek – hodnoty vnitřních signálů na připojení. Průběhy je možné sledovat v panelu dat. [9]

Při použití LFSR musíme zadat počáteční stav (v panelu označen „I“), nastavit zpětnou vazbu struktury (v panelu označen „S“) a určit délku zkušební posloupnosti. Podle LFSR můžeme simulovat BIST koncept ve dvou směrech. Built-In-Logic-Block Observer (BILBO) a Circular-Self-Test-Path (CSTP). [9]

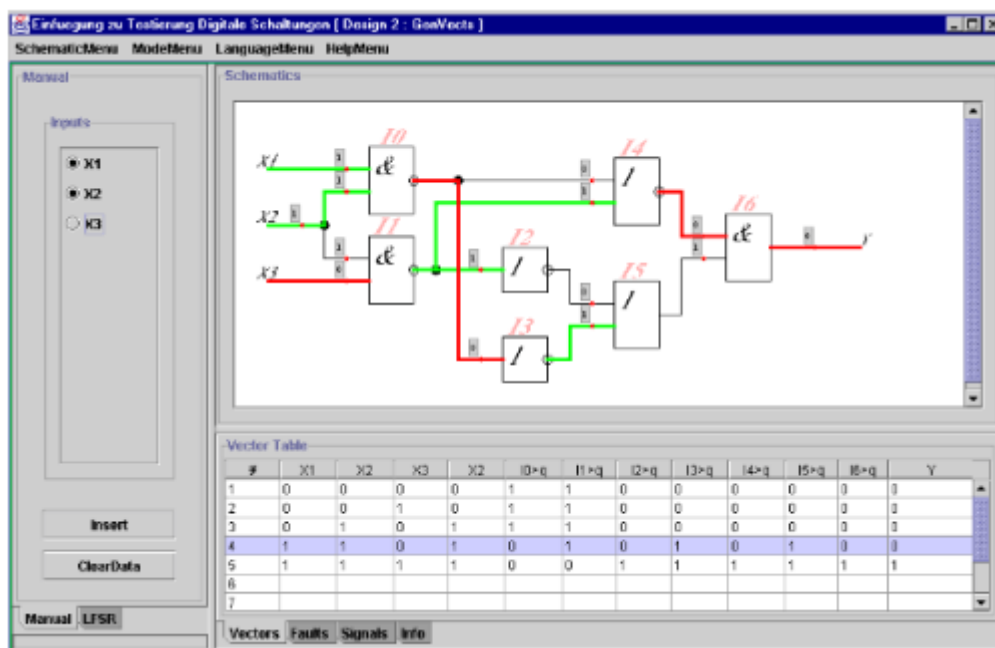
V režimu BILBO pseudo-random test sekvence je generován samostatně LFSR na základě nastavení na sub-panelu I a S. [9]

V režimu CSTP pseudo-random test sekvence je generována oběma LSFR a okruhy vybranými v menu schématu. Změnou nastavení na panelu S můžeme napodobit různé reakce struktur vybraných BIST architekturou. [9]

4.3.9 Test generátor

V režimu test generátor, si vybereme cíl chyby ve schématu a vytvoříme krok po kroku vlastní aktivní části v obvodu pro vyjádření chyb v této části a pro šíření klamných signálů na výstup, klikáním na potřebné hodnoty jednotlivých kolonek na linkách. Z těchto konečných hodnot můžeme odvodit vstupní vektor. Barvy na linkách nám pomáhají pochopit aktuální stavy. [9]

- Aktivní vady a aktivní cesty jsou označeny červenou a zelenou linkou
- Nesrovnalosti hodnot signálu jsou zvýrazněny modrou barvou



Obr. 22 Ukázka schématu v Diagnostic tester [9]

4.3.10 Chyba simulace

V simulačním režimu poruch, tabulka chyb je vyobrazena v data panelu pro všechny vektory vytvořené v daném okamžiku. Výběrem vektorových dat na panelu jsou všechny zjištěné závady v tomto vektoru zdůrazněny barvami na panelu se schématem. [9]

4.3.11 Diagnostika

V režimu diagnostiky poruch musíme nejprve vytvořit tabulku chyb spuštěním simulátoru chyb pro již dříve vytvořený testovací vektor. Data vstupují do režimu diagnostiky poruch vložení náhodné poruchy do obvodu. Lze zkoumat dvě diagnostické strategie kombinační a sekvenční diagnostiku. [9]

Po určení kombinační strategie jeden vektor nebo podmnožina vektorů mohou být vybrány a použity v obvodu chyb (napodobovat experimentální test). Diagnostic tester ukazuje výsledky experimentu v posledním sloupci tabulky (Y - znamená, že zkušební vektor je správný, N – znamená, že zkušební vektor je chybný). Porucha v tabulce zobrazuje také podmnožinu podezření závad. Pro zlepšení diagnostického rozlišení lze použít další testovací vektor a opakovaně tak provádět experimenty. [9]

Sekvenční diagnostika je založena na řízeném snímání. Testovací model se spustí a jeho chování se zobrazuje. Kliknutím na příslušné kolonky provádíme sondovací test a jsou měřeny skutečné hodnoty v obvodu. Chybné signály jsou zvýrazněny červenou čarou,

správné signály jsou zobrazeny šedou barvou. Nalezení chyby je buď na hradle s vadnými vstupy. Pokud jsou správné vstupní hodnoty (chyba je v hradle), nebo mohou být vadné vstupy (jsou vadné vstupy). [9]

Hlavním bodem v určení strategie diagnostiky je pokusit se lokalizovat závadu snadným testovacím vektorem, nebo provádět kontrolní měření řady signálů. [9]

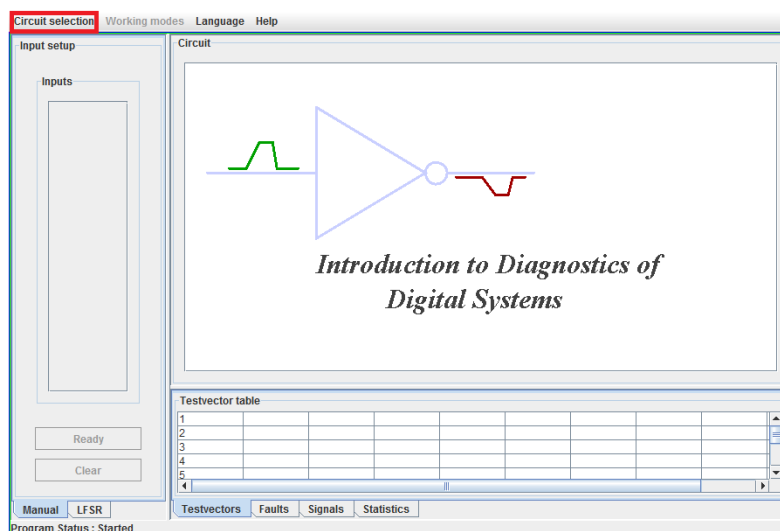
5 CVIČENÍ

Veškeré cvičení se provádí v diagnostickém programu Diagnostic tester. Postupně se budou volit jednotlivé obvody a na nich vyhledávat závady. Většina obvodů má 5 vstupů a 2 výstupy, počet hradel se liší v rozsahu 8-10. V následujícím popisu bude rozebráno jak správně testovat a provádět jednotlivé kroky abychom dosáhli zdárného konce.

5.1 Postup při provádění diagnostiky

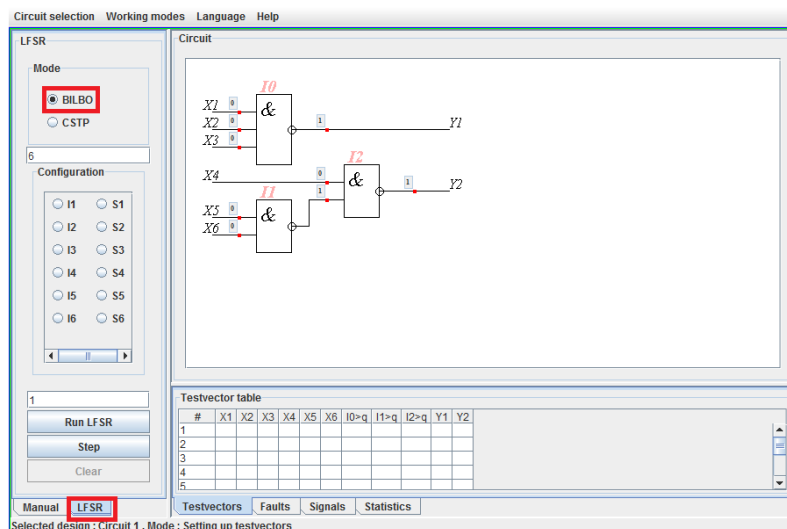
5.1.1 Pseudonáhodně generovaný test pomocí LFSR

- Spustíme si Diagnostic tester a v menu vybereme obvod, na kterém budeme provádět diagnostické testy. [10]



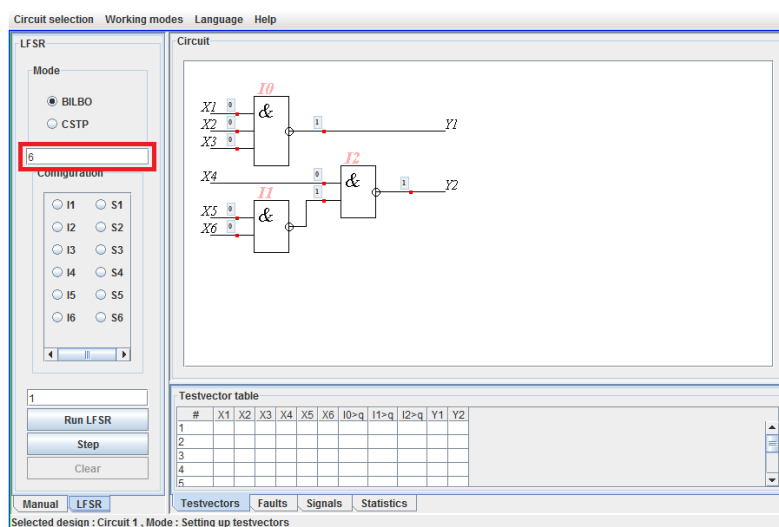
Obr. 23 Výběr obvodu

- Poté si na ovládacím panelu zvolíme LSFR BILBO(Build – In –Logic – Block- Observer) režim. [10]



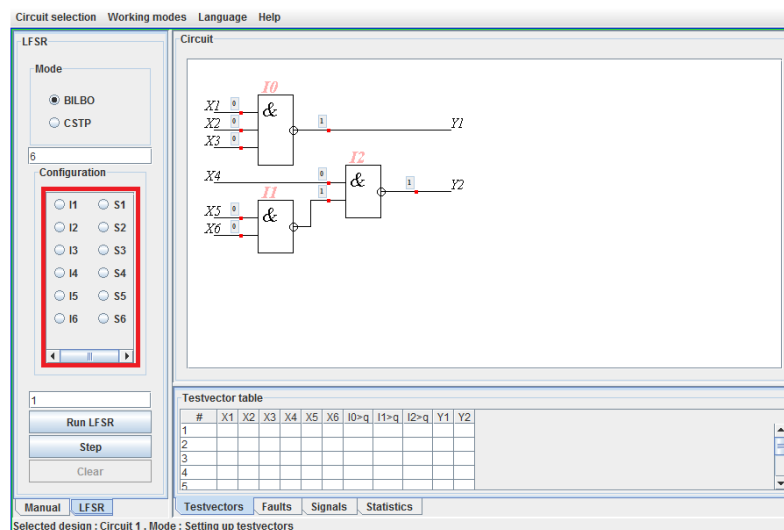
Obr. 24 Výběr LSFR BILBO

- Zvolte délku LSFR (bitů), tuto hodnotu je lepší ponechat přednastavenou v Diagnostic testeru, protože odpovídá počtu vstupů daného obvodu. [10]



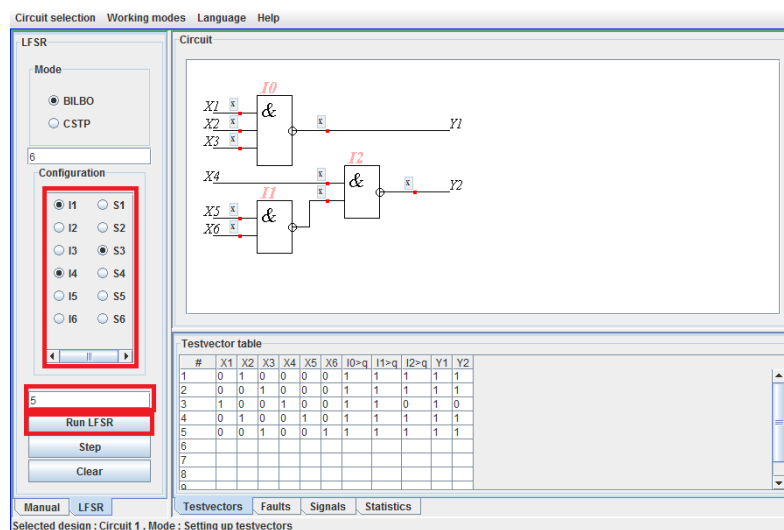
Obr. 25 Volba vstupů

- Další volbou je výběr konfigurace LSFR – zadání počtu vstupních stavů „I“ a zpětné vazby „S“. [10]



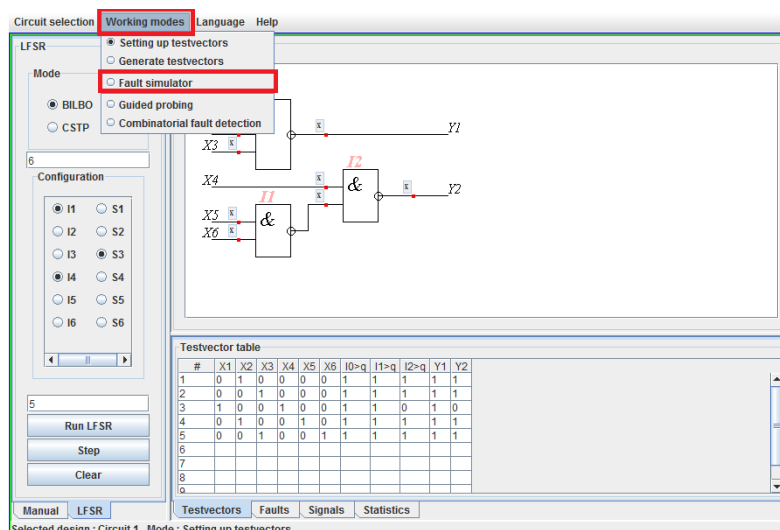
Obr. 26 Volba počátečních stavů

- Je třeba vložit testovací vektory pro vytvoření generace, vhodné je volit krok s pěti testovanými vektory. Poté stisknete tlačítko Run LSFR. [10]



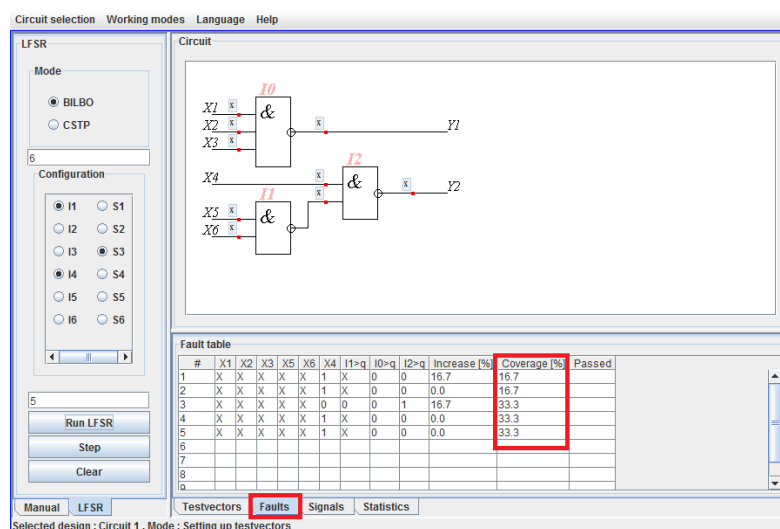
Obr. 27 Vložení vektorů a spuštění

- Poté v nabídce v části Working mode vybereme možnost Fault simulator pro vyplnění tabulky chyb. [10]

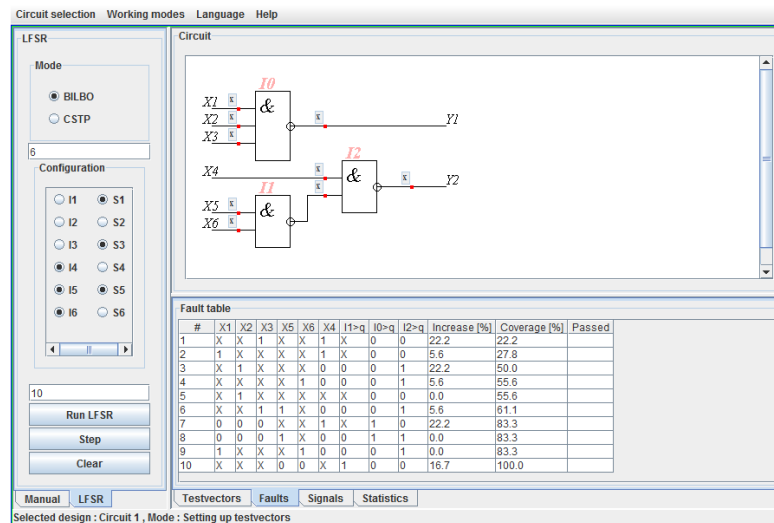


Obr. 28 Volba Fault simulator

- V tabulce vidíme výsledky a musíme dosáhnout pokrytí 100%, opakujeme tedy předešlé kroky s volbou vektoru a místo počtu 5 počet navýšíme a navýšíme i počet vstupních hodnot, dokud nedosáhneme 100% pokrytí simulovaného testu. [10]



Obr. 29 Tabulka chyb s vyobrazeným pokrytím



Obr. 30 Vytvoření 100% krytí testu

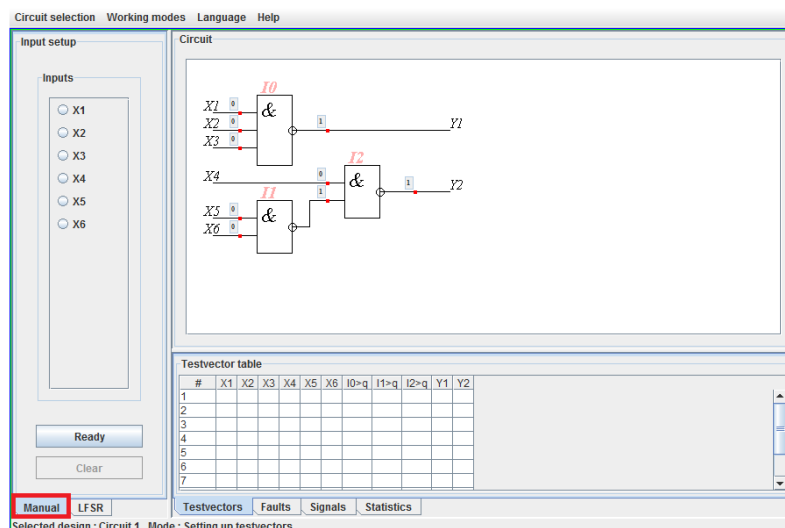
- Při provádění změn se vždy musíme přepnout zpět do režimu Setting up testvectors, který najdeme v hlavním menu v části Working modes. [10]

5.1.2 Studie tabulky poruch

- Hledejte 5 a více chyb ve snadno testovaných chybách a potom všechny těžce testované chyby. Všimněte si, že jsou některé chyby v obvodu, které můžeme testovat pouze na základě jedinečných testovacích vektorů. Tyto chyby budeme považovat za těžce testované chyby. [10]
- Zapište si vektor, který obsahuje těžce testovanou chybu.
- Pokud je to možné pokuste se najít redundantní testovaný vektor a smažte jej. Budeme uvažovat, že se jedná o nadbytečný vektor, který zvyšuje chybu pokrytí. Stejným způsobem, testovaný vektor může být redundantní, pokud ostatní testované vzory v tabulce chyb pokrytí jsou testovány právě tímto vektorem. Zrušením některých nadbytečných vektorů budeme zjednodušovat množství našich zkoušek. [10]

5.1.3 Manuální (deterministický) zkušební vzor generování

- Na ovládacím panelu zvolte možnost manual



Obr. 31 Výběr manuálního vkládání vektorů

- Můžeme provést dva způsoby, použitím testovacího vzoru přímo z panelu nebo volbou Generate testvectors v nabídce Working modes za účelem zjištění speciální poruchy, kterou chceme vložit do obvodu. Pokud si zvolíme druhou možnost, je nutné propagovat tuto poruchu na výstup a pozorovat vyprodukované hodnoty. To lze provést kliknutím na hodnoty kontrolních bodů a odhadovat hodnotu, kterou je třeba zvolit. Za účelem propagovat poruchy na výstup je třeba, aby bylo odhaleno hradlo, které způsobuje chybu. Zatímco se šíří konkrétní chyba, zkuste testem odhalit co nejvíce závad, mohou být použity stejné zkušební vektory. Pokračujte v generování testu vektorů, dokud nedosáhnete žádoucích 100% pokrytí poruchami. [10]
- Po vložení vektoru, který vám pokryje 100% poruchy, vyzkoušejte možnost zrušení některých z nich. Pravděpodobně neztratíte chybu pokrytím. Pokuste se sestavit co nejkratší kompletní test (100% pokrytí chybami). Poté zhodnoťte náklady (kvalitu) vámi vytvořeného testu. [10]

5.1.4 Guided probing diagnosis

- Po simulaci testovacích vektorů z předešlých kroků, vyberte Guided probing z menu Working modes a přepněte se do tohoto režimu. Nyní můžete vkládat náhodné poruchy do obvodu. Vyberte vektor označený jako 'N' a klikněte na testovací bod pro měření hodnoty signálu. Pamatuje si, že princip guided-probing testování je zpětné vyhledání chyb z primárních

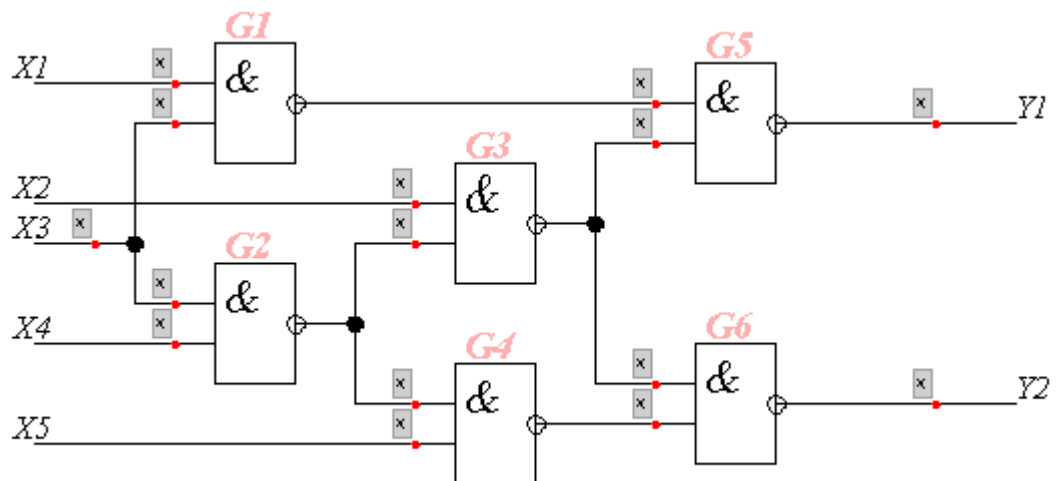
výstupů, kde byly chyby nalezeny a vedou k jejich zdroji. Jedná se o nejvíce časově náročnou část z guided-probe testování, jde o pohyb sondy a měli byste se pokusit snížit počet sondovaných linek. Snažte se o co nejmenší počet kliků. Formulovat některá pravidla a metody pro minimalizaci počtu sond (nejsou uvedeny v příkladu). Sestrojte diagnostický strom pro konkrétní vektor. Počet kliknutí se zaznamenává na dolní liště Diagnostic testeru pod tabulkami. [10]

5.1.5 Kombinační diagnostika (fault table)

- Vyberte v menu Combinational fault detection z Working modes. Opět, stejně jako v předchozím kroku, měli byste si vložit náhodně chyby do obvodu. [10]
- Kliknutím na vektory, vyberte (držením tlačítka Ctrl můžete označit více vektorů najednou) menší počet testovacích vektorů, vhodně dle uvážení, aby bylo možno najít danou chybu. [10]
- Pokud nemůžete lokalizovat poruchu i přesto že byly vybrány všechny vektory, sestrojte jeden nebo více dalších diagnostických testovacích vektorů. [10]

5.1.6 Příklad

- V následujícím příkladu si předvedeme jednotlivé popsané kroky a jak se jimi řídit. Náš obvod má 5 vstupů 2 výstupy a 6 hradel. V menu je pod označením ISCAS C17. [10]



Obr. 32 Příklad testovaného obvodu [10]

5.1.7 Pseudonáhodné generování testu pomocí LFSR

- Pokud vybereme všechna potřebná nastavení, pokračujeme dalším krokem
- Prvních 5 vygenerovaných vektorů LFSR dalo chybu 66,7% což není špatné, protože jsme se zabývali více než polovinou všech možných závad, pouze o pěti testovaných vektorech. Ale naším cílem je dosáhnout 100% a proto nám dalších 5 testovacích vektorů zvýší pravděpodobnost dosažení 100%. Po simulaci 10 testovacích vektorů, máme štěstí, protože jsme dosáhli 100% pokrytí poruch. Nyní můžeme jít na další krok. Zkušební vektory tabulky a vyprodukované chyby jsou uvedeny níže. [10]

Testvector table														
#	x1	x2	x3	x4	x5	x3	g1>out	g2>out	g3>out	g4>out	g5>out	g6>out	y2	y1
1	0	1	0	0	0	0	1	1	0	1	1	1	1	1
2	0	0	1	0	0	1	1	1	1	1	0	0	0	0
3	0	0	0	1	0	0	1	1	1	1	0	0	0	0
4	1	0	0	0	1	0	1	1	1	0	0	1	1	0
5	0	1	0	0	0	0	1	1	0	1	1	1	1	1
6	1	1	1	0	1	1	0	1	0	0	1	1	1	1
7	1	1	1	1	0	1	0	0	1	1	1	0	0	1
8	0	1	1	1	1	1	1	0	1	1	0	0	0	0
9	1	0	1	1	1	1	0	0	1	1	1	0	0	1
10	1	1	0	1	1	0	1	1	0	0	1	1	1	1

Obr. 33 Tabulka testovaných vektorů [10]

Fault table																		
#	x3	g1>inp2	x1	x4	g2>inp1	g3>inp2	x2	x5	g4>inp1	g5>inp2	g1>out	g4>out	g6>inp1	g6>out	g5>out	Increase [%]	Coverage	
1	X	X		X	X	X	0	0	X	X	1	X	X	1	0	0	20.0	20.0
2	X	X		1	X	X	X	1	1	X	0	0	0	0	1	1	30.0	50.0
3	X	X		X	X	X	X	1	1	X	0	0	0	0	1	1	0.0	50.0
4	1	1		X	X	X	X	1	0	0	0	0	1	X	0	1	16.7	66.7
5	X	X		X	X	X	0	0	X	X	1	X	X	1	0	0	0.0	66.7
6	X	X		X	1	X	X	X	X	X	X	X	X	X	0	0	3.3	70.0
7	0	0		0	0	0	1	X	X	X	X	1	0	0	1	0	23.3	93.3
8	0	X		1	0	0	1	X	X	1	0	0	0	0	1	1	3.3	96.7
9	0	0		0	0	0	X	X	X	1	X	1	0	0	1	0	0.0	96.7
10	1	X		X	X	1	0	0	X	X	1	X	X	X	0	0	3.3	100.0

Obr. 34 Tabulka chyb [10]

5.1.8 Studie tabulky chyb

Nyní najdeme nějaké snadné chyby v testu a nějaké těžce testované chyby. Pro první můžeme předvést například: g5>out (sa-0), g6>out (sa-0 a sa-1), g6>inp1 (sa-0), g4>out (sa-0). Domníváme se, že snadně testované chyby jako chyby můžou být testovány maximálním počtem zkušebních vzorů v porovnání s počtem vektorů, které mohou pokrýt jiné chyby. Takže, g5>out (sa-0) je testovaný 6 testovacími vektory. Jelikož neexistují žádné další vady, které mohou být testovány 6 testovacími vektory, vybrali jsme chyby, které jsou testované 5 vzory. Proto, g6>out (sa-0 a sa-1), g6>inp1 (sa-0), a g4>out (sa-0) jsou testovány 5 testovacími vektory. Pro těžce testované chyby můžeme vybrat například: x4 (sa-1), g2>inp1 (sa-1). Pouze jedinečný testovací vektor detekuje zde dvě chyby. Existuje více těžce testovaných chyb, které nemůžeme ukázat, ale v ostatních variantách obvodů by mělo být možné najít všechny. [10]

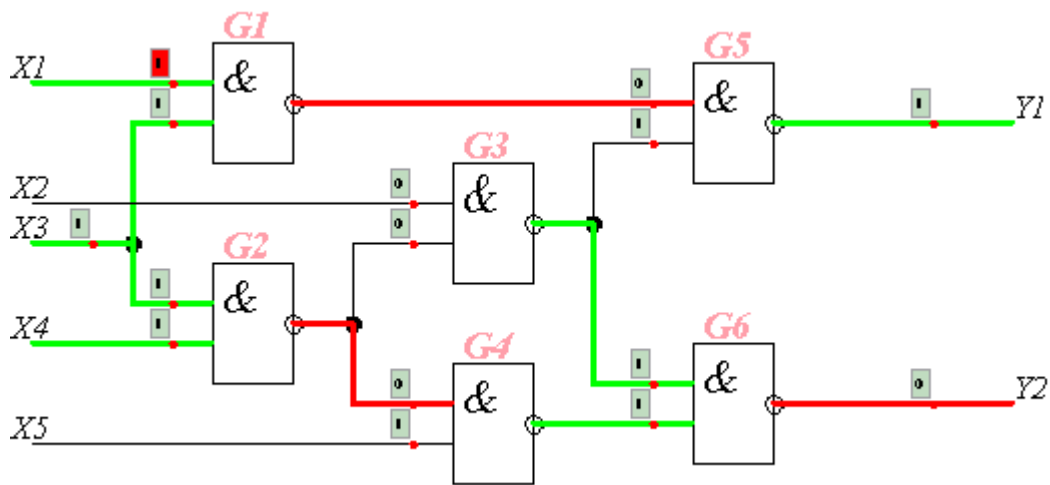
Nyní napíšeme testovací vektor, který obsahuje těžce testovanou chybu. Pro x4 (sa-1) – 11101, pro g2>inp1 (sa-1) – 11011. Tyto hodnoty jsme vyčetli z panelu vektorů. [10]

Je jasné, že třetí, pátý a devátý testovací vzor nezvyšují chybu krytí (v tabulce sloupec Increase[%]) a proto tyto vektory můžeme snadno odstranit. Zbytek testovacích vektorů budeme muset obejít, protože musíme vyzkoušet některé unikátní chyby. [10]

5.1.9 Manuální (deterministický) zkušební vzor generování – příklad

Předpokládejme, že jsme zvolili Generate testvectors z menu v části Working modes. Vyberte chybu x1 nastavte na (sa-0) a vložte ji do obvodu. Nyní musíme propagovat tuto chybu na výstup Y1. Proto, g1>inp2 musí mít hodnotu '1' a g5>inp2 musí mít také hodnotu '1'. Při použití těchto hodnot, vytváříme otevření bran g1 a g5 přes které chyba x1 (sa-0) je dále šířena. Musíme se také snažit pokrýt co nejvíce závad, jak můžeme. Například pokud jdeme na vstup x5 '0', nemohli jsme testovat x4, g2>inp1,

protože použitím této hodnoty, vytvoříme neotevřené hradlo NAND, tak by tato chyba neměla vliv na výstup hradla. [10]



Obr. 35 Ukázka schématu v módu Generate testvectors [10]

Předpokládejme, že máme 100% pokrytí chyb. Nyní bychom měli ověřit možnosti odstranění nadbytečných vektorů. Podívejme se na tabulku chyb. Můžeme vidět že, sedmý vektor nezvýší chybu pokrytí, proto jej můžeme odstranit. Také můžeme odstranit druhý a třetí testovací vektor, protože jiné zkušební vektory v tabulce chyb by mohly testovat stejnou chybu, tyto dva modely můžeme otestovat. Díky tomuto kroku jsme snížili množství testů z 9 na 6 testovacích vektorů. [10]

Fault table																	
#	x3	g1>inp2	x1	x4	g2>inp1	g3>inp2	x2	x5	g4>inp1	g5>inp2	g1>out	g4>out	g6>inp1	g6>out	g5>out	Increase [%]	Coverage [%]
1	X	X	1	X	X	X	1	1	X	0	0	0	0	1	1	30.0	30.0
2	0	0	0	X	X	X	1	1	X	X	1	0	0	1	0	16.7	46.7
3	0	0	0	0	0	X	X	X	1	X	1	0	0	1	0	10.0	56.7
4	0	0	0	1	X	X	X	0	0	X	1	1	X	0	0	16.7	73.3
5	1	X	X	X	1	0	0	X	X	1	X	X	X	0	0	16.7	90.0
6	0	0	0	0	0	1	X	X	1	X	1	0	0	1	0	3.3	93.3
7	X	X	X	X	X	0	0	X	X	1	X	X	X	0	0	0.0	93.3
8	1	1	X	X	X	X	1	0	0	0	0	1	X	0	1	3.3	96.7
9	X	X	X	X	X	0	0	X	X	1	X	X	1	0	0	3.3	100.0

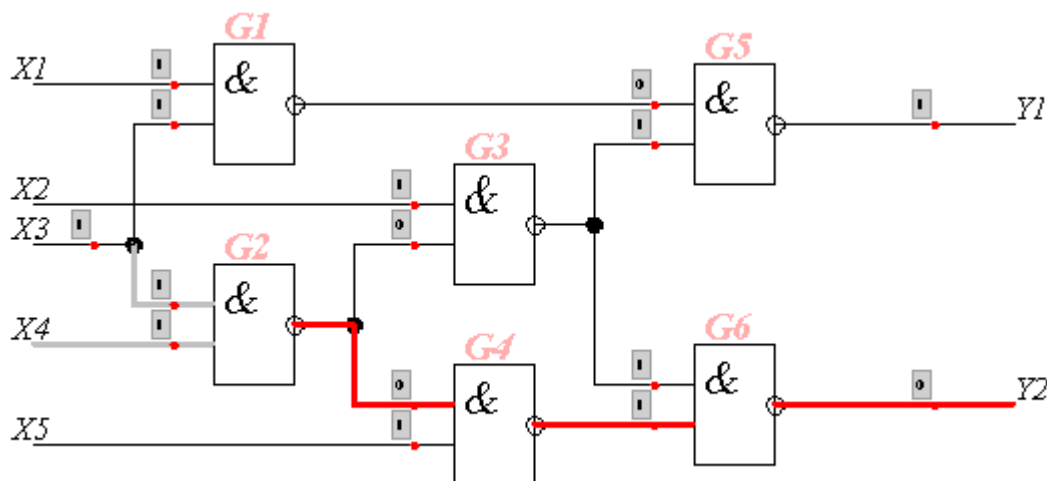
Fault table																	
#	x3	g1>inp2	x1	x4	g2>inp1	g3>inp2	x2	x5	g4>inp1	g5>inp2	g1>out	g4>out	g6>inp1	g6>out	g5>out	Increase [%]	Coverage [%]
1	X	X	1	X	X	X	1	1	X	0	0	0	0	1	1	30.0	30.0
2	0	0	0	1	X	X	X	0	0	X	1	1	X	0	0	33.3	63.3
3	1	X	X	X	1	0	0	X	X	1	X	X	X	0	0	16.7	80.0
4	0	0	0	0	0	1	X	X	1	X	1	0	0	1	0	13.3	93.3
5	1	1	X	X	X	X	1	0	0	0	1	X	X	0	1	3.3	96.7
6	X	X	X	X	X	0	0	X	X	1	X	X	1	0	0	3.3	100.0

Obr. 36 Tabulka chyb úpravou a po úpravě [10]

5.1.10 Guided probing diagnosis – příklad

Vložme náhodně poruchy do obvodu. Pak vybereme alespoň jeden vektor označený jako 'N' (neprošel, což znamená, že byla zjištěna závada na výstupu). Nyní musíme

přesunout sondu tak, abychom zpětně došli z výstupu na primární vstup. Snažíme se použít co nejmenší počet sond. Vyzkoušíme jeden z výstupů Y2. Měli jsme štěstí, protože na tomto výstupu je pozorovatelná chyba (červená čára Y2 označuje toto místo). Nyní musíme zkontrolovat jeden ze vstupů hradla G6. Použijeme druhý vstup G4>out. I tento vstup je červený. Dalším krokem je jít dále a kontrolovat jeden ze vstupů hradla G4. V tomto případě můžeme s jistotou říct, že X5 není vadný, protože G4>inp1 má hodnotu '0', která vytvoří NAND vstup X5 nepozorovatelným. Proto, jediný vstup G4>inp1 může ovlivnit výstupy hradel. Proto vybereme tento vstup a zjistíme, že je vadný. Nyní je čas použít sondu na jeden vstup hradla G2. Vybereme vstup G2>inp1. Šedá linka znázorňuje, že zde není chyba. Musíme zkontrolovat druhý vstup hradla G2. Náhodně vložená chyba by mohla být právě tam. Po kliknutí na druhý vstup hradla G2, zjistíme, že je tato linka také šedá. Uvědomili jsme si, že náš úsudek byl špatný. Nyní můžeme s jistotou říct, že zdrojem závad je vstup hradla G4>inp1. Potřebovali jsme 5 sond k lokalizaci této poruchy. Od této chvíle můžeme zastavit náš diagnostický proces, protože jsme zpětně zjistili zavinění z primárního výstupu Y2, kde byla závada pozorována až k jeho zdroji G4>inp1. [10]



Obr. 37 Technika zjištění chyb pomocí sond [10]

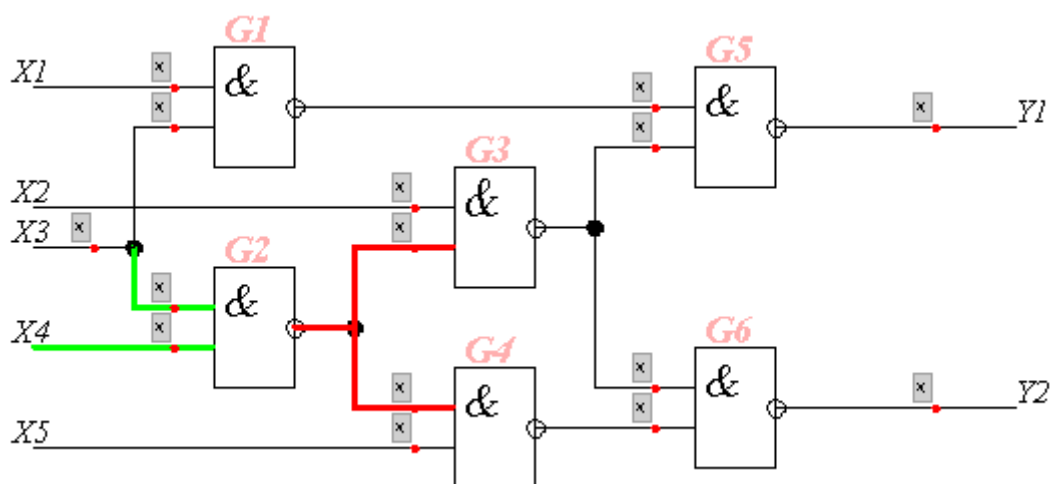
5.1.11 Kombinační diagnostika (fault table) – příklad

Vložme do obvodu náhodnou chybu. Nyní klikáním na vektory, musíme vybrat nižší počet testovaných vektorů, tak aby dostatečně lokalizovali náhodně vloženou chybu. Na tento příklad slouží obrázek níže, který znázorňuje fault table a daný obvod. V tomto případě je vybráno všech 6 vektorů, nemůžeme lokalizovat náhodně vložené chyb

(diagnostické rozlišení je příliš nízké a jsou zde momentálně 4 chyby). Fault table ukazuje, že stále existují 4 chyby, X4 (sa-0), G2>inp1 (sa-0), G3>inp2 (sa-1) a G4>inp1 (sa-1). Pouze jeden z nich obsahuje skutečnou chybu. Pro tento případ, musíme přidat takový model, který detekuje některé z těchto podezřelých 4 chyby, ale zároveň není možné detekovat další chyby. [10]

Fault table																		
#	x3	g1>inp2	x1	x4	g2>inp1	g3>inp2	x2	x5	g4>inp1	g5>inp2	g1>out	g4>out	g6>inp1	g6>out	g5>out	Increase [%]	Coverage [%]	Passed
1	X	X	1	X	X	X	1	1	X	0	0	0	0	1	1	30.0	30.0	Y
2	0	0	0	1	X	X	X	0	0	X	1	1	X	0	0	33.3	63.3	Y
3	1	X	X	X	1	0	0	X	X	1	X	X	X	0	0	16.7	80.0	Y
4	0	0	0	0	0	1	X	X	1	X	1	0	0	1	0	13.3	93.3	N
5	1	1	X	X	X	X	1	0	0	0	0	1	X	0	1	3.3	96.7	Y
6	X	X	X	X	X	0	0	X	X	1	X	X	1	0	0	3.3	100.0	Y

Obr. 38 Fault table [10]

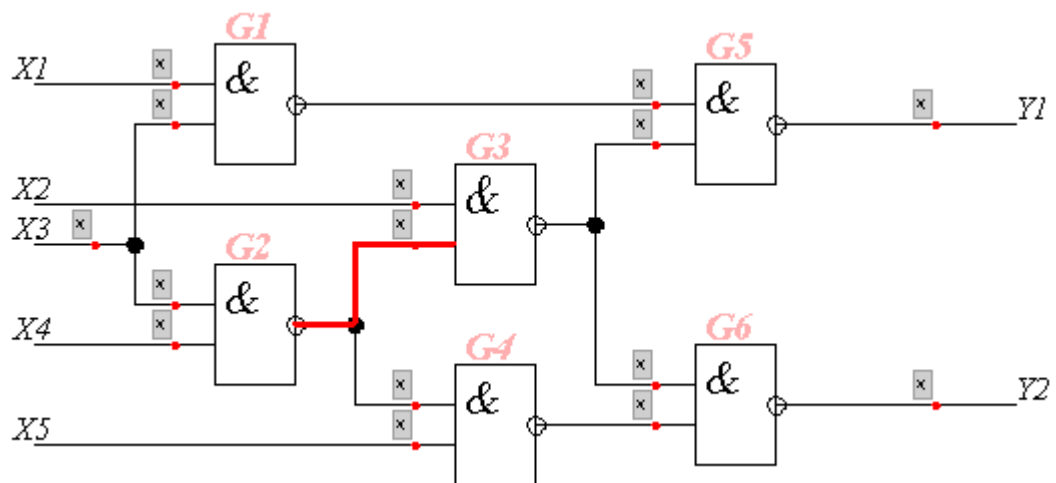


Obr. 39 Obvod odpovídající fault table [10]

Tímto způsobem za použití dalšího diagnostického test vektoru můžeme snížit seznam podezřelých poruch. Potom můžeme doufat, že najdeme náhodně vloženou chybu. Přidáme další testovací vektor 10111. Tento testovací model testuje 3 vady, X4 (sa-0), G2>inp1 (sa-0) a G4>inp1 (sa-1). Jako výsledek v nové fault table můžeme vidět, že poslední vektor, který jsme přidali je označen jako 'Y' (prošel, což znamená, že chyba nebyla zjištěna na vstupu) a proto tyto 3 zmíněné chyby jsou mimo podezření. Lze snadně předpokládat, že náhodně vložená chyba je umístěna na G3>inp2 a hodnoty sa-1. Obrázek níže zobrazuje novou fault table s přidáním sedmým vektorem a obvod, kde je náhodně vložená chyba graficky znázorněna. [10]

Fault table																		
#	x3	g1>inp2	x1	x4	g2>inp1	g3>inp2	x2	x5	g4>inp1	g5>inp2	g1>out	g4>out	g6>inp1	g6>out	g5>out	Increase (%)	Coverage (%)	Passed
1	X	X		1	X	X		1	1	X	0	0	0	1	1	30.0	30.0	Y
2	0	0		0	1	X	X	0	0	X	1	1	X	0	0	33.3	63.3	Y
3	1	X	X	X	1	0	0	X	X	1	X	X	X	0	0	16.7	80.0	Y
4	0	0	0	0	0	1	X	X	1	X	1	0	0	1	0	13.3	93.3	N
5	1	1		X	X	X	X	1	0	0	0	1	X	0	1	3.3	96.7	Y
6	X	X		X	X	X	0	0	X	X	1	X	X	1	0	3.3	100.0	Y
7	0	0	0	0	0	0	X	X	X	1	X	1	0	0	1	0.0	100.0	Y


Obr. 40 Nově vložený vektor do fault table [10]



Obr. 41 Obvod zobrazující náhodně vloženou chybu [10]

6 ZADÁNÍ PROTOKOLŮ


6.1 Zadání protokolu č. 1

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 1:</i>	Seznámení se s programem Diagnostic tester

Úkol:

- 1) Prostudujte teorii o programu Diagnostic tester. Seznamte se se základními ovládacími prvky programu. Popište jednotlivé části pracovních módů, které program obsahuje.
- 2) Pomocí uvedeného příkladu v teorii si vyzkoušejte funkčnost a ovladatelnost programu Diagnostic tester.
- 3) Popište, jaké problémy jste měli při první práci s daným programem.


6.2 Zadání protokolu č. 2

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 2:</i>	Měření na obvodu Circuit 1

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Tyto hodnoty vložte do protokolu.
- 2) V tabulce chyb určete vektory, které nemohou způsobit navýšení chyby a vektory, které obsahují snadně rozpoznatelnou chybu. Tyto vektory vypište.
- 3) Přepněte se do módu vkládání vektorů a veškerá data vymažte. Spusťte Generate testvectors a vložte chybu na pozici X5 a hodnotu nastavte na 0. Sestavte vektor tak aby se vygenerovaná chyba dostala až na patřičný výstup Y2 vámi vytvořený vektor vypište.


6.3 Zadaní protokolu č. 3

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 3:</i>	Vyhledávání chyb pomocí sond

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spusťte Guided probing a nechejte vložit náhodnou chybu do obvodu. V tabulce chyb se vyznačí hodnoty, které jsou chybné. Takto vyznačenou tabulku vložte do protokolu.
- 3) Nyní vyhledejte vadnou část obvodu a určete, kde nastala pravděpodobně chyba. Snažte se dosáhnout co možná nejmenšího počtu měřených bodů. Schéma obvodu vložte do protokolu a vzniklou chybu popište.


6.4 Zadání protokolu č. 4

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 4:</i>	Vyhledávání chyb pomocí kombinační detekce chyb

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. V tabulce chyb se vyznačí hodnoty, které jsou chybné. Takto vyznačenou tabulku vložte do protokolu.
- 3) Označte všechny části tabulky tak aby se vyznačila pouze jediná vložená chyba do obvodu. Pokud to není možné, vložte další testovací vektor tak, aby zajistil vyšší pokrytí obvodu a vyloučil ostatní chyby a dosáhli jsme vyznačení jediné náhodně vložené hodnoty do obvodu. Takto vytvořený vektor, tabulku chyb a obvod zobrazující závadu vložte do obvodu.


6.5 Zadání protokolu č. 5

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 5:</i>	Měření na obvodu Circuit 3

Úkol:

- 1) Na obvodu Circuit 3 proveďte vhodnou volbu vstupních proměnných v režimu Manual tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.6 Zadání protokolu č. 6

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 6:</i>	Měření na obvodu Circuit 4

Úkol:

- 1) Na obvodu Circuit 4 provedte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.7 Zadání protokolu č. 7

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 7:</i>	Měření na obvodu Circuit 5

Úkol:

- 1) Na obvodu Circuit 5 provedte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.8 Zadání protokolu č. 8

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 8:</i>	Měření na obvodu Circuit 6

Úkol:

- 1) Na obvodu Circuit 6 provedte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.9 Zadání protokolu č. 9

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 9:</i>	Měření na obvodu Circuit 7

Úkol:

- 1) Na obvodu Circuit 7 provedte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.10 Zadání protokolu č. 10

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 10:</i>	Měření na obvodu t1

Úkol:

- 1) Na obvodu t1 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.


6.11 Zadání protokolu č. 11

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 11:</i>	Měření na obvodu d1

Úkol:

- 1) Na obvodu d1 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

6.12 Zadání protokolu č. 12

 Univerzita Tomáše Bati ve Zlíně Fakulta aplikované informatiky	
<i>Předmět:</i>	Diagnostika číslicových systémů
<i>Protokol č. 12:</i>	Měření na d obvodech

Úkol:

- 1) Vyberte si nějaký z d obvodů a proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

7 VYPRACOVÁNÍ PROTOKOLŮ

7.1 Vypracování protokolu č. 1

Univerzita Tomáše Bati ve Zlíně FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 1	Seznámení se s programem Diagnostic tester	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Prostudujte teorii o programu Diagnostic tester. Seznamte se se základními ovládacími prvky programu. Popište jednotlivé části pracovních módů, které program obsahuje.
- 2) Pomocí uvedeného příkladu v teorii si vyzkoušejte funkčnost a ovladatelnost programu Diagnostic tester.
- 3) Popište, jaké problémy jste měli při první práci s daným programem.

Řešení:

- 1) Program se skládá z menu a pracovních oken. Okna jsou rozdělena do 3 částí. První je vektorová část kde si volíme ruční zadávání vektorů anebo část LFSR, kde si volíme zadání vstupních hodnot a zpětné vazby a následně kolik vektorů má být vygenerováno. Druhé okno zobrazuje sestavený obvod, jeho jednotlivé části, hradla, linky s hodnotami, které na nich jsou umístěny, vstupní a výstupní části. Poslední je tabulkové menu, kde jsou zobrazeny naše vygenerované vektory, dále tabulka chyb kde se zobrazují výstupní hodnoty navýšení chyb a pokrytí chybami. Signály na vstupech a výstupech z jednotlivých hradel a statistiku.

Menu je rozděleno na čtyři části. Circuit selection je databáze vložených obvodů do programu. Working mode představuje pracovní módy, které Diagnostic tester obsahuje. Language zde si můžeme zvolit námi vyhovující jazyk. Help nápověda. Working mode je rozdělen na 5 částí.

- Setting up testvectors – v tomto módu můžeme do obvodu vkládat jednotlivé vektory a dokonce i tyto vektory mazat.
 - Generate testvectors – Při jeho spuštění máme možnost náhodného vložení chyby do obvodu Random fault generation, pokud toto políčko odtrhneme máme možnost zvolit si Fault type a Fault location jedná se o vložení proměnné 0 anebo 1 a na jakou pozici tuto námi vytvořenou chybu chceme vložit
 - Fault simulator – při vytvoření vektorů nám chybová simulace slouží k vytvoření tabulky chyb kde je zobrazeno pokrytí chybami. Po vytvoření testovacích vektorů je vhodné spustit tento mód, abychom mohli vytvořit další vektory nebo již další vektory nepřidávali
 - Guided probing – testování pomocí sondy. Stejně jako Generate testvectors máme možnost vložení náhodné chyby do obvodu anebo si chybu můžeme do obvodu sami vložit. Při vložení je v tabulce chyb zobrazeno, jestli daný vektor prošel znakem Y na zeleném pozadí anebo N na červeném pozadí ve sloupečku Passed. Následně klepáním na jednotlivé výstupy se nám zobrazí, jestli je chyba právě na tomto výstupu a zpětně pokračujeme po obvodu, dokud nelokalizujeme umístění chyby.
 - Combinatorial fault detection – kombinační detekce chyb se spouští tak, že vybereme veškeré položky v tabulce chyb a pokud se nám v obvodu zobrazí jen 1 chybová linka, našli jsme poruchu v obvodu. Pokud se nám toto nepodaří, musíme přidat testovací vektory tak aby se dala chyba jednoznačně identifikovat. Má opět možnost vložení náhodné chyby anebo námi definované chyb.
- 2) Postupovali jsme dle uvedeného příkladu a následně se seznamovali s funkcemi a fungováním daného programu.
- 3) Při prvním spuštění bylo obtížné vytvořit vektory tak aby bylo dosaženo 100% pokrytí chybami v sekci LFSR a proto jsem do obvodu vkládal vektory manuálně dle příkladu, abych pochopil fungování tohoto programu. Při vložení vektoru

pomocí generate testvectors mi dělalo problém vhodně chybu dopravit na výstup. Odhalení chyby z tabulky chyb je dosti složité bez použití Guided probing a nebo Combinatorial fault detection. Při použití těchto módů se chyba daleko snadněji vyhledá.

7.2 Vypracování protokolu č. 2

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 2	Měření na obvodu Circuit 1	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Tyto hodnoty vložte do protokolu.
- 2) V tabulce chyb určete vektory, které nemohou způsobit navýšení chyby a vektory, které obsahují snadně rozpoznatelnou chybu. Tyto vektory vypište.
- 3) Přepněte se do módu vkládání vektorů a veškerá data vymažte. Spusťte Generate testvectors a vložte chybu na pozici X5 a hodnotu nastavte na 0. Sestavte vektor tak aby se vygenerovaná chyba dostala až na patřičný výstup Y2 vámi vytvořený vektor vypište.

Řešení:

- 1) Pomocí LFSR vkládání vektorů jsem vložil 6 vektorů. Tyto dostatečně stačí, protože bylo dosaženo 100% pokrytí chybami. Poté jsem spustil mód Fault simulator aby došlo k vyplnění tabulky chyb. Tabulka vektorů a tabulka chyb je zobrazena níže.

Testvector table											
#	X1	X2	X3	X4	X5	X6	I0>q	I1>q	I2>q	Y1	Y2
1	1	0	1	1	0	1	1	1	0	1	0
2	1	1	0	1	0	1	1	1	0	1	0
3	1	1	1	0	1	0	0	1	1	0	1
4	0	1	1	1	0	1	1	1	0	1	0
5	0	0	1	1	1	0	1	1	0	1	0
6	1	0	0	1	1	1	1	0	1	1	1
7											

Fault table												
#	X1	X2	X3	X5	X6	X4	I1>q	I0>q	I2>q	Increase [%]	Coverage [%]	Passed
1	X	1	X	1	X	0	0	0	1	33.3	33.3	
2	X	X	1	1	X	0	0	0	1	5.6	38.9	
3	0	0	0	X	X	1	X	1	0	33.3	72.2	
4	1	X	X	1	X	0	0	0	1	5.6	77.8	
5	X	X	X	X	1	0	0	0	1	5.6	83.3	
6	X	X	X	0	0	X	1	0	0	16.7	100.0	
7												

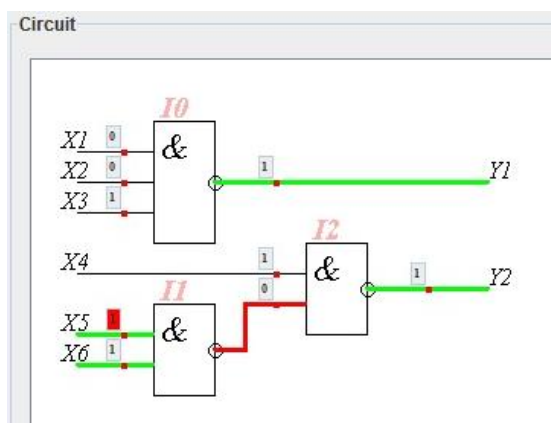
Vektor 6 splňuje podmínku pro pokrytí 100% chyb.

- 2) Jedná se o první 2 vektory 101101 a 110101 jelikož jejich výstupní hodnoty jsou naprosto stejné a rozpoznání, který z nich může obsahovat chybu je v tento moment nemožné. Proto tyto dva vektory můžeme odstranit, jelikož jejich výstupní hodnoty jsou naprosto totožné.

Fault table												
#	X1	X2	X3	X5	X6	X4	I1>q	I0>q	I2>q	Increase [%]	Coverage [%]	Passed
1	X	1	X	1	X	0	0	0	1	33.3	33.3	
2	X	X	1	1	X	0	0	0	1	5.6	38.9	
3	0	0	0	X	X	1	X	1	0	33.3	72.2	
4	1	X	X	1	X	0	0	0	1	5.6	77.8	
5	X	X	X	X	1	0	0	0	1	5.6	83.3	
6	X	X	X	0	0	X	1	0	0	16.7	100.0	
7												

Vektory, které nemohou způsobit navýšení chyby, v tomto případě zde nejsou.

- 3) Vytvořený vektor je 001111, chyba nastane v přechodu mezi hradly I1 a I2 ze vstupu X5.



7.3 Vypracování protokolu č. 3

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 3	Vyhledávání chyb pomocí sond	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spustěte Guided probing a nechejte vložit náhodnou chybu do obvodu. V tabulce chyb se vyznačí hodnoty, které jsou chybné. Takto vyznačenou tabulku vložte do protokolu.
- 3) Nyní vyhledejte vadnou část obvodu a určete, kde nastala pravděpodobně chyba. Snažte se dosáhnout co možná nejmenšího počtu měřených bodů. Schéma obvodu vložte do protokolu a vzniklou chybu popište.

Řešení:

- 1) Při vkládání vektorů pomocí LFSR jsem zvolil vytvoření 10 náhodných vektorů. Díky tomuto množství bylo dosaženo pokrytí chybami 100%. Po vytvoření vektorů jsem spustil Fault simulator aby došlo k vyplnění tabulky chyb. Tabulka vytvořených vektorů a tabulka chyb jsou zobrazeny níže.

Testvector table												
#	X1	X2	X3	X4	X5	X6	I0>q	I1>q	I2>q	Y1	Y2	
1	1	1	0	1	0	1	1	1	0	1	0	
2	1	1	1	0	1	0	0	1	1	0	1	
3	0	1	1	1	0	1	1	1	0	1	0	
4	0	0	1	1	1	0	1	1	0	1	0	
5	1	0	0	1	1	1	1	0	1	1	1	
6	0	1	0	0	1	1	1	0	1	1	1	
7	1	0	1	0	0	1	1	1	1	1	1	
8	1	1	0	1	0	0	1	1	0	1	0	
9	1	1	1	0	1	0	0	1	1	0	1	
10	0	1	1	1	0	1	1	1	0	1	0	

Fault table													
#	X1	X2	X3	X5	X6	X4	I1>q	I0>q	I2>q	Increase [%]	Coverage [%]	Passed	
1	X	X	1	1	X	0	0	0	1	33.3	33.3		
2	0	0	0	X	X	1	X	1	0	33.3	66.7		
3	1	X	X	1	X	0	0	0	1	5.6	72.2		
4	X	X	X	X	1	0	0	0	1	5.6	77.8		
5	X	X	X	0	0	X	1	0	0	16.7	94.4		
6	X	X	X	X	X	X	X	0	0	0.0	94.4		
7	X	1	X	X	X	1	X	0	0	5.6	100.0		
8	X	X	1	X	X	0	0	0	1	0.0	100.0		
9	0	0	0	X	X	1	X	1	0	0.0	100.0		
10	1	X	X	1	X	0	0	0	1	0.0	100.0		

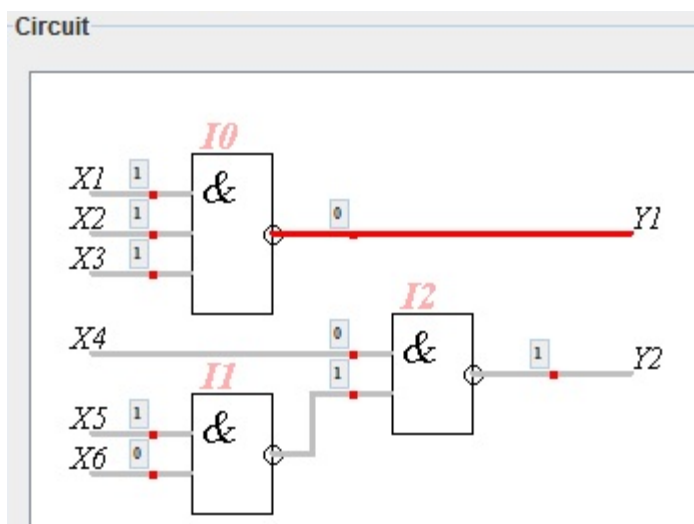
Vektory 8, 9 a 10 lze vymazat, protože vektor 7 obsahuje pokrytí chybami 100% a zbytek již nenavýší množství chyb.

- 2) V menu v části Working modes jsem vybral možnost Guided probing a vybral jsem možnost náhodného vložení chyby do obvodu dle zadání. Následující tabulka vyznačuje vektory obsahující chybu.

Fault table													
#	X1	X2	X3	X5	X6	X4	I1>q	I0>q	I2>q	Increase [%]	Coverage [%]	Passed	
1	X	X	1	1	X	0	0	0	1	33.3	33.3		
2	0	0	0	X	X	1	X	1	0	33.3	66.7		
3	1	X	X	1	X	0	0	0	1	5.6	72.2		
4	X	X	X	X	1	0	0	0	1	5.6	77.8		
5	X	X	X	0	0	X	1	0	0	16.7	94.4		
6	X	X	X	X	X	X	X	0	0	0.0	94.4		
7	X	1	X	X	X	1	X	0	0	5.6	100.0		

Tabulka ukazuje, že chyba je ve druhém vektoru. Jeho hodnota je 111010.

- 3) V obvodu jsem postupoval od výstupů, tak že jsem na každý klikl a postupně klikal na ostatní linie v obvodu, dokud se chyba zastaví a dále se v obvodu nebude zobrazovat. Pokud nastane chyba, linka se označí červenou barvou. Linky bez chyb jsou označeny šedou barvou.



Na spodní liště Diagnostic testeru se započítává, kolikrát bylo do obvodu kliknuto. V tomto případě došlo k devíti kliknutím. Na schématu je vidět, že chyba je na výstupu hradla I0, ale vstupy jsou v pořádku. Tudíž chyba je v obvodu I0 a k jejímu odstranění by bylo nutné tuto jednotku vyměnit.

7.4 Vypracování protokolu č. 4

Univerzita Tomáše Bati ve Zlíně FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 4	Vyhledávání chyb pomocí kombinační detekce chyb	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 1 proveďte vhodnou volbu vstupních proměnných a zpětných vazeb v režimu LFSR tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. V tabulce chyb se vyznačí hodnoty, které jsou chybné. Takto vyznačenou tabulku vložte do protokolu.
- 3) Označte všechny části tabulky tak aby se vyznačila pouze jediná vložená chyba do obvodu. Pokud to není možné, vložte další testovací vektor tak, aby zajistil vyšší pokrytí obvodu a vyloučil ostatní chyby a dosáhli jsme vyznačení jediné náhodně vložené hodnoty do obvodu. Takto vytvořený vektor, tabulku chyb a obvod zobrazující závadu vložte do obvodu.

Řešení:

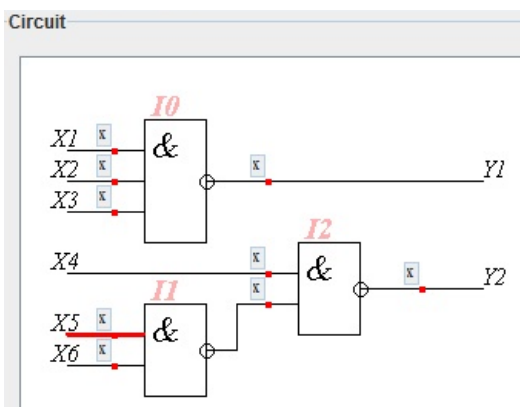
- 1) Vytvořil jsem celkem 7 vektorům, které vytvoří dostatečné pokrytí chybami. Následně byl spuštěn mód Fault simulator aby došlo k vyplnění tabulky chyb.

[illegible]

- 2) Po spuštění módu Combinatorial fault detection a vložení náhodné chyby jsem provedl vybrání všech položek v tabulce chyb, kde se vyznačí sloupce, ve kterých dochází k chybám.

[illegible]

- 3) Nebylo potřeba vytvářet další vektor k dosažení zpřesnění umístění chyby. Použité vektory nám určily, kde se chyba nachází a jedná se o vstup X5. Chyba je vyznačena níže ve schématu i v tabulce.

[illegible]

7.5 Vypracování protokolu č. 5

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 5	Měření na obvodu Circuit 3	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 3 proveďte vhodnou volbu vstupních proměnných v režimu Manual tak, aby bylo vytvořeno 100% pokrytí chybami. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

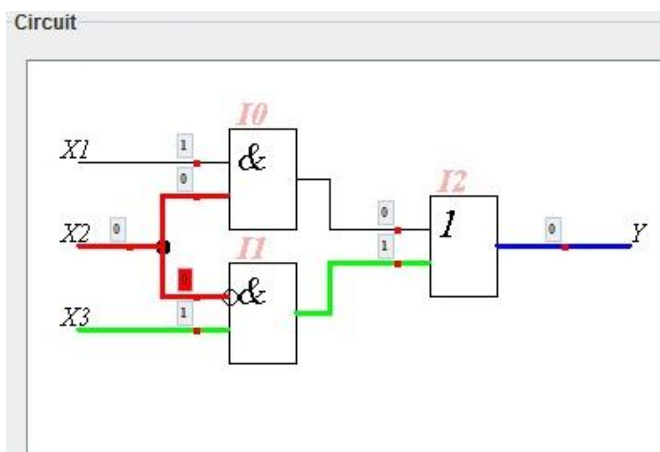
- 1) Na vytvoření 100% pokrytí stačilo vložit 6 vektorů. Jelikož obvod má 3 vstupy vyčerpal jsem všechny možnosti a kombinace vstupů.

Testvector table									
#	X1	X2	X3	X2	I0>q	I1>q	I2>q	Y	
1	1	0	0	0	0	0	0	0	
2	0	1	0	1	0	0	0	0	
3	0	0	1	0	0	1	1	1	
4	1	0	1	0	0	1	1	1	
5	0	1	1	1	0	0	0	0	
6	1	1	0	1	1	0	1	1	
7									
8									
9									
10									

Následně spuštěný mód Fault simulator vyplní tabulku chyb a vektor 110 představuje pokrytí chybami 100%. Další vektory není třeba vytvářet.

Fault table											
#	X2	X1	I0>b	I1>a	X3	I0>q	I1>q	I2>q	Increase [%]	Coverage [%]	Passed
1	1	X	1	X	1	1	1	1	37.5	37.5	
2	X	1	X	X	X	1	1	1	6.3	43.8	
3	1	X	X	1	0	X	0	0	25.0	68.8	
4	X	X	X	1	0	X	0	0	0.0	68.8	
5	0	1	X	0	X	1	1	1	12.5	81.3	
6	0	0	0	X	X	0	X	0	18.8	100.0	
7											
8											
9											
10											

- 2) Náhodná chyba je vložena do jedné z větví na vstupu X2



Výsledný vektor má hodnotu 101.

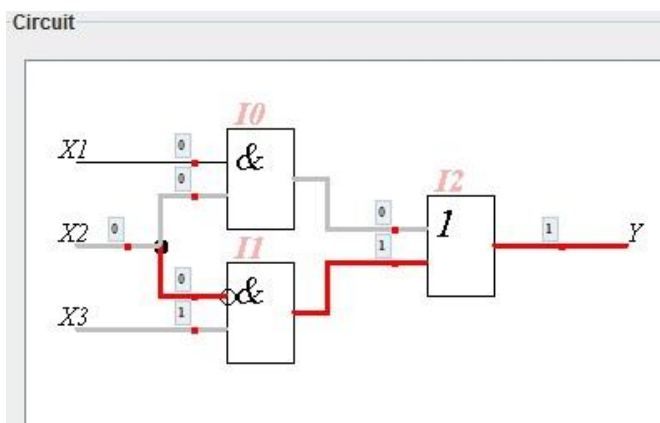
- 3) Odstraněním čtvrtého vektoru 101 dosáhneme zjednodušení tabulky.

Fault table											
#	X2	X1	I0>b	I1>a	X3	I0>q	I1>q	I2>q	Increase [%]	Coverage [%]	Passed
1	1	X	1	X	1	1	1	1	37.5	37.5	
2	X	1	X	X	X	1	1	1	6.3	43.8	
3	1	X	X	1	0	X	0	0	25.0	68.8	
4	X	X	X	1	0	X	0	0	0.0	68.8	
5	0	1	X	0	X	1	1	1	12.5	81.3	
6	0	0	0	X	X	0	X	0	18.8	100.0	
7											
8											
9											
10											

- 4) V módu Guided probing jsem nechal do obvodu vložit náhodnou chybu. Tabulka chyb vypíše, které vektory jsou bezchybné a které chybu obsahují.

Fault table											
#	X2	X1	I0>b	I1>a	X3	I0>q	I1>q	I2>q	Increase [%]	Coverage [%]	Passed
1	1	X	1	X	1	1	1	1	37.5	37.5	✓
2	X	1	X	X	X	1	1	1	6.3	43.8	✓
3	1	X	X	1	0	X	0	0	25.0	68.8	✗
4	0	1	X	0	X	1	1	1	12.5	81.3	✓
5	0	0	0	X	X	0	X	0	18.8	100.0	✓
6											
7											
8											
9											
10											

Následně jsem provedl kontrolní měření v obvodu. Kdy jsem pozoroval chybu na výstupu Y a zpětně hledal původní místo, kde chyba vznikla.



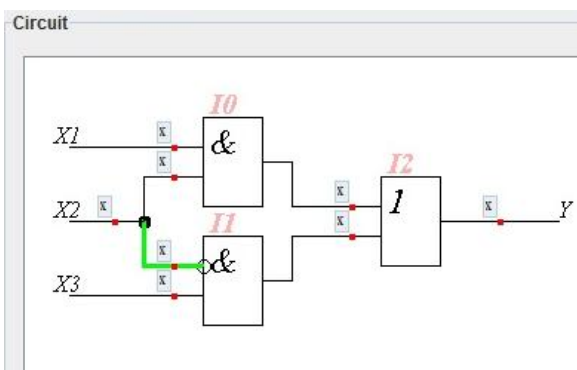
Chyba je stejně, jako ve druhém kroku kdy jsme nechali vložit chybu do obvodu v módu Generat testvectors. Na zjištění této chyby jsem potřeboval ověřit 7 kontrolních bodů. Vadným místem je tedy uzel na X2.

- 5) Při kombinačním hledání chyb jsme nechali vložit náhodnou chybu do obvodu. Jelikož daný obvod vykazoval více poruch při prvním měření. Přidal jsem do obvodu další testovací vektor, který měření upřesnil. Jedná se o vektor 101.

Testvector table									
#	X1	X2	X3	X2	I0>q	I1>q	I2>q	Y	
1	1	0	0	0	0	0	0	0	
2	0	1	0	1	0	0	0	0	
3	0	0	1	0	0	1	1	1	
4	0	1	1	1	0	0	0	0	
5	1	1	0	1	1	0	1	1	
6	1	0	1	0	0	1	1	1	
7									
8									
9									
10									

Následné označení všech vektorů v tabulce chyb nám v obvodu vyznačí místo, kde se chyba nachází.

Fault table												
#	X2	X1	I0>b	I1>a	X3	I0>q	I1>q	I2>q	Increase [%]	Coverage [%]	Passed	
1	1	X	1	X	1	1	1	1	37.5	37.5	Y	
2	X	1	X	X	X	1	1	1	6.3	43.8	Y	
3	1	X	X	1	0	X	0	0	25.0	68.8	Y	
4	0	1	X	0	X	1	1	1	12.5	81.3	Y	
5	0	0	0	X	X	0	X	0	18.8	100.0	Y	
6	X	X	X	1	0	X	0	0	0.0	100.0	Y	
7												
8												
9												
10												



Chyba je opět ve stejné části jako v krocích 2 a 4. Je to způsobeno jednoduchým zapojením obvodu.

7.6 Vypracování protokolu č. 6

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 6	Měření na obvodu Circuit 4	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 4 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

- 1) Zvolil jsem si metodu vkládání vektorů manuálně, protože obvod má jen 3 vstupy a vytvořit všechny možné kombinace není náročné.

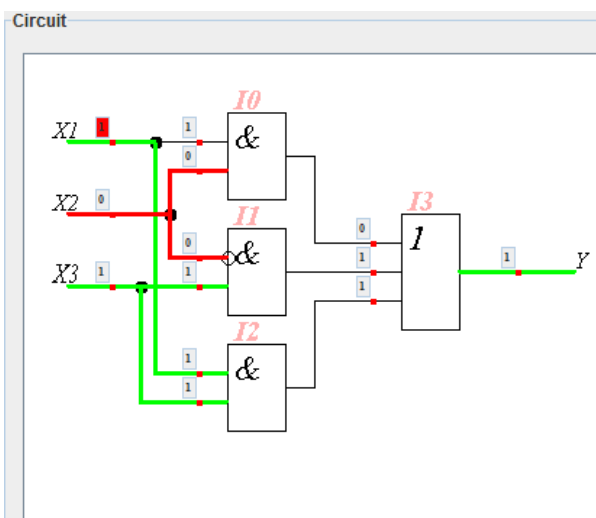
Testvector table												
#	X1	X2	X3	X1	X2	X3	I0>q	I1>q	I2>q	I3>q	Y	
1	0	0	0	0	0	0	0	0	0	0	0	
2	1	0	0	1	0	0	0	0	0	0	0	
3	0	1	0	0	1	0	0	0	0	0	0	
4	0	0	1	0	0	1	0	1	0	1	1	
5	1	1	0	1	1	0	1	0	0	1	1	
6	0	1	1	0	1	1	0	0	0	0	0	
7	1	0	1	1	0	1	0	1	1	1	1	
8	1	1	1	1	1	1	1	0	1	1	1	
9												
10												

V tabulce chyb ovšem nedošlo k vytvoření 100% pokrytí chybami.

Fault table																
#	X1	X2	X3	I0>a	I0>b	I1>a	I1>b	I2>a	I2>b	I0>q	I1>q	I2>q	I3>q	Increase [%]	Coverage [%]	
1	X	X	1	X	X	X	1	X	X	1	1	1	1	23.1	23.1	
2	X	1	1	X	1	X	1	X	1	1	1	1	1	11.5	34.6	
3	1	X	X	1	X	X	X	X	X	1	1	1	1	7.7	42.3	
4	X	1	0	X	X	1	0	X	X	X	0	X	0	19.2	61.5	
5	0	0	X	0	0	X	X	X	X	0	X	X	0	19.2	80.8	
6	1	0	X	1	X	0	X	1	X	1	1	1	1	7.7	88.5	
7	X	X	0	X	X	X	X	X	X	X	X	X	0	0.0	88.5	
8	0	X	X	X	X	X	X	X	X	X	X	X	0	0.0	88.5	
9																
10																

I přes využití všech možných kombinací vstupů se nepodařilo dosáhnout 100% pokrytí. A tudíž odhalovat chyby na takovémto obvodu je daleko složitější a bylo by potřeba obvod rozšířit o další vstup anebo využít nějakých přesnějších metod pro vytvoření vstupních vektorů.

- 2) I při vytvoření náhodné chyby v obvodu se chybu nepodaří dovést na výstup. Obvod se začne chovat jako by žádnou chybu neobsahoval a funguje správně.



Chyba je hned na vstupu X1 a pokud hodnoty pozměníme na opačné, tedy za uzlem použijeme místo 1 hodnotu 0, označí se nám tato cesta modře, což znamená špatně

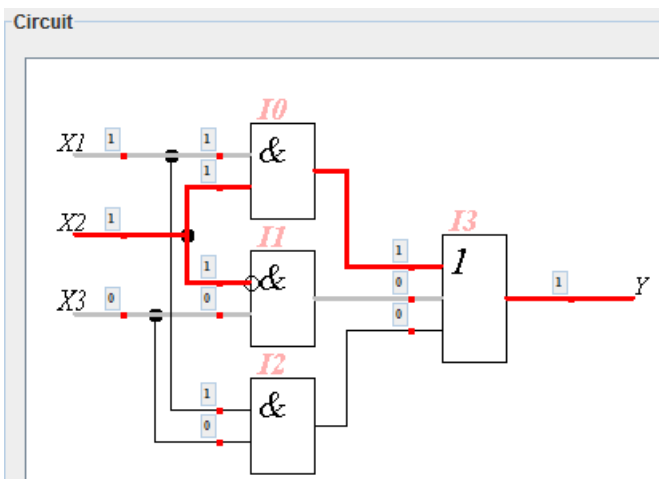
zvolenou hodnotu. Chyba tedy zůstane na své pozici a v obvodu se dále nešíří. Výsledný vektor je tedy 101.

- 3) Odstranit můžeme poslední 2 vektory, které nezpůsobují navýšení chyb. Jedná se o vektor 101 a 111.

Fault table																
#	X1	X2	X3	I0>a	I0>b	I1>a	I1>b	I2>a	I2>b	I0>q	I1>q	I2>q	I3>q	Increase [%]	Coverage [%]	Passed
1	X	X	1	X	X	X	1	X	X	1	1	1	1	23.1	23.1	
2	X	1	1	X	1	X	1	X	1	1	1	1	1	11.5	34.6	
3	1	X	X	1	X	X	X	X	X	1	1	1	1	7.7	42.3	
4	X	1	0	X	X	1	0	X	X	X	0	X	0	19.2	61.5	
5	0	0	X	0	0	X	X	X	X	0	X	X	0	19.2	80.8	
6	1	0	X	1	X	0	X	1	X	1	1	1	1	7.7	88.5	
7																
8																

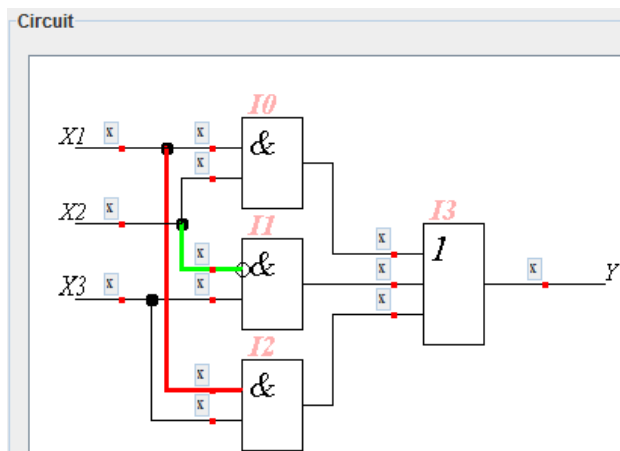
- 4) Při hledání chyb pomocí sondy v módu Guided probing se náhodně vložená chyba v obvodu dala pomocí sondy vystopovat. V tabulce chyb se vyobrazily 2 vložené chyby a následovným měřením od výstupu jsem zjistil, že chyba je vložena hned na vstupu X2.

Fault table																
#	X1	X2	X3	I0>a	I0>b	I1>a	I1>b	I2>a	I2>b	I0>q	I1>q	I2>q	I3>q	Increase [%]	Coverage [%]	Passed
1	X	X	1	X	X	X	1	X	X	1	1	1	1	23.1	23.1	
2	X	1	1	X	1	X	1	X	1	1	1	1	1	11.5	34.6	
3	1	X	X	1	X	X	X	X	X	1	1	1	1	7.7	42.3	
4	X	1	0	X	X	1	0	X	X	X	0	X	0	19.2	61.5	
5	0	0	X	0	0	X	X	X	X	0	X	X	0	19.2	80.8	
6	1	0	X	1	X	0	X	1	X	1	1	1	1	7.7	88.5	
7																
8																

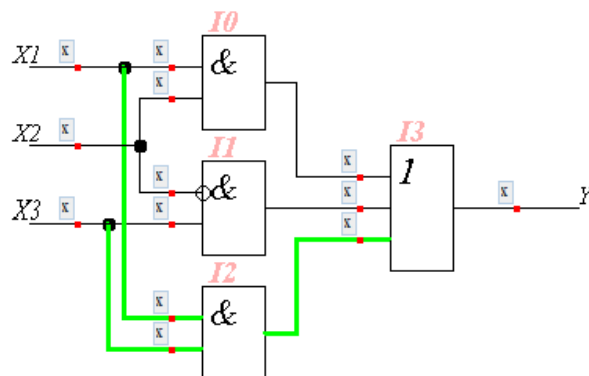


K určení chyby jsem potřeboval použít 10 měřících bodů.

- 5) Při kombinovaném vyhledání chyb se v obvodu vyznačily 2 větve jedna jako špatná druhá jako správná.



Jelikož nejde určit, o jakou chybu se jedná a kde přesně je chyba umístěna provedl jsem rozšíření o smazané vektory 101 a 111. A následně jsem provedl měření znovu.



Fault table																	
#	X1	X2	X3	I0>a	I0>b	I1>a	I1>b	I2>a	I2>b	I0>q	I1>q	I2>q	I3>q	Increase [%]	Coverage [%]	Passed	
1	X	X	1	X	X	X	1	X	X	1	1	1	1	23.1	23.1	Y	
2	X	1	1	X	1	X	1	X	1	1	1	1	1	11.5	34.6	Y	
3	1	X	X	1	X	X	X	X	X	1	1	1	1	7.7	42.3	Y	
4	X	1	0	X	X	1	0	X	X	X	0	X	0	19.2	61.5	Y	
5	0	0	X	0	0	X	X	X	X	0	X	X	0	19.2	80.8	Y	
6	1	0	X	1	X	0	X	1	X	1	1	1	1	7.7	88.5	Y	
7	0	X	X	X	X	X	X	X	X	X	X	X	0	0.0	88.5	Y	
8	X	X	0	X	X	X	X	X	X	X	X	X	0	0.0	88.5	Y	
9																	

Jak je vidět i po vložení odstraněných vektorů není tato metoda schopna určit vloženou chybu a proto je měření dosti nepřesné. Pro přesné zjištění chyb bychom museli použít jiné a přesnější měřící metody. Ty ovšem tento program nenabízí.

7.7 Vypracování protokolu č. 7

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 7	Měření na obvodu Circuit 5	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 5 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

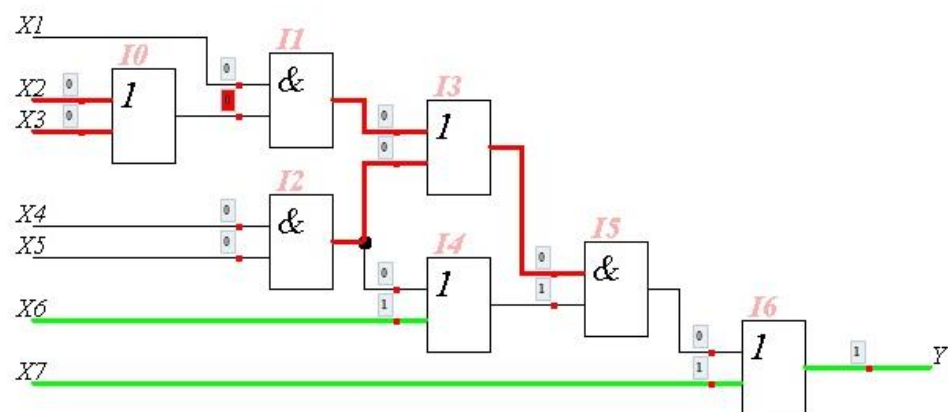
- 1) U obvodu Circuit 5 jsem zvolil metodu vkládání vektorů LFSR jelikož obvod obsahuje 7 vstupních proměnných a jejich sestavování manuálně by bylo náročné. Vektory jsem vkládal několikrát abych dosáhl co největšího pokrytí chybami, ale nakonec bylo dosaženo jen 90% pokrytí s pomocí 30-ti vloženými vektory.

Testvector table																
#	X1	X2	X3	X4	X5	X6	X7	I0>q	I1>q	I2>q	I3>q	I4>q	I5>q	I6>q	Y	
1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1	
2	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
3	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1	
4	0	0	1	0	1	1	1	1	0	0	0	1	0	1	1	
5	1	0	0	1	0	1	1	0	0	0	0	1	0	1	1	
6	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	
7	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	
8	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	
9	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
10	1	1	0	0	0	1	0	1	1	0	1	1	1	1	1	
11	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	
12	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	
13	0	0	0	1	1	0	0	0	0	1	1	1	1	1	1	
14	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	
15	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	
16	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	
17	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	
18	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0	
19	1	1	0	1	0	0	0	1	1	0	1	0	0	0	0	
20	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	
21	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	
22	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	
23	1	1	1	0	1	1	0	1	1	0	1	1	1	1	1	
24	0	1	1	1	0	1	1	1	0	0	0	1	0	1	1	
25	0	0	1	1	1	0	1	1	0	1	1	1	1	1	1	

Fault table																	
#	X2	X3	X1	I0>q	X4	X5	I1>q	I3>b	I4>a	X6	I3>q	I4>q	I5>q	X7	I6>q	Increase [%]	Coverage [%]
1	X	X	X	X	0	0	X	0	X	X	0	0	0	X	0	23.3	23.3
2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0.0	23.3
3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0.0	23.3
4	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	3.3	26.7
5	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	26.7
6	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	26.7
7	X	X	1	X	X	X	1	1	X	X	1	X	1	1	1	23.3	50.0
8	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	50.0
9	X	X	X	X	1	X	X	X	X	X	X	X	1	1	1	3.3	53.3
10	0	X	0	0	X	X	0	X	X	0	0	0	0	X	0	16.7	70.0
11	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	70.0
12	X	X	X	X	X	1	X	X	X	X	X	X	1	1	1	3.3	73.3
13	X	X	X	X	0	0	X	0	0	X	0	0	0	X	0	3.3	76.7
14	X	X	X	X	1	X	1	1	X	X	1	X	1	1	1	0.0	76.7
15	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	76.7
16	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	76.7
17	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0.0	76.7
18	X	X	X	X	X	X	X	X	1	1	X	1	1	1	1	10.0	86.7
19	X	X	X	X	X	1	X	X	1	1	X	1	1	1	1	0.0	86.7
20	X	X	X	X	1	X	X	X	X	X	X	X	1	1	1	0.0	86.7
21	X	0	0	0	X	X	0	X	X	0	0	0	0	X	0	3.3	90.0
22	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0.0	90.0
23	X	X	0	0	X	X	0	X	X	0	0	0	0	X	0	0.0	90.0
24	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	90.0

Jak je vidět v tabulce chyb od vektoru 21 nedochází k navýšení chyb a tudíž ostatní hodnoty jsou zbytečné, jelikož obvod nijak neovlivní.

- 2) Náhodně vložená chyba v módu Generate testvectors se mi podařila dovést až na obvod I5. Jak znázorňuji níže zobrazené schéma zapojení. Z výstupu obvodu I5 se chyba již dále nešíří. Chyba se tedy v obvodu ztrácí. Výsledný vektor má hodnotu 0000011.



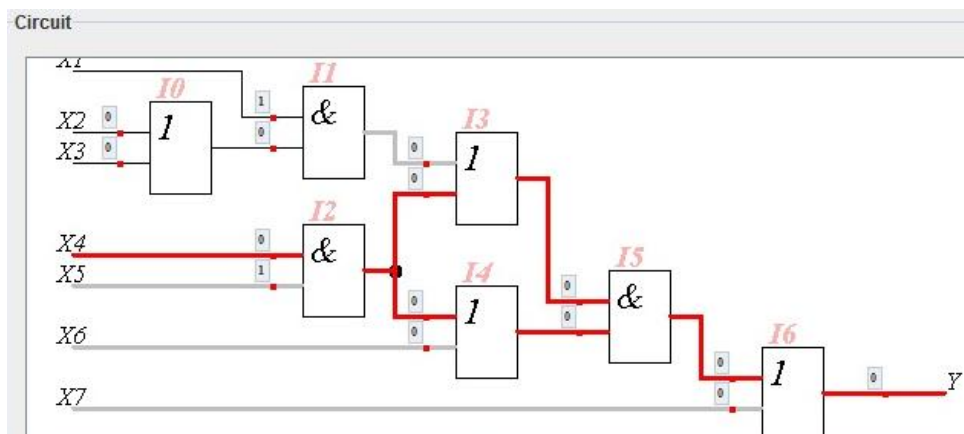
- 3) Při odstranění vektorů, které již chybu v obvodu neovlivní, došlo k razantnímu zkrácení. Po odstranění se původní množství vektorů zmenšilo na pouhých 9.

	X2	X3	X1	I0>q	X4	X5	I1>q	I3>b	I4>a	X6	I3>q	I4>q	I5>q	X7	I6>q	Increase [%]	Coverage [%]
	X	X	X	X	0	0	X	0	X	X	0	0	0	X	0	23.3	23.3
	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	3.3	26.7
	X	X	1	X	X	X	1	1	X	X	1	X	1	1	1	23.3	50.0
	X	X	X	X	1	X	X	X	X	X	X	X	1	1	1	3.3	53.3
	0	X	0	0	X	X	0	X	X	0	0	0	0	X	0	16.7	70.0
	X	X	X	X	X	1	X	X	X	X	X	X	1	1	1	3.3	73.3
	X	X	X	X	0	0	X	0	0	X	0	0	0	X	0	3.3	76.7
	X	X	X	X	X	X	X	X	1	1	X	1	1	1	1	10.0	86.7
	X	0	0	0	X	X	0	X	X	0	0	0	0	X	0	3.3	90.0

- 4) Při vyhledání chyb pomocí sondy jsem nechal vložit náhodnou chybu do obvodu a ta byla následně zobrazena v tabulce chyb.

	X2	X3	X1	I0>q	X4	X5	I1>q	I3>b	I4>a	X6	I3>q	I4>q	I5>q	X7	I6>q	Increase [%]	Coverage [%]	Pass
	X	X	X	X	0	0	X	0	X	X	0	0	0	X	0	23.3	23.3	✓
	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	3.3	26.7	✓
	X	X	1	X	X	X	1	1	X	X	1	X	1	1	1	23.3	50.0	✓
	X	X	X	X	1	X	X	X	X	X	X	X	1	1	1	3.3	53.3	✗
	0	X	0	0	X	X	0	X	X	0	0	0	0	X	0	16.7	70.0	✓
	X	X	X	X	X	1	X	X	X	X	X	X	1	1	1	3.3	73.3	✓
	X	X	X	X	0	0	X	0	0	X	0	0	0	X	0	3.3	76.7	✓
	X	X	X	X	X	X	X	X	1	1	X	1	1	1	1	10.0	86.7	✓
	X	0	0	0	X	X	0	X	X	0	0	0	0	X	0	3.3	90.0	✓

Při měření na kontrolních bodech jsem dosáhl z výstupu, až na místo kde byla chyba vložena.

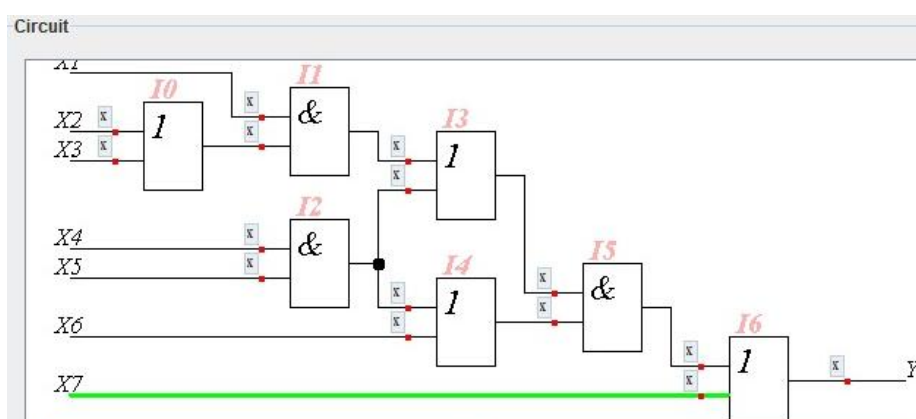


Jednalo se o vstup X4 jak je vidět ve schématu. K určení tohoto místa jsem potřeboval provést 11 kontrolních měření.

- 5) Při použití kombinovaného vyhledání chyb došlo k nepřesnostem.

	X2	X3	X1	I0>q	X4	X5	I1>q	I3>b	I4>a	X6	I3>q	I4>q	I5>q	X7	I6>q	Increase [%]	Coverage [%]	Pass
	X	X	X	X	0	0	X	0	X	X	0	0	0	X	0	23.3	23.3	Y
	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	3.3	26.7	Y
	X	X	1	X	X	X	1	1	X	X	1	X	1	1	1	23.3	50.0	Y
	X	X	X	X	1	X	X	X	X	X	X	X	1	1	1	3.3	53.3	Y
	0	X	0	0	X	X	0	X	X	X	0	0	0	X	0	16.7	70.0	Y
	X	X	X	X	X	1	X	X	X	X	X	X	1	1	1	3.3	73.3	Y
	X	X	X	X	0	0	X	0	0	X	0	0	0	X	0	3.3	76.7	Y
	X	X	X	X	X	X	X	X	1	1	X	1	1	1	1	10.0	86.7	Y
	X	0	0	0	X	X	0	X	X	0	0	0	0	X	0	3.3	90.0	Y

Tabulka chyb obsahuje jeden chybný vektor.



Ale schéma zapojení nedetekuje žádnou chybu v obvodu. Tudíž je tato metoda nepřesná a na zjištění chyb v takovémto druhu zapojení by bylo vhodnější využít

metodu Guided probing. I když je zapotřebí provádět zpětná kontrolní měření pomocí sondy.

7.8 Vypracování protokolu č. 8

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 8	Měření na obvodu Circuit 6	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 6 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

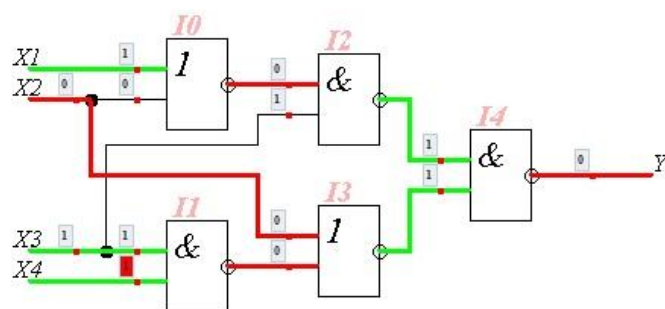
- 1) Při sestavování vektorů jsem použil metodu LFSR. Obvod obsahuje 4 vstupy a metoda LFSR je při vytváření vektorů rychlejší. Pro vytvoření jsem použil 20 vektorů.

Testvector table													
#	X1	X2	X3	X4	X2	X3	I0>q	I1>q	I2>q	I3>q	I4>q	Y	
1	0	1	1	1	1	1	0	0	1	0	1	1	
2	0	0	1	1	0	1	1	0	0	1	1	1	
3	1	0	0	1	0	0	0	1	1	0	1	1	
4	0	1	0	0	1	0	0	1	1	0	1	1	
5	0	0	1	0	0	1	1	1	0	0	1	1	
6	1	0	0	1	0	0	0	1	1	0	1	1	
7	1	1	0	0	1	0	0	1	1	0	1	1	
8	0	1	1	0	1	1	0	1	1	0	1	1	
9	0	0	1	1	0	1	1	0	0	1	1	1	
10	0	0	0	1	0	0	1	1	1	0	1	1	
11	0	0	0	0	0	0	1	1	1	0	1	1	
12	1	0	0	0	0	0	0	1	1	0	1	1	
13	0	1	0	0	1	0	0	1	1	0	1	1	
14	1	0	1	0	0	1	0	1	1	0	1	1	
15	1	1	0	1	1	0	0	1	1	0	1	1	
16	0	1	1	0	1	1	0	1	1	0	1	1	
17	1	0	1	1	0	1	0	0	1	1	0	0	
18	0	1	0	1	1	0	0	1	1	0	1	1	
19	1	0	1	0	0	1	0	1	1	0	1	1	
20	0	1	0	1	1	0	0	1	1	0	1	1	

Fault table																
#	X2	X3	X1	I0>b	I1>a	X4	I0>q	I2>b	I3>a	I1>q	I2>q	I3>q	I4>q	Increase [%]	Coverage [%]	
1	X	X	X	X	X	X	X	X	0	X	X	1	0	11.5	11.5	
2	X	X	1	1	X	X	0	0	X	X	1	X	0	19.2	30.8	
3	X	1	X	X	1	X	X	X	X	0	X	1	0	11.5	42.3	
4	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	42.3	
5	X	X	X	X	X	X	X	X	X	X	X	X	0	0.0	42.3	
6	X	1	X	X	1	X	X	X	X	0	X	1	0	0.0	42.3	
7	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	42.3	
8	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	42.3	
9	X	X	1	1	X	X	0	0	X	X	1	X	0	0.0	42.3	
10	X	X	X	X	1	X	X	X	X	0	X	1	0	0.0	42.3	
11	X	X	X	X	X	X	X	X	X	0	X	1	0	0.0	42.3	
12	X	X	X	X	X	X	X	X	X	0	X	1	0	0.0	42.3	
13	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	42.3	
14	X	X	X	X	X	1	X	X	X	0	X	1	0	3.8	46.2	
15	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	46.2	
16	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	46.2	
17	1	0	0	X	0	0	1	X	1	1	0	0	1	42.3	88.5	
18	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	88.5	
19	X	X	X	X	X	1	X	X	X	0	X	1	0	0.0	88.5	
20	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	88.5	

Jak je vidět ani 20 vytvořených vektorů nestačí k vytvoření pokrytí chybami 100%. Dosáhl jsem jen 88.5% v tomto obvodu. Jak je vidět poslední 3 vektory již nijak hodnotu chyb nijak nezvyšují.

- 2) Při použití Generate testvectors byla do obvodu vložena náhodná chyba. I když je obvod poměrně složitý a obsahuje uzly, podařilo se mi chybu vyprodukovat až na výstup obvodu. Výsledný vektor má hodnotu 1011.



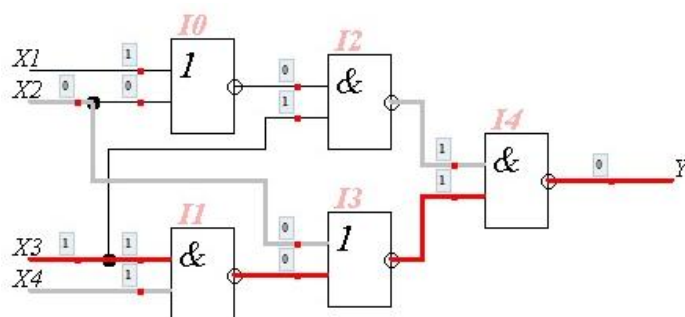
- 3) Po zkrácení vektorů, které nezvyšují chyby v obvodu, došlo ke značnému zjednodušení. Z původních 20-ti vektorů je výsledný počet pouhých 5.

Fault table															
#	X2	X3	X1	I0>b	I1>a	X4	I0>q	I2>b	I3>a	I1>q	I2>q	I3>q	I4>q	Increase [%]	Coverage [%]
1	X	X	X	X	X	X	X	X	0	X	X	1	0	11.5	11.5
2	X	X	1	1	X	X	0	0	X	X	1	X	0	19.2	30.8
3	X	1	X	X	1	X	X	X	X	0	X	1	0	11.5	42.3
4	X	X	X	X	X	1	X	X	X	0	X	1	0	3.8	46.2
5	1	0	0	X	0	0	1	X	1	1	0	0	1	42.3	88.5
6															
7															

- 4) Pomocí metody Guided probing byla do obvodu náhodně vložena chyba, která byla v tabulce chyb zvýrazněna.

Fault table																
#	X2	X3	X1	I0>b	I1>a	X4	I0>q	I2>b	I3>a	I1>q	I2>q	I3>q	I4>q	Increase [%]	Coverage [%]	Passed
1	X	X	X	X	X	X	X	X	0	X	X	1	0	11.5	11.5	Y
2	X	X	1	1	X	X	0	0	X	X	1	X	0	19.2	30.8	Y
3	X	1	X	X	1	X	X	X	X	0	X	1	0	11.5	42.3	Y
4	X	X	X	X	X	1	X	X	X	0	X	1	0	3.8	46.2	Y
5	1	0	0	X	0	0	1	X	1	1	0	0	1	42.3	88.5	N
6																

Ve schématu jsem provedl následovně zpětné vyhledání chyby z výstupu až k místu kde byla chyba vložena.



Chyba byla umístěna na vstup X3 jak je zobrazeno ve schématu. Na její určení jsem provedl 9 kontrolních měření pomocí sondy.

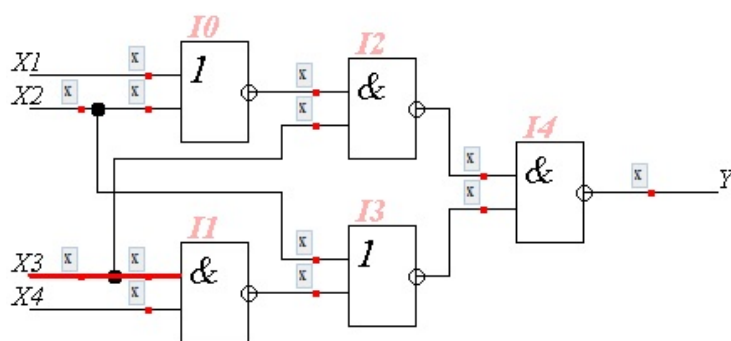
- 5) Metoda Combinatorial fault detection vyhledá chybu pomocí sestavených vektorů. Jenže v tomto případě je vektorů málo a proto jsem musel rozšířit tabulku o další vektory, aby byla chyba v obvodu jednoznačně odhalena.

Testvector table													
#	X1	X2	X3	X4	X2	X3	I0>q	I1>q	I2>q	I3>q	I4>q	Y	
1	0	1	1	1	1	1	0	0	1	0	1	1	
2	0	0	1	1	0	1	1	0	0	1	1	1	
3	1	0	0	1	0	0	0	1	1	0	1	1	
4	1	0	1	0	0	1	0	1	1	0	1	1	
5	1	0	1	1	0	1	0	0	1	1	0	0	
6	1	1	1	0	1	1	0	1	1	0	1	1	
7	1	1	0	1	1	0	0	1	1	0	1	1	
8	1	0	1	1	0	1	0	0	1	1	0	0	
9	0	1	1	1	1	1	0	0	1	0	1	1	

Tabulka chyb vypadá následovně.

Fault table																	
#	X2	X3	X1	I0>b	I1>a	X4	I0>q	I2>b	I3>a	I1>q	I2>q	I3>q	I4>q	Increase [%]	Coverage [%]	Passed	
1	X	X	X	X	X	X	X	X	0	X	X	1	0	11.5	11.5	✓	
2	X	X	1	1	X	X	0	0	X	X	1	X	0	19.2	30.8	✓	
3	X	1	X	X	1	X	X	X	X	0	X	1	0	11.5	42.3	✗	
4	X	X	X	X	X	1	X	X	X	0	X	1	0	3.8	46.2	✓	
5	1	0	0	X	0	0	1	X	1	1	0	0	1	42.3	88.5	✓	
6	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	88.5	✓	
7	X	X	X	X	X	X	X	X	X	X	X	1	0	0.0	88.5	✓	
8	1	0	0	X	0	0	1	X	1	1	0	0	1	0.0	88.5	✓	
9	X	X	X	X	X	X	X	X	0	X	X	1	0	0.0	88.5	✓	

A následně je zobrazena v obvodu a představuje ji chyba na vstupu X3.



7.9 Vypracování protokolu č. 9

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 9	Měření na obvodu Circuit 7	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu Circuit 7 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

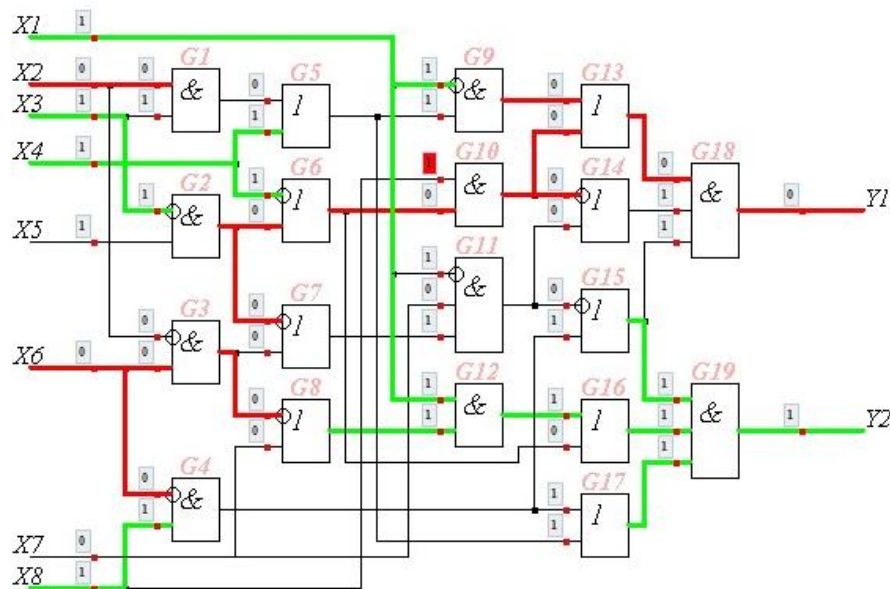
- 1) Obvod Circuit 7 obsahuje celkem 8 vstupů a vhodnější metoda pro vkládání vektorů je LFSR. Při vytváření vektorů jsem zvolil právě tuto metodu a zadal jsem pro vytvoření celkem 30 testovacích vektorů.

Testvector table																			
#	x1	x2	x3	x4	x5	x6	x7	x8	x1	x2	x3	x4	x5	x6	x7	x8	g1>out	g2>out	g3>out
1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	0	0	1
2	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0
3	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1
4	0	0	0	1	0	1	1	1	0	0	0	1	1	1	1	1	0	0	1
5	1	0	0	0	1	0	1	1	1	0	0	0	0	1	1	0	1	0	1
6	1	1	0	0	0	1	0	1	1	1	0	0	1	0	1	0	0	0	1
7	0	1	1	0	0	0	1	0	0	1	1	0	0	1	0	1	0	0	1
8	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1
9	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1
10	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	1
11	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	1
12	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1
13	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1
14	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1
15	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1
16	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	1	0	0	1
17	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1
18	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1
19	0	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0	1	0	0
20	1	0	1	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1	1
21	0	1	0	1	0	0	1	1	0	1	0	1	0	1	1	0	0	0	1
22	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1

Fault table									
g13>out	g14>out	g18>inp3	g19>inp1	g16>out	g17>out	g18>out	g19>out	Increase [%]	Coverage [%]
1	X	X	0	0	0	1	0	20.0	20.0
X	1	X	0	0	0	1	0	16.0	36.0
X	X	1	X	X	X	1	1	6.0	42.0
X	X	1	X	X	X	1	1	4.0	46.0
X	1	X	0	0	0	1	0	4.0	50.0
X	1	X	X	X	1	1	1	7.0	57.0
X	X	1	1	X	X	1	1	4.0	61.0
0	0	0	X	1	X	0	1	15.0	76.0
0	0	0	0	0	0	0	0	1.0	77.0
1	X	X	X	X	1	1	1	1.0	78.0
X	X	X	X	X	X	1	1	0.0	78.0
0	0	0	0	0	0	0	0	5.0	83.0
X	1	X	0	0	0	1	0	0.0	83.0
1	X	X	X	X	1	1	1	0.0	83.0
1	X	X	0	0	0	1	0	4.0	87.0
0	0	0	X	1	X	0	1	2.0	89.0
0	0	0	X	1	X	0	1	1.0	90.0
1	X	X	0	0	0	1	0	0.0	90.0
1	X	X	X	X	1	1	1	0.0	90.0
X	1	X	X	X	1	1	1	2.0	92.0
0	0	0	X	1	X	0	1	0.0	92.0
1	X	X	0	0	0	1	0	2.0	94.0

Dosažné pokrytí chybami je jen 94%. Proto jsem vektory zkrátil z původních 30-ti na 22 jelikož zbylých 8 již nenavýší množství chyb a pokrytí zůstane naprosto stejné.

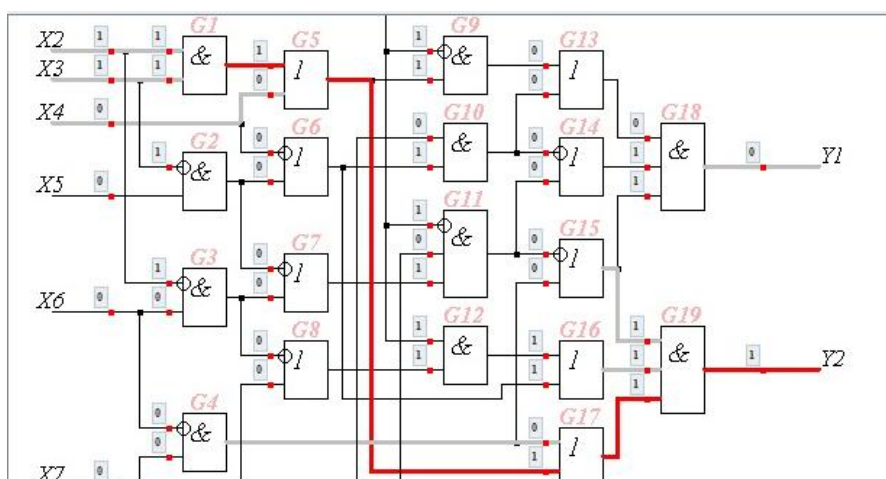
- 2) Vloženou chybu pomocí módu Generate testvectors se mi podařilo dovést až na výstup Y1. Obvod je dosti složitý a nepřehledný a vložená chyba je za uzlem ze vstupu X8. Chyba se tedy v obvodu neztrácí a lze ji dovést až na výstup. Výsledný vektor tedy je 10111001.



- 3) Celkový počet vektorů, které ovlivňují chyby v obvodu, jsem dospěl k číslu 16. Přebytné vektory, které nenavýšily chybu v obvodu, jsem odstranil.

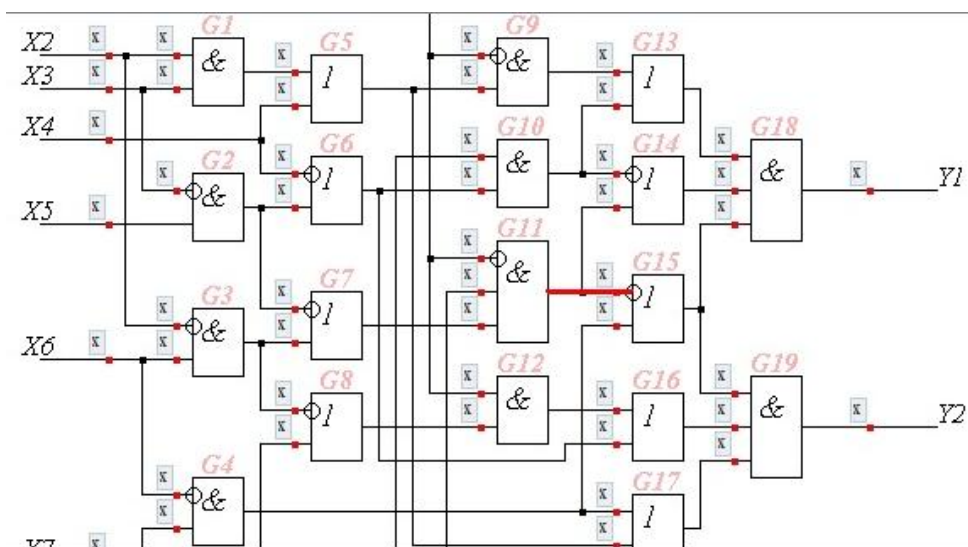
g17>inp2	g13>out	g14>out	g18>inp3	g19>inp1	g16>out	g17>out	g18>out	g19>out	Increase [%]	Coverage [%]
0	1	X	X	0	0	0	1	0	20.0	20.0
0	X	1	X	0	0	0	1	0	16.0	36.0
X	X	X	1	X	X	X	1	1	6.0	42.0
X	X	X	1	X	X	X	1	1	4.0	46.0
X	X	1	X	0	0	0	1	0	4.0	50.0
1	X	1	X	X	X	1	1	1	7.0	57.0
X	X	X	1	1	X	X	1	1	4.0	61.0
X	0	0	0	X	1	X	0	1	15.0	76.0
0	0	0	0	0	0	0	0	0	1.0	77.0
1	1	X	X	X	X	1	1	1	1.0	78.0
X	0	0	0	0	0	0	0	0	5.0	83.0
0	1	X	X	0	0	0	1	0	4.0	87.0
X	0	0	0	X	1	X	0	1	2.0	89.0
X	0	0	0	X	1	X	0	1	1.0	90.0
1	X	1	X	X	X	1	1	1	2.0	92.0
0	1	X	X	0	0	0	1	0	2.0	94.0

- 4) Díky metodě Guided probing jsem odhalil vloženou závadu v obvodu. Ta byla umístěna na obvodu G1 a k jejímu určení jsem potřeboval 11 kontrolních měření pomocí sondy.



- 5) Kombinovaná metoda v tomto případě byla dosti přesná, i když nedošlo ke 100% pokrytí chybami a do obvodu bylo vloženo více chyb, které jsou vyobrazeny v tabulce chyb se díky této metodě podařilo najít vloženou závadu v obvodu. Je označeno celkem 7 vadných vektorů. V jiných případech by závada zůstala neodhalená, ale tady v tomto obvodu se jedná o jednu závadu která ovlivňuje právě těchto 7 vektor.

Fault table											
b2	g13>out	g14>out	g18>inp3	g19>inp1	g16>out	g17>out	g18>out	g19>out	Increase [%]	Coverage [%]	Passed
1	X	X	0	0	0	0	1	0	20.0	20.0	
X	1	X	0	0	0	0	1	0	16.0	36.0	
X	X	1	X	X	X	X	1	1	6.0	42.0	Y
X	X	1	X	X	X	X	1	1	4.0	46.0	Y
X	1	X	0	0	0	0	1	0	4.0	50.0	Y
X	1	X	X	X	X	1	1	1	7.0	57.0	Y
X	X	1	1	X	X	X	1	1	4.0	61.0	Y
0	0	0	X	1	X	0	1	1	15.0	76.0	Y
0	0	0	0	0	0	0	0	0	1.0	77.0	
1	X	X	X	X	1	1	1	1	1.0	78.0	Y
0	0	0	0	0	0	0	0	0	5.0	83.0	Y
1	X	X	0	0	0	0	1	0	4.0	87.0	
0	0	0	X	1	X	0	1	1	2.0	89.0	
0	0	0	X	1	X	0	1	1	1.0	90.0	
X	1	X	X	X	X	1	1	1	2.0	92.0	Y
1	X	X	0	0	0	0	1	0	2.0	94.0	



Ve schématu vidíme, že závada je na výstupu obvodu G11. Tudiž je tento obvod špatný a pro odstranění chyby by bylo zapotřebí tento obvod odstranit.

7.10 Vypracování protokolu č. 10

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 10	Měření na obvodu t1	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu t1 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spusťte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spusťte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spusťte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

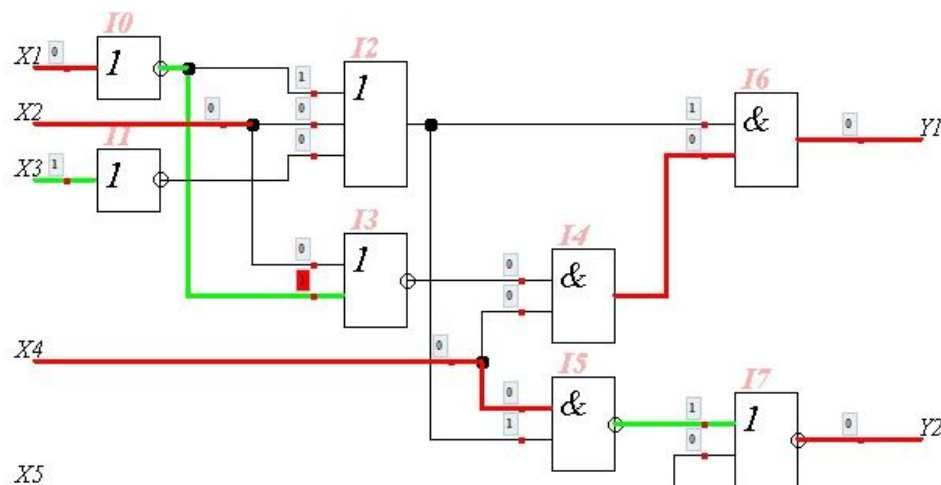
Řešení:

- 1) Obvod t1 má celkem 5 vstupů. Zadávání vektorů jsem provedl pomocí funkce LFSR a nechal jsem vytvořit 16 testovacích vektorů, které stačí na vytvoření 100% pokrytí chybami.

Testvector table																	
#	X1	X2	X3	X4	X5	X2	X4	I0>q	I1>q	I2>q	I3>q	I4>q	I5>q	I6>q	I7>q	Y2	Y1
1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	0	0
2	1	1	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0
3	0	1	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0
4	1	0	1	1	0	0	1	0	0	0	1	1	1	0	0	0	0
5	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0
6	0	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0
7	1	0	0	1	0	0	1	0	1	1	1	1	0	1	1	1	1
8	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0
9	0	1	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0
10	0	0	1	1	0	0	1	1	0	1	0	0	0	0	1	1	0
11	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0
13	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
14	1	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0
15	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0
16	1	1	1	1	0	1	1	0	0	1	0	0	0	0	1	1	0

Fault table															
b	I1>q	I3>a	I3>b	I3>q	I4>b	I5>a	I5>b	I6>a	I4>q	I5>q	X5	I7>q	I6>q	Increase [%]	Coverage [%]
1	X	X	X	X	X	X	X	1	X	X	X	1	1	18.4	18.4
X	0	X	1	X	X	X	X	X	1	X	0	1	1	13.2	31.6
X	X	X	X	X	X	X	X	X	1	X	X	1	1	0.0	31.6
1	X	X	X	X	X	X	1	1	X	0	X	1	1	10.5	42.1
X	X	X	1	X	X	X	X	X	1	X	0	1	1	0.0	42.1
X	X	X	X	X	X	X	X	X	1	X	X	1	1	0.0	42.1
0	1	1	0	0	0	0	0	0	0	1	1	0	0	39.5	81.6
X	X	X	X	X	X	X	X	X	1	X	X	1	1	0.0	81.6
X	X	X	X	X	X	1	X	X	1	0	X	1	1	5.3	86.8
X	X	0	1	X	0	0	X	1	1	1	1	0	1	7.9	94.7
X	X	0	1	X	X	X	X	1	X	0	1	1	1	0.0	94.7
X	X	X	X	X	X	X	X	X	1	X	X	1	1	0.0	94.7
X	X	X	X	X	X	1	X	X	1	0	X	1	1	0.0	94.7
X	X	X	X	1	1	X	X	1	0	X	1	1	1	2.6	97.4
X	X	X	X	X	1	X	X	1	0	X	1	1	1	0.0	97.4
X	0	X	1	X	0	0	X	1	1	1	1	0	1	2.6	100.0

- 2) Náhodně vložená chyba v režimu Generate testvector je vyznačena ve schématu a nachází se na výstupu obvodu I0. Vektor díky, kterému je možné sledovat chybu až na výstup má hodnotu 00100.



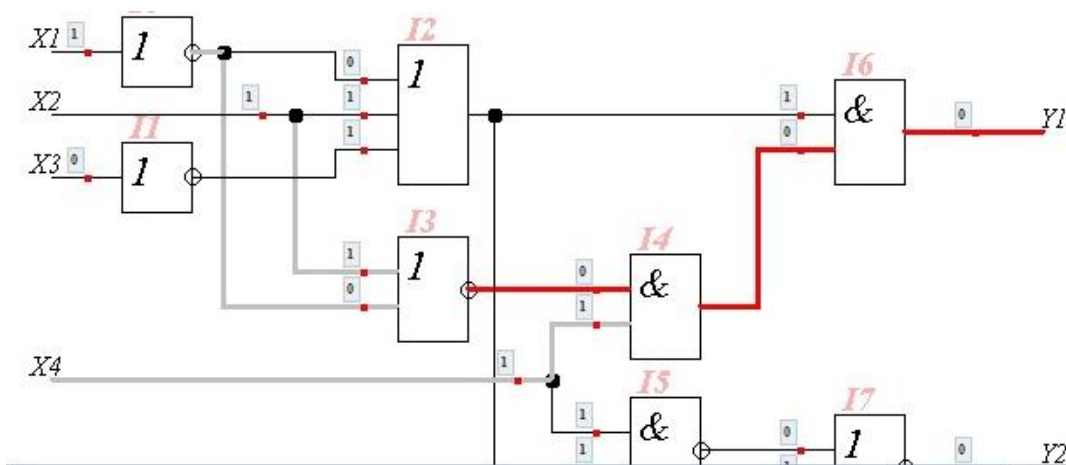
- 3) Po odstranění přebytečných vektorů vypadá výsledná tabulka chyb následovně.

Fault table															
b	I1>q	I3>a	I3>b	I3>q	I4>b	I5>a	I5>b	I6>a	I4>q	I5>q	X5	I7>q	I6>q	Increase [%]	Coverage [%]
1	X	X	X	X	X	X	X	1	X	X	X	1	1	18.4	18.4
X	0	X	1	X	X	X	X	X	1	X	0	1	1	13.2	31.6
1	X	X	X	X	X	X	1	1	X	0	X	1	1	10.5	42.1
0	1	1	0	0	0	0	0	0	0	1	1	0	0	39.5	81.6
X	X	X	X	X	X	1	X	X	1	0	X	1	1	5.3	86.8
X	X	0	1	X	0	0	X	1	1	1	0	1	1	7.9	94.7
X	X	X	X	X	1	1	X	X	1	0	X	1	1	2.6	97.4
X	0	X	1	X	0	0	X	1	1	1	0	1	1	2.6	100.0

- 4) Při vyhledávání chyby pomocí sondy byla do obvodu zanesena náhodná chyba. V tabulce chyb byly vyznačeny poškozené vektory.

Fault table															
I1>q	I3>a	I3>b	I3>q	I4>b	I5>a	I5>b	I6>a	I4>q	I5>q	X5	I7>q	I6>q	Increase [%]	Coverage [%]	Passed
1	X	X	X	X	X	X	1	X	X	X	1	1	18.4	18.4	
X	0	X	1	X	X	X	X	1	X	0	1	1	13.2	31.6	
1	X	X	X	X	X	1	1	X	0	X	1	1	10.5	42.1	
0	1	1	0	0	0	0	0	0	1	1	0	0	39.5	81.6	
X	X	X	X	X	1	X	X	1	0	X	1	1	5.3	86.8	
X	X	0	1	X	0	0	X	1	1	0	1	1	7.9	94.7	
X	X	X	X	1	1	X	X	1	0	X	1	1	2.6	97.4	
X	0	X	1	X	0	0	X	1	1	1	0	1	2.6	100.0	

Abychom zjistili, kde se závada nachází, musíme jít postupně od výstupu až k místu kde se nachází závada. Provedeme měření pomocí sondy na kontrolních bodech.

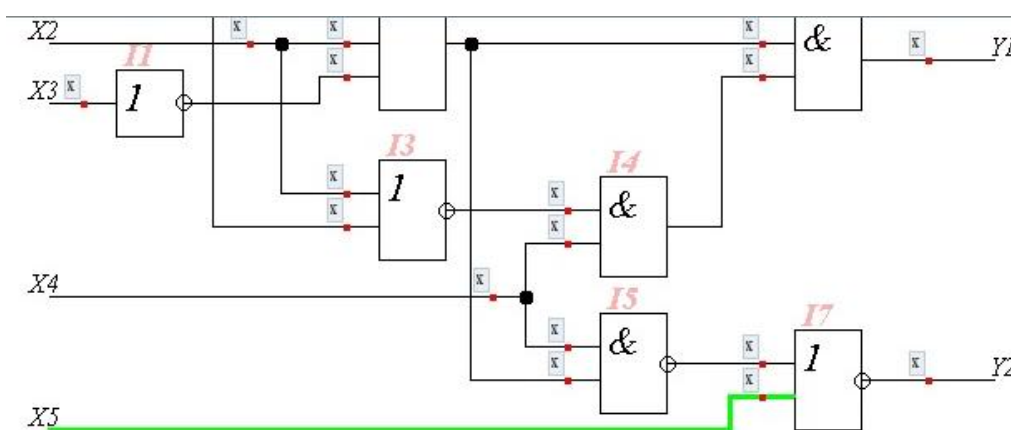


Ve schématu je znázorněn postup kontrolovaných tras. Je vidět kde závada vznikla. Jedná se o výstup obvodu I3. Aby byla chyba odstraněna, museli bychom tento obvod vyměnit.

- 5) V režimu kombinační diagnostiky poruch je vložena do obvodu náhodná chyba. V tabulce chyb je vyobrazený vadný vektor. Jak je vidět jedná se pouze o jeden vadný vektor a proto není potřeba rozšiřovat tabulku o další vektory, aby došlo ke zpřesnění testu.

Fault table																
I1>q	I3>a	I3>b	I3>q	I4>b	I5>a	I5>b	I6>a	I4>q	I5>q	X5	I7>q	I6>q	Increase [%]	Coverage [%]	Passed	
1	X	X	X	X	X	X	1	X	X	X	1	1	18.4	18.4	✓	
X	0	X	1	X	X	X	X	1	X	0	1	1	13.2	31.6	✗	
1	X	X	X	X	X	1	1	X	0	X	1	1	10.5	42.1	✓	
0	1	1	0	0	0	0	0	0	1	1	0	0	39.5	81.6	✓	
X	X	X	X	X	1	X	X	1	0	X	1	1	5.3	86.8	✓	
X	X	0	1	X	0	0	X	1	1	1	0	1	7.9	94.7	✓	
X	X	X	X	1	1	X	X	1	0	X	1	1	2.6	97.4	✓	
X	0	X	1	X	0	0	X	1	1	1	0	1	2.6	100.0	✓	

Výsledné vyobrazení vektoru ve schématu nám ukazuje, že závada by měla být na vstupu X5. Jenomže trasa není označena barvou červené, ale zelené což značí že tento druh testování není vhodný pro odhalení vložené závady do tohoto daného obvodu.



7.11 Vypracování protokolu č. 11

Univerzita Tomáše Bati ve Zlíně			
FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 11	Měření na obvodu d1	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Na obvodu d1 proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

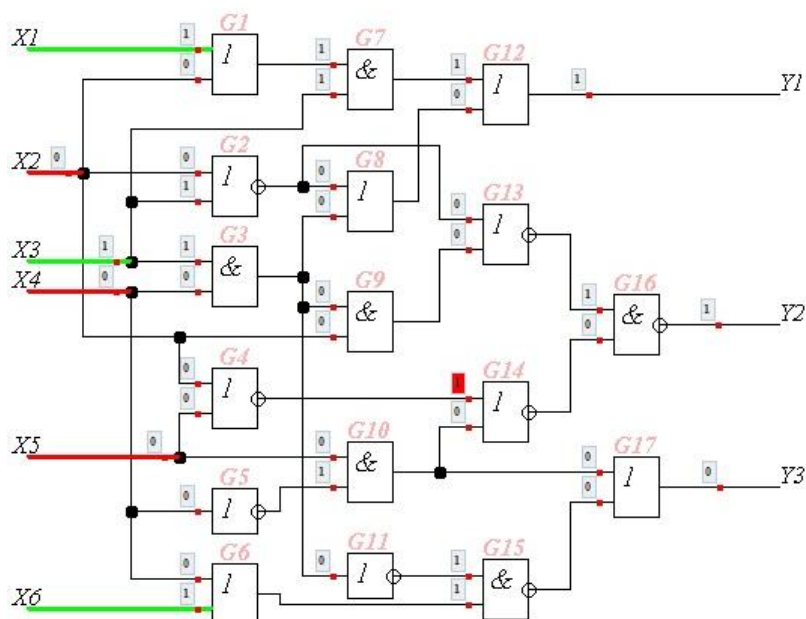
- 1) Tento obvod je dosti složitý. Skládá se z 6 vstupů a pro vkládání vektorů jsem zvolil metodu LFSR. Pro vytvoření co největšího procenta pokrytí jsem použil vytvoření 30 vektorů a dosažená přesnost je 100%.

testvector table																				
#	x1	x2	x3	x4	x5	x6	x2	x3	x4	x5	G1>q	G2>q	G3>q	G4>q	G5>q	G6>q	G7>q	G8>q	G9>q	G11>q
1	1	1	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	1	1	0
2	1	1	1	1	1	0	1	1	1	1	1	0	1	0	0	1	1	1	1	0
3	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0
4	0	0	1	1	1	1	0	1	1	1	0	0	1	0	0	1	0	1	0	0
5	1	0	0	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	0	1
6	1	1	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1
7	1	1	1	0	0	1	1	1	0	0	1	0	0	0	1	1	1	0	0	1
8	0	1	1	1	0	0	1	1	1	0	1	0	1	0	0	1	1	1	1	0
9	1	0	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0
10	1	1	0	1	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1
11	0	1	1	0	1	1	1	1	0	1	1	0	0	0	1	1	1	0	0	1
12	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	1	1	0	0
13	0	1	0	1	1	0	1	0	1	1	1	0	0	0	0	1	0	0	0	1
14	0	0	1	0	1	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1
15	0	0	0	1	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	1
16	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	1
17	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	0	1	0	1
18	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0	1
19	1	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1
20	0	1	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	0	1
21	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0	0
22	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1
23	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	1	0	1
24	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	1	0	0	0	1
25	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1
26	1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1
27	0	1	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1
28	1	0	1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	0	0	1
29	1	1	0	1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1
30	1	1	1	0	1	0	1	1	0	1	1	0	0	0	1	0	1	0	0	1

Fault table												
	G6>q	G4>q	G14>b	G17>a	G15>q	G13>q	G14>q	G17>q	G16>q	G12>q	Increase [%]	Coverage [%]
X	X	X	X	X	0	1	X	0	0	0	17.9	17.9
X	X	X	X	X	0	1	X	0	0	0	1.3	19.2
X	X	X	X	X	0	1	X	0	0	0	0.0	19.2
X	1	1	X	X	0	0	0	0	1	0	20.5	39.7
0	X	X	1	1	1	1	X	1	0	0	15.4	55.1
X	X	0	0	X	X	X	1	0	0	1	17.9	73.1
0	1	1	1	1	0	0	0	1	1	0	11.5	84.6
X	X	X	X	X	0	1	X	0	0	0	0.0	84.6
X	1	1	X	X	0	0	0	0	1	0	0.0	84.6
0	1	1	1	1	1	0	0	1	1	1	0.0	84.6
X	X	0	0	X	X	X	1	0	0	0	1.3	85.9
X	0	X	X	X	0	X	1	0	0	0	3.8	89.7
0	1	1	1	1	1	0	0	1	1	1	1.3	91.0
X	X	0	0	X	X	X	1	0	0	1	3.8	94.9
0	X	X	1	1	1	X	X	1	0	0	0.0	94.9
X	X	X	X	X	X	X	X	0	0	0	0.0	94.9
0	X	X	1	1	1	X	X	1	0	0	0.0	94.9
1	X	X	X	X	0	X	X	0	0	0	3.8	98.7
1	1	1	1	X	0	0	0	0	1	1	0.0	98.7
1	1	1	1	X	0	0	0	0	1	0	0.0	98.7
X	0	X	X	X	0	X	1	0	0	0	0.0	98.7
0	X	X	1	1	1	1	X	1	0	0	0.0	98.7
X	X	X	0	X	X	X	X	0	0	0	0.0	98.7
0	1	1	1	1	1	0	0	1	1	1	0.0	98.7
1	0	X	X	X	0	X	1	0	0	1	0.0	98.7
0	X	X	1	1	1	X	X	1	0	0	0.0	98.7
X	X	0	X	X	X	X	1	0	0	1	0.0	98.7
0	0	X	1	1	1	X	1	1	0	0	1.3	100.0
0	1	1	1	1	1	0	0	1	1	1	0.0	100.0
X	X	0	X	X	X	X	1	0	0	0	0.0	100.0

- 2) Vložená chyba v režimu Generate testvector je na výstupu obvodu G4. Problém nastal s posunem poruchy v obvodu. Tato porucha se nedá v obvodu posouvat a

zůstává jen na jednom místě. I po několika různých pokusech a změnách hodnot se mi nepodařilo chybu v obvodu rozšířit. Výsledný vektor tedy je 101001.

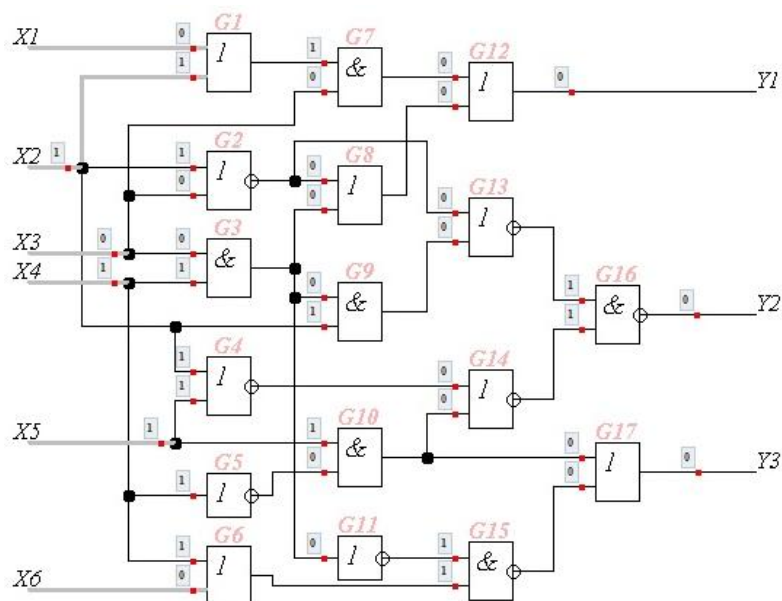


- 3) Po zjednodušení se celkový počet použitých vektorů snížil na 12.

Fault table													
>q	G6>q	G4>q	G14>b	G17>a	G15>q	G13>q	G14>q	G17>q	G16>q	G12>q	Increase [%]	Coverage [%]	
X	X	X	X	X	0	1	X	0	0	0	17.9	17.9	
X	X	X	X	X	0	1	X	0	0	0	1.3	19.2	
X	1	1	X	0	0	0	0	1	0	0	20.5	39.7	
0	X	X	1	1	1	X	1	0	0	0	15.4	55.1	
X	X	0	0	X	X	1	0	0	0	1	17.9	73.1	
0	1	1	1	1	1	0	0	1	1	0	11.5	84.6	
X	X	0	0	X	X	1	0	0	0	0	1.3	85.9	
X	0	X	X	0	X	1	0	0	0	0	3.8	89.7	
0	1	1	1	1	0	0	1	1	1	1	1.3	91.0	
X	X	0	0	X	X	1	0	0	0	1	3.8	94.9	
1	X	X	X	0	X	X	0	0	0	0	3.8	98.7	
0	0	X	1	1	X	1	1	0	0	0	1.3	100.0	

- 4) Metoda vyhledávání chyb pomocí sondy. Je v tomto obvodu neúčinná. Obvod je natolik složitý, že se mi nepodařilo danou chybu nalézt a využil jsem k prověření všech 36 měřících bodů. Tabulka chyb nám zobrazuje jasně chybné vektory, ale zpětné vyhledání pomocí sond je zde neúčinné. Metoda Guided probing tedy selhala.

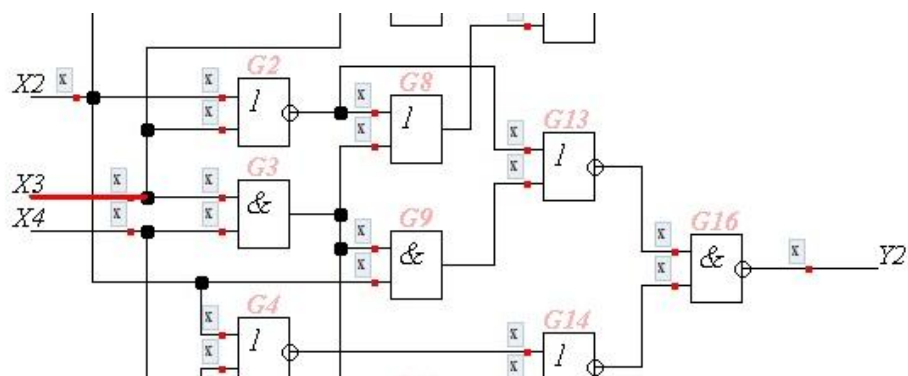
Fault table													
>q	G6>q	G4>q	G14>b	G17>a	G15>q	G13>q	G14>q	G17>q	G16>q	G12>q	Increase [%]	Coverage [%]	Pass
X	0	X	X	0	X	1	0	0	0	0	3.8	89.7	✓
0	1	1	1	1	0	0	1	1	1	1	1.3	91.0	✓
X	X	0	0	X	X	1	0	0	0	1	3.8	94.9	✓
1	X	X	X	0	X	X	0	0	0	0	3.8	98.7	✓
0	0	X	1	1	X	1	1	0	0	0	1.3	100.0	✓



- 5) Metoda Combinatorial fault detection je pro tento obvod daleko vhodnější a náhodně vloženou chybu dokázala odhalit, aniž by bylo nutné přidávat další testovací vektory na zpřesnění testu.

Fault table													
j>q	G6>q	G4>q	G14>b	G17>a	G15>q	G13>q	G14>q	G17>q	G16>q	G12>q	Increase [%]	Coverage [%]	Passed
X	X	X	X	X	0	1	X	0	0	0	17.9	17.9	<div></div>
X	X	X	X	X	0	1	X	0	0	0	1.3	19.2	<div></div>
X	1	1	X	0	0	0	0	0	1	0	20.5	39.7	<div></div>
0	X	X	1	1	1	1	X	1	0	0	15.4	55.1	<div></div>
X	X	0	0	X	X	X	1	0	0	1	17.9	73.1	<div></div>
0	1	1	1	1	1	0	0	1	1	0	11.5	84.6	<div></div>
X	X	0	0	X	X	X	1	0	0	0	1.3	85.9	<div></div>
X	0	X	X	0	X	X	1	0	0	0	3.8	89.7	<div></div>
0	1	1	1	1	1	0	0	1	1	1	1.3	91.0	<div></div>
X	X	0	0	X	X	X	1	0	0	1	3.8	94.9	<div></div>
1	X	X	X	0	X	X	X	0	0	0	3.8	98.7	<div></div>
0	0	X	1	1	1	X	1	1	0	0	1.3	100.0	<div></div>

Výsledná závada byla způsobena na vstupu X3. Zobrazení této závady představuje níže přiložené schéma obvodu s vyznačenou závadou.



7.12 Vypracování protokolu č. 12

Univerzita Tomáše Bati ve Zlíně FAKULTA APLIKOVANÉ INFORMATIKY			
Jméno:		Ročník:	
Předmět:	Diagnostika číslicových systémů	Skupina:	
		Naměřeno:	
Protokol č. 12	Měření na d obvodech	Odevzdáno:	
		Hodnocení:	

Úkol:

- 1) Vyberte si nějaký z d obvodů a proveďte vložení testovacích vektorů tak, aby bylo vytvořeno 100% pokrytí chybami. Použijte buď metodu Manual nebo LFSR. Vektory vložte do protokolu.
- 2) Spustěte Generate testvectors a vložte do obvodu náhodnou chybu. Snažte se danou chybu dovést až na výstup obvodu. Sestavený vektor vypište a přidejte schéma.
- 3) Odstraňte vektory, které nenavýšují chybu pokrytí.
- 4) Spustěte Guided probing a vložte do obvodu náhodnou chybu. Pomocí sondy určete místo, kde nastala porucha. Snažte se dosáhnout co nejmenšího počtu měření sondou. Schéma s nalezenou chybou a tabulku chyb vložte do protokolu.
- 5) Spustěte Combinatorial fault detection a nechejte vložit náhodnou chybu do obvodu. Vyhledejte náhodně vloženou chybu, případně přidejte další testovací vektor na vyšší pokrytí a zpřesnění testu. Tabulku chyb a obvod vložte do protokolu.

Řešení:

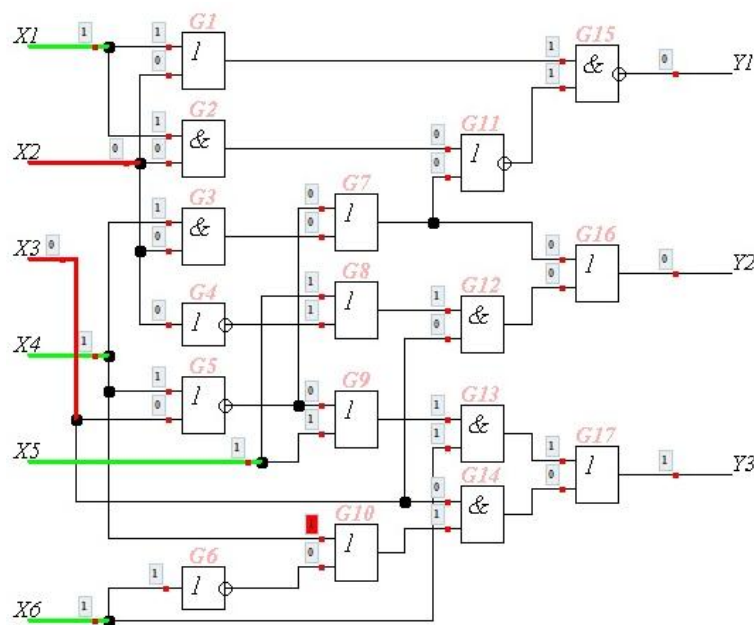
- 1) Zvolil jsem si obvod d2. Nechal jsem vytvořit 20 testovacích vektorů v režimu LFSR. Obvod má 6 vstupů a jeho zapojení je dosti komplikované a proto je vhodnější použít metodu LFSR než metodu manuálního vkládání.

Testvector table																					
#	x1	x2	x3	x4	x5	x6	x1	x2	x3	x4	x5	x6	G1>q	G2>q	G3>q	G4>q	G5>q	G6>q	G7>q	G8>q	G9>q
1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	0	1	0	0
2	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1
4	0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	1	1
5	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	0	0	0	1	1
6	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	1
7	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0
8	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	0
9	1	0	1	1	1	0	1	0	1	1	1	0	1	0	0	1	0	1	0	1	1
10	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1
11	0	1	1	0	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	1
12	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0	1	0	0	0	1	0
13	0	1	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1	1	1
14	0	0	1	0	1	1	0	0	1	0	1	1	0	0	0	1	0	0	0	1	1
15	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0
16	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1
17	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
18	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	1	1
19	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	1	0	1
20	0	1	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0

Těchto 20 vektorů stačí na vytvoření 100% pokrytí chybami v tabulce chyb.

Fault table												
i	G11>q	G14>a	G10>q	G13>q	G14>q	G16>a	G12>q	G17>q	G16>q	G15>q	Increase [%]	Coverage [%]
1	0	0	X	0	0	X	0	0	0	0	18.3	18.3
1	0	0	X	0	0	X	X	0	0	0	0.0	18.3
1	X	X	X	X	X	X	X	0	0	0	1.2	19.5
X	X	X	X	X	X	X	0	0	0	0	8.5	28.0
0	X	X	0	X	1	1	0	1	1	1	29.3	57.3
1	X	X	0	X	0	X	0	0	0	0	3.7	61.0
1	X	1	1	1	1	1	1	1	1	0	25.6	86.6
1	0	0	X	0	0	X	0	0	0	0	0.0	86.6
0	0	0	X	0	0	X	0	0	0	1	0.0	86.6
1	X	X	0	X	0	X	0	0	0	0	0.0	86.6
0	X	X	0	X	X	X	0	0	0	1	3.7	90.2
0	0	0	X	0	0	X	0	0	0	1	2.4	92.7
1	1	X	1	1	0	X	1	0	0	0	3.7	96.3
X	X	X	0	X	X	0	0	0	0	0	0.0	96.3
X	1	X	1	1	1	1	1	1	1	0	0.0	96.3
X	1	X	1	1	0	X	1	0	0	0	0.0	96.3
X	X	X	0	X	0	X	0	0	0	0	1.2	97.6
1	1	X	1	1	0	X	1	0	0	0	0.0	97.6
1	1	X	1	1	0	X	1	0	0	0	0.0	97.6
0	0	0	X	0	1	1	0	1	1	1	2.4	100.0

- 2) Funkcí Generate testvector jsem nechal do obvodu vložit chybu. Chyba byla umístěna na vedlejší větev vstupu X4. I po několika různých druzích kombinace vektorů se mi nepodařilo rozšířit tuto chybu dále do obvodu natož na výstup. Obvod se příliš složitý a tuto chybu se mi nepodařilo dostat dále než z původního místa.



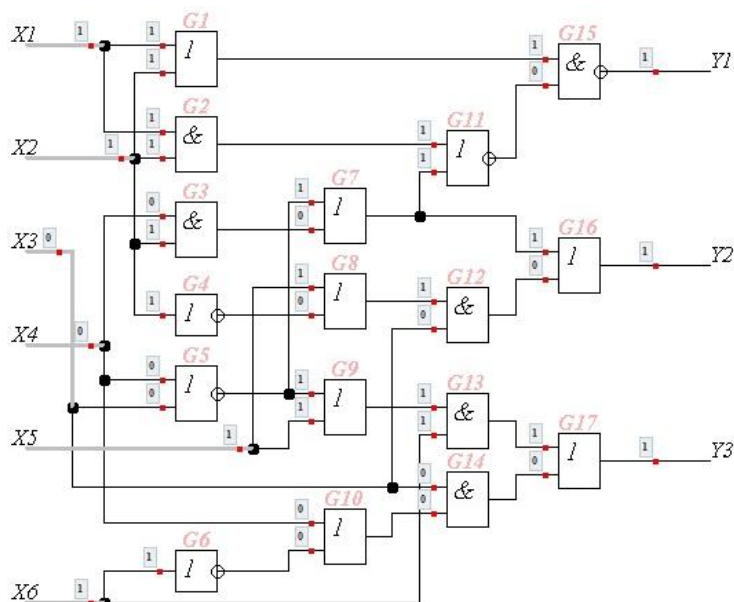
- 3) Po odstranění přebytečných vektorů, které nenavýšují chybu v obvodu, se mi podařilo celkový počet vložených vektorů zredukovat z 20 na 11.

Fault table												
i	G11>q	G14>a	G10>q	G13>q	G14>q	G16>a	G12>q	G17>q	G16>q	G15>q	Increase [%]	Coverage [%]
1	0	0	X	0	0	X	0	0	0	0	18.3	18.3
1	X	X	X	X	X	X	X	0	0	0	1.2	19.5
X	X	X	X	X	X	X	0	0	0	0	8.5	28.0
0	X	X	0	X	1	1	0	1	1	1	29.3	57.3
1	X	X	0	X	0	X	0	0	0	0	3.7	61.0
1	X	1	1	1	1	1	1	1	1	0	25.6	86.6
0	X	X	0	X	X	0	0	0	0	1	3.7	90.2
0	0	0	X	0	X	0	0	0	0	1	2.4	92.7
1	1	X	1	1	0	X	1	0	0	0	3.7	96.3
X	X	X	0	X	0	X	0	0	0	0	1.2	97.6
0	0	0	X	0	1	1	0	1	1	1	2.4	100.0

- 4) Metodou Guided probing jsem nechal do obvodu zavést náhodnou chybu. Její přítomnost představuje tabulka chyb, která obsahuje vadný vektor.

Fault table													
i	G11>q	G14>a	G10>q	G13>q	G14>q	G16>a	G12>q	G17>q	G16>q	G15>q	Increase [%]	Coverage [%]	Passed
1	0	0	X	0	0	X	0	0	0	0	18.3	18.3	Y
1	X	X	X	X	X	X	X	0	0	0	1.2	19.5	Y
X	X	X	X	X	X	X	0	0	0	0	8.5	28.0	Y
0	X	X	0	X	1	1	0	1	1	1	29.3	57.3	Y
1	X	X	0	X	0	X	0	0	0	0	3.7	61.0	N
1	X	1	1	1	1	1	1	1	1	0	25.6	86.6	Y
0	X	X	0	X	X	X	0	0	0	1	3.7	90.2	Y
0	0	0	X	0	X	0	0	0	0	1	2.4	92.7	Y
1	1	X	1	1	0	X	1	0	0	0	3.7	96.3	Y
X	X	X	0	X	0	X	0	0	0	0	1.2	97.6	N
0	0	0	X	0	1	1	0	1	1	1	2.4	100.0	Y

Následně jsem se pokoušel zjistit, kde v obvodu se daná chyba nalézá pomocí měření kontrolních bodů. Ale tato metoda byla neúčinná a závadu se mi nepodařilo nalézt.



Využil jsem ke kontrole všech 34 měřených míst v tomto obvodu, ale závadu se mi nepodařilo metodou Guided probing nalézt.

- 5) Použil jsem tedy metodu Combinatorial fault detection, ale ani tato metoda nevedla k nalezené náhodně vložené závadě. I když odhalila vadné vektory v tabulce chyb.

Fault table													
t	G11>q	G14>a	G10>q	G13>q	G14>q	G16>a	G12>q	G17>q	G16>q	G15>q	Increase [%]	Coverage [%]	Passed
1	0	0	X	0	0	X	0	0	0	0	18.3	18.3	
1	X	X	X	X	X	X	0	0	0	0	1.2	19.5	Y
X	X	X	X	X	X	X	0	0	0	0	8.5	28.0	Y
0	X	X	0	X	1	1	0	1	1	1	29.3	57.3	Y
1	X	X	0	X	0	X	0	0	0	0	3.7	61.0	Y
1	X	1	1	1	1	1	1	1	1	0	25.6	86.6	Y
0	X	X	0	X	X	X	0	0	0	1	3.7	90.2	Y
0	0	0	X	0	X	0	0	0	0	1	2.4	92.7	
1	1	X	1	1	0	X	1	0	0	0	3.7	96.3	Y
X	X	X	0	X	0	X	0	0	0	0	1.2	97.6	Y
0	0	0	X	0	1	1	0	1	1	1	2.4	100.0	
1	0	0	X	0	0	X	0	0	0	0	0.0	100.0	
1	X	X	X	X	X	X	X	0	0	0	0.0	100.0	Y
1	0	0	X	0	X	X	0	0	0	0	0.0	100.0	
1	1	X	1	1	0	X	1	0	0	0	0.0	100.0	Y

Na nalezení chyby v tomto obvodu bychom museli použít jiné metody, které jsou přesnější ale také časově náročnější. Obě metody na vyhledávání chyb Guided probing a Combinatorial fault detection selhaly.

ZÁVĚR

Cílem diplomové práce bylo vypracovat laboratorní úlohy pro implementaci software na test generation a fault diagnosis. Základem bylo nejen důkladné seznámení s programem Diagnostic tester, ale seznámení se s definicemi a problematikami spojenými s identifikací poruch v integrovaných obvodech.

Práce seznamuje s problematikou poruch v integrovaných obvodech. Princip vzniku chyb během výrobního procesu. Vyhledávání závad v elektrických zařízeních. Různé druhy metod potřebné pro vyhledání závady.

Dále práce popisuje principy modelů a simulací během kterých se zkoumají uměle vyvolané chyby v obvodech. Test generation, který zajišťuje vytváření náhodných chyb do předem připravených modelů. Poruchy diagnostik, kdy nejsme schopni předem určit možnou vzniklou poruchu a následně zvolit vhodnou a především efektivní metodu, která nám pomůže chybu odhalit.

Práce obsahuje návod pro simulační program, který dokáže na předem definovaný obvod vložit závadu a následně pomocí různých druhů diagnostických metod tuto závadu odhalit. Tento návod společně s vytvořenými protokoly bude sloužit jako výukový materiál pro výuku studentů, aby pochopili princip odhalování takto vzniklých závad.

Elektronika nás obklopuje každý den. Neustále dochází k technologickému vývoji a ke vzniku nových poruch. S novou technologií vznikají i nové závady, které se musejí rychle odstranit a proto je nutné takovéto závady umět identifikovat a snažit se jim předcházet. Technologický vývoj se nezastaví a bezpečnostní průmysl není výjimkou. Proto je nutné znát i závady spojené s instalací nových bezpečnostních zařízení a snažit se předcházet zbytečným chybám.

ZÁVĚR V ANGLIČTINĚ

The objective of this thesis was to develop laboratory tasks for the implementation of software to test generation and fault diagnosis. The basis was not only thoroughly familiar with the program Diagnostic tester, but familiarity with the definitions and issues related to identification of defects in integrated circuits.

This work introduces the issue of failures in integrated circuits. Principle of error rise during the manufacturing process. Finding faults in electrical equipment. Various types of methods needed to find defects.

Further describes the principles of modeling and simulation during which examined artificially induced error in the circuits. Test generation which ensures a random error in the pre-models. Disorders of diagnostics where we can not predict the potential of failure and select an appropriate and effective method above which will help us detect the error.

The work contains instructions for the simulation program that is able to insert predefined circuit fault and the using various types of diagnostic methods to detect the fault. This guide, together with established protocols will be used as teaching material for learning students to understand the principle of detecting faults arising as follows.

Electronics encompass us every day. Constantly these are technological developments and the emergence of new faults. With the new technology creates new faults which must be removed quickly and therefore must be able to identify such defects and try to prevent them. Technological development is not stop and the security industry is no exception. Therefore it is necessary to know the faults associated with installing new security equipment and try to avoid unnecessary mistakes.

SEZNAM POUŽITÉ LITERATURY

- [12] ANTOŠOVÁ, Marcela; DAVÍDEK, Vratislav. *Číslicová technika*. Vyd. 1. České Budějovice: KOOP, 2003. 288 s. ISBN 80-7232-206-0.
- [2] *Wikipedia : The Free Encyclopedia* [online]. 2001 [cit. 2011-03-07]. Diagnosis. Dostupné z WWW: <<http://en.wikipedia.org/wiki/Diagnosis>>.
- [3] *Introduction to Design for Testability* [online]. 2004 [cit. 2011-04-21]. DILDIS. Dostupné z WWW: <<http://www.pld.ttu.ee/diagnostika/introduction.html>>.
- [4] *Introduction to Design for Testability* [online]. 2004 [cit. 2011-04-21]. DILDIS. Dostupné z WWW: <<http://www.pld.ttu.ee/diagnostika/theory/faultdiagnosis.html#31>>.
- [5] NEUMANN, P. *Diagnostika číslicových obvodů*. (přednáška) Zlín: Univerzita Tomáše Bati ve Zlíně, 2010.
- [6] *Introduction to Design for Testability* [online]. 2004 [cit. 2011-04-21]. DILDIS. Dostupné z WWW: <<http://www.pld.ttu.ee/diagnostika/theory/fault.html>>.
- [7] *Introduction to Design for Testability* [online]. 2004 [cit. 2011-04-21]. DILDIS. Dostupné z WWW: <<http://www.pld.ttu.ee/diagnostika/theory/testgeneration.html>>.
- [8] *Applet on Basics of Test and Diagnostics* [online]. 2004 [cit. 2011-04-22]. Java Applets. Dostupné z WWW: <<http://www.pld.ttu.ee/applets/td/>>.
- [9] *Applet on Basics of Test and Diagnostics* [online]. 2004 [cit. 2011-04-22]. Java Applets. Dostupné z WWW: <http://www.pld.ttu.ee/applets/td/td_doc.html>.
- [10] *Applet on Basics of Test and Diagnostics* [online]. 2004 [cit. 2011-04-22]. Java Applets. Dostupné z WWW: <http://www.pld.ttu.ee/applets/td/td_exercises.html>.
- [11] UBAR, R., WUTTKE, H.-D. *Action Based Learning System for Teaching Digital Electronics and Test*. Proc. Of 3rd European Workshop on Microelectronics Education, Aix-en-Provence, France, 2000, p.65-66, ISBN 978-0-7923-6456-6.
- [12] UBAR, R., ORASSON, E., EVARTSON, T. *Java Applet for Self-Learning of Digital Test Issues*. 13th EAEEIE Conference, York, Great Britain, 2002, ISBN 1-85911-008-8.
- [13] DAVEDZE, S., JUTMAN, A., SUDNITSON, A., UBAR, R. *Web-based training systém for teaching basics of RT-level Digital Design, Test, and Design for Test*, in Proc. Of 9th International Conference of Mixed Design of Integrated Circuits and Systems (MIXDES 2002), Wroclaw, Poland, June 20-22, 2002, pp.699-704

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UUT	Unit Under Test
FEF	Functionally Equivalent Faults
RU	Replaceable Units
FD	Fault Dictionary
FN	Fault Nodes
TN	Test Nodes
P	Passed
F	Failed
OUT	Výstup
IN	Vstup
Sa-0	Stuck at 0
Sa-1	Stuck at 1
SSF	Single Stuck Fault
MSF	Multiple Stuck Fault
TG	Test Generation
PO	Primary Output
LSFR	Linear Feedback Shift Register
ATPG	Automated Test Pattern Generator
BILBO	Built In Logick Block Observer
CSTP	Circular Self Test Path

SEZNAM OBRÁZKŮ

<i>Obr. 1 Příklad tabulkových poruch [4]</i>	20
<i>Obr. 2 Příklad poruchy adresáře [4].....</i>	20
<i>Obr. 3 Minimalizace diagnostických dat [4]</i>	21
<i>Obr. 4 Místo poruchy v edge-pin testování [4]</i>	22
<i>Obr. 5 Odlišení závad [4]</i>	23
<i>Obr. 6 Snížení počtu sondování [4]</i>	25
<i>Obr. 7 Diagnostický strom [4].....</i>	25
<i>Obr. 8 Porucha v obvodu [6].....</i>	31
<i>Obr. 9 Nejistitelná porucha [6].....</i>	32
<i>Obr. 10 Porucha maskování [6]</i>	36
<i>Obr. 11 Kruhové poruchy maskování [6]</i>	37
<i>Obr. 12 Chyba aktivace [7]</i>	39
<i>Obr. 13 Porucha množení [7].....</i>	39
<i>Obr. 14 Odůvodnění řádku [7]</i>	40
<i>Obr. 15 Aktivace poruchy [8]</i>	43
<i>Obr. 16 Odůvodnění přidělených hodnot [8]</i>	43
<i>Obr. 17 Tabulka chyb [8]</i>	44
<i>Obr. 18 Model edge-pin testování [8]</i>	45
<i>Obr. 19 Panel vložení vektoru [9]</i>	46
<i>Obr. 20 Panel pro schéma [9]</i>	47
<i>Obr. 21 Zobrazení průběhu dat [9]</i>	48
<i>Obr. 22 Ukázka schématu v Diagnostic tester [9].....</i>	50
<i>Obr. 23 Výběr obvodu.....</i>	52
<i>Obr. 24 Výběr LSFR BILBO.....</i>	53
<i>Obr. 25 Volba vstupů</i>	53
<i>Obr. 26 Volba počátečních stavů.....</i>	54
<i>Obr. 27 Vložení vektorů a spuštění</i>	54
<i>Obr. 28 Volba Fault simulator</i>	55
<i>Obr. 29 Tabulka chyb s vyobrazeným pokrytím</i>	55
<i>Obr. 30 Vytvoření 100% krytí testu</i>	56
<i>Obr. 31 Výběr manuálního vkládání vektorů.....</i>	57
<i>Obr. 32 Příklad testovaného obvodu [10]</i>	59

<i>Obr. 33 Tabulka testovaných vektorů [10]</i>	59
<i>Obr. 34 Tabulka chyb [10]</i>	60
<i>Obr. 35 Ukázka schématu v módu Generate testvectors [10]</i>	61
<i>Obr. 36 Tabulka chyb úpravou a po úpravě [10]</i>	61
<i>Obr. 37 Technika zjištění chyb pomocí sond [10]</i>	62
<i>Obr. 38 Fault table [10]</i>	63
<i>Obr. 39 Obvod odpovídající faul table [10]</i>	63
<i>Obr. 40 Nově vložený vektor do fault table [10]</i>	64
<i>Obr. 41 Obvod zobrazující náhodně vloženou chybu [10]</i>	64

SEZNAM TABULEK

<i>Tab. 1 Přehled nejčastěji používaných čísel v číselných soustavách [1]</i>	13
<i>Tab. 2 Logické funkce jedné nezávislé proměnné [1]</i>	14
<i>Tab. 3 Logické funkce dvou vstupních proměnných [1]</i>	15
<i>Tab. 4 Logické funkce dvou vstupních proměnných [1]</i>	17
<i>Tab. 5 Přehled pravidel zjednodušení [6]</i>	33

SEZNAM PŘÍLOH

1x CD – Diplomová práce ve formátu „pdf“

– Diagnostic tester ve formátu „vbs“

– Webový odkaz na Diagnostic tester ve formátu „txt“