

Josef Strnadel

Analýza a zlepšení testovatelnosti číslicového obvodu na úrovni meziregistrových přenosů

Disertační práce

Fakulta informačních technologií
Vysoké učení technické v Brně
Brno, Česká republika

Ediční rada Fakulty informačních technologií Vysokého učení technického v Brně (FIT VUT v Brně):

doc. Ing. Jaroslav Zendulka, CSc.
Ústav informačních systémů FIT VUT v Brně
předseda

prof. Ing. Tomáš Hruška, CSc.
Ústav informačních systémů FIT VUT v Brně
člen

Ing. Adam Herout, Ph.D.
Ústav počítačové grafiky a multimédií FIT VUT v Brně
člen

Ing. Radek Kočí, Ph.D.
Ústav inteligentních systémů FIT VUT v Brně
člen

prof. RNDr. Alexander Meduna, CSc.
Ústav informačních systémů FIT VUT v Brně
člen

doc. Ing. Lukáš Sekanina, Ph.D.
Ústav počítačových systémů FIT VUT v Brně
člen

Mgr. Barbora Selingerová
knihovna FIT VUT v Brně
člen

© 2008 Fakulta informačních technologií, Vysoké učení technické v Brně
Disertační práce

Obálku navrhla Dagmar Hejduková

Vydala Fakulta informačních technologií, Vysoké učení technické v Brně,
Brno, Česká republika

Tisk MJ servis, spol. s r.o.

ISBN 978-80-214-3599-5

Poděkování

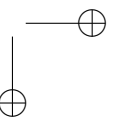
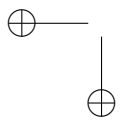
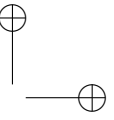
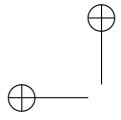
Tato publikace vznikla za podpory projektu CEZ MSM 0021630528 *Výzkum informačních technologií z hlediska bezpečnosti*.

Činnosti související s touto prací však probíhaly po několik let (zpočátku na ÚIVT FEI VUT v Brně, později na ÚPSY FIT VUT v Brně) a byly zahájeny a v různých etapách dokončeny za finanční podpory v současné době již ukončených projektů GAČR GA102/01/1531 *Formální přístupy v diagnostice číslicových obvodů - Verifikace testovatelného návrhu*, FRVŠ FR1754/2002/G1 *Evoluční přístupy pro zvýšení testovatelnosti číslicových obvodů*, GAČR GA102/04/0737 *Moderní metody syntézy číslicových systémů* a GAČR GP102/05/P193 *Optimalizační postupy v diagnostice číslicových systémů*.

Dále bych na tomto místě rád poděkoval všem, kteří mě jakkoli podnítili a ovlivnili ve výzkumu týkajícího se tématu mé disertační práce a také těm, kteří mi v rámci řešení svých vědecko-výzkumných či studijních aktivit byli jakkoli nápomocni s ověřováním principů publikovaných v této práci. Zejména bych rád poděkoval svému školiteli doc. Ing. Zdeňku Kotáskovi, CSc. za příkladné odborné vedení v oblasti mého doktorského studia a to především za cenné rady a připomínky, které mi k tématu mé disertační práce poskytl. Dále bych rád poděkoval zejména Ing. Tomáši Pečenkovi, Ph.D., Ing. Richardu Růžičkovi, Ph.D. a doc. Ing. Lukáši Sekaninovi, Ph.D.

V neposlední řadě bych rád poděkoval všem blízkým za to, že byli ochotni obětovat společný čas pro dokončení této práce; také děkuji za jejich trpělivost a podporu při dokončování této práce. Díky patří zejména mé manželce synovi, rodičům a sourozencům.

Josef Strnadel, autor
Brno, březen 2008



Předmluva

Hlavním předmětem této práce je analýza testovatelnosti číslicového obvodu popsaného na úrovni meziregistrových přenosů a využití jejích výsledků ve vybraných oblastech souvisejících s diagnostikou číslicových obvodů. Práce předpokládá, že obvod popsaný na úrovni meziregistrových přenosů je tvořen dvěma částmi - strukturálním popisem obvodových datových cest a řadičem ovládajícím tok dat těmito cestami - a že pro směrování toku dat obvodem je zvolena propojovací strategie multiplexovaných datových cest. Další propojovací strategie mohou být předmětem dalšího výzkumu. Tato práce se zaměřuje pouze na problematiku související s testovatelností obvodových datových cest, ale řadičem se blíže nezabývá. V části zabývající se přehledem současného stavu v oblasti analýzy testovatelnosti práce shrnuje hlavní nedostatky existujících metod analýzy testovatelnosti. V práci je ukázáno, že většinu ze zmíněných nedostatků lze odstranit, je-li každý prvek uložený v knihovně obvodových prvků kromě informací týkajících se návrhového popisu jeho rozhraní, činnosti atp. vybaven také informacemi usnadňujícími jak jeho diagnostiku, tak také diagnostiku systému, jehož je tento prvek součástí. V této práci je tato informace popsána pomocí prostředků zavedeného matematického modelu. Tento model je rozšířením výchozího modelu, dříve publikovaného jiným autorem, o nové prostředky nezbytné pro popis a modelování obecnější koncepce tzv. transparentních cest, než jakými jsou příliš přísné a často používané koncepce, obvykle založené na tzv. koncepci I-cest. Výhodou rozšířeného modelu je možnost modelování a analýzy většího počtu obvodových datových cest, tj. i těch cest, které jsou z pohledu přísnějších koncepcí pro účely přenosu diagnostických dat nepoužitelné, přestože ve skutečnosti použitelné jsou. Hlavním cílem rozšířeného modelu je poskytnout prostředky potřebné pro formální popis vztahů a dílčích algoritmů pro analýzu testovatelnosti. Navržená metoda analýzy testovatelnosti je založena na číselném ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti vnitřních částí obvodu a obvodu jako celku a jelikož není svázána s žádnou z technik návrhu pro snadnou testovatelnost, lze její výsledky považovat za univerzálně použitelné při řešení problémů souvisejících s testovatelností číslicových ob-

VI

vodů popsaných na úrovni meziregistrových přenosů. Metoda není popsána pomocí v současnosti nejednotných pojmů z oblasti diagnostiky, ale pomocí modelem přesně definovaných prostředků. Za jeden z hlavních přínosů této práce je možno považovat transformaci problému analýzy testovatelnosti na matematický problém. Tím je problém prohledávání dvou orientovaných grafů zkonstruovaných na základě modelu daného obvodu, a to grafu datového toku testovacích vzorků a grafu datového toku odezev. Výhody transformace řešeného problému na problém matematický jsou zřejmé - matematicky popsaný problém je jednoznačně definován, k jeho řešení je možné využít řadu již existujících principů a algoritmů a v neposlední řadě je možno poměrně snadno dokázat významné vlastnosti algoritmu řešícího tento problém. Důkazům významných vlastností navržené metody je věnován prostor v závěrečné části práce, kde jsou také výsledky dosažené touto metodou shrnuty a srovnány s výsledky existujících metod. Z výsledků vyplývá, že díky detailnější analýze obvodových datových cest poskytuje metoda založená na rozšířené koncepci transparentních datových cest přesnější informaci o testovatelnosti daného obvodu, než jakou poskytují metody řešící stejný problém na téže úrovni abstrakce.

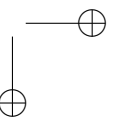
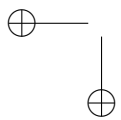
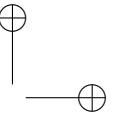
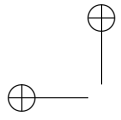
Obsah

| | | |
|----------|---|----|
| 1 | Úvod | 1 |
| 1.1 | Motivace k výzkumu | 3 |
| 1.2 | Cíle výzkumu | 4 |
| 1.3 | Struktura disertační práce | 5 |
| 2 | Související oblasti | 7 |
| 2.1 | Modelování číslicových systémů | 7 |
| 2.1.1 | Klasifikace modelů | 7 |
| 2.1.2 | Úrovně popisu | 8 |
| 2.2 | Návrh číslicových systémů | 9 |
| 2.2.1 | Specifikace a popis systému | 9 |
| 2.2.2 | Syntéza | 11 |
| 2.2.3 | Implementace systému | 12 |
| 2.2.4 | Výroba | 12 |
| 2.2.5 | Testování, expedice a provoz | 13 |
| 2.3 | Diagnostika číslicových obvodů | 14 |
| 2.3.1 | Základní pojmy | 15 |
| 2.3.2 | Detekce a lokalizace poruch | 16 |
| 2.3.3 | Návrh pro snadnou testovatelnost | 19 |
| 2.3.4 | Principy vestavěné diagnostiky | 31 |
| 2.4 | Shrnutí | 33 |
| 3 | Blízká témata | 35 |
| 3.1 | Návrh pro snadnou testovatelnost s využitím techniky scan ... | 35 |
| 3.1.1 | Vybrané varianty scan buněk | 36 |
| 3.1.2 | Typy scan technik | 38 |
| 3.1.3 | Částečný scan | 40 |
| 3.2 | Analýza zpětnovazebních smyček | 41 |
| 3.2.1 | Aplikace v oblasti návrhu a diagnostiky číslicových obvodů | 43 |
| 3.2.2 | Přehled metod | 45 |

VIII Obsah

| | | |
|----------|--|------------|
| 3.2.3 | Shrnutí | 46 |
| 3.3 | Analýza testovatelnosti | 47 |
| 3.3.1 | Úroveň hradel | 48 |
| 3.3.2 | Úroveň meziregistrových přenosů | 49 |
| 3.3.3 | Vyšší úrovně popisu | 55 |
| 3.3.4 | Shrnutí | 58 |
| 3.4 | Hierarchický test | 59 |
| 3.4.1 | Přehled základních pojmů koncepce transparentnosti .. | 62 |
| 3.5 | Shrnutí | 71 |
| 4 | Model obvodu popsaného na úrovni meziregistrových přenosů | 73 |
| 4.1 | Model struktury číslicového obvodu | 74 |
| 4.1.1 | Model rozhraní prvků | 74 |
| 4.1.2 | Model spojů a datového toku | 89 |
| 4.2 | Model transparentních režimů a datových cest | 90 |
| 4.2.1 | Model transparentních režimů | 90 |
| 4.2.2 | Model transparentních datových cest | 96 |
| 4.3 | Shrnutí | 104 |
| 5 | Metoda analýzy testovatelnosti | 107 |
| 5.1 | Vztahy pro ohodnocení říditelnosti a pozorovatelnosti | 107 |
| 5.1.1 | Vztahy pro lokální ohodnocení | 108 |
| 5.1.2 | Vztahy pro globální ohodnocení | 116 |
| 5.2 | Algoritmus analýzy testovatelnosti | 117 |
| 5.2.1 | Komentář k prohledávání grafů | 117 |
| 5.2.2 | Komentář k principu navržené metody | 118 |
| 5.2.3 | Hlavní blok algoritmu | 119 |
| 5.2.4 | Blok pro ohodnocení testovatelnosti | 121 |
| 5.2.5 | Blok pro ohodnocení říditelnosti | 121 |
| 5.2.6 | Blok pro ohodnocení pozorovatelnosti | 125 |
| 5.3 | Informace pro zlepšení testovatelnosti | 128 |
| 5.4 | Shrnutí | 130 |
| 6 | Ověření algoritmu analýzy testovatelnosti | 133 |
| 6.1 | Důkazy vybraných vlastností | 133 |
| 6.1.1 | Správnost a časová složitost algoritmu | 133 |
| 6.2 | Experimentální výsledky a příklady aplikace metody | 136 |
| 6.2.1 | Komentář k benchmarkovým obvodům | 137 |
| 6.2.2 | Analýza testovatelnosti | 138 |
| 6.2.3 | Návrh pro snadnou testovatelnost pomocí techniky scan .. | 145 |
| 6.2.4 | Generování benchmarkových obvodů | 154 |
| 6.3 | Shrnutí | 156 |

| | |
|--|-----|
| 7 Závěr | 157 |
| 7.1 Shrnutí výsledků práce | 157 |
| 7.2 Přínos práce | 158 |
| 7.3 Možné směry navazujícího výzkumu | 159 |
| Reference | 161 |
| Tabulky | 171 |
| Trojúhelníky čísel | 181 |



Seznam obrázků

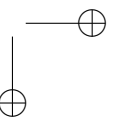
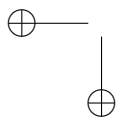
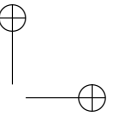
| | | |
|----|--|----|
| 1 | Ilustrace k úrovním popisu | 8 |
| 2 | Postup vzniku číslicového čipu | 10 |
| 3 | Testovací body | 23 |
| 4 | Testovací body a úspora vývodů obvodu pomocí adresace bodů | 23 |
| 5 | Testovací body a úspora vývodů obvodu pomocí sdílení datových cest | 24 |
| 6 | Segmentace obvodů | 26 |
| 7 | Ilustrace k principu strukturovaného návrhu | 27 |
| 8 | Ilustrace principu strukturovaného návrhu | 28 |
| 9 | Hraniční snímání | 29 |
| 10 | Struktura obvodu využívajícího RAS..... | 31 |
| 11 | Klopný obvod D a vybrané varianty scan buněk | 37 |
| 12 | Zobecněná scan buňka a zobecněný scan registr | 38 |
| 13 | Ilustrace k technice sériový scan (rozvody hodin nejsou zakresleny) | 39 |
| 14 | Ilustrace k analýze zpětnovazebních smyček..... | 42 |
| 15 | Ilustrace k využití analýzy zpětnovazebních smyček v diagnostice | 44 |
| 16 | Ilustrace k příkladu 2 | 45 |
| 17 | Ilustrace rozdílu mezi hierarchickým a nehierarchickým generováním testu | 59 |
| 18 | Ilustrace k principu hierarchického testu..... | 60 |
| 19 | Bližší pohled na hierarchické testování | 63 |
| 20 | Ilustrace k T-režimu a I-režimu | 65 |
| 21 | Příklad prvků nemajících I-režim resp. T-režim | 66 |
| 22 | Ilustrace k transparentním diagnostickým cestám | 69 |
| 23 | Příklad jednoduchého RTL obvodu (Diffeq) - schéma bylo vygenerováno nástrojem Synopsys Design Analyzer na základě popisu chování tohoto obvodu v jazyce VHDL | 75 |

XII Seznam obrázků

| | | |
|----|---|-----|
| 24 | Obvodový prvek MUX1 - vlevo schematická značka, vpravo struktura prvku | 77 |
| 25 | Propojení rozhraní prvků | 89 |
| 26 | Příklad k virtuálním portům a transparentním režimům | 91 |
| 27 | Průměrný čas CPU potřebný pro analýzu testovatelnosti | 137 |
| 28 | Ilustrace k zapojení scan registrů do samostatných scan řetězců . | 142 |
| 29 | Schéma obvodu Bert | 143 |
| 30 | Schéma obvodu Diffeq | 143 |
| 31 | Schéma obvodu Paulin | 144 |
| 32 | Schéma obvodu Tseng | 144 |
| 33 | Ilustrace k výběru a rozmístění registrů do scan řetězců | 146 |
| 33 | Vizualizace rozložení testovatelnosti pro techniku scan a obvod <i>Diffeq</i> | 151 |
| 33 | Evoluce nejlepších řešení při použití genetického algoritmu | 153 |

Seznam tabulek

| | | |
|----|---|-----|
| 1 | Přehled vybraných benchmarkových sad a jejich rozčlenění podle předpokládaných oblastí aplikace | 138 |
| 2 | Tabulka hodnot s_{1n} a s_{2n} pro vybraná n | 149 |
| 2 | Výsledky prohledávání stavového prostoru techniky scan pro obvod <i>Diffeq</i> | 150 |
| 3 | Ohodnocení diagnostických vlastností obvodu NLoops | 171 |
| 4 | Ohodnocení diagnostických vlastností obvodu NLoops1 | 172 |
| 5 | Ohodnocení diagnostických vlastností obvodu NLoops2 | 173 |
| 6 | Ohodnocení diagnostických vlastností obvodu NLoops3 | 174 |
| 7 | Ohodnocení diagnostických vlastností obvodu NLoops4 | 175 |
| 8 | Ohodnocení diagnostických vlastností obvodu Bert | 176 |
| 9 | Ohodnocení diagnostických vlastností obvodu Diffeq | 177 |
| 10 | Ohodnocení diagnostických vlastností obvodu Paulin | 178 |
| 11 | Ohodnocení diagnostických vlastností obvodu Tseng | 179 |
| 12 | Část trojúhelníku STIRLINGových čísel 2. řádu a BELLOvých čísel pro $1 \leq k \leq 10$ | 181 |
| 13 | Část trojúhelníku LAHOvých čísel a a_k čísel pro $1 \leq k \leq 10$ | 182 |



Použité zkratky a symboly

Zkratka Význam

| | |
|------|--|
| ASIC | Aplikačně-specifický integrovaný obvod (z angl. Application Specific Integrated Circuit) |
| BIST | Vestavěné samočinné testování (z angl. Built-In Self-Test) |
| CAD | Počítačem podporovaný návrh (z angl. Computer Aided Design) |
| CTF | Činitel přenosu říditelnosti (z angl. Controlability Transfer Factor) |
| CUA | Analyzovaný obvod (z angl. Circuit Under Analysis) |
| CUA | Testovaný obvod (z angl. Circuit Under Test) |
| DFT | Návrh pro snadnou testovatelnost (z angl. Design For Testability) |
| FAS | Množina zpětnovazebních hran (z angl. Feedback Arc Set) |
| FVS | Množina zpětnovazebních uzlů (z angl. Feedback Vertex Set) |
| GAL | Úroveň hradel (z angl. Gate Level) |
| GT | Generátor testu |
| GTKO | Generátor testu kombinačních obvodů |
| GTSO | Generátor testu sekvenčních obvodů |
| HDL | Jazyk pro popis hardwaru (z angl. Hardware Description Language) |
| HW | Hardware |
| ISL | Úroveň instrukční sady (z angl. Instruction Set Level) |
| LFSR | Posuvný registr s lineární zpětnou vazbou (z angl. Linear Feedback Shift Register) |
| MFAS | Nejmenší množina zpětnovazebních hran (z angl. Minimum Feedback Arc Set) |
| MFT | Modelování pro testovatelnost (z angl. Modeling For Testability) |
| MFVS | Nejmenší množina zpětnovazebních uzlů (z angl. Minimum Feedback Vertex Set) |
| OTF | Činitel přenosu pozorovatelnosti (z angl. Observability Transfer Factor) |
| RAS | Snímání pomocí libovolného přístupu (z angl. Random Access Scan) |
| RTL | Úroveň meziregistrových přenosů (z angl. Register-Transfer Level) |
| SAS | Snímání pomocí sériového přístupu (z angl. Serial Access Scan) |
| SFT | Syntéza pro snadnou testovatelnost (z angl. Synthesis For Testability) |

XVI Seznam tabulek

| | |
|------|---|
| STL | Strukturální úroveň (z angl. Structural Level) |
| SW | Software |
| SYL | Systémová úroveň (z angl. System Level) |
| TA | Analýza testovatelnosti (z angl. Testability Analysis) |
| TPG | Generátor testovacích vzorků (z angl. Test Pattern Generator) |
| TRL | Úroveň tranzistorů (z angl. Transistor Level) |
| VHDL | z angl. Very High Speed Integrated Circuits HDL |

Symbol Význam

| | |
|------------------------------------|--|
| $ M $ | Počet prvků množiny M |
| \mathbb{N} | Množina přirozených čísel (včetně nuly) |
| \mathbb{R} | Množina reálných čísel |
| $\mathbb{R}_{\langle 0,1 \rangle}$ | Množina reálných čísel ležících v intervalu $\langle 0, 1 \rangle$ |
| \emptyset | Prázdná množina |
| ∞ | Nekonečno |
| $A \setminus B$ | Rozdíl množin A a B |
| \square | Konec zvláštního textu (definice, věty, důkazu, příkladu atp.) |
| R^+ | Tranzitivní uzávěr binární relace R |
| R^* | Reflexivní a tranzitivní uzávěr binární relace R |
| $Def(R)$ | Definiční obor binární relace R |
| $Im(R)$ | Obor hodnot binární relace R |
| $E(G)$ | Množina hran grafu G |
| $V(G)$ | Množina uzlů (vrcholů) grafu G |
| $\sigma(G)$ | Incidence grafu G |
| $=$ | Rovnost |
| $:=$ | Přiřazení |

1

Úvod

Problematika testování je jednou z nejdůležitějších, ale také nejnákladnějších a nejnáročnějších částí návrhového cyklu elektronického systému. Klasický přístup k testování obvodu spočívá v nalezení a aplikaci takových testovacích vektorů, pomocí nichž bude možné detekovat či také lokalizovat výskyt fyzické poruchy ve vyrobeném elektronickém systému.

Neustálý pokrok v technologii výroby elektronických systémů umožňuje realizovat velmi rozsáhlé a složité návrhy, což díky hnacím silám¹ konkurenčního průmyslového vývoje vede mj. k soustavnému tlaku na návrh nových metod zabývajících se problematikou testování. Příčinou tohoto tlaku je zejména skutečnost, že současné metody, zabývajících se problematikou testování elektronických systémů, jsou v jistém smyslu ”pozadu” za současnými návrhovými trendy. Zatímco elektronické systémy jsou obvykle navrhovány na vysokých úrovních abstrakce², současné prakticky použitelné metody spojené s testováním na těchto úrovních jsou stále ve stádiu výzkumu či ověřování. V rámci tématu této práce nebude řešena problematika testování obecného elektronického systému na obecné úrovni abstrakce, ale jak vyplývá z pozdějšího textu, omezíme se pouze na jistou, modelem přesně vymezenou, podmnožinu číslicových obvodů (systémů) popsaných na tzv. úrovni meziregistrových přenosů. Obvod na této úrovni popisu lze získat i jinak, avšak vzhledem k tomu, že takový obvod bývá výstupem vysokoúrovňové syntézy a vstupem logické syntézy, pak můžeme předpokládat, že obvod na této úrovni popisu získáme právě v této etapě návrhového cyklu obvodu. Z toho plyne, že právě tato etapa je oblastí předpokládané aplikace postupů navržených v této práci. Obvod popsaný na úrovni meziregistrových přenosů se obvykle skládá ze dvou částí, a to z popisu struktury obvodových datových cest, což je část obvodu, kterou se tato práce zabývá, a řadiče ovládacího toku datovými cestami. Tato

¹ mezi něž patří např. cena, výkon a čas nutný k uvedení výrobku na trh

² jmenujme např. algoritmický popis chování systému jazykem pro popis hardwaru nebo tzv. návrh ”systému na čipu” obvykle předpokládající zapouzdření celého systému do jednoho křemíkového čipu

2 1 Úvod

práce se však bude věnovat pouze problematice spojené s testovatelností obvodových datových cest a nutnost existence příslušného radiče bude jen mlčky předpokládána a nebude dále zdůrazňována.

Při řešení problémů spojených s testováním číslicových obvodů lze vysledovat několik vývojových směrů [ABF90]. Jeden z prvních směrů vývoje v oblasti generování testů vycházel ze znalosti obvodové struktury a vedl k výpočetně náročným algoritmům, schopným však nalézt ”velmi kvalitní” testy³. Vzhledem k tomu, že použití těchto algoritmů bylo díky jejich velké časové složitosti prakticky omezeno na jednodušší návrhy, vznikly techniky tzv. hierarchického generování testu založené na modulární dekompozici obvodu. Jejich cílem bylo rozdělit problém generování globálního testu na problém generování lokálních testů dílčích modulů a na problém jejich následného překladač na globální test. Tento problém lze také chápat jako problém vytvoření souboru testovacích dat pro dílčí moduly a problém způsobu přenosu těchto dat obvodovou strukturou, v níž jsou moduly umístěny. Ukázalo se však, že pro velmi rozsáhlé a složité obvody a při nevhodné volbě zrnitosti modulů může být i tento přístup výpočetně náročný a že je navíc třeba řešit další problém, a to problém volby zrnitosti modulů. Jako alternativa ke strukturálně založeným metodám proto vznikaly další metody, např. pseudonáhodné generování testu či metody umožňující generovat test na základě popisu funkce nebo chování. Velkou nevýhodou těchto metod je jejich přílišná abstrakce od testované obvodové struktury, což obecně vede na generování méně kvalitních testů, než v případě metod založených na analýze struktury obvodu.

Odstranění nedokonalostí metod generování testů či zvýšení pravděpodobnosti generování kvalitnějšího testu lze dosáhnout např. použitím vhodné kombinace několika metod generování testu, modifikací původního návrhu za účelem zlepšení jeho testovatelnosti⁴ - obvykle pomocí metod návrhu pro snadnou testovatelnost⁵ nebo pomocí metod syntézy pro snadnou testovatelnost⁶. Kromě zlepšení testovatelnosti obvodu je tato modifikace ve většině případů příčinou nárůstu plochy, počtu vývodů a změny dalších - např. dynamických či energetických - parametrů obvodu.

Modifikace obvodové struktury za účelem zlepšení testovatelnosti obvodu se stala nerozlučnou částí moderních návrhů číslicových obvodů. Aby bylo možné splnit návrhová omezení kladená na výsledný obvod⁷ a současně byla nalezena modifikace vyznačující se vysokou testovatelností⁸, je nutné, aby proces provádějící tyto modifikace byl informován o kvalitě dané obvodové modifikace z hlediska testovatelnosti a plnění návrhových omezení. Patříčně

³ tj. krátké testy dosahující vysokého pokrytí poruch

⁴ angl. testability

⁵ angl. design for testability, DFT [ABF90]

⁶ angl. synthesis for testability, SFT [SS03]

⁷ např. maximální přípustná plocha, maximální přípustný počet vývodů, maximální příkon, maximální doba aplikace testu, maximální přijatelná doba návrhu obvodu

⁸ tj. co nejlepší testovatelností dosažitelnou při respektování návrhových omezení

informace o testovatelnosti obvodu obvykle poskytuje tzv. analýza testovatelnosti. Pomocí jejích výsledků je pak možné nalézt přijatelný kompromis mezi návrhovými omezeními a diagnostickými vlastnostmi obvodu.

1.1 Motivace k výzkumu

Existující metody analýzy testovatelnosti číslicových obvodů na úrovni meziregistrových přenosů jsou buď založeny na využití nějakého pravděpodobnostního modelu přenosu diagnostických dat obvodovou strukturou nebo na využití modelu vybraných vlastností obvodových prvků umožňujících detailnější a přesnější analýzu obvodových datových cest. Jelikož každá metoda analýzy testovatelnosti je velmi úzce spjata se způsobem generování testu [Uba04], jsou výsledky metod patřících do první skupiny prakticky použitelné pouze pro případy, kdy je uvažováno pseudonáhodné generování testu. Výsledky metody z druhé skupiny jsou pak vhodné v případech, kdy je uvažováno deterministické generování testu.

Právě druhá skupina metod souvisí s výzkumem prováděným v rámci této práce. K hlavním nevýhodám současných metod z této skupiny patří např. velká časová složitost, která je v rozporu s obecným předpokladem malé časové složitosti metod analýzy testovatelnosti [ABF90], vlivem používaného modelu nepřiliš přesná analýza obvodových datových cest či v případě požadavku univerzální použitelnosti výsledků analýzy úzká svázanost některých metod s konkrétní technikou návrhu pro snadnou testovatelnost.

K dlouhodobým nevýhodám metod zabývajících se problematikou testovatelnosti patří nejednotnost a mnohdy značná rozdílnost pojmů z této oblasti⁹. Nestandardizace faktorů ovlivňujících testovatelnost - tj. faktorů na jejichž základě má být testovatelnost obvodu určena - je příčinou různých výkladů a definic testovatelnosti a s ní souvisejících pojmů, což vede k nejednotnosti a roztržitosti metod z této oblasti. Je tedy snahou pojmy z této oblasti standardizovat a zahrnout bezesporné a jednoznačné matematické definice těchto pojmů do připravovaného standardu IEEE P1522 [SK04].

Je nutno poznamenat, že přestože se cíle existujících metod analýzy testovatelnosti z důvodu nejednotnosti definic liší, vyznačují se tyto metody jedním společným znakem, který lze označit jako snahu poskytnout dostatečně přesnou informaci o testovatelnosti daného obvodu. Většinou se jedná o odhalení obtížně testovatelných částí na základě ohodnocení jistých vlastností obvodu. Obvykle se předpokládá, že touto informací bude podstatně ovlivněn způsob modifikace obvodové struktury vedoucí ke zlepšení testovatelnosti.

Výzkum k tématu této práce byl zaměřen na návrh metody analýzy testovatelnosti číslicových obvodů na úrovni meziregistrových přenosů s předpokladem generování deterministického testu, tj. na návrh metody patřící do druhé skupiny a byl motivován zejména výše uvedenými nedostatky existujících metod z této skupiny.

⁹ podrobněji viz strana 19

1.2 Cíle výzkumu

Hlavním cílem výzkumu bylo významnou měrou přispět k odstranění výše uvedených nedostatků tím, že v této práci bude ukázáno, že lze vytvořit metodu poskytující přesnější informaci o testovatelnosti obvodu a pracující s přijatelnou časovou složitostí. Taková metoda je v práci představena, jsou shrnuty a dokázány její významné vlastnosti, je ověřena na vhodně zvolené sadě testovacích obvodů a výsledky pomocí ní dosažené jsou srovnány s výsledky existujících metod řešících stejný problém na téže úrovni abstrakce. Zdůrazníme zde, že tato práce se zabývá pouze problematikou testovatelnosti obvodových datových cest využívajících ke směrování toku dat strategii multiplexovaných datových cest na úrovni meziregistrových přenosů.

Jak z podrobnějšího textu vyplývá, přesnost analýzy datových cest podstatně závisí na přesnosti poskytnuté informace o transparentních vlastnostech konkrétních obvodových prvků. V této práci bude ukázáno, jak lze tuto informaci popsat a jak ji lze použít k ohodnocení testovatelnosti obvodu. Snaha o matematický popis takové informace vychází z předpokladu, že zejména pro efektivní průběh analýzy testovatelnosti či generování testu je velmi výhodné pokud každý obvodový prvek uložený v knihovně obvodem používaných prvků je kromě popisu svého rozraní, funkce atd. požadovaného např. návrhovým systémem vybaven také informací usnadňující řešení problémů spojených jak s diagnostikou prvku samotného tak s diagnostikou celku, jehož je prvek součástí. Cíle výzkumu prováděného v rámci této práce lze shrnout do následujících bodů:

- rozšíření stávající koncepce transparentnosti a formálního modelu [Růž02], včetně formálního popisu informace potřebné pro analýzu testovatelnosti číslicového obvodu popsaného na úrovni meziregistrových přenosů,
- návrh a formální popis vztahů pro ohodnocení testovatelnosti číslicového obvodu popsaného na úrovni meziregistrových přenosů,
- návrh a popis metody analýzy testovatelnosti s využitím prostředků rozšířeného modelu,
- ukázka aplikace navržené metody analýzy testovatelnosti a jejích výsledků při řešení vybraných problémů souvisejících s diagnostikou číslicových obvodů,
- shrnutí a dokázání významných vlastností navržené metody, srovnání výsledků s výsledky metod řešících stejný problém.

Cílem této práce ukázat, že je možné navrhnout metodu analýzy testovatelnosti, která bude díky detailnější analýze obvodových datových cest poskytovat přesnější výsledky, než jaké poskytují dosavadní metody, a to za přijatelné časové složitosti a s využitím dodatečné matematicky popsané informace o diagnostických vlastnostech obvodových prvků.

1.3 Struktura disertační práce

Dříve, než budou uvedeny kapitoly tvořící hlavní text této práce, je vhodné každou z těchto kapitol alespoň stručně představit.

Hlavní text práce začíná kapitolou 2. Ta je věnována přehledu základních pojmů z oblasti návrhu a diagnostiky číslicových systémů. V úvodní podkapitole jsou zmíněny základní způsoby a úrovně popisu číslicových systémů, z nichž je z hlediska této práce významná úroveň meziregistrových přenosů - jak bude z pozdějšího textu patrné, právě pro tuto úroveň je konstruován jak matematický model číslicového systému (kapitola 4), tak na jeho základě popsaná metoda analýzy testovatelnosti (kapitola 5). Druhá podkapitola se věnuje základním pojmům z oblasti návrhu, jakými jsou přehled prostředků pro popis číslicových systémů a přehled etap vzniku číslicového obvodu. Poslední podkapitola je pojata jako úvod do problematiky související s testováním číslicových systémů a její hlavní snahou bylo zmínit základní pojmy z oblasti diagnostiky číslicových systémů.

Kapitola 3 se věnuje tématům blízké souvisejícím s problematikou řešenou v rámci této práce za účelem usnadnění jejího zařazení do kontextu témat z oblasti diagnostiky číslicových obvodů. První podkapitola se věnuje návrhu pro snadnou testovatelnost s využitím techniky scan. Druhá podkapitola se věnuje náročnosti výběru registrů do scan řetězců a představuje tento problém z matematického pohledu. V třetí podkapitole je představen přehled existujících metod pro analýzu testovatelnosti. Hlavní snahou této podkapitoly bylo ukázat, že v současné době neexistuje přesná definice testovatelnosti a že obecně bývá testovatelnost chápána jako charakteristika zohledňující různé náklady spojené s testováním číslicového obvodu. Poslední podkapitola je kromě základního přehledu motivací, základních pojmů a principů z oblasti tzv. hierarchického testu věnována zejména pohledu na tzv. transparentní režimy a cesty. Účelem této podkapitoly je seznámit čtenáře se základními myšlenkami a principy koncepce transparentnosti a s nastíněním způsobu využití této koncepce pro analýzu testovatelnosti publikovanou v této práci.

Cílem kapitoly 4 je představit rozšíření výchozího modelu [Růž02], založeného na tzv. koncepci I-cest, o prostředky, pomocí kterých je možno popsat v této práci navrženou metodu analýzy testovatelnosti číslicového obvodu na úrovni meziregistrových přenosů. Rozšíření výchozího modelu tedy v žádném případě není samoučelné, ale bylo vyvoláno potřebou modelovat další skutečnosti týkající se struktury obvodu, jeho podčástí a vlastností. Za nejdůležitější rozšíření výchozího modelu lze považovat zejména modelování bitových složek portů a spojů, rozlišení bran portů, modelování tzv. virtuálních portů, zobecnění modelu transparentních režimů a cest.

Kapitola 5 se věnuje principům analýzy testovatelnosti navržené v rámci této práce. První podkapitola se kromě popisu vztahů pro ohodnocení snadnosti konstrukce tzv. systémů transparentních cest v obvodu, vztahů pro ohodnocení říditelnosti a pozorovatelnosti virtuálních portů a bran v obvodu a vztahů pro ohodnocení říditelnosti, pozorovatelnosti a testovatel-

6 1 Úvod

nosti portů obvodových prvků obvodu zabývá popisem vztahů pro ohodnocení testovatelnosti obvodového návrhu jako celku. Druhá podkapitola je věnována popisu navrženého algoritmu analýzy testovatelnosti. Ten je kvůli přehlednosti rozčleněn na několik dílčích algoritmů popisujících konstrukci množin, nutných pro ohodnocení diagnostických vlastností obvodu pomocí vztahů zavedených v první podkapitole. Všechny algoritmy jsou založeny na prohledávání orientovaných grafů G_S a G_I , jejichž struktura je dána jednak strukturou daného obvodu a jednak diagnostickými vlastnostmi obvodových prvků tvořících strukturu obvodu. Třetí podkapitola komentuje možné využití výsledků navržené metody analýzy testovatelnosti.

V první podkapitole kapitoly 6 jsou představeny a dokázány některé z významných vlastností navržené metody analýzy testovatelnosti, zejména správnost ohodnocování diagnostických vlastností ve smyslu zavedených definic a složitost algoritmu analýzy testovatelnosti. Ve druhé podkapitole jsou představeny výsledky dosažené pomocí navržené analýzy testovatelnosti a to v oblastech samostatné analýzy testovatelnosti, využití výsledků analýzy testovatelnosti ke zlepšení testovatelnosti obvodu s použitím techniky scan a generování benchmarkových obvodů. Dosažené výsledky jsou shrnuty a jsou srovnány s výsledky metod řešících stejný problém.

Závěrečná kapitola 7 se věnuje zhodnocení hlavních výsledků, cílů a přínosů této práce a nastínění možných směrů navazujícího výzkumu.

2

Související oblasti

2.1 Modelování číslicových systémů

Volba modelu je velmi důležitá pro návrh, výrobu a testování číslicového systému. Způsob, jakým reprezentujeme systém, významně ovlivňuje jak simulaci prováděnou za účelem verifikace návrhu a syntézu, tak modelování a simulaci poruch a v neposlední řadě i způsob generování testu pro tento systém a hledání vhodného způsobu aplikace vygenerovaného testu [ABF90].

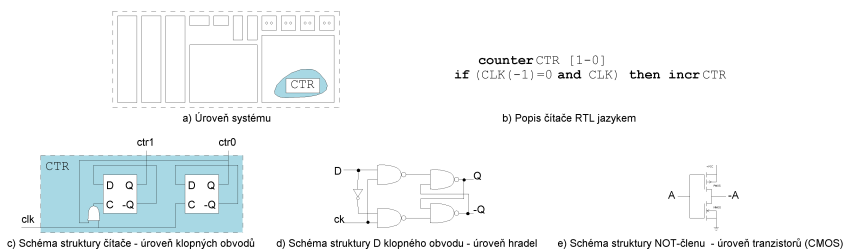
2.1.1 Klasifikace modelů

Na libovolné úrovni abstrakce můžeme na číslicový systém pohlížet jako na tzv. *černou skříňku*, využívající informace na vstupech k produkovaní odezvy na výstupech. Funkci systému pak lze popsat vstupně-výstupním zobrazením, modelujícími operaci prováděnou systémem; v takovém případě hovoříme o tzv. *funkčním popisu (modelu)* systému - může se jednat o jistou logickou funkci, popis transformace dat soustavou rovnic atp. Spojíme-li funkční popis (tj. oblast hodnot) s časovou oblastí, získáme *popis systému na úrovni chování*¹. Pro obvody lišící se časováním, ale provádějící stejnou funkci, pak může být použit též funkční model. Funkcí a časováním se také lze zabývat samostatně, např. během verifikace návrhu či generování testu.

Kromě výše popsaných způsobů popisu (zabývajících se popisem vnější funkce systému, nikoliv však jeho vnitřní strukturou) existuje tzv. *strukturální popis (model)*. Ten chápe systém jako strukturu tvořenou vzájemně propojenými podsystémy, obvykle nazývanými *komponenty* nebo *prvky*. Strukturální popis bývá často hierarchický, tzn. že komponenty mohou být tvořeny vzájemně propojenými podkomponentami atd. Komponenty, které již nelze takto popsat, jsou nazývány primitivní prvky - jejich popis je obvykle uložen v nějaké návrhové knihovně. *Příklad 1*: blokový diagram počítačového systému

¹ někdy označovanou pojmem behaviorální úroveň popisu

8 2 Související oblasti



Obrázek 1. Ilustrace k úrovním popisu

je strukturálním modelem tvořeným komponentami jako CPU, RAM, vstupně-výstupní zařízení apod. *Příklad 2:* Schéma obvodu může být tvořeno komponentami AND, OR, dekodéry, funkčními jednotkami apod.

Strukturální model s sebou vždy nese (implicitně nebo explicitně) informaci o funkci jeho komponent; čistě funkčním popisem jsou obvykle reprezentovány pouze velmi malé obvody. Moderní návrhové systémy a prostředky (např. VHDL) umožňují kombinovat v jednom návrhu různé způsoby popisu, čímž je výrazně usnadněna zejména znovupoužitelnost dříve vytvořených návrhů, ale také návrh často principiálně velmi odlišných součástí systému. Důsledkem toho je, že popisy, které bývají navenek označovány jako funkční či behaviorální, často obsahují i jistou strukturální informaci. V návrhu z praxe se téměř vždy setkáme s kombinací různých popisů.

2.1.2 Úrovně popisu

Podle toho, s jakou abstrakcí popisu systému pracujeme, můžeme rozlišit několik úrovní popisu systému - počet a pojmenování těchto úrovní není jednotný a často se v literatuře liší. [Coh01] rozlišuje následující úrovně popisu číslicových systémů:

- úroveň systémová² - snahou tohoto popisu je modelovat systém pomocí tzv. zdrojů (např. sběrnice, procesory, paměti) např. za účelem analýzy efektivnosti systému s ohledem na využití zdrojů při různých úlohách nebo definice komunikačního protokolu mezi zdroji a algoritmů činnosti jednotlivých zdrojů,
- úroveň instrukční sady³ - tato úroveň popisu je vhodná zejména k definici a simulaci instrukcí procesoru. Instrukce jsou v simulaci vybírány a dekódovány s ohledem na formát instrukcí. K simulaci provádění instrukcí je obvykle použito aritmetických a logických operací,
- úroveň strukturální⁴ - tato úroveň popisuje systém pomocí vzájemně propojených komponent. Každá komponenta může být tvořena několika

² angl. System Level, SYL

³ angl. Instruction Set Level, ISL

⁴ angl. Structural Level, STL

menšími, vzájemně propojenými podkomponentami atd., až dokud se nejedná o primitivní prvek, uložený v knihovně,

- úroveň meziregistrových přenosů⁵ - pro systém na této úrovni je typické, že je složen ze dvou částí a to datových cest a obvodového řadiče ovládajícího tok dat těmito cestami [Mär92]. Datová část pak obvykle sestává z funkčních bloků oddělených registry či pamětmi a z propojovacích prvků - obvykle multiplexorů a sběrnic. Obvodový řadič bývá realizován stavovým automatem. Velmi často je tento typ popisu používán jako vstup logické syntézy. Protože způsob propojení funkčních bloků, paměťových prvků a propojovacích prvků je znám, lze popis na této úrovni chápat jako speciální případ strukturálního popisu,
- úroveň hradel⁶ - jedná se o další speciální případ strukturálního popisu. Návrh je tentokrát tvořen vzájemným propojením nízkoúrovňových prvků - obvykle logických hradel či klopných obvodů. Tento popis bývá výstupem logické syntézy nebo nástroje provádějícího rozmístění a je použit buď jako dokumentace k netlistu nebo jako model struktury návrhu pro účely simulace,
- úroveň tranzistorů⁷ - opět se jedná o speciální případ strukturálního popisu - návrh je tentokrát tvořen vzájemným propojením tranzistorů (např. unipolární technologie CMOS). Na této úrovni popisu nebývají obvody současných složitostí a rozměrů navrhovány přímo návrhářem - tento popis je obvykle součástí automatizované implementace systému, jejímž hlavním cílem je příprava masky k výrobě čipu. I na této velmi nízké úrovni popisu je však někdy prováděna (implementačně velmi přesná, avšak také výpočetně náročná) simulace a i pro tuto úroveň existují např. modely poruch.

2.2 Návrh číslicových systémů

2.2.1 Specifikace a popis systému

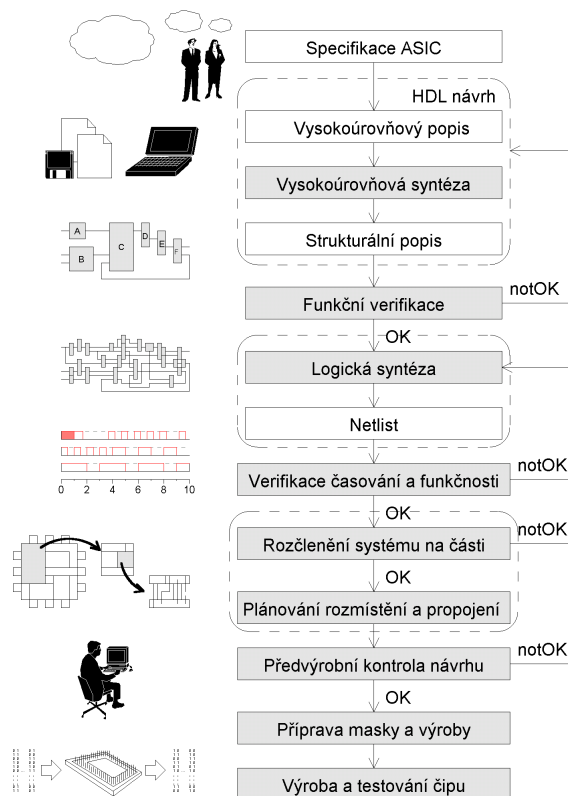
Návrh číslicového obvodu obvykle začíná systémovou specifikací, zachycující počáteční požadavky kladené na výsledný obvod - zejména se jedná o celkové vstupně-výstupní chování obvodu (tj. jeho funkci), strukturu rozhraní obvodu, předpokládanou velikost plochy obvodu, požadovaný pracovní kmitočet obvodu a další parametry. Z této prvotní specifikace pak obvykle vychází specifikace dílčích podsystémů, z nichž každý může být dále zjemněn a popsán pomocí menších bloků atd. Máme-li již některé z podsystémů resp. bloků k dispozici (např. ve vlastní knihovně prvků), nemusíme se jejich návrhem zabývat

⁵ angl. Register-Transfer Level, RTL

⁶ angl. Gate Level, GAL

⁷ angl. Transistor Level, TRL

10 2 Související oblasti



Obrázek 2. Postup vzniku číslicového čipu

znovu⁸, čímž lze některé návrhové kroky vynechat a tím výrazně uspořit čas pro celkový návrh číslicového obvodu.

Vlastní návrhová práce pak začíná popisem systému, některého jeho podsystému resp. bloku (např. pomocí některého z jazyků pro popis hardwaru⁹ (hardware description language, HDL)). HDL jazyky vznikly proto, aby jejich použití vedlo k zefektivnění návrhů - spolu s popisem návrhu totiž poskytují prostředky pro jeho dokumentaci, simulaci a verifikaci. Historicky starší způsob popisu (obvodové schéma - např. na úrovni hradel) sice rovněž umožňoval návrh daného systému - a to formou popisu propojení jeho podčástí, avšak vzhledem k jeho grafické reprezentaci byl sice vhodným pro návrháře-člověka, ale už ne příliš vhodný pro další automatizované zpracování např. v CAD nástroji. Obvodové schéma bylo obvykle nutné transformovat na počítačově zpracovatelný *netlist*¹⁰, který už byl a který kromě

⁸ takový postup se označuje termínem znovupoužití návrhu (angl. design reuse)

⁹ např. VHDL (Very high-speed integrated circuit HDL) či Verilog

¹⁰ tj. popis propojení logických buněk - viz níže

binárně či ASCII zakódovaného schématu obsahoval i informace o jednotlivých podčástech obvodu. Nicméně, ani v netlistu, který lze chápat jako informačně obohacenou verzi popisu schématu, nebylo možné přehledně a kvalitně daný obvod zdokumentovat a zaznamenat do něj všechny ve schématu chybějící informace. Proto - zejména nedostatky historicky dřívějších způsobů popisů při návrzích složitějších obvodů - přispěly k velkému rozšíření jazyků HDL, které těmito nedostatky netrpí a umožňují tak rychlý a kvalitní návrh i velmi složitých obvodů. Současné HDL jazyky umožňují popis obvodu prakticky na kterékoliv úrovni abstrakce, včetně kombinace několika úrovní popisů v jednom návrhu (často je např. kombinován strukturální popis s popisem chování).

2.2.2 Syntéza

Po dokončení popisu systému pomocí některého z existujících prostředků (nejčastěji pomocí některého HDL jazyka) a jeho uložení v daném datovém formátu obvykle následuje *logická syntéza*, což je proces provádějící transformaci HDL popisu na netlist. Jelikož oba v současné době nejrozšířenější HDL jazyky byly původně vyvinuty pro jiný účel¹¹ než pro jaký jsou v současné době používány a navíc oba vznikly dlouho před tím, než se objevily první komerční nástroje pro logickou syntézu, byli návrháři omezeni pouze na jisté podmnožiny HDL jazyků, tj. podmnožiny velmi blízké výsledné hardwarové reprezentaci obvodu. Tzn., že návrhář musel mít jasnou představu nejen o funkci navrhovaného obvodu, ale i o tom, jak konkrétně bude tento obvod implementován na cílové platformě.

Se zvyšující se abstrakcí popisu obvodů se objevují vysokoúrovňové či behaviorální nástroje pro syntézu. Tyto nástroje již umožňují syntetizovat i obvody navržené s vysokou měrou abstrakce od cílové hardwarové implementace (tj. např. zápisem algoritmu popisujícího transformaci mezi vstupy a výstupy daného obvodu). Typické je, že HDL popis může být kombinován s jinými popisy - např. stavovým automatem, tabulkami logických funkcí či schématem obvodu na úrovni hradel. Samozřejmě i v dnešní době je výhodou - zejména pro efektivnost implementace - má-li návrhář co nejjasnější představu o tom, jak bude jeho popis implementován. U moderních - často složitých a rozsáhlých - návrhů je však mnohdy tento stav nereálný a proto je obvyklé, že implementace je automatizována a z podstatné části ovlivněna nástrojem pro *vysokoúrovňovou syntézu*. Ten je tvořen knihovnou buněk¹² a nástrojem pro logickou syntézu¹³. Po úspěšné simulaci behaviorálního HDL popisu je nástrojem pro logickou syntézu obvykle vygenerován netlist¹⁴ reprezentující

¹¹ VHDL byl původně navržen pro dokumentaci a popis obvodu, Verilog pro simulaci

¹² tj. knihovnou prvků (logických buněk) sestavených z technologicky nezávislých členů - např. NAND a invertorů

¹³ většina výrobců nástrojů dodává jen programové vybavení a většina prodejců číslicových obvodů jen knihovny buněk

¹⁴ nejčastěji ve formátu EDIF, avšak existují i nástroje umožňující zapsat netlist v HDL

strukturální popis daného obvodu - obvykle se jedná o popis obvodu na úrovni meziregistrových přenosů, tj. na úrovni, pro kterou je vytvořena metoda analýzy testovatelnosti publikovaná v této práci. Z tohoto pohledu je možné konstatovat, že metoda navržená v této práci je mj. uplatnitelná právě v této části návrhového cyklu obvodu. Po vygenerování netlistu je provedena další simulace a její výsledky jsou porovnány s výsledky předchozí (behaviorální) simulace.

2.2.3 Implementace systému

Po dokončení syntézy je možné provést vlastní *implementaci systému* [Smi97], na jejímž konci získáme veškeré podklady nutné pro zahájení výroby čipu. Nejdříve (není-li to dáno již samotným popisem systému) je nutno rozčlenit složitý systém na několik podčástí nebo bloků (paměti RAM, ROM, řadiče paměti cache, bloky ALU, FPU, MMU, DMA apod.) tak, aby byly tyto podčásti co nejméně vzájemně propojeny, a aby plocha žádná z nich nepřesáhla jistou maximální hodnotu. Poté může být zahájeno tzv. *plánování čipu*¹⁵. Vstupem plánování čipu je *hierarchický netlist* (tj. *logický popis* číslicového obvodu), popisující propojení bloků a propojení logických buněk uvnitř bloků. Cílem plánování čipu je rozmístit bloky tak, aby více propojené bloky byly blíže u sebe. Zároveň je vyhrazen prostor pro rozvod hodin a napájení a rozhodnuto o počtu a umístění I/O a napájecích vývodů. Výstupem je *plán čipu*¹⁶, tj. *fyzický popis* číslicového obvodu.

Dalším krokem je tzv. *rozmístování*¹⁷. Cílem je vyhradit v blocích propojovací oblasti a rozmístit logické buňky v blocích tak, aby byla minimalizována plocha číslicového obvodu a kritická zpoždění v blocích. Jelikož je tento rozmístování úzce spjato s plánováním čipu, bývají někdy oba tyto kroky kombinovány do jednoho celku. Snahou je, aby po nich následující krok (tzv. *propojování*) proběhl co nejúspěšněji. Cílem *propojování*¹⁸ je zajistit umístění spojů tak, aby byla minimalizována plocha nutná k propojení, minimalizovat součet délek použitých spojů a jejich ploch, minimalizovat zpoždění kritické cesty a počet vrstev, přes něž spoje procházejí. Až po ukončení této fáze lze jednoznačně určit fyzické rozměry čipu, počet a podobu vrstev masky a přesné umístění spojů.

2.2.4 Výroba

K samotné výrobě čipů [Mue01] se používá proces zvaný *fotolitografie*. Nejprve je odpařováním na *plátku* křemíku vytvořena izolační vrstva oxidu křemičitého. Následuje vytvoření polovodičové vrstvy a nanesení vrstvy odolné

¹⁵ angl. floorplanning nebo také chip planning

¹⁶ angl. floorplan

¹⁷ angl. placement

¹⁸ angl. routing

vůči světlu. Poté je na vzniklý fotocitlivý povrch promítnut obraz některé z vrstev čipu. Přitom se v projektoru používá speciální *maska*, která odpovídá dané vrstvě čipu. Jakmile světlo procházející maskou osvítil povrch plátku, vytvoří v něm fotografický obraz dané vrstvy. Plátek je pomocí krokovacího stroje pootočen a stejná maska je použita k osvětlení části povrchu plátku příslušející jinému čipu. Po dokončení osvětlování je celý povrch omyt leptavým roztokem, čímž se z povrchu smyje fotorezistentní vrstva. Na povrchu tak zůstane pouze obraz jednotlivých vrstev a jejich propojení. Následně je na plátek nanášena další vrstva polovodičového materiálu, která je pokryta fotorezistentním materiálem. Vzniklý povrch je ozářen projektorem, v němž je umístěna maska pro další vrstvu čipu. Tento postup se opakuje, dokud nejsou vytvořeny všechny požadované vrstvy čipu.

V konečné fázi jsou vytvořeny *metalizační vrstvy* definující kovová propojení mezi jednotlivými tranzistory a případnými dalšími součástkami.

Na dokončeném kruhovém plátku je obvykle vytvořeno tolik čipů, kolik je maximálně možné. Protože však čipy bývají čtyřhranné a plátky kruhové, vždy vzniká na hranách plátku nějaký odpad. Poznamenejme, že z daného plátku o jistém průměru lze použitím modernější projekční technologie vyrobit větší počet návrhově totožných procesorů¹⁹.

Z hlediska kvality je nutné uvést, že zdaleka ne všechny procesory na plátku jsou vyrobeny bezchybně. Poměr počtu dobrých čipů ku celkovému počtu se nazývá *výtěžnost* - ta bývá při náběhu nové výrobní linky kolem 50%. Po plném rozjezdu sériové výroby daného čipu může výtěžnost dosáhnout až 90%. Tato čísla jsou však obecná a přibližná, neboť skutečné údaje o výtěžnosti pochopitelně každý výrobce tají.

2.2.5 Testování, expedice a provoz

Po dokončení plátku je každý v něm obsažený čip otestován pomocí speciálních sond - obvykle uspořádaných do matice - sloužících pro generování testovacích vektorů resp. sběr odezev na ně. Je-li na základě analýzy odezev čip vyhodnocen jako špatný, je označen barevnou skvrnou. Tento typ testu²⁰ se nazývá *produkční test* a je prováděn speciálním testovacím strojem; vlastní generování testovacích vektorů a vyhodnocování odezev je řízeno testovacím programem vyvinutým výrobcem a/nebo objednavatelem číslicového obvodu.

¹⁹ např. z plátku o (obvyklém) průměru 200 mm, tj. ploše 31416 mm² bylo možné vyrobit okolo 150 procesorů PentiumII (7.5 mil. tranzistorů) při použití projekční technologie 0.35 μm; při použití modernější technologie 0.25 μm lze z téhož plátku vyrobit dvojnásobný počet procesorů PentiumII nebo 300 procesorů PentiumIII (ten má asi 28.1 mil. tranzistorů) technologií 0.18 μm. V současnosti již existuje technologie pod 0.09 μm s plátky o větším průměru, čímž je umožněna ještě větší produkce procesorů

²⁰ testem kombinačního resp. sekvenčního obvodu je myšlena množina resp. posloupnost testovacích vektorů a analýza získaných odezev

Následně jsou neoznačené čipy vyříznuty buď vysoce výkonným laserem nebo diamantovou pilou a znovu otestovány. Během tzv. *zapouzdrění* je čip umístěn do pouzdra a speciální strojem jsou ze zlatých drátků vytvořeny spojky mezi vývody čipu a pouzdra.

Následuje *expedice* neoznačených čipů. V některých případech jsou (zejména u velkých odběratelů, např. implementujících dané čipy do svých zařízení) čipy otestovány ještě vlastním vstupním testem, aby nedošlo zabudováním vadných čipů ke znehodnocení celého zařízení obsahujícího čip - např. desky plošných spojů již částečně osazené jinými prvky. Pokud test odhalí chybu až v desce, obvykle bývá nákladné provést výměnu vadné komponenty; je-li chyba způsobena čipem, bývá tento zaslán výrobcí k opakovanému otestování. Zde se zjišťuje, zda se jedná o poruchu neodhalenou jeho produkčním testem (v tom případě je produkční test nedostatečný) či zda se jedná o poruchu způsobenou špatným pájením, elektrostatickým či mechanickým poškozením atd. Snahou je zjistit, na které ze stran existují nedostatky v používaných postupech.

Je zřejmé, že detekce a odstraňování poruch na úrovni dílčího zařízení (např. desky) je nákladná záležitost. Ještě nákladnější [FA97] je však provádět tuto činnost až na úrovni, kdy je vada odhalena přímo koncovým uživatelem vyrobeného zařízení nebo servisní firmou. V případě vadné desky obvykle lze - z hlediska koncového uživatele - považovat za nejlevnější odstranění poruchy její vyjmutí ze systému, uplatnění reklamace u prodejce a její nahrazení novou deskou. Pro výrobce však taková výměna znamená jisté dodatečné výdaje a opatření. Je-li snahou úspěšného výrobce dodávat bezporuchová zařízení na trh, pak jeho snahou musí také být zjistit příčinu vzniku poruchy a příčinu neodhalení této poruchy produkčním testem.

2.3 Diagnostika číslicových obvodů

Již bylo zmíněno, že nefunkčnost čipu může být způsobena závadou vzniklou při jeho výrobě nebo po jejím ukončení; stejně tak může být způsobena chybou při návrhu. Problematikou chyb vzniklých při návrhu obvodu se zabývá samostatná disciplína nazývaná *verifikace návrhu*. Je to činnost, během níž se zjišťuje, zda navržený obvod realizuje správně funkce, pro něž byl navržen. Verifikace návrhu se provádí před zahájením výroby obvodu, takže nápravu návrhové chyby je možné realizovat opakováním jisté části návrhu. Nedochází tak ke zvýšení ceny v důsledku přípravy podkladů pro výrobu čipu a jeho následné výroby. Verifikace návrhu se však tato práce nedotýká.

Ale i když je po verifikaci návrhu konstatováno, že je obvod navržen správně, není tím ještě zaručena bezchybná funkce čipu, který bude podle tohoto návrhu vyroben. V etapě výroby obvodu nebo později²¹ totiž mohou být

²¹ příčinou výrobní vady bývá nepřesnost technologického postupu, k příčinám mimovýrobní vady patří např. destrukční, degradační mechanismus nebo mimořádný jev

do fyzické struktury čipu zaneseny *poruchy*, jejichž projevem může být chybná funkce čipu. Avšak, není-li část čipu s poruchou v dané aplikaci používána, nemusí se chybné chování projevit, přestože se v něm porucha vyskytuje. Není ale možné předpokládat, že chybná část čipu nebude používána - naopak, je nutné uvažovat, že čip bude používán v celém rozsahu svých funkcí, i když některé z nich daná aplikace nevyužije. Je tedy snahou spolehlivě a zároveň efektivně ověřit, že vyrobený či dodaný čip neobsahuje poruchu. To je možné zjistit během tzv. *testování* čipu. Je výhodné, pokud prostředky vyvinuté pro samostatné testování obvodu v etapě výroby jsou použitelné i při použití obvodu ve větším logickém celku, např. na desce či jako podsystému jiného obvodu.

Problémy souvisejícími s testováním číslicových obvodů se zabývá *diagnostika číslicových obvodů*. Ta se však neomezuje pouze na testování obvodů, ale zasahuje také do jejich návrhu, a to zejména do raných etap popisu a syntézy obvodu. Cílem návrhu, ale hlavně syntézy je, aby jejím výstupem byl obvod plnící požadovanou funkci, který by zároveň vyhovoval různým návrhovým kritériím. Mezi ně patří např. cena obvodu, počet vývodů, dynamické parametry a plocha na křemíkovém plátku. Z hlediska uplatnění požadavků diagnostiky je dalším kritériem *testovatelnost* (viz strana 19) obvodu. Tu je možno chápat především jako snahu o zvýšení říditelnosti/pozorovatelnosti (viz strana 21) vnitřních uzlů obvodu. Pokud jsou tyto principy uplatněny již během etapy popisu obvodu, hovoříme o *návrhu pro snadnou testovatelnost*²² (viz kapitola 2.3.3, strana 19) a jsou-li uplatněny během syntézy, pak hovoříme o *syntéze pro snadnou testovatelnost* [ABF90, Ait95, Cox95, Lee97].

2.3.1 Základní pojmy

Diagnostika se zabývá analýzou technického stavu číslicového obvodu či systému (např. desky); jejím cílem je zjistit na základě odezev systému na vstupní stimuly (tzv. *testovací vektory*²³), zda je, či není v systému porucha, popř. i ve které části systému tato porucha nastala. V této formulaci jsou obsaženy dvě základní úlohy diagnostiky, a to detekce poruchy a její lokalizace. Původní snahou bylo používat přitom bezdemontážních postupů a vystačit si pouze s přístupem k vývodům čipu, tzv. *primárním vstupům* a *primárním výstupům*. Brzy se však ukázalo, že s tímto omezením není možné vždy vystačit²⁴ a že je nutné si tuto úlohu usnadnit přístupem k jistým vnitřním bodům systému, k tzv. *testovacím bodům*. Tento přístup je obvykle podmíněn zásahem do původního návrhu. Z tohoto pohledu pak už nehovoříme pouze o návrhu, ale o návrhu pro snadnou testovatelnost - blíže viz kapitola 2.3.3,

²² angl. design for testability, DFT

²³ kromě tohoto "klasického" způsobu testování existují i jiné, netradiční přístupy jako např. měření tepelného spektra, měření sekundární emise, testování plazmovým obloukem či měření klidového odběru proudu

²⁴ zejména při testování sekvenčních obvodů, kdy existuje problém s nastavením resp. zjištěním vnitřního stavu

strana 19 - ten je vykoupěn změnou parametrů obvodu, např. změnou dynamických parametrů, nárůstem plochy či počtu vývodů obvodu.

Tyto principy spolu s principy autonomního testování číslicových systémů využívajícího vestavěné diagnostiky tvoří moderní formu diagnostické disciplíny, kdy již diagnostika vystupuje v problematice boje s poruchami aktivně. Další formou aktivního diagnostického přístupu je navrhování *bezpečných obvodů*, na které navazuje navrhování *systémů odolných proti poruchám*.

V této práci budeme předpokládat, že je test založen na ”klasickém” testování využívajícím posloupností binárních testovacích vzorků a odezev. V takovém případě tedy nutně musí existovat nějaký zdroj testovacích vektorů a analyzátor správnosti odezev. Obvod, který je testován, budeme nazývat *testovaný obvod*²⁵.

Při úvahách o testování číslicového obvodu musíme vyřešit dva problémy - vytvoření testovací posloupnosti (tzv. *generování testu*) a způsob *aplikace testu*. Test je možné vytvořit buď manuálně např. návrhářem obvodu nebo pomocí programových prostředků pro automatické generování testu, tzv. *generátory testu* (GT). Ty jsou dvojího typu: *generátory testu pro kombinační obvody* (GTKO) a *generátory testu pro sekvenční obvody* (GTSO). Při aplikaci testu, jak je chápán v této práci²⁶, jsou řešeny tyto problémy:

- způsob nastavení testovacích vektorů na vstupy komponent tvořících obvod,
- přenos odezev na testovací vektory do takových bodů obvodu, kde je možné tyto odezvy snímat (a případně vyhodnocovat).

Diagnostika se klasifikuje podle několika hledisek. V první řadě jde o způsob získávání testovacích vektorů a o způsob vyhodnocování správnosti odezev. Generování vstupní testovací posloupnosti probíhá buď předem (*off-line*) nebo v průběhu testu (*on-line*), a to buď programovými, nebo technickými prostředky, vyhodnocování správnosti odezev probíhá buď srovnáváním se správnými odezvami, čtenými z paměti, nebo srovnáváním s kombinacemi, získanými z etalonu, který se testuje současně²⁷.

Nejčastější klasifikace forem diagnostiky vychází z časového uspořádání testování, kdy rozlišujeme *diagnostiku periodickou* a *průběžnou*; ze vztahu zkoušeč - testovaná jednotka pak rozeznáváme *diagnostiku vnější* a *vnitřní*.

2.3.2 Detekce a lokalizace poruch

Každý test je spojen s detekcí popř. lokalizací pouze jistého druhu poruch; hovoříme pak o detekčním resp. lokalizačním testu. K základní úloze diagnostiky tedy patří i *modelování poruch* - cílem je vytvořit jednak co

²⁵ angl. circuit under test, CUT

²⁶ předpokládáme generování deterministického, tzv. hierarchického testu [YMV99, MH90, PVS93, LP97, YM99] pro RTL obvod založeného na generování dílčích lokálních testů pro komponenty obvodu a jejich transformaci na globální test

²⁷ tzv. komparační testování

nejpřesnější model poruch (přesnost modelu), ale i model, se kterým by se snadno a rychle pracovalo (použitelnost modelu). Je zřejmé, že tyto dva požadavky jsou protichůdné, proto je výsledkem kompromisní řešení. Nejjednodušší a nejrozšířenější je model poruch typu trvalá 0 resp. trvalá 1. Předpokládá, že libovolná porucha v obvodu vznikne přerušením vodivé cesty, nebo vytvořením nechtěné vodivé cesty na napájecí napětí nebo zem. Z dalších modelů poruch zmiňme např. modelování zkratů mezi vodiči, modelování poruch p-n přechodů obvodů DTL a TTL či model poruch zpoždění.

Základní úloha diagnostiky najít pro testovanou jednotku *úplný*²⁸ a pokud možno *minimální test*²⁹ se řeší různými způsoby [ABF90]. V naprosté většině diagnostických metod se činí předpoklad jediné poruchy - předpoklad vícenásobné poruchy vede na metody v praxi nepoužitelné.

Náklady na vytvoření testu představují pro každý obvod jednorázový náklad, jednou vygenerovaný test se používá opakovaně pro všechny obvody vyrobené dle téhož návrhu. Při velkých sériích vyráběných obvodů se pak náklady na vygenerování testu přepočtené na jeden obvod snižují. Cena obvodu je však ovlivněna dalším faktorem - cenou za implementaci zkoušeče, pomocí něhož je test aplikován a dobou, po níž je pro daný test využíván. Zde dochází opět ke konfliktu. Chceme-li snížit náklady na aplikaci testu, musíme často zjednodušit test, což ovšem obvykle znamená, že dosáhneme menšího pokrytí poruch.

Detekce poruch

Cílem *detekčního testu* nad daným modelem poruch je zjistit, zda v daném obvodu existuje porucha popsatelná tímto modelem poruch. Princip odhalení konkrétní poruchy kombinačního obvodu spočívá v nastavení takového testovacího vzorku na dané místo obvodu, který je schopen tuto poruchu odhalit. Detekční test - v ideálním případě úplný - pak sestává z množiny testovacích vektorů vedoucích k odhalení všech jednotlivých poruch z daného modelu poruch. Jelikož cílem tohoto testu není rozlišit, která konkrétní porucha nastala, ale zda nějaká nastala, je možné provést rozklad množiny poruch, v němž poruchy odhalitelné týmž testovacím vektorem budou náležet téže rozkladové třídě. Hovoříme pak o *ekvivalenci poruch* a ekvivalenční třídy nazýváme *redukovaným seznamem poruch*. Metody generování detekčních testů lze rozdělit na:

- metody pro kombinační obvody - pro kombinační obvody jsou nejlépe zpracované *metody strukturální*, které vycházejí z tzv. *D-algoritmu* [ABF90] a hledají úplný test respektující topologii obvodu. Existují však i např.

²⁸ důležitým parametrem detekčního testu je pokrytí poruch, tedy číslo udávající procentuálně počet poruch, které je test schopen detekovat. Test s pokrytím poruch 100% se označuje jako úplný

²⁹ s využitím redukovaného seznamu poruch lze nalézt tzv. *minimální test*, tj. úplný test tvořený minimálním počtem testovacích vektorů

metody generující (tzv. *funkční*) *test* na základě popisu funkce obvodu. Nevýhodou však je, že pro týž obvod funkční test obvykle nedosahuje pokrytí poruch srovnatelného se strukturálním testem; to je způsobeno skutečností, že funkční test uvažuje pouze funkci obvodu a abstrahuje od jeho cílové struktury. Proto existují i kombinované metody, kdy je při generování funkčního testu zohledněna informace o obvodové struktuře,

- metody pro sekvenční obvody - metody generování testů pro sekvenční obvody rovněž dělíme na strukturální a funkční. Vzhledem k existenci vnitřního stavu u těchto obvodů však použití těchto metod vede na značně větší objem dat než u kombinačních obvodů. Proto je jejich použití prakticky obvykle omezeno na méně složité obvody. U složitějších obvodů je pak snaha tyto generátory využívat při testu co nejméně (vzhledem k dlouhým posloupnostem testovacích vektorů) a co nejvíce je zastoupit generátory méně náročnými - to je však obvykle možné až např. s využitím technik návrhu pro snadnou testovatelnost nebo pomocí prostředků vestavěné diagnostiky. Obecně platí, že generování testu pro sekvenční obvody je podstatně náročnější než generování testu pro kombinační obvody. Někdy se o obou výše jmenovaných metodách hovoří jako o metodách *deterministických*,
- metody pseudonáhodného generování testů - kromě výše uvedených způsobů generování testu existuje ještě tzv. *pseudonáhodné generování testu*. To je použitelné jak pro kombinační tak sekvenční obvody a v jeho základní formě není nutná znalost struktury ani funkce obvodu. Takový test je obvykle tvořen posloupnostmi testovacích vzorků, které mají ve své základní podobě neměnné pořadí kroků a jistou periodicitu opakování. Základem pseudonáhodného generátoru je např. *posuvný registr s lineární zpětnou vazbou*³⁰ či *celulární automat*. Obvykle však pomocí základního pseudonáhodného generátoru není možné dosáhnout tak vysokého pokrytí poruch jako při použití deterministických metod, respektujících strukturu a/nebo funkci daného obvodu. Tento nedostatek pak lze vyřešit např. kombinací obou metod či např. použitím adaptivního pseudonáhodného generátoru s možností nastavení pravděpodobnosti jedničky na jeho výstupech.

Lokalizace poruch

Narozdíl od detekčních testů umožňují *lokalizační testy* navíc rozlišit, která porucha nastala, čímž dojde i k lokalizaci místa poruchy³¹. Pro technickou praxi jsou významné, protože poskytují informaci o umístění poruchy v číslicovém systému a s požadovanou přesností lokalizace určují buď vyměnitelnou porouchanou součástku pro následující opravu nebo poskytují podklady pro sledování spolehlivosti jednotlivých komponent systému pro řízení jakosti výroby. Kromě základní ekvivalence poruch (její třídy tvoří redukovaný seznam poruch) vzniká při lokalizaci i *ekvivalence topologická*.

³⁰ angl. linear feedback shift register, LFSR

³¹ např. špatného hradla, klopného obvodu, komponenty, jejich vývodů či spojů

Při tvorbě lokalizačního testu se obvykle vychází z tzv. *matice poruch*. Z ní je vytvořen *lokalizační strom*, reprezentující rozklady množiny poruch - cílem je dosažení nulového rozkladu, tj. aby bylo lokalizačním testem možno rozlišit, která konkrétní porucha nastala. Z lokalizačního stromu je pak pro každou poruchu možné určit, jakou konkrétní lokalizační posloupností testovacích vektorů je tato porucha lokalizovatelná.

Na základě lokalizačního stromu lze sestavit tzv. *slovník poruch*, v němž pro každou lokalizační posloupnost nalezneme její interpretaci, tj. příslušnou lokalizovanou poruchu či bezporuchový stav. Jelikož pro složitější obvody může slovník poruch nabývat velkých rozměrů, lze tento slovník zjednodušit např. redukcí na slovník s jednobitovými výsledky, víceúrovňově organizovaný slovník nebo fázově rozdělený slovník.

Kompresie diagnostických dat

Minimalizace slovníku poruch představuje kompresi diagnostických dat pro potřeby lokalizace poruch. S růstem složitosti číslicových obvodů se však zvětšuje také objem diagnostických dat (tj. testovacích vektorů a odezev) potřebných pro detekci poruch do té míry, že komprese se stala nezbytností i zde. Nejčastější metody komprese lze rozdělit na metody čítání událostí, příznakovou analýzu a spektrální metody; podrobněji se však metodám komprese věnovat nebudeme.

2.3.3 Návrh pro snadnou testovatelnost

Snahou algoritmů pro generování testu je vygenerovat pro daný číslicový obvod co nejlepší test, tj. co nejkratší sekvencí testovacích vzorků dosáhnout co největšího pokrytí poruch v daném obvodu. Kromě pokrytí poruch, které je významným ukazatelem kvality testu, lze nalézt i ukazatele nákladů spojených s testem - mezi ně patří zejména náklady na generování testovacích vzorků, náklady na simulaci poruch a generování informace o lokalizaci poruch, náklady na vybavení nutné pro test a náklady na samotný proces testování, jehož ukazatelem je např. doba aplikace testu. Protože náklady spojené s testováním prvku mohou být vysoké tak, že mohou přesáhnout i nejvyšší přípustné náklady na návrh a výrobu obvodu, je důležité, abychom byli schopni uchovat je v rozumných mezích. Jedním ze způsobů, jak toho lze dosáhnout, je využít tzv. *návrhu pro snadnou testovatelnost*, jemuž je věnována tato podkapitola.

Testovatelnost

Testovatelnost je chápána jako charakteristika zohledňující různé náklady spojené s testováním navrhovaného číslicového obvodu - obvykle je chápána jako

ukazatel snadnosti tvorby a provádění efektivních testů³² číslicového obvodu. Obecná definice testovatelnosti sice v současné době neexistuje, avšak existuje řada dílčích, konkrétně aplikačně orientovaných definic a z nich vycházejících metod, z nichž každá je obvykle konstruována pro jistou konkrétní úroveň popisu obvodu a pro zohlednění vybrané podmnožiny nákladů spojených s jeho testováním.

Nejednotnost a mnohdy značná rozdílnost přístupů zabývajících se problematikou testovatelnosti byla podnětem pro standardizaci pojmů z této oblasti. Od roku 1998 pracuje standardizační skupina IEEE č. 20 zabývající se problematikou diagnostiky elektronických systémů na standardu IEEE P1522. Ten se týká standardizace pojmů, zejména měř a vlastností z oblasti testovatelnosti a diagnostiky elektronických systémů. Snahou standardu je poskytnout bezesporné a jednoznačné definice těchto a dalších pojmů. Definice by měly být založeny na vhodných standardizovaných modelech. Pro standard IEEE P1522 by měly být charakteristické zejména následující vlastnosti [SK04]:

- technologická nezávislost - pojmy tohoto standardu by měly být použitelné na libovolnou cílovou technologii,
- implementační nezávislost - míry definované v tomto standardu by neměly být závislé na žádném konkrétním programovém prostředí,
- bezespornost a jednoznačnost definic - všechny definice budou zapsány jak pomocí přirozeného jazyka (angličtiny), tak formálně matematickými prostředky. Dojde-li ke sporu, bude pro jeho vyřešení rozhodující matematický zápis definice,
- měřitelnost definic - míry definované v tomto standardu musí být odvozeny ze standardizovaných modelů a musí vycházet z měřitelných údajů,
- stanovení podpůrných informací - standard by měl jasně definovat, jaké konkrétní podpůrné informace jsou pro jednotlivé definované míry nezbytné, tj. jaké informace musí obsahovat resp. o jaké dodatečné informace je nutno doplnit popis návrhu daného systému, aby bylo možné standardizované míry na takto popsaný systém aplikovat.

Další³³ budou zahrnuty ve zvláštní příloze ke standardu nebo budou součástí doporučení pro praxi. Standardizační proces [SK04] standardu IEEE P1522 zatím není ukončen, avšak je již v jedné z konečných fází příprav standardu.

I když jednotná definice testovatelnosti není v současnosti k dispozici, lze obecně konstatovat, že v dílčích definicích je *zlepšením testovatelnosti* obvodu obvykle myšlena redukce jistých nákladů spojených s testováním tohoto obvodu. K faktorům ovlivňujícím testovatelnost obvodu jsou v současné době zařazovány zejména *řiditelnost (ovladatelnost)*, *pozorovatelnost* a *předvídatel-*

³² za efektivní je obvykle považován krátký test s vysokým pokrytím poruch

³³ tj. matematicky nedefinované pojmy a skutečnosti nesplňující výše uvedené vlastnosti

nost [ABF90]. Postup, který se používá k ohodnocení testovatelnosti obvodu se nazývá *analýza testovatelnosti* - viz kapitola 3.3, strana 47.

Řiditelnost resp. pozorovatelnost bývá chápána jako schopnost ovlivnit³⁴ resp. změřit³⁵ hodnotu signálu v daném místě v obvodu. Je snahou říditelnost resp. pozorovatelnost číselně ohodnotit; hodnota říditelnosti resp. pozorovatelnosti pak obvykle vyjadřuje míru snadnosti nastavení resp. zjištění hodnoty signálu v daném místě obvodu. Předvídatelnost je obvykle chápána jako schopnost odhadu výstupní odezvy obvodu na základě vstupního podnětu [ABF90]. Mezi faktory ovlivňující předvídatelnost patří např. dostupnost vnitřních stavů obvodu, hazardy, oscilátory v obvodu atp. Protože předvídatelnost má smysl uvažovat spíše v případech, kdy je plánováno pseudonáhodné generování testu, předvídatelností se tato práce dále blíže nezabývá.

Řiditelnost resp. pozorovatelnost bývá někdy ohodnocena s ohledem na to, zda test bude pseudonáhodný nebo deterministický. Např. pro pseudonáhodný generátor testu je málo pravděpodobné, že bude schopen nastavit výstup desetivstupového hradla AND na hodnotu 1, zatímco deterministický test založený na D-algoritmu je schopen tento problém vyřešit jediným přístupem do tabulky. Proto se také někdy hovoří o *pseudonáhodné říditelnosti* resp. *pozorovatelnosti*. Ta je obecně obtížná u prvku, vyžadujícího nastavení konkrétní vstupní hodnoty resp. posloupnosti hodnot pro nastavení resp. zjištění jeho stavu. Obecně platí, že obvod či jeho vnitřní prvek (komponenta) je *obtížně říditelný* resp. *pozorovatelný*, je-li pro nastavení resp. zjištění jeho stavu nutná dlouhá posloupnost vstupních vzorků. Typicky *obtížně říditelné* jsou např. obvody se zpětnými vazbami, oscilátory, generátory hodin, čítače, navíc i typicky *obtížně pozorovatelné* jsou sekvenční obvody, obvody s globálními zpětnými vazbami, obvody pro detekci a korekci chyb apod. Obecně lze učinit tyto závěry [ABF90]:

- sekvenční obvody jsou obtížněji testovatelné než kombinační,
- řídicí logika je obtížněji testovatelná než logika datových cest,
- náhodná logika je obtížněji testovatelná než strukturované, sběrniceově orientované návrhy,
- asynchronní obvody jsou obtížněji testovatelné než synchronní.

Poznamenejme, že definice testovatelnosti a metody analýzy testovatelnosti existují i pro analogové a smíšené obvody. Tato práce se však bude věnovat pouze jisté podmnožině číslicových obvodů³⁶.

Techniky návrhu pro snadnou testovatelnost

V tomto oddíle budou uvedeny některé z nejčastěji používaných technik návrhu pro snadnou testovatelnost (zkráceně budeme tyto techniky označovat pojmem *DFT techniky*).

³⁴ např. nastavit signál v datové cestě na danou hodnotu

³⁵ např. zjistit hodnotu signálu v daném uzlu

³⁶ tato podmnožina bude vymezena modelem popsáním v kapitole 4

Obecným cílem DFT technik je, aby v důsledku jejich použití během etapy návrhu obvodu byl vytvořen návrh splňující vstupní specifikaci a vyznačující se snadnou testovatelností výsledného obvodu.

DFT techniky mají kromě pozitivního dopadu z hlediska testovatelnosti také negativní důsledky. Mezi ně patří např. zvětšení plochy čipu, které vede jednak ke zvýšení příkonu a tím k většímu zahřívání obvodu a zvýšení požadavků na odvod tepla a jednak ke snížení výtěžnosti s důsledkem vyšší ceny čipu, zvýšení počtu vývodů obvodu a zhoršení dynamických vlastností obvodu. Během návrhu pro snadnou testovatelnost je proto nutné volit vhodný kompromis mezi negativními důsledky použití DFT technik a testovatelností. Jde vlastně o kompromis mezi požadavky návrháře a diagnostika, který poněkud ztrácí na významu v dnešní době, kdy jsou pro účely nalezení vhodného kompromisu používány automatizované CAD nástroje.

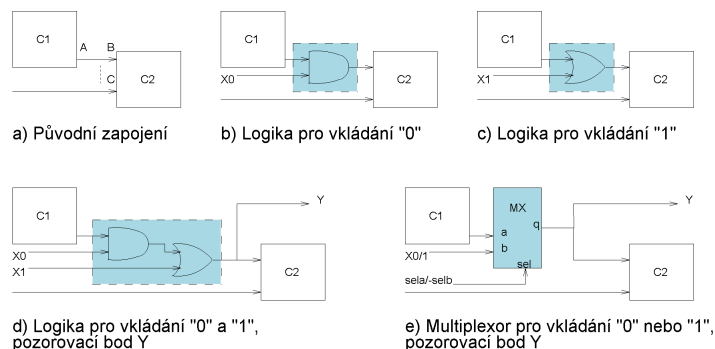
Následující text bude věnován principům některých DFT technik [ABF90]. DFT techniky se obvykle dělí na dvě skupiny. Do první z nich patří tzv. *ad-hoc techniky*, tzn. techniky použitelné v konkrétních speciálních případech, druhá skupina pak obsahuje obecně použitelné techniky - tzv. *techniky strukturovaného návrhu*. Tyto metody doznaly největšího rozvoje v minulém desetiletí a jsou použitelné prakticky na libovolné úrovni popisu číslicového systému. Vybrané ad hoc techniky jsou uvedeny pouze z ilustrativních důvodů a jako ukázka postupů, které se při dnešním stupni integrace používají velmi zřídka. Druhou skupinou metod jsou metody strukturovaného návrhu.

1. Vybrané ad hoc techniky

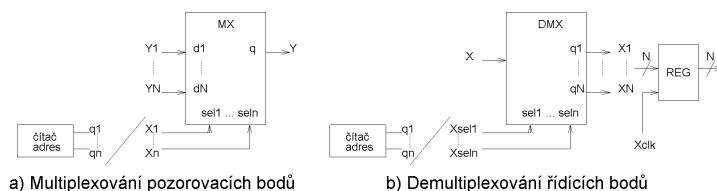
- a) Testovací body - jejich cílem je zlepšení říditelnosti a/nebo pozorovatelnosti daného místa obvodu. Rozlišují se dva typy testovacích bodů - *řídící* resp. *pozorovací body* používané ke zlepšení říditelnosti resp. pozorovatelnosti v daném místě obvodu. Příklady 1-bitových testovacích bodů jsou uvedeny na obrázku 3. řídícím bodem je obvykle chápáno obvodové místo (uzel) ovládané přídavným primárním vstupem, určeným k nastavení hodnoty "logická 0" (viz obrázek 3b, primární vstup X0) resp. "logická 1" na daný uzel (viz obrázek 3c, primární vstup X1). Obdobně pozorovacím bodem bývá chápán uzel sledovatelný přídavným primárním výstupem (viz obrázek 3, body d a e, primární výstup Y), umožňujícím sledovat hodnotu vyskytující se na daném uzlu. Pro účely vkládání libovolné logické hodnoty na dané místo obvodu je třeba většího počtu primárních vstupů a složitější logiky (viz obrázek 3d, e).

Nevýhodou této techniky je nárůst plochy, způsobený vložením jednoúčelové logiky testovacích bodů do původní obvodové struktury; hlavní nevýhodou této techniky je však skutečnost, že počet primárních vstupů a výstupů (tj. vývodů obvodu) rezervovaných pro účely testu, roste lineárně s počtem použitých řídících a pozorovacích bodů. Tuto nevýhodu je možné částečně zmírnit - a to za cenu přidání další logiky:

2.3 Diagnostika číslicových obvodů 23



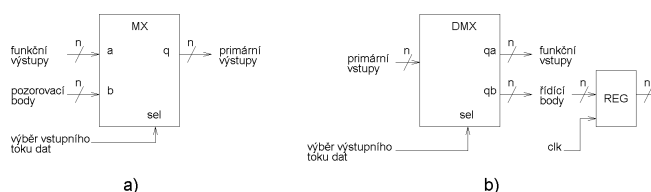
Obrázek 3. Testovací body



Obrázek 4. Testovací body a úspora vývodů obvodu pomocí adresace bodů

- i. a obrázku 4a, je ke snížení počtu primárních výstupů pro účely sledování hodnot na pozorovacích bodech Y_1, \dots, Y_N použit multiplexor - místo N primárních výstupů rezervovaných pro pozorovací body je třeba pouze jeden primární výstup Y a $n = \lceil \log N \rceil$ primárních vstupů X_1, \dots, X_n sloužících pro adresaci pozorovacích bodů. Nevýhodou této modifikace je fakt, že v jednom okamžiku lze sledovat hodnotu pouze jednoho pozorovacího bodu, což vede k prodloužení doby testu. Je-li počet primárních vstupů pro účely testu i po této modifikaci příliš velký či je-li třeba pro malou sérii testovacích vzorků sledovat dlouhou posloupnost hodnot na různých pozorovacích bodech, je vhodné pro řízení adresových vstupů multiplexoru použít čítač,
- ii. obdobně (viz obrázek 4b) lze použít demultiplexor pro snížení počtu primárních vstupů rezervovaných k ovládání hodnot na řídicích bodech X_1, \dots, X_N . Místo původního počtu primárních vstupů je díky demultiplexoru třeba pouze jeden primární vstup pro nastavení hodnoty na adresovaný řídicí bod a $n = \lceil \log N \rceil$ primárních vstupů sloužících pro adresaci řídicích bodů; je-li pro adresování použit čítač, lze primární vstupy, určené pro adresaci řídicích bodů, uspořít. Vyžaduje-li to test, lze na výstup demultiplexoru připojit registr, který umožní nastavit hodnoty na několik řídicích bodů současně,

24 2 Související oblasti



Obrázek 5. Testovací body a úspora vývodů obvodu pomocí sdílení datových cest

iii. dalším způsobem, jak snížit počet obvodových vývodů rezervovaných pouze pro účely testu, je zavedení časového sdílení vybraných vývodů obvodu - na obrázku 5a, je do obvodové struktury vložen multiplexor sloužící k přepínání toku dat buď z funkční logiky obvodu nebo z vybraných pozorovacích bodů na dané primární výstupy. Obdobný účel plní demultiplexor na obrázku 5b. Ten slouží k přepínání toku dat z daných primárních vstupů buď na funkční vstupy obvodu nebo na vybrané řídicí body. Pro uchování hodnot na řídicích bodech může být použit přídatný registr.

Tato technika snadno ilustruje fakt, že aplikace DFT techniky vede ke zlepšení testovatelnosti obvodu za cenu jisté modifikace jeho struktury a že je třeba dosáhnout přijatelného kompromisu mezi přídatnou logikou (tj. plochou), počtem vývodů obvodu a dobou aplikace testu. Výběr konkrétních testovacích bodů je dán zkušeností návrháře/diagnostika - dle [ABF90] lze za typické kandidáty na řídicí body považovat např.:

- řídicí, adresové a datové vodiče u sběrníkových systémů,
- povolovací vstupy paměťových prvků,
- synchronizační vstupy a vstupy pro nastavení/nulování klopných obvodů, čítačů, posuvných registrů a dalších paměťových prvků,
- body ležící v globálních zpětných vazbách,
- adresové vstupy multiplexorů a demultiplexorů.

Obdobně [ABF90], vhodnými kandidáty na pozorovací body mohou být např.:

- body, které se vyznačují značným větvením,
- body ležící v globálních zpětných vazbách,
- výstupy vícevstupových prvků,
- výstupy paměťových prvků,
- vodiče adresových, řídicích a datových sběrnic.

b) inicializace - snahou této techniky je zajistit (např. při zapnutí napájení či jistou posloupností vzorků), aby bylo možné nastavit sekvenční části obvodu do požadovaného stavu. Možnost snadné inicializace sekvenčních částí obvodu totiž obecně vede ke kratším testům, protože odpadá nutnost generování testovacích vzorků, majících význam výhradně/jedině pro inicializaci dané sekvenční části obvodu; v ideálním

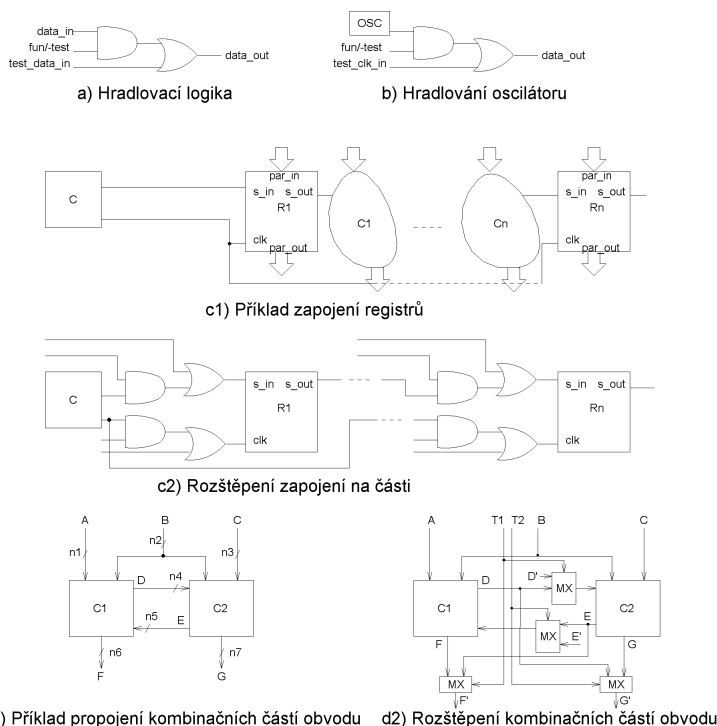
případě je pak možné i pro sekvenční obvody použít kombinačního generátoru testu. Kvůli snadné inicializaci je tedy vhodné používat sekvenční prvky vybavené alespoň jedním inicializačním³⁷ vstupem, použít vlastní inicializační logiku či např. některou z technik strukturovaného návrhu. Nevýhodou je, že pokud se v obvodové části sloužící pro inicializaci vyskytne porucha, obvykle je tato porucha velmi obtížně detekovatelná,

- c) segmentace obvodů - cílem této techniky je fyzicky rozdělit daný systém do několika dílčích částí tak, aby byla zajištěna co nejlepší testovatelnost každé z nich a výsledně i systému jako celku. Technika spočívá v hradlování obtížně říditelných uzlů obvodu a následném zajištění jejich externí říditelnosti během tzv. *testovacího režimu činnosti* obvodu. Na obrázku 6b, je uveden příklad hradlování výstupu oscilátoru - hradlovací logika umožňuje (při $fun/-test = 0$) blokování tohoto výstupu a jeho nahrazení signálem z *test_clk.in* sloužícím pro účely testu. Na obrázku 6, část c1, je uvedeno zapojení n registrů. Taková a obdobná zapojení sekvenčních prvků (např. čítače) jsou obecně obtížně testovatelná, jelikož k jejich testu je obvykle potřeba dlouhých posloupností testovacích vzorků. Proto je snahou taková zapojení rozčlenit na několik částí (viz obrázek 6c2) tak, aby tato modifikace přispěla k co nejefektivnějšímu testu. Avšak i spojením kombinačních obvodů může vzniknout obtížně testovatelná struktura (viz obrázek 6d1); ke zlepšení její testovatelnosti lze opět použít segmentace obvodů - v tomto případě se jedná o rozštěpení struktury na dvě samostatně testovatelné části, $C1$ a $C2$ (viz obrázek 6d2) pomocí multiplexorů. Nevýhodou této techniky je, že vlivem přídavné logiky může vést ke značné degradaci výkonu systému,
- d) omezení redundance - snahou je vyhnout se při návrhu používání redundantní logiky - ta může způsobit zavedení chyb velmi obtížně detekovatelných statickým testem. Pokud se totiž v redundantní logice objeví porucha, může tato porucha ovlivnit testovací vzorky resp. odezvy příslušející danému statickému testu. Problémem však je, jak algoritmicky rozlišit, která logika je skutečně redundantní a nepotřebná³⁸, a která je sice redundantní, ale úmyslně (např. logika pro zvýšení spolehlivosti daného zařízení pomocí bezpečnostních kódů apod.) [ABF90]. Problémům, které tato logika způsobuje při testu, je také možné se vyhnout bez nutnosti jejího odstanění - např. pomocí testovacích bodů či segmentace obvodů,
- e) rozbití globálních zpětných vazeb - v obvodové struktuře se mohou vyskytovat dva typy globálních zpětných vazeb; první se vyznačuje neexistencí sekvenčního prvku ve své datové cestě, druhý typ naopak sekvenční prvek v datové cestě obsahuje. Globální zpětnou vazbou

³⁷ obvykle tzv. nulovacím či resetovacím

³⁸ např. v důsledku špatně zvoleného návrhového postupu

26 2 Související oblasti



Obrázek 6. Segmentace obvodů

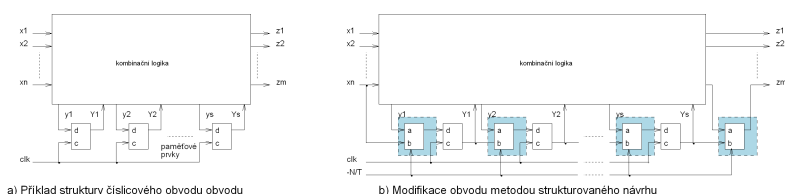
je obvykle chápán každý cyklus (kružnice) obsahující větší obvodové celky (např. registry, funkční jednotky); za globální zpětnou vazbu nejsou považovány např. zpětné vazby mezi NAND resp. NOR členy klopného obvodu typu R-S. Zlepšení pozorovatelnosti resp. říditelnosti místa ležícího v datové cestě globální zpětné vazby lze dosáhnout např. vložení pozorovacího bodu resp. vložení řídicího bodu či multiplexoru. Je-li zajištěna testovatelnost dané zpětné vazby, hovoříme často o "rozbití" této zpětné vazby. Dovoluje-li to struktura obvodu, lze v obou případech pro rozbití zpětných vazeb použít některé z technik strukturovaného návrhu (viz níže).

2. Princip technik strukturovaného návrhu

Uvažujme sekvenční číslicový obvod podle obrázku 7a. Takový obvod sestává jednak z kombinační logiky a jednak z bloku paměťových prvků obsahujících stav obvodu (automatu). Okamžitá hodnota výstupních signálů závisí na hodnotách vstupních logických proměnných a na stavu paměťových prvků. Jeden krok testu tohoto obvodu by sestával z těchto akcí [ZK99]:

- a) nastavení požadovaného vnitřního stavu obvodu a zajištění testovacího vzorku na vstupech kombinační sítě,

2.3 Diagnostika číslicových obvodů 27



Obrázek 7. Ilustrace k principu strukturovaného návrhu

- b) provedení jednoho kroku činnosti obvodu generováním synchronizačních pulsů na clk,
- c) kontrola výsledku testu, tj. odezvy na testovací vzorek a cílového vnitřního stavu obvodu.

Je zřejmé, že některé činnosti značně komplikují provedení testu, pokud je jejich provedení vůbec možné. Platí to zejména o nastavení požadovaného stavu obvodu a zjištění aktuálního stavu obvodu. Předpokládejme, že paměťové prvky tohoto obvodu nejsou říditelné/pozorovatelné. Zlepšení říditelnosti a pozorovatelnosti pak můžeme dosáhnout např. tím, že údajovému vstupu každého z klopných obvodů předřadíme multiplexor, na jehož vstup *a* přivedeme výstupy kombinační sítě, na vstup *b* výstup předcházejícího klopného obvodu (obrázek 7b). V režimu $-N/T = 1$ tvoří jednotlivé klopné obvody paralelní registr, do něhož se ukládají výstupy y_1, \dots, y_s kombinační sítě. V režimu $-N/T = 0$ jsou jednotlivé klopné obvody zapojeny jako posuvný registr se vstupem x_n a výstupem z_m . Tato úprava umožňuje:

- vložit před provedením kroku testu do posuvného registru požadovaný vzorek,
- zkontrolovat na výstupu z_m odezvu na vstupní vzorek tvořený kombinací $x_1, \dots, x_n, Y_1, \dots, Y_s$

Použitím některé ze strukturálních metod na sekvenční obvod se tedy oddělí kombinační logika od sekvenčních prvků³⁹ a problém testování kombinačních sítí, které jsou součástí sekvenčního obvodu, se zjednoduší. Zjednodušení spočívá zejména v tom, že pro generování testu takto upraveného obvodu je možné použít generátory používané pro kombinační obvody⁴⁰. Pokud nebyl obvod podle některé z metod strukturovaného návrhu upraven, musí být pro vytvoření testu použit GTSO⁴¹. Obecně platí, že vytvoření testu pro sekvenční obvod je výpočetně náročnější než je tomu u kombinačního obvodu.

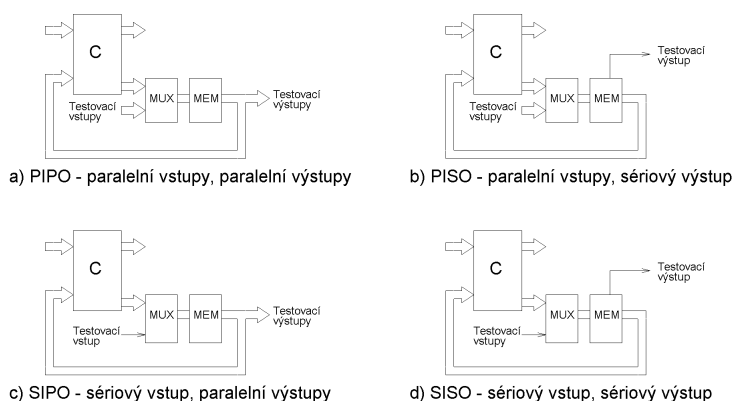
V odborné literatuře byla principům strukturovaného návrhu věnována výrazná pozornost již v 80. letech. Tento typ DFT technik byl přijat všemi

³⁹ odtud název metody strukturovaného návrhu

⁴⁰ tzv. generátory testu kombinačních obvodů, GTKO

⁴¹ tzv. generátor testu sekvenčních obvodů

28 2 Související oblasti



Obrázek 8. Ilustrace principu strukturovaného návrhu

významnými světovými výrobci číslicových obvodů a v současné době existuje celá řada snímacích⁴² technik, založených na těchto a obdobných principech.

Největší popularity dosáhl strukturovaný návrh v systémech IBM pod názvem LSSD (Level-Sensitive Scan Design) [TS82]. Další obdobné metody strukturovaného návrhu se označují Scan/Set (metoda sledování/nastavení) využívaná v 80. letech především firmou Sperry Univac a metoda Scan Path (metoda přístupové cesty), která našla největší uplatnění ve výrobcích firmy Nippon Electric Company. Na principech metod strukturovaného návrhu jsou založeny i současné metody, techniky a nástroje návrhu pro snadnou testovatelnost - např. VirtualScan [Syn04] společnosti Syn-test Technologies či hraniční snímání dle standardu IEEE 1149.1. Podle způsobu zápisu informace do paměťových prvků se tyto metody označují jako metody sériové a souhrnně jsou označovány zkratkou SAS⁴³. Naopak paralelní metodou je metoda RAS⁴⁴ firmy Fujitsu, příp. její modifikace metoda ARAS (Amdahl RAS).

Obě existují čtyři základní testovací strategie (viz obrázek 8). Základními jsou strategie *SISO* využívající sériového přísunu testovacích vzorků a sériového sledování odezev a strategie *PIPO* využívající paralelního přísunu testovacích vzorků a paralelního sledování odezev. Kombinací strategií *SISO* a *PIPO* získáme testovací strategie *SIPO* resp. *PISO*, vyznačující se sériovým přísunem testovacích vzorků a paralelním sledováním odezev resp. paralelním přísunem testovacích vzorků a sériovým sledováním odezev.

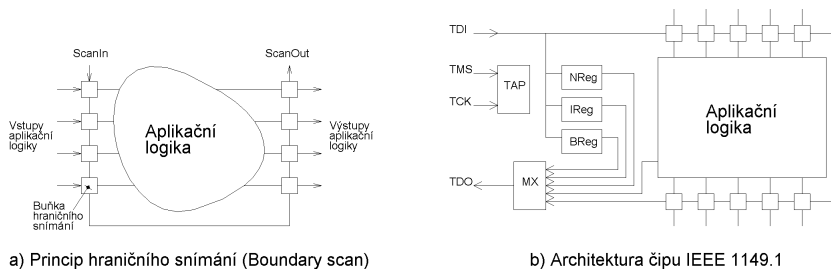
Stručná charakteristika vybraných metod [ABF90]:

⁴² tzv. scan

⁴³ z angl. serial access scan

⁴⁴ z angl. random access scan

- a) SAS - princip této metody spočívá v rozčlenění obvodu na kombinační části oddělené paměťovými prvky vytvářejícími posuvný registr, který v režimu testu umožňuje sériový přenos diagnostických dat z primárního vstupu obvodu resp. na primární výstup obvodu.



Obrázek 9. Hraniční snímání

Během normálního režimu činnosti plní každá paměťová buňka původní funkci. Jelikož při použití této metody je diagnostický režim od funkčního režimu striktně oddělen, výsledné uspořádání obvodu obecně neumožňuje uchovávat data a výsledky výpočtu v paměťových prvcích nezměněné - sériové vložení resp. čtení diagnostických dat je u SAS destruktivní.

Na obrázku 9 je uveden příklad tzv. hraničního snímání⁴⁵ - aplikační logika zde představuje původní logiku čipu (včetně případných DFT struktur), tj. logiku před aplikací prostředků IEEE 1149.1. Původní vývody čipu jsou přístupné přes tzv. buňky hraničního snímání. Ty během normálního režimu plní funkci původních vývodů a během režimu testu pracují jako paměťové buňky posuvného registru, což umožňuje sériovou cestou (ze ScanIn) přivádět testovací vzorky na původní vstupy obvodu resp. odvádět odezvy (na ScanOut) z původních výstupů obvodu. Testovací logika IEEE 1149.1 obsahuje zejména tyto bloky:

- instrukční registr (IREG) - jedná se o posuvný registr, do kterého je sériově nahrána instrukce určující typ požadované operace a registry pro testovací data. Důležité je, že každá instrukce zaručuje existenci sériové cesty z TDI na TDO. IEEE 1149.1 požaduje implementaci instrukcí BYPASS (možnost zkrácení sériové cesty z TDI na TDO přes 1-bitový registr BREG), EXTEST (instrukce pro umožnění externího testování obvodu. Sériová cesta z TDI na TDO je tvořena všemi buňkami hraničního snímání příslušejícími danému obvodu a je využívána pro sériový přísun testovacích vzorků resp. snímání odezev) a SAMPLE (umožňuje během normál-

⁴⁵ IEEE 1149.1 Boundary Scan Standard

30 2 Související oblasti

ní operace obvodu uložit hodnoty vyskytující se na jeho vývodech) a doporučuje implementaci nepovinných instrukcí INTEST nebo RUNBIST,

- registry testovacích dat - jedná se o několik posuvných registrů (nejméně však dva - BREG a posuvný registr tvořený buňkami hraničního snímání). Podle aktuální instrukce jsou vybrané z nich sériovou cestou naplněny testovacími vzorky nebo jinými požadovanými daty. Po provedení instrukce jsou výsledky uloženy do registru testovacích dat a mohou být sériovou cestou snímány,
- řadič TAP - jedná se o synchronní stavový automat, generující řídicí a hodinové signály nutné pro činnost ostatních bloků IEEE 1149.1. TAP pracuje s těmito signály: vstupní testovací hodiny (TCK - tento signál je nezávislý na systémových hodinách obvodu, což umožňuje, aby byly testovací operace synchronizovány např. v případě testování několika prvků na desce), vstup pro výběr testovacího režimu (TMS - vstup sloužící pro změnu vnitřního stavu řadiče TAP, tj. pro změnu instrukce či reset řadiče do počátečního stavu), vstup testovacích dat (TDI - data přiváděná na tento sériový vstup jsou, v závislosti na instrukci, buď ukládána do registru testovacích dat nebo do instrukčního registru), výstup testovacích dat (TDO - v závislosti na instrukci jsou na tento sériový výstup přiváděna data buď z registru testovacích dat nebo z instrukčního registru), nepovinný vstup pro asynchronní reset testu (-TRST - přivedením hodnoty logická 0 na tento vstup bude TAP asynchronně uveden do počátečního stavu).

Vzhledem k významu SAS techniky pro praxi a k jejímu vztahu k této práci se SAS technice bude podrobněji věnovat kapitola 3.1, strana 35,

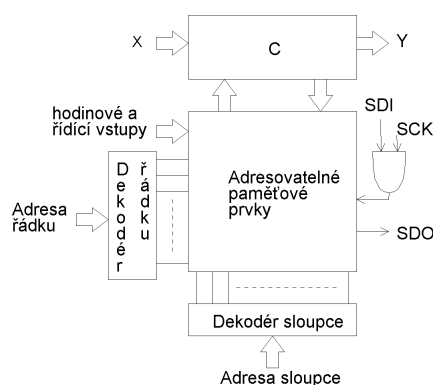
- b) stínový registr - princip této metody spočívá ve zdvojení registrů vybraných pro ovládání/snímání vnitřního stavu obvodu (obrázek 13d, strana 39); během sledování stavu obvodu je obsah původního registru (RegA) paralelně zkopírován do stínového registru (RegB) a sériovou cestou sledován na primárním výstupu obvodu (ScanOut).

Obdobně při nastavování stavu obvodu je nejprve sériovou cestou z primárního vstupu (ScanIn) nastaven obsah stínového registru (RegB) a následně je paralelně zkopírován do původního registru (RegA).

Výhodou této metody je, že během přenosu diagnostických dat nedochází k ovlivnění dat v původních datových cestách obvodu; nevýhodou je nutnost vložení přídatného registru,

- c) RAS - paměťové prvky nejsou propojeny, aby utvářely posuvný registr, ale jsou konstruovány jako adresovatelné buňky paměti s náhodným přístupem (obrázek 10). Hlavní výhodou je tedy rychlý přístup k paměťovým prvkům. I zde je striktně oddělen normální režim od režimu diagnostického (např. jako u SAS metod), avšak během nastavování obsahu konkrétního paměťového prvku nedochází k destrukci hodnot uložených v ostatních paměťových prvcích a při sledování hodnoty

paměťového prvku dokonce nedochází k destrukci hodnot v žádném z paměťových prvků.



Obrázek 10. Struktura obvodu využívajícího RAS

Během normálního režimu činnosti plní každá paměťová buňka původní funkci. V diagnostickém režimu je zvolený paměťový prvek adresován a pomocí datového vstupu *SDI* a hodinového vstupu *SCK* naplněn novou hodnotou resp. je jeho hodnota sledována na výstupu *SDO*. Nevýhodou RAS techniky je nutnost vybavit obvod, používající tuto techniku, logikou pro adresování paměťových buněk.

2.3.4 Principy vestavěné diagnostiky

Kromě DFT technik, používaných převážně pro účely vnějšího testu, existují techniky tzv. *vestavěného autonomního (samočinného) testování*⁴⁶. Jedná se o techniky, které nepředpokládají přísun velkého množství testovacích vzorků z primárních vstupů obvodu resp. sledování velkého množství odezev na primárních výstupech obvodu, ale umožňují, aby obvod obsahoval podčásti pro jeho autonomní testování. Obecně, při použití BIST má každá testovaná jednotka (komponenta, prvek) svůj vestavěný zkoušeč, který komunikuje s okolím na úrovni velice jednoduchých povelů (start/stop, dobře/špatně). Test může být proveden kdykoliv⁴⁷, protože testovanou jednotku není třeba ke zkoušeči připojovat, jak tomu je v případě externího testu. Navíc lze tímto způsobem testovat i několik jednotek současně, protože vestavěné zkoušeče daných jednotek jsou vzájemně nezávislé; zbývá pouze vyhodnotit celkový výsledek samočinného testování těchto jednotek. Kromě možnosti nezávislého testování jednotek patří k dalším výhodám vestavěného zkoušeče např. menší množství diagnostických dat přenášených během testu, vyšší

⁴⁶ angl. built-in self-test, BIST

⁴⁷ záleží na typu BIST

rychlost testování, vyšší pokrytí poruch, ochrana intelektuálního vlastnictví, podpora znovupoužití testu při opakovaném použití návrhu. Hlavní nevýhodou je cena obvodového vybavení věnovaného na realizaci vestavěné diagnostiky zvláště pro každou testovanou jednotku.

DFT a BIST jsou slučitelné a lze tedy, zejména za účelem dosažení efektivnějšího testu, oba typy technik kombinovat - tato kombinace je dokonce přímo podporována standardem (IEEE 1149.1) hraničního snímání, viz strana 29.

Jelikož tato práce předpokládá, že obvod bude testován vnějším testem, z oblasti BIST technik budou uvedeny pouze nejzákladnější pojmy. BIST techniky je možné rozdělit na následující skupiny [ABF90]:

- *On-line BIST* - autonomní testování obvodu probíhá za podmínek pro běžnou funkci obvodu, tj. obvod není třeba vybavovat diagnostickým režimem činnosti. On-line BIST lze rozdělit na:
 - *souběžný* - je typem, kdy je testování umožněno zároveň s původní činností obvodu. Obvykle je toho dosaženo pomocí kódovacích technik (bezpečnostní detekční a korekční kódy), duplikace a porovnávání,
 - *nesouběžný* - lze využít tehdy, není-li vyžadováno souběžné testování a postačuje-li obvod testovat v jeho klidovém stavu, tj. stavu, kdy dokončil provádění své normální činnosti a čeká na podnět k zahájení další činnosti. Technicky bývá tento typ testu zajištěn pomocí softwarového kódu (makrokód) a/nebo diagnostického firmware kódu (mikrokód). Průběh testu lze přerušit a umožnit tak zahájení další činnosti obvodu,
- *off-line BIST* - řeší způsob autonomního testování v situaci, kdy obvod nepracuje v normálním režimu činnosti, tzn. je nutná existence tzv. režimu testu. Nevýhodou oproti on-line BIST je obecně pozdější odhalení případné poruchy, protože off-line BIST test není spouštěn tak často jako on-line BIST test; výhodou je nižší cena off-line BIST řešení. Testování pomocí off-line BIST je obvykle uskutečňováno pomocí
 - vestavěných generátorů testovacích vzorků⁴⁸ (nejčastěji lineární zpětnovazebný posuvný registr (LFSR⁴⁹) nebo celulární automat),
 - části pro kompresi odezev (např. výpočet parity, syndromu, počítání přechodů $0 \leftrightarrow 1$, příznaková analýza⁵⁰ či použití víceúčelového prvku BILBO⁵¹),
 - analyzátorů výstupních odezev⁵²
 - a/nebo diagnostických mikrokódů.

Off-line BIST lze rozdělit na:

⁴⁸ angl. test pattern generator, TPG

⁴⁹ angl. linear feedback shift register, LFSR

⁵⁰ obvykle založená na CRC (angl. cyclic redundancy checking/code) pomocí LFSR

⁵¹ angl. built-in logic block observer, BILBO

⁵² angl. output response analyzer, ORA

- *funkční* - prováděný funkční test (viz strana 18) vychází z funkčního popisu obvodu a obvykle vysokoúrovňového modelu poruch,
- *strukturální* - test vznikl na základě předchozí analýzy obvodové struktury, jedná se tedy o strukturální test (viz strana 17). Pro generování testu resp. kompresi odezev je obvykle používán LFSR.

Na BIST jsou kladeny obdobné požadavky jako v případě použití DFT technik; zejména se požaduje co nejkratší doba testu, vysoké pokrytí poruch, minimální nárůst plochy a počtu vývodů způsobený implementací BIST, nenáročnost a jednoduchost implementace testovacích struktur pro BIST.

Na závěr zmiňme, že kromě výše uvedených i dalších metod návrhu pro snadnou testovatelnost a principů vestavěné diagnostiky (které lze považovat za souhrn dodatečných prostředků pro zlepšení testovatelnosti číslicového systému, tedy za jistý zásah do původního návrhu) byly dokonce vytvořeny tzv. MFT⁵³ zásady (viz např. [PVA92]). Jejich respektování při popisu číslicového systému ve VHDL by mělo vést k výraznému zlepšení testovatelnosti celého systému a tedy k minimalizaci nezbytnosti použití technik návrhu pro snadnou testovatelnost. Hlavní z těchto zásad jsou:

- oddělovat řídicí část od datové,
- sdružovat do registrů co nejvíce klopných obvodů řízených stejným hodinovým signálem,
- zamezit vícenásobnému použití pomocných proměnných,
- zajistit přiřazení datových signálů pro libovolnou kombinaci řídicích signálů a současně zabránit redundantním podmínkám.

Další zásady vyžadují bližší znalost VHDL a proto zde nejsou uvedeny.

2.4 Shrnutí

Tato kapitola se věnovala přehledu základních pojmů z oblasti návrhu a diagnostiky číslicových systémů. V úvodní podkapitole byly zmíněny základní způsoby a úrovně popisu číslicových systémů, z nichž je z hlediska této práce významná úroveň meziregistrových přenosů - jak bude z pozdějšího textu patrné, právě pro tuto úroveň je konstruován jak matematický model číslicového systému (kapitola 4), tak na jeho základě popsána metoda analýzy testovatelnosti (kapitola 5). Volba úrovně popisu číslicového systému je významná nejen z hlediska pohledu na daný systém, ale také z hlediska analýzy a návrhu daného systému. Základním pojmům z oblasti návrhu, jakými jsou přehled prostředků pro popis číslicových systémů (na zvolených úrovních abstrakce) a přehled etap vzniku číslicového obvodu, se zabývala druhá podkapitola. V ní lze mj. nalézt informaci o možnosti vysokoúrovňového návrhu číslicových systémů využívajícího knihovny prvků, informaci o vysokoúrovňové syntéze,

⁵³ z angl. modeling for testability

34 2 Související oblasti

jejímž výsledkem je strukturální popis daného obvodu, či informaci o nezbytnosti povýrobního testování vyrobených obvodů. Model číslicového systému používaný v této práci lze z pohledu vysokoúrovňové syntézy považovat za její výsledný produkt; veškeré postupy navržené pro tento model lze pak z tohoto úhlu pohledu považovat za postupy aplikované v oblasti mezi vysokoúrovňovou a logickou syntézou. Předpokládáme přitom, že knihovna prvků, používaná při vysokoúrovňové syntéze bude pro každý prvek, kromě informací nezbytných pro návrh a implementaci systému, obsahovat také informace související s jeho testováním (např. test daného prvku, informaci pro usnadnění generování testu, usnadnění analýzy testovatelnosti nebo informaci usnadňující návrh či syntézu pro snadnou testovatelnost). Informace, kterou je třeba ke každému prvku pro tyto účely z pohledu této práce dodat, je popsána v podkapitole 4.2.

Poslední podkapitola byla pojata jako úvod do problematiky související s testováním číslicových systémů a její hlavní snahou bylo zmínit základní pojmy z oblasti diagnostiky číslicových systémů. Podstatným bylo zejména seznámení s úvodem k testovatelnosti číslicových systémů důležitým z hlediska tématu této práce a s vybranými souvisejícími oblastmi, zejména se základy problematiky generování testů a několika principy zajištění snadné testovatelnosti číslicových obvodů. Některé z těchto pojmů budou v dalším textu běžně používány a bude na ně navázáno.

3

Blízká témata

Předchozí text byl věnován stručnému představení oblastí, do kterých spadá problematika řešená v této práci; jeho cílem bylo představit základní pojmy z oblasti návrhu a diagnostiky číslicových systémů. Tato kapitola bude věnována zejména tématům, které velmi úzce souvisejí s touto prací. Hlavním cílem této kapitoly je přiblížit vybrané pojmy, principy a metody, s nimiž tato práce souvisí to tak, aby bylo možné správně zařadit problematiku řešenou v této práci a aby toto přiblížení přispělo zejména ke snazšímu popisu a pochopení následujících kapitol, které se problematice řešené v rámci této práce věnují.

3.1 Návrh pro snadnou testovatelnost s využitím techniky scan

Prvním tématem, kterému se budeme v této kapitole věnovat, je tzv. *scan technika* - toto téma již bylo stručně zmíněno v části uvádějící přehled technik návrhu pro snadnou testovatelnost (strana 21 a následující, blíže pak od strany 26), v následujícím textu mu bude věnováno více prostoru. Jedná se o techniku velmi populární, což je dáno zejména její univerzální použitelností na mnoha úrovních návrhu a také při různých návrhových stylech. V této práci bude tato technika použita k ověření a zhodnocení navržených postupů.

Vhodnou aplikací této techniky lze vždy oddělit kombinační část od sekvenční pro zajištění snadné říditelnosti (ovladatelnosti) a pozorovatelnosti (sledovatelnosti) vnitřního stavu obvodu a dosáhnout tak toho, že budeme do jisté míry schopni zastoupit generátor testu pro sekvenční obvody generátorem testu pro kombinační obvody. Obvykle jsou rozlišovány dvě základní varianty techniky scan¹ lišící se ve způsobu přístupu k paměťovým prvkům vybraným pro test (tzv. *scan buňkám*) obvodu - technika RAS, založená na přímé adresaci scan buněk a technika SAS (sériový scan), založená na řetězení

¹ pro bližší zařazení obou technik viz strana 29 resp. strana 30

scan buněk do posuvných (tzv. *scan*) registrů, umožňujících sériové ovládání resp. sledování stavů scan buněk zřetězených (zařazených) do tohoto registru. Struktura scan buněk umožňuje, aby tyto během tzv. *normálního režimu* činnosti plnily svou původní funkci a během tzv. *režimu testu* sloužily k diagnostickým účelům, tj. v případě techniky SAS, aby tvořily scan registr, umožňující přenos diagnostických dat mezi vnitřními uzly obvodu a vývody obvodu. Obecně může být v obvodě scan registrů více a jejich řetězením lze pak vytvářet tzv. *scan řetězy*. Nebude-li uvedeno jinak, budeme v dalším textu pod označením scan resp. scan technika rozumět pouze techniku sériový scan.

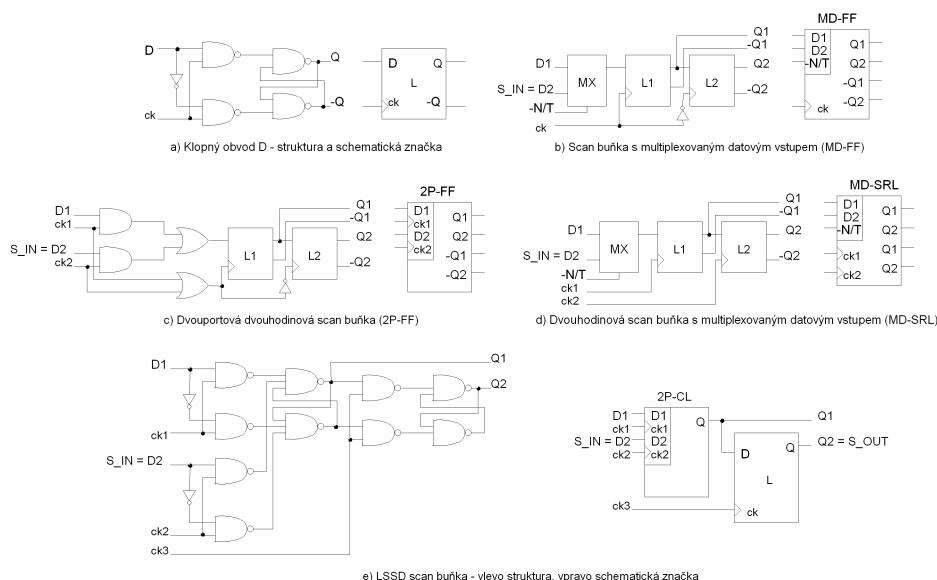
V této kapitole budou stručně prezentovány základní pojmy týkající se techniky sériový scan. Budou představeny vybrané varianty scan buněk pro techniku sériový scan, uvedeny základní typy scan technik a nastíněny typické problémy spojené s aplikací této techniky.

3.1.1 Vybrané varianty scan buněk

Na obrázku 11a, je příklad struktury a schematická značka klopného obvodu D, používaného k ilustraci struktur vybraných scan buněk. Značení na obrázku 11 je provedeno tak, že vstup D1 resp. D2 představuje vstup dat v normálním resp. testovacím režimu činnosti buňky a výstup Q1 resp. Q2 představuje výstup dat v normálním resp. testovacím režimu činnosti buňky. Následuje stručný komentář k vybraným variantám scan buněk [ABF90]:

- MD-FF buňka - v závislosti na hodnotě adresového vstupu multiplexoru MX (obrázek 11b) jsou do paměťové buňky ukládána data normálního režimu činnosti ($-N/T = 1$) či režimu testu ($-N/T = 0$). Paměťová buňka je tvořena dvěma D klopnými obvody, zapojenými jako řídicí (L1) a řízený (L2), má dva datové vstupy, jeden hodinový vstup a jeden vstup pro přepínání režimu,
- 2P-FF buňka - tato varianta scan buňky (obrázek 11c) je vhodná zejména tehdy, je-li potřeba oddělit hodiny normálního režimu od hodin režimu testu. Kromě dvou datových vstupů má tato buňka dva hodinové vstupy (ck1 jsou hodiny normálního režimu, ck2 jsou hodiny režimu testu), sloužící zároveň k přepínání režimu činnosti buňky. Buňka je tvořena dvěma D klopnými obvody, zapojenými jako řídicí (L1) a řízený (L2),
- MD-SRL buňka - tato buňka (obrázek 11d) má dva datové vstupy, jeden vstup pro přepínání režimu a oddělené hodinové vstupy, jeden pro řízení řídicího (L1) klopného obvodu a jeden pro řízení řízeného (L2) klopného obvodu. Pro správnou funkci se předpokládá, že hodinové signály se nepřekrývají,
- LSSD buňka - je základním prvkem obvodů realizovaných na bázi LSSD. Je založena na bezhazardovém, hladinově řízeném klopném obvodu. Při aktivních hodinách je stav klopného obvodu citlivý na hodnotu příslušného datového vstupu. Pro zajištění správné činnosti buňky je nutné, aby se signály na vstupech ck1 a ck3 resp. ck2 a ck3 nepřekrývaly. Její zapojení

3.1 Návrh pro snadnou testovatelnost s využitím techniky scan 37



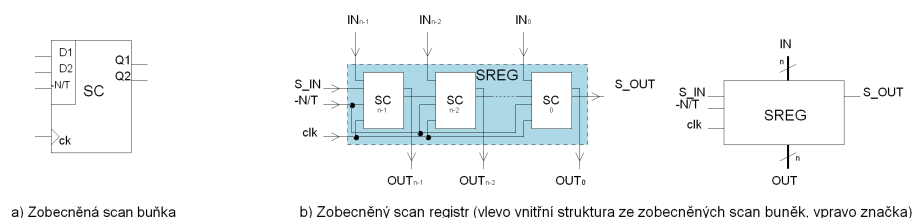
Obrázek 11. Klopný obvod D a vybrané varianty scan buněk

a schematická značka jsou zobrazeny na obrázku 11e. Sestává ze dvou klopných obvodů, které jsou zapojeny jako řídicí a řízený. Nepřekrývající se signály na hodinových vstupech ck1 resp. ck2 slouží k uložení normálních resp. testovacích dat do řídicího klopného obvodu, signál na hodinovém vstupu ck3 slouží k řízení řízeného klopného obvodu [WP83].

Jednotlivé varianty scan buněk jsou odlišné svou vnitřní strukturou, rozhraním, některými vlastnostmi i oblastmi použitelnosti. Na základě předchozích příkladů scan buněk však je možné konstatovat, že pro scan buňku jsou charakteristické následující skutečnosti [ABF90]:

- vstup(y) zajišťující přepínání mezi normálním a testovacím režimem činnosti buňky (vstup -N/T a/nebo hodinové vstupy),
- vývod pro vstup dat normálního režimu (vstup D1),
- vývod pro vstup dat testovacího režimu (vstup D2),
- vývod pro výstup dat normálního režimu (výstup Q1),
- vývod pro výstup dat testovacího režimu (výstup Q2),
- vstup(y) pro zajištění změny obsahu paměti scan buňky (povolovací a hodinové vstupy).

Pro účely této práce proto můžeme od konkrétní varianty scan buňky abstrahovat a pracovat s modelem zobecněné scan buňky (viz obrázek 12a). Obdobně můžeme abstrahovat i od konkrétního scan registru a místo něj můžeme použít zobecněný scan registr (viz obrázek 12b), tvořený zobecněnými scan



Obrázek 12. Zobecněná scan buňka a zobecněný scan registr

buňkami a zobecněný scan řetěz, tvořený zobecněnými scan registry. Nebude-li uvedeno jinak, tak v dalším textu již budeme uvažovat pouze zobecněné scan registry tvořené zřetěžením zobecněných scan buněk.

3.1.2 Typy scan technik

Podle toho, zda lze k prvkům registru scan přistupovat² sériově či nikoliv, rozlišujeme dva druhy techniky scan - *sériové*³ a *nesériové*. Dále, podle počtu paměťových prvků, využívaných touto technikou - hovoříme o úplném resp. částečném scanu a podle toho, zda je při implementaci techniky scan použito původních paměťových resp. přídavných prvků, hovoříme o integrovaném resp. izolovaném scanu. Následující dělení bude vycházet ze způsobu přístupu k paměťovým prvkům [ABF90].

Sériový scan

- *Sériový integrovaný scan* - při aplikaci tohoto druhu scan techniky jsou všechny resp. pouze některé původní paměťové prvky obvodu zařazeny do registru scan - ten plní kromě své původní funkce (v normálním režimu) i funkci posuvného registru (v testovacím režimu) pro nastavení resp. sledování stavu obvodu. V případě, že jsou do registru scan zařazeny všechny paměťové prvky obvodu, hovoříme o technice *úplný sériový integrovaný scan* nebo krátce *úplný scan*⁴ (viz obrázek 13b), jsou-li do registru scan zařazeny pouze některé paměťové prvky obvodu (alespoň jeden, avšak ne všechny), pak hovoříme o technice *částečný sériový integrovaný scan* nebo krátce *částečný scan*⁵ (viz obrázek 13c).

Testování s využitím techniky scan pak probíhá v těchto krocích [ABF90]:

1. sériové nastavení požadovaného vnitřního stavu obvodu,
2. zajištění testovacího vzorku na vstupech kombinační logiky,

² tj. měnit resp. zjišťovat vnitřní stav paměťových prvků

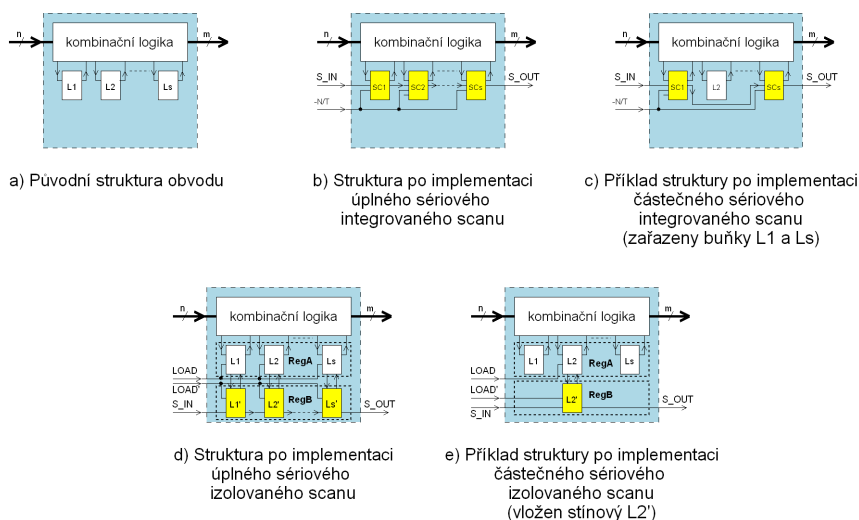
³ vybrané buňky prezentované v předchozím odstavci (3.1.1) se vztahovaly právě k tomuto druhu scan techniky

⁴ angl. full scan

⁵ angl. partial scan

3.1 Návrh pro snadnou testovatelnost s využitím techniky scan 39

3. pozorování odezvy z výstupů kombinační logiky,
4. zjištění cílového stavu obvodu, tj. sériové pozorování obsahu scan registrů.



Obrázek 13. Ilustrace k technice sériový scan (rozvody hodin nejsou zakresleny)

- *Sériový izolovaný scan* - oproti sériovému integrovanému scanu jsou paměťové prvky obsažené ve scan registru vloženy do obvodové struktury dodatečně a nejsou tedy součástí původní datové cesty obvodu. To znamená, že tento druh techniky má spíše vlastnosti ad-hoc technik, protože paměťové prvky byly do obvodové struktury přidány, aby plnily pouze funkci řídicích resp. pozorovacích bodů.

Nevýhodou této techniky je jednak nutnost vložit do obvodové struktury zcela nové paměťové prvky⁶ nepatřící do původní struktury obvodu a jednak skutečnost, že ani tato cenově nákladná technika není (v případě jejího nevhodného použití⁷ zárukou úplné a snadné říditelnosti resp. pozorovatelnosti vnitřního stavu a uzlů obvodu a zárukou vyloučení nutnosti použití generátoru testu pro sekvenční obvody.

Jsou-li pomocí nově vložených paměťových prvků (viz obrázek 13d) řízeny resp. pozorovány všechny původní paměťové prvky, pak hovoříme o technice *úplný sériový izolovaný scan* - v tomto případě bývá scan registr označován pojmem *stínový registr*; jinak hovoříme o technice *částečný sériový izolovaný scan* (viz obrázek 13e).

Testování pomocí stínového registru pak probíhá v těchto krocích [ABF90]:

⁶ sloužící pouze pro účely testu a ne pro činnost obvodu - viz obrázek 13d, e)

⁷ tj. v případě nevhodné volby paměťových prvků pro řízení resp. pozorování

40 3 Blízká témata

1. sériové vložení testovacího vzorku do stínového registru,
2. paralelní vložení testovacího vzorku ze stínového registru do původních paměťových prvků obvodu,
3. aplikace testovacího vzorku,
4. paralelní vložení odezvy z původních paměťových prvků do stínového registru,
5. sériové pozorování obsahu stínového registru.

Stejně probíhá i testování pomocí techniky částečný izolovaný scan - jediná odlišnost je v počtu nově vložených paměťových prvků, který je menší než v případě použití stínového registru.

Ve srovnání s integrovaným scanem představuje izolovaný scan dražší řešení, avšak je využitelný při některých formách on-line testování či testování v reálném čase.

Nesériový scan

Tato varianta se liší od výše popsaných sériových variant tím, že paměťové prvky tvoří posuvný registr, ale jsou uspořádány do tvaru bitově adresovatelné RAM paměti - blíže viz strana 30 a obrázek 10.

3.1.3 Částečný scan

Techniku částečný scan (v dalším textu budeme pod tímto označením rozumět částečný sériový integrovaný scan, viz odstavec 3.1.2, strana 38) je možné považovat za zobecněnou scan techniku, protože počítá (oproti technice úplný scan) s možností výběru paměťových prvků do registru scan. Cílem tohoto výběru je dosáhnout přijatelného kompromisu mezi cenou aplikace techniky scan (nárůstem plochy obvodu, nárůstem počtu vývodů, změnou dynamických parametrů atd.) a přínosem této techniky pro snadnou testovatelnost daného obvodu. Kvalita kompromisu je dána vybranou podmnožinou paměťových prvků, jejich rozmístěním do scan registrů a pořadím scan registrů ve scan řetězech. Lze konstatovat, že z hlediska diagnostických vlastností obvodu

- má výběr paměťových prvků do registrů scan a rozmístění paměťových prvků ve scan registrech vliv zejména na eliminaci globálních zpětnovazebních smyček, což vede na snazší použití generátoru testu sekvenčních obvodů resp. k jeho úplnému nahrazení a zastoupení generátorem testu kombinačních obvodů a tím k obecně kratší délce testu a
- rozmístění scan registrů do scan řetězů a jejich uspořádání ve scan řetězech má kromě dopadu na délku testu také dopad zejména na nárůst počtu vývodů obvodu, přidaných pro účely aplikace testu daného obvodu pomocí techniky scan.

Hlavní proud metod zabývajících se výběrem klopných obvodů pro částečný scan je možné rozdělit na tyto oblasti:

- *metody založené na využití výsledků analýzy obvodové struktury* - např. [CA90, CP91, STCA94, TB95, TOP95, XP96, PFR01]. Tyto metody se obvykle snaží vhodným výběrem klopných obvodů do řetězců scan pokrýt všechny kružnice obvodového grafu. Kromě toho, že tento problém byl shledán NP-úplným (viz např. [Kar72, Spe89, PFR01]), bylo ukázáno (např. [CP90]), že i když docílíme pokrytí všech globálních zpětných vazeb klopnými obvody, není tím obecně zaručeno, že tím dosáhneme 100% pokrytí poruch, tj. detekce (či lokalizace) všech poruch v daném modelu poruch,
- *metody založené na využití výsledků analýzy testovatelnosti* - např. [KKS95, PA95, Buk00, Růž02]. Tyto metody (přehled vybraných metod viz podkapitola 3.3, strana 47) jsou efektivnější z hlediska výpočetní složitosti, ale jelikož s sebou většinou přinášejí další abstrakce od obvodové struktury, tak zejména u obvodů se složitější strukturou obvykle nevedou k dosažení tak vysokého pokrytí poruch jako u metod založených na (detailnější) strukturální analýze,
- *metody založené na použití generátoru testu* - např. [HKTMSV88, CP91, IPS95, XP96, FCR96, BF96, TIF02]. Tyto metody vybírají klopné obvody do částečného scanu např. tak [XLR99], aby co nejvíce poruch mohlo být detekováno už generátorem testu pro kombinační obvody. Cílem je minimalizovat počet poruch detekovatelných pouze generátorem testu pro sekvenční obvody. Pomocí těchto metod lze dosáhnout velmi dobrých výsledků - nutností je však kooperace metody s generátorem(y) testu. Nevýhodou je pouze výpočetní náročnost těchto metod - ta je dána výpočetní náročností zvolené metody pro generování testu.

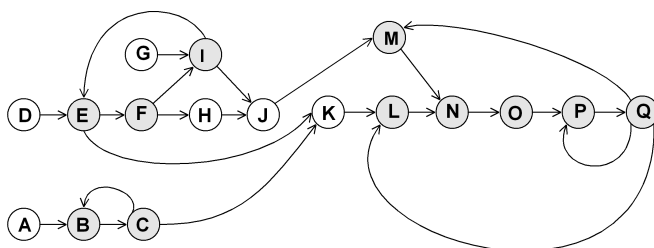
Z dalších metod zmiňme např. symbolické techniky [FCV98] či metody založené na zlepšení dosažitelnosti těžce dostupných stavů obvodu [BF96, SH99]. Z předchozího textu plyne, že výběr vhodné množiny registrů do scan řetězců má podstatný vliv na eliminaci zpětnovazebních smyček a tím i na míru zastoupení generátoru testu sekvenčních obvodů generátorem testu kombinačních obvodů. Kromě toho z něj vyplývá, že důležitým není jen výběr vhodné množiny registrů, ale je jím také způsob jejich rozmístění do scan řetězců a jejich uspořádání ve scan řetězcích. Další skupina metod (např. [YBP02, SGT03]) souvisejících s technikou částečný scan se zabývá problematikou vhodného rozmístění vybraných registrů ve scan řetězcích. Častým cílem takových metod je dosažení co nejlepšího kompromisu mezi danými návrhovými omezeními, a to zkoumáním výhodnosti různých variant rozmístění, omezených jistou množinou kandidátních scan registrů.

3.2 Analýza zpětnovazebních smyček

Podstatná část metod, řešících problematiku částečného scanu na základě výsledků analýzy obvodové struktury, vychází z grafové reprezentace struk-

tury obvodu. Problém je obvykle definován jako tzv. (M)FVS resp. (M)FAS problém nad grafem a snahou těchto metod je jej co nejefektivněji řešit.

V teorii grafů je FVS^8 resp. FAS^9 taková podmnožina uzlů resp. hran grafu $G = (V, E)$, která z každé kružnice daného grafu obsahuje alespoň jeden uzel resp. hranu. Odstranění všech uzlů náležejících FVS resp. hran náležejících FAS z grafu G tedy způsobí, že G se stane acyklickým. Problém nalezení množin FVS resp. FAS či jejich podmnožin byl formulován jak pro neorientované, tak pro orientované grafy a bývá označován jako FVS problém¹⁰ resp. FAS problém¹¹. Oba¹² pak bývají souhrnně označovány zkratkou FSP¹³.



Obrázek 14. Ilustrace k analýze zpětnovazebních smyček

V praktických aplikacích bývá množina kružnic, v níž se hledá FVS resp. FAS, zúžena na jistou podmnožinu $C \subset V(G)$ resp. $C \subset E(G)$ či bývá požadováno, aby váha¹⁴ w množiny FVS resp. FAS byla minimální. Může být např. požadováno nalezení minimálního počtu paměťových prvků resp. spojů, jejichž odstraněním¹⁵ z G a tím i k odstranění všech kružnic grafu G . V takovém případě hovoříme o množině $MFVS^{16}$ resp. o množině $MFAS^{17}$. V souvislosti s otázkou nalezení množin MFVS resp. MFAS pak hovoříme o MFVS problému resp. MFAS problému. Souhrnně bývají tyto problémy

⁸ množina zpětnovazebních uzlů, z angl. *feedback vertex set*

⁹ množina zpětnovazebních hran, z angl. *feedback arc set*

¹⁰ zkráceně FVSP

¹¹ zkráceně FASP

¹² a někdy také některé další obdobné problémy

¹³ *problém nalezení množiny zpětnovazebních prvků*, z angl. *feedback set problem*

¹⁴ $w: V(G) \rightarrow \mathbb{N}$ resp. $w: E(G) \rightarrow \mathbb{N}$ je nezáporná funkce definovaná na množině $V(G)$ resp. $E(G)$

¹⁵ v oblasti diagnostiky je odstraněním obvykle myšleno zlepšení jejich testovatelnosti. Odstranění kružnice bývá někdy označováno jako "rozbití kružnice". Z pohledu diagnostiky je jím myšleno zajištění možnosti řídit resp. pozorovat datový toku v dané kružnici - např. pomocí testovacího bodu či scan registru umístěného ve vhodném místě kružnice

¹⁶ *nejmenší množina zpětnovazebních uzlů*, z angl. *minimum feedback vertex set*

¹⁷ *nejmenší množina zpětnovazebních hran*, z angl. *minimum feedback arc set*

označovány zkratkou MFSP¹⁸. Pro zdůraznění příslušnosti množin ke grafu G pak hovoříme o množinách $FVS(G)$, $FAS(G)$, $MFVS(G)$, $MFAS(G)$ atp. Zřejmě platí $MFVS(G) \subseteq FVS(G)$, $MFAS(G) \subseteq FAS(G)$.

Příklad 1:

Nechť je dán graf $G = (V, E)$ z obrázku 14, kde $V(G) = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q\}$ a $E(G) = \{(A, B), (B, C), (C, B), (C, K), (D, E), (E, F), (F, I), (G, I), (I, E), (E, K), (F, H), (H, J), (I, J), (J, M), (K, L), (M, N), (L, N), (N, O), (O, P), (P, Q), (Q, P), (Q, L), (Q, M)\}$.

V nejjednodušším případě pak $FVS(G) = \{B, C, E, F, I, L, M, N, O, P, Q\}$. Zřejmě, $MFVS(G)$ je tříprvková množina $\{a, b, c\}$, kde $a \in \{B, C\}$, $b \in \{E, F, I\}$, $c \in \{P, Q\}$.

□

Z předchozího příkladu je zřejmé, že zatímco pro nalezení množiny $FVS(G)$ resp. $FAS(G)$ postačuje z každého cyklu (kružnice) grafu G nalézt alespoň jeden uzel resp. hranu a vložit jej resp. ji jako prvek do množiny $FVS(G)$ resp. $FAS(G)$ (v nejjednodušším případě to znamená nalézt množinu uzlů resp. hran vyskytujících se v cyklech (kružnicích) grafu G , tj. ve zpětných vazbách), tak pro nalezení množin $MFVS(G)$ resp. $MFAS(G)$ již toto nepostačuje - navíc je totiž nutné odstranit z množin $FVS(G)$ resp. $FAS(G)$ prvky tak, aby váha w množiny $MFVS(G) \subseteq FVS(G)$ resp. $MFAS(G) \subseteq FAS(G)$ byla minimální. Problém nalezení množin $MFVS(G)$ a $MFAS(G)$ však není triviální a patří mezi tzv. NP-problémy (viz např. [Kar72, Spe89, PFR01]).

K oblastem zabývajícím se (M)FSP patří oblast návrhu VLSI obvodů [MBIR87], dokazování správnosti programů [Flo67], kryptografie [Gus88], zotavení z uváznutí v operačních systémech [LL79] a další - přehled lze nalézt např. v [Spe89, PFR01].

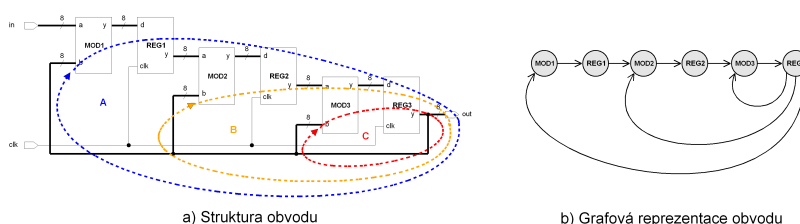
3.2.1 Aplikace v oblasti návrhu a diagnostiky číslicových obvodů

Jednou z prvních publikovaných prací zabývajících se řešením (M)FSP s aplikací v oblasti návrhu obvodů je [Joh75]. Práce se zabývá situacemi, kdy existence cyklů v obvodu vede ke vzniku souběhu. Těmto situacím lze zamezit např. tak, že do každého cyklu obvodu je vložen registr - to však znamená jisté dodatečné zpoždění v datové cestě obvodu, které je přímo úměrné počtu dodatečně vložených registrů. Cílem práce [Joh75] bylo zabránit vzniku souběhů (zajištěním ustálených hodnot pomocí vložených registrů) a současně minimalizovat počet vložených registrů tak, aby celkové zpoždění v datové cestě obvodu bylo co nejmenší, což vedlo na řešení MFSP problému.

Jednou z mnoha dalších oblastí, v níž probíhal intenzivní výzkum týkající se (M)FSP, a která souvisí s touto disertační prací, je oblast testování VLSI

¹⁸ *problém nalezení nejmenší množiny zpětnovazebních prvků*, z angl. *minimum feedback set problem*

obvodů [MBIR87, BG89, KW90, LR90, STCA94]. Vychází se z toho, že obvod lze modelovat orientovaným grafem, jehož uzly reprezentují obvodové prvky (na úrovni hradel např. hradla provádějící nějakou Booleovskou funkci, na úrovni meziregistrových přenosů např. funkční jednotky, multiplexory, paměťové prvky atp.) a orientované hrany vyjadřují kromě existence spojů mezi vývody prvků také směr toku dat po těchto spojích. Problém pak bývá zadán např. jako problém výběru uzlů reprezentujících paměťové prvky (např. klopné obvody či registry) do množiny MFVS tak, aby byl zajištěn co nejmenší nárůst plochy a počtu vývodů způsobený úpravou struktury obvodu pro účely jeho snadného testování např. pomocí techniky scan (v tomto případě je tato technika založena na modifikaci paměťových prvků náležících MFVS na jejich tzv. scan verze a na jejich rozmístění do scan řetězců) a zároveň aby bylo dosaženo přijatelné testovatelnosti obvodu. Problematikou analýzy zpětných vazeb, s výsledky využitelnými pro zlepšení testovatelnosti číslicového obvodu, jsem se zabýval také, a to zejména v pracích [FZ02, Str03c, Str03a]. Hlavní směr výzkumu k tématu této práce se však nakonec ubíral jiným směrem, než jakým je řešení MFSP.



Obrázek 15. Ilustrace k využití analýzy zpětnovazebních smyček v diagnostice

Příklad 2:

Nechť je dán graf $G = (V, E)$ z obrázku 15, část b, kde $V(G) = \{MOD1, MOD2, MOD3, REG1, REG2, REG3\}$ a $E(G) = \{(MOD1, REG1), (REG1, MOD2), (MOD2, REG2), (REG2, MOD3), (MOD3, REG3), (REG3, MOD1), (REG3, MOD2), (REG3, MOD3)\}$.

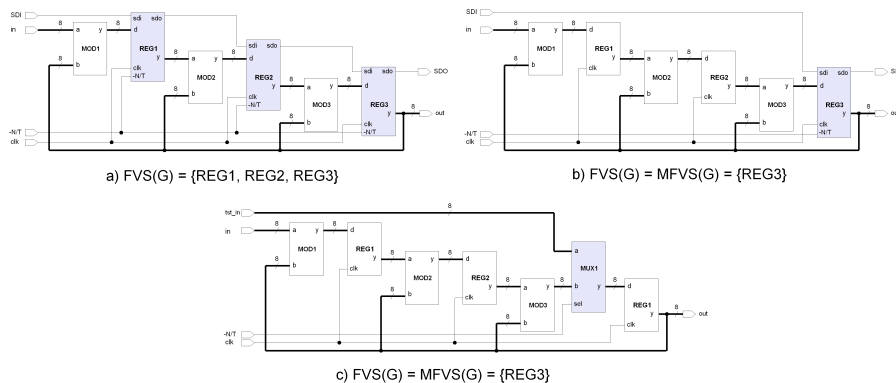
Ze struktury grafu G je zřejmé, že v nejjednodušším případě platí $FVS(G) = V(G)$. Pokud předpokládáme, že pro odstranění cyklů je nutno ke každému prvku $FVS(G)$ buď vložit testovací bod nebo takový prvek např. modifikovat pro diagnostické účely, pak je snadnost určení množiny $FVS(G)$ v tomto případě vykoupena příliš velkou cenou za odstranění cyklů - cena je zde dána zejména nárůstem plochy obvodu, způsobeným modifikacemi obvodové struktury za účelem odstranění cyklů.

Předpokládejme nyní, že pro odstranění cyklů bude využita pouze podmnožina $X = \{x | x \in V(G) \wedge x \text{ je paměťový prvek}\}$, v tomto příkladě $X = \{REG1,$

$REG2, REG3$ }. V nejjednodušším případě pak $FVS(G) = X$. Použijeme-li techniku sériový scan, pak každý prvek z množiny X - v tomto příkladu každý registr v obvodu - je modifikován na scan registr a dané scan registry jsou vhodně zřetězeny do scan řetězů (např. dle obrázku 16a). Je zřejmé, že zúžením množiny uzlů vhodných pro odstranění cyklů na jistou podmnožinu $Y \subseteq V(G)$ lze výrazně snížit cenu za acykličnost grafu. Pokud však nebude každý cyklus grafu G obsahovat alespoň jeden prvek množiny Y , graf G nebude acyklický a množinu Y bude nutné rozšířit o nové prvky tak, aby bylo dosaženo acykličnosti.

Nejnižší ceny za odstranění kružnic lze dosáhnout, pokud $Y = FVS(G) = MFVS(G)$; problém nalezení množiny $MFVS(G)$ však patří mezi NP problémy. Množina prvků $MFVS(G)$ může být také zúžena na jistou podmnožinu prvků $V(G)$ - zde např. na paměťové prvky - avšak vždy platí, že jsou odstraněny všechny kružnice grafu G a že váha w množiny $MFVS(G)$ je minimální. Zřejmě v tomto příkladu $MFVS(G) = \{MOD3\}$ nebo $MFVS(G) = \{REG3\}$. Zúžíme-li množinu prvků využitelných pro odstranění kružnic na množinu registrů, pak $MFVS(G) = \{REG3\}$ a pro rozbití kružnic v G postačí zlepšit testovatelnost datové cesty v okolí $REG3$ např. modifikací registru $REG3$ na scan registr (viz obrázek 16b) nebo např. vložením multiplexoru do datové cesty dle obrázku 16c.

□



Obrázek 16. Ilustrace k příkladu 2

3.2.2 Přehled metod

Ačkoli obecný MFSP nelze vyřešit v polynomiálním čase, existují algoritmy pracující v polynomiálním čase, řešící MFSP omezené na jisté speciální třídy grafů. Bylo tedy a vždy je výhodné takové třídy grafů přesně popsat, aby

bylo pro každý další graf G možné stanovit, zda do některé takové třídy patří (a zda $\text{MFSP}(G)$ je řešitelný v polynomiálním čase) či nikoliv.

Jedna z vůbec prvních prací zabývajících se návrhem MFSP algoritmu pro speciální třídu (redukovatelných) grafů byla práce [Sha76]. Z velkého množství dalších prací zmiňme pouze odkazy na algoritmy [SW75, CWS85, Lia94, LT97]. V [LL88] je prezentován efektivní algoritmus využívající tzv. kontrakčních operací - pomocí nich lze redukovat strukturu grafu a přitom neporušit jeho vlastnosti důležité pro řešení MFSP. Přestože původně byl princip kontrakce grafu [LL88] prakticky použitelný pouze na jistou třídu grafů, jeho výchozí myšlenka byla natolik významná, že měla velký vliv na řadu pozdějších prací zabývajících se řešením MFSP. Na principu kontrakce je založena většina později vzniklých heuristických a aproximačních algoritmů pro řešení MFSP.

Velkou skupinu algoritmů řešících MFSP tvoří již zmíněné aproximační algoritmy - např. [MS81, RBYR98, BG94, BG96, FACW98]. Tyto algoritmy se nesnaží MFSP vyřešit pro jistou speciální třídu grafů, ale pro případy grafů, pro něž není znám polynomiální algoritmus se snaží co nejvíce přiblížit přesnému (optimálnímu) řešení MFSP. Pro účely porovnávání kvalit různých aproximačních algoritmů je stanoveno několik kritérií a definováno několik tzv. aproximačních schémat - přehled např. v [PFR01]. Ačkoli aproximační algoritmy garantují řešení MFSP jisté kvality, existuje řada případů, v nichž je vhodnější použít pro řešení MFSP nějakou heuristickou metodu - viz např. [PMP99].

Ještě zmiňme tzv. přesné algoritmy, řešící MFSP pro obecné grafy. Tato skupina algoritmů je nejmenší a typickým zástupcem této skupiny je algoritmus pro nalezení FVS s minimální kardinalitou v daném orientovaném grafu [SW75] (algoritmus pracuje v exponenciálním čase pro obecné orientované grafy, pro některé typy orientovaných grafů v polynomiálním čase). Pozdější metody se snažily časovou složitost snížit použitím kontrakčních operací - viz např. studie [CA90] řešící problematiku použití techniky částečný scan, metoda "dvojitá redukce" [TOP95] řešící speciální variantu FSP - nalezení maximálně váhově ohodnoceného uzlově indukovaného orientovaného podgrafu.

3.2.3 Shrnutí

Existuje několik činitelů ovlivňujících efektivitu přístupů založených na řešení MFSP; empirické postřehy jsou shrnuty např. v [SH01], přičemž a k nejdůležitějším z nich patří:

- není nezbytné pokrýt ty cykly, které jsou snadno testovatelné (tj. není třeba rozbíjet ty cykly, jejichž datová cesta již je testovatelná - např. díky vhodnému umístění multiplexoru v datové cestě cyklu),
- zejména ve složitějších obvodech obsahujících cykly velké délky může být z hlediska výsledné testovatelnosti výhodnější rozbít takovýto cyklus pomocí několika testovacích bodů, scan registrů apod.,

- sama přítomnost cyklů v obvodu nutně neznamená, že generování testu bude složité a že nepůjde provést generátorem testu pro kombinační obvodu.¹⁹

Pokrytí cyklů obvodu klopnými obvody založené pouze na řešení MFSP je známo jako efektivní způsob výběru klopných obvodů do částečného scanu. Avšak, v posledních letech se objevují publikace - např. [SH99, SH01, TIF02, XP04] - které ukazují, že lze dosáhnout vyššího pokrytí poruch než navrhuje klasické MFSP přístupy, a to výběrem menšího počtu klopných obvodů do částečného scanu. Tento fakt vede např. k tomu, že vznikají metody, obsahující klasické MFSP přístupy o informace získané pomocí generátoru testu či o vhodné modely chování obvodových prvků tvořících strukturu obvodu. Přístup typu "všechny cykly jsou si rovné" je nahrazen přístupem typu "pokryj jen ty cykly, u nichž je to nutné".

3.3 Analýza testovatelnosti

Již bylo uvedeno (odstavec 2.3.3), že v současné době neexistuje přesná definice testovatelnosti a že obecně bývá testovatelnost chápána jako charakteristika zohledňující různé náklady spojené s testováním číslicového obvodu, a to jako ukazatel efektivnosti tvorby a aplikace testu. Hovoří se pak např. o testovatelnosti poruchy nebo o testovatelnosti uzlu v obvodu a porucha nebo uzel jsou označeny za testovatelné, existuje-li nějaký postup (závisející na typu testu, např. přivedení testovacího vzorku detekujícího danou poruchu na jistý uzel obvodu a odvedení odezvy za účelem jejího vyhodnocení), jímž jsme schopni poruchu v daném uzlu odhalit. V případě testovatelnosti poruchy resp. uzlu je také snahou nějakým způsobem, obvykle pomocí tzv. měr testovatelnosti ohodnotit snadnost provedení tohoto postupu.

Rozdílnost definic testovatelnosti však vede k odlišnému chápání některých pojmů, které jsou pro testovatelnost obvodu podstatné a také měř, používaných pro jejich ohodnocení. K odstranění těchto nevýhod (zejména ke sjednocení definic) by pak mělo výraznou měrou přispět dokončení standardu IEEE P1522²⁰.

K nejčastějším faktorům, na jejichž základě je v současnosti testovatelnost obvodu odhadována, patří říditelnost, pozorovatelnost a předvídatelnost (viz strana 19). Postup, který umožňuje testovatelnost číselně ohodnotit, se nazývá analýza testovatelnosti. Jelikož jednotná definice testovatelnosti v současnosti neexistuje, liší se dosavadní přístupy k tomuto problému jak svými cíly, tak úrovněmi abstrakce popisu obvodu, na nichž jsou použitelné.

¹⁹ je zřejmé, že tento druh problému (tj. zda daný cyklus je netestovatelný kombinačním generátorem a je jej nutné pokrýt klopnými obvody) může být vyřešen pouze pomocí vhodné analýzy funkce obvodu resp. jeho prvků

²⁰ více informací o tomto standardu viz text začínající na straně 20

Mezi nejčastější cíle metod pro analýzu testovatelnosti patří ohodnocení snadnosti nastavování resp. pozorování hodnot vyskytujících se na uzlech obvodu, detekce obtížně testovatelných částí obvodu, odhad zvolených diagnostických vlastností obvodu a další. Existují metody navržené pro obvody popsané na úrovni hradel, pro obvody popsané na úrovni RT, pro obvody na vyšší úrovni popisu, ale i metody víceúrovňové, provádějící analýzu na několika úrovních popisu. Následující text bude věnován přehledu vybraných metod pro analýzu testovatelnosti.

3.3.1 Úroveň hradel

Většina prvních prací zabývajících se problematikou analýzy testovatelnosti byla orientována na úroveň hradel. Testovatelnost je pomocí metod patřících do této skupiny obvykle ohodnocena pomocí několika tzv. měř testovatelnosti. Těchto měř bývá obecně několik a obvykle vycházejí z tzv. koncepce říditelnosti a pozorovatelnosti. Jejich snahou je pro každý uzel obvodu číselně ohodnotit snadnost ovládní resp. pozorování hodnoty vyskytující se na tomto uzlu tak, že uzly, které jsou blíže primárním vstupům resp. výstupům obvodu jsou lépe říditelné resp. pozorovatelné (tj. mají lepší hodnotu říditelnosti resp. pozorovatelnosti). Přestože jsou tyto míry založené na ”měření vzdálenosti uzlů od vývodů obvodu” velmi jednoduché, ukázalo se [Lar00], že s jejich pomocí je možno velmi přesně určit obtížně testovatelné části daného návrhu.

Metoda SCOAP

SCOAP (Sandia Controllability Observability Analysis Program) [Gol79] je pravděpodobně nejznámější metodou pro analýzu testovatelnosti. Tato metoda je navržena pro obvody popsané na úrovni logických členů a ve známost vstoupila zejména díky její popularizaci v řadě publikací.

Metoda SCOAP je založena na tzv. mírách říditelnosti a pozorovatelnosti. Ty reflektují obtížnost ovládní resp. pozorování konkrétních jednobitových hodnot na vnitřních bodech obvodu, přičemž vyšší hodnoty říditelnosti resp. pozorovatelnosti indikují větší obtížnost této činnosti.

Pro každý uzel x obvodu je metodou SCOAP vypočteno šest hodnot vyjadřujících míru jeho říditelnosti a pozorovatelnosti:

- kombinační
 - říditelnost nuly na uzlu x (značená $CC^0(x)$),
 - říditelnost jedničky na uzlu x (značená $CC^1(x)$),
 - pozorovatelnost uzlu x (značená $CO(x)$),
- sekvenční
 - říditelnost nuly na uzlu x (značená $SC^0(x)$),
 - říditelnost jedničky na uzlu x (značená $SC^1(x)$),
 - pozorovatelnost uzlu x (značená $SO(x)$).

Kombinační míry závisejí na počtu uzlů, které musí být nastaveny pro řízení resp. pozorování hodnoty na uzlu x , sekvenční míry na počtu časových kroků potřebných pro řízení resp. pozorování hodnoty na uzlu x . Na začátku algoritmu SCOAP je říditelnost všech primárních vstupů kombinačních obvodů nastavena na hodnotu 1 a sekvenčních obvodů na hodnotu 0. Pozorovatelnost všech primárních výstupů je nastavena na nulu a pro každý uzel obvodu je vypočtena hodnota říditelnosti. Hodnoty říditelnosti uzlů jsou následně použity pro výpočet hodnot pozorovatelnosti uzlů.

Sekvenční hodnoty jsou počítány obdobným způsobem. Reprezentují délky vstupních posloupností nutných pro řízení a pozorování daného uzlu. Znamená to, že kombinační prvky nemají na sekvenční míry vliv. U sekvenčního prvku je sekvenční říditelnost jeho výstupu dána sekvenční říditelností jeho vstupů a sekvenční hloubkou prvku.

Časová složitost analýzy testovatelnosti pomocí SCOAP je lineární funkcí počtu hradel, což z metody SCOAP činí atraktivní nástroj pro posuzování testovatelnosti na úrovni logických členů.

3.3.2 Úroveň meziregistrových přenosů

Metody analýzy testovatelnosti, které pracují na úrovni meziregistrových přenosů, v podstatné většině vycházejí z nějakého pravděpodobnostního modelu toku dat obvodovou strukturou. Každý prvek patřící do struktury obvodu bývá chápán jako překážka, snižující pravděpodobnost ovládnutí resp. pozorování hodnoty uzlů vyskytujících se v jejím okolí.

Analýza testovatelnosti prezentovaná v [CM89] je založena na výpočtu kombinační říditelnosti (CC), kombinační pozorovatelnosti (CO), sekvenční říditelnosti (SC) a sekvenční pozorovatelnosti (SO). Hodnoty CC a SC jsou navíc počítány zvlášť pro úroveň logické 1 a zvlášť pro úroveň logické 0. Každý uzel v obvodu lze tedy ohodnotit celkem šesti hodnotami - CC_0 , CC_1 , CO , SO , SC_0 a SC_1 . Hodnota CC_0 resp. CC_1 vyjadřuje pravděpodobnost nastavení daného uzlu na hodnotu logická 0 resp. 1. Hodnota SC_0 resp. SC_1 představuje odhad délky posloupnosti nutné pro nastavení daného uzlu na hodnotu logická 0 resp. 1. Hodnota CO určuje, s jakou pravděpodobností změna hodnoty na daném uzlu vyvolá změnu na primárních výstupech obvodu a hodnota SO je odhadem počtu časových jednotek nutných k vyvolání této změny hodnoty na primárních výstupech v důsledku změny hodnoty na daném uzlu.

Obdobný přístup byl prezentován např. v [KP90, Gu96]. Každý uzel v obvodu je ohodnocen hodnotami kombinační říditelnosti (CC), kombinační pozorovatelnosti (CO), sekvenční říditelnosti (SC) a sekvenční pozorovatelnosti (SO). Tyto hodnoty jsou vypočteny pomocí dvou dalších hodnot, CTF a OTF - ty jsou definovány pro každou funkční jednotku a mají následující význam. CTF je činitel přenosu říditelnosti a vyjadřuje vztah mezi říditelností vstupů a říditelností výstupů této jednotky - CTF vyjadřuje, s jakou pravděpodobností je možno nastavit na výstup funkční jednotky požadovanou hodnotu, jsou-li na vstupech funkční jednotky generovány

náhodné hodnoty. *OTF* je činitel přenosu pozorovatelnosti a vyjadřuje vztah mezi pozorovatelností vstupů a pozorovatelností výstupů této jednotky - *OTF* vyjadřuje, s jakou pravděpodobností je možno hodnotu z daného vstupu funkční jednotky přenést na výstup funkční jednotky, jsou-li na ostatních vstupech funkční jednotky generovány náhodné hodnoty. Hodnoty obou činitelů leží v intervalu $\langle 0; 1 \rangle$, kde krajní hodnota 0 resp. 1 představuje nejhorší resp. nejlepší vlastnosti z hlediska pravděpodobnosti přenosu konkrétních dat strukturou dané funkční jednotky. Z podobných principů vycházejí také např. práce [FMLB97] nebo [Buk00] s metodou implementovanou v nástroji IDAT.

Metoda TMEAS

Tato metoda [JES76, Gra79] byla původně navržena pro obvody popsané na úrovni RT, ale lze ji aplikovat i na obvody popsané na úrovni hradel. Hodnoty říditelnosti a pozorovatelnosti se pohybují v intervalu $\langle 0; 1 \rangle$ a vyjadřují snadnost ovládní resp. pozorování vnitřních uzlů obvodu. Tato metoda může být shrnuta do následujících bodů:

1. Pro každý uzel obvodu s je určena říditelnost $CY(s)$ a pozorovatelnost $OY(s)$ (vyřešením soustavy rovnic o neznámých $CY(s_i)$ a $OY(s_i)$, $i = 1, \dots$, počet uzlů obvodu):
 - a) Necht' komponenta p má vstupy x_1, \dots, x_n a výstupy z_1, \dots, z_m . Pak $CY(z_j) = CTF \times \frac{1}{n} \sum_{i=1}^n CY(x_i)$, kde CTF je činitel přenosu říditelnosti pro danou komponentu. $CTF = \frac{1}{m} \sum_{j=1}^m (1 - \frac{|N_j(0) - N_j(1)|}{2^n})$, kde $N_j(0)$ resp. $N_j(1)$ je počet vstupních kombinací, pro něž z_j nabývá hodnoty 0 resp. 1. Pro všechny výstupy daného prvku platí, že mají stejnou hodnotu CTF .
 - b) Obdobně je vypočtena hodnota OY pro každý vstup x_i , a to podle vztahu $OY(x_i) = OTF \times \frac{1}{m} \sum_{j=1}^m OY(z_j)$, kde OTF je činitel přenosu pozorovatelnosti pro danou komponentu. $OTF = \frac{1}{n} \sum_{i=1}^n \frac{NS_i}{2^n}$, kde NS_i je počet vstupních kombinací, v nichž změna x_i způsobí změnu hodnoty na výstupu. Pro všechny vstupy daného prvku platí, že mají stejnou hodnotu OTF .
2. Pro každou z k větví b_1, \dots, b_k každého větvení s je vypočítána její říditelnost CY podle vztahu $CY = \frac{CY(s)}{(1 + \log k)}$ a její pozorovatelnost OY podle vztahu $OY = 1 - \prod_{i=1}^k (1 - OY(b_i))$.
3. Sekvenční prvky jsou modelovány přidáním zpětné vazby na podprvky reprezentující vnitřní stav daného sekvenčního prvku.

Metoda CAMELOT

Metoda CAMELOT (Computer-Aided MEasure for LOgic Testability) [RGRB80, Ben84] vznikla vylepšením metody TMEAS.

1. Řiditelnost výstupu y daného prvku je počítána z hodnot řiditelností těch jeho vstupů x_1, \dots, x_k , na nichž výstup y závisí, podle vztahu $CY(y) = CTF(y) \times f(CY(x_1), \dots, CY(x_k))$, kde $CTF = 1 - \left| \frac{N(0) - N(1)}{N(0) + N(1)} \right|$, kde $N(0)$ resp. $N(1)$ jsou počty vstupních kombinací, pro které výstup y nabývá hodnoty 0 resp. 1. Stejně jako u metody TMEAS platí $0 \leq CTF \leq 1$ a navíc také:
 - a) $CTF = 1$ pokud $N(0) = N(1)$,
 - b) $CTF = 0$ pokud $N(0) = 0$ nebo $N(1) = 1$,
 - c) každý výstup y má vlastní množinu vstupů, na kterých závisí a vlastní hodnotu CTF .
2. Pozorovatelnost vstupu x daného prvku je počítána z hodnot pozorovatelnosti daného výstupu y , hodnot CY vstupů (x_1, \dots, x_k) nezbytných pro zajištění pozorovatelnosti z x na y a $OTF(x - y)$ podle vztahu $OY(x) = OTF(x - y) \times OY(y) \times g(x_1, \dots, x_k)$, kde $OTF(x - y)$ je OTF ohodnocující snadnost propagace poruchy ze vstupu x na výstup y . Rovněž se zavádí zápis $N(SP : x - y)$ resp. $N(IP : x - y)$, vyjadřující počet citlivých cest pro šíření poruchy ze vstupu x na výstup y resp. počet způsobů, jimiž je možné šíření poruchy ze vstupu x na výstup y blokovat. Pro daný vstup x a výstup y je možné vyjádřit $OTF(x - y)$ vztahem $OTF(x - y) = \frac{N(SP : x - y)}{N(SP : x - y) + N(IP : x - y)}$.
3. Testovatelnost $TY(x)$ vnitřního bodu x obvodu je určena z hodnot $CY(x)$ a $OY(x)$ vztahem $TY(x) = CY(x) \times OY(x)$, celková testovatelnost obvodu *cir* je rovna aritmetickému průměru hodnot OY vnitřních bodů obvodu.

Princip metody CAMELOT je možné shrnout do následujících kroků:

1. Načtení dat obvodu a inicializace hodnot
2. Pro každý uzel je vypočtena:
 - a) hodnota CY (postup směrem od primárních vstupů na primární výstupy)
 - b) hodnota OY (postup směrem od primárních výstupů na primární vstupy)
 - c) hodnota TY
3. Výpočet hodnoty TY pro obvod a vyhodnocení výsledků

Metoda CoPS

Základem metody CoPS (Cost-based Partial Scan) [PSP93, PSP95] je výpočet nákladů spojených s aplikací testu pro obvod popsáný na úrovni hradel s využitím DFT techniky částečný scan. Pro každý vodič v obvodě je metodou spočítána tzv. cena detekovatelnosti poruchy, zohledňující cenu řiditelnosti, sekvenční hloubku a cenu pozorovatelnosti.

Cena řiditelnosti vodiče v pro hodnotu h je definována jako minimální počet synchronizačních pulzů potřebných pro nastavení logické hodnoty h na vodič v , cena pozorovatelnosti vodiče v je dána cenou nastavitelnosti cesty

z vodiče v na některý z primárních výstupů a sekvenční hloubka vodiče v pro hodnotu h je rovna počtu klopných obvodů, které jsou součástí cesty c , kde c je nejkratší cesta pro nastavení hodnoty h (tj. přenos hodnoty h z primárních vstupů) na v resp. nejkratší cesta pro pozorování hodnoty vyskytující se na v (tj. přenos této hodnoty na primární výstupy).

Na základě výše uvedených dílčích cen je pro každý vodič a poruchu určena cena detekovatelnosti, reflektující obtížnost aplikace testu jednotlivých uzlů v obvodu. Celková nákladová funkce je pak definována jako součet cen detekovatelnosti uzlů obvodu. Je možné ji využít jako míru obtížnosti aplikace testu obvodu pro danou množinu poruch a pro stanovení tzv. citlivosti, tj. velikosti změny celkové nákladové funkce zařazením klopného obvodu do scanu. Pomocí citlivosti je pak doporučeno, které klopné obvody je výhodné zařadit do scanu.

Podobná metoda je publikována v [KKS95]. Pracuje se vztahy pro výpočet hodnot říditelnosti a pozorovatelnosti uzlů obvodu. Tyto hodnoty jsou pro každý uzel obvodu určeny pomocí generátoru testovacích vektorů, který metoda používá ke zjištění doby nutné pro generování testu. Hodnota říditelnosti uzlu je pak určena tak, že je tento uzel modelován jako dostupný na primárních výstupech (tj. uzel je chápán pozorovatelným) a vyhodnotí se doba potřebná pro generování testu pro konkrétní poruchu v tomto uzlu. Obdobně se postupuje při výpočtu hodnoty pozorovatelnosti uzlu - v tomto případě je uzel modelován jako dostupný na primárních vstupech obvodu (tj. uzel je chápán říditelným). Metoda [KKS95] vychází

Velkou nevýhodou metod založených na analýze citlivosti je jejich velká časová složitost. Algoritmus pro analýzu citlivosti musí být opakovaně spouštěn. Nejprve pro obvod, v němž není aplikována žádná z technik návrhu pro snadnou testovatelnost, poté při každé aplikaci těchto technik. Vzhledem k velké časové složitosti nejsou metody založené na tomto principu prakticky používány [ABF90].

Metoda symbolické analýzy testovatelnosti

Metoda symbolické analýzy testovatelnosti (STA) byla původně konstruována pro obvody na úrovni meziregistrových přenosů [IGB98, SRJ98], později však byla její použitelnost rozšířena i na obvody popsané na systémové úrovni [SRJ99]. V obou případech je však metodika zaměřena na analýzu testovatelnosti, jejíž výsledky napomohou zejména k co nejvhodnějšímu zabudování techniky samočinného testování²¹.

[IGB98] předpokládá, že jejím vstupem budou datové cesty obvodu a řadič obvodu. Na základě těchto informací je zkonstruován graf (TCDF²², graf řídicího a datového toku testu), jehož uzly reprezentují operace dílčích modulů obvodu a hrany jsou ohodnoceny proměnnými reprezentujícími registry v obvodu. Z TCDF a informace o funkci, která je uchována spolu s každým typem

²¹ angl. built-in self test(ing), BIST

²² angl. test control/data flow

modulu obvodu, je pro každý modul obvodu odvozena množina symbolických cest (označovaná pojmem prostředí testu²³), nezbytných k otestování vybraných operací a proměnných z TCDF. Pro účely vytvoření prostředí testu jsou pro každou proměnnou z TCDF zavedeny následující vlastnosti:

- obecná říditelnost C_g proměnné - ohodnocení schopnosti nastavit proměnnou na libovolnou hodnotu (pomocí dat z primárních vstupů obvodu)
- obecná pozorovatelnost C_v proměnné - ohodnocení schopnosti sledovat libovolnou hodnotu proměnné na některém z primárních výstupů obvodu
- konstantní říditelnost C_q proměnné - ohodnocení schopnosti nastavit proměnnou na konstantní hodnotu q (pro speciální konstanty existují speciální případy C_q , např. C_1 pro schopnost nastavení proměnné na hodnotu 1, C_0 pro schopnost nastavení proměnné na hodnotu 0, C_a1 pro schopnost nastavení proměnné na hodnotu odpovídající jedničkovému binárnímu vektoru, popř. C_a0 pro schopnost nastavení proměnné na hodnotu odpovídající nulovému binárnímu vektoru)
- ověřitelnost V hodnoty proměnné - ohodnocení schopnosti nastavit nebo sledovat hodnotu proměnné

Jak říditelnost, pozorovatelnost, tak ověřitelnost mohou nabývat pouze hodnoty 0 nebo 1, podle toho, zda proměnná nemá či má danou schopnost. Každé z vlastností proměnné je přiřazena informace o čase (zadaném počtem hodinových taktů), kdy je tato vlastnost po dané proměnné požadována. Např. zápis $C_g(2)$ pro danou proměnnou znamená, že během druhého taktu hodin potřebujeme nastavit hodnotu této proměnné na libovolnou²⁴ hodnotu.

Symboličnost metody spočívá v tom, že pomocí zavedených vlastností metoda nepracuje s konkrétními hodnotami testovacích vzorků, proměnných a odezev, ale tyto hodnoty nahrazuje symbolickým zápisem. Nad ním pak provádí předem definovaná transformační pravidla tak, aby bylo možné splnit pokud možno všechny podmínky nutné pro nastavení a pozorování požadovaných hodnot dané proměnné v daných časových okamžicích, tj. tak, aby bylo možné otestovat co největší počet proměnných z TCDF. Dojde-li při snahách o splnění podmínek pro testování proměnných ke kolizi (např. pokud pro testování proměnné x potřebujeme v čase t nastavit proměnnou y na konstantní hodnotu $c0$ a zároveň pro testování proměnné z potřebujeme v čase t nastavit proměnnou y na konstantní hodnotu $c1$), tj. při hledání daného prostředí testu, jsou pomocí zpětného vyhledávání vybrána a aplikována jiná transformační pravidla. V případě, že při stávajícím chování a struktuře obvodu není možné nalézt prostředí testu daného modulu, je pro řešení této situace do obvodové struktury vložen přídatný multiplexor. Místa pro vložení multiplexorů jsou vybrána na základě hodnoty počítadla

²³ angl. test environment

²⁴ tj. v okamžiku analýzy blíže neurčenou

špatnosti²⁵ proměnných z TCDF; snahou je, aby vložení těchto multiplexorů nezpůsobilo nežádoucí změnu kritické cesty obvodu.

Prostředí testu je pak využito k otestování modulu nebo registru pomocí pseudonáhodného generátoru testovacích vektorů, který bude pro tento účel připojen na primární vstupy obvodu a pomocí příznakového analyzátoru, který bude pro tento účel připojen na primární výstupy obvodu. Zároveň je nezbytné, aby byl každý modul v obvodu testovatelný pomocí pseudonáhodného generátoru testovacích vzorků - není-li tomu tak, musí být knihovna modulů modifikována tak, aby byl tento požadavek splněn. V posledním kroku je proveden návrh a syntéza takového řadiče vestavěného testu, který je schopen provést testy všech prostředí testu a architektura tohoto řadiče je zabudována do stávající obvodové struktury.

Navazující práce [SRJ98] se mj. zaměřuje na aplikačně specifické integrované obvody, aplikačně specifické programovatelné procesory, aplikačně specifické instrukční procesory popsané na úrovni meziregistrových přenosů a zavádí prostředky pro popis prostředí testu pomocí regulárních výrazů, popisuje symbolickou analýzu testovatelnosti jako proces pro řešení rovnic nad regulárními výrazy a ukazuje, že úplnost prostředí testu (tj. schopnost prostředí testu k uchování informace nutné pro nastavení resp. sledování hodnoty dané proměnné v daném čase) má podstatný vliv na generování symbolického testu. Prostředí testu daného modulu je zde chápáno jako množina řetězů mikroinstrukcí, které popisují všechny možné symbolické cesty pro testování daného modulu, tj. všechny cesty pro nastavování testovacích vzorků a všechny cesty pro sledování odezev.

Z poslední doby zmiňme např. práce [ARME02, ARME03], kde byla metoda symbolické analýzy testovatelnosti využita při syntéze pro snadnou testovatelnost s využitím samočinného testování - cílem bylo minimalizovat plochu, kterou zabírají obvody pro podporu samočinného testování a dosáhnout testovatelnosti co největšího počtu modulů v obvodu. Pro optimalizaci tohoto problému bylo použito simulovaného žlhaní.

Přístup založený na třídách registrů a analýze I-cest

Další metoda pro analýzu testovatelnosti, spolu s metodikou pro výběr registrů do řetězce scan, modelem aplikace testu pomocí Petriho sítě a analýzou kolizí při přenosu diagnostických dat, je popsána v [Růž02].

Metoda analýzy testovatelnosti [Růž02] je analýzou datových cest obvodu a je konstruována pro obvody na úrovni meziregistrových přenosů. Předpokládá, že struktura obvodu je tvořena propojením tří typů prvků - multiplexorů, registrů a funkčních jednotek. Pro otestování daného obvodového prvku, tj. pro přenos diagnostických dat tohoto prvku je důležité, aby ve struktuře obvodu existovaly dvě cesty (tzv. I-cesty) - cesta pro přenos testovacích vzorků z primárních vstupů obvodu a cesta pro přenos odezev na

²⁵ angl. badness count

primární výstupy obvodu. Proto je výhodné, aby obvodové prvky (zejména funkční jednotky) dokázaly pracovat v tzv. *I-režimu*, tj. režimu činnosti, kdy jsou schopny přenést data z některého ze svých vstupů na své vybrané výstupy. V [Růž02] jsou výše uvedené skutečnosti, tj. zejména struktura číslicového obvodu popsaného na úrovni meziregistrových přenosů, transparentní vlastnosti obvodových prvků a navržených algoritmů popsány formálně pomocí pojmů diskrétní matematiky. Veškeré objekty, které jsou předmětem analýzy (obvodové prvky, spoje atd.), jsou podle svých vlastností sdruženy do množin; další vlastnosti a vztahy mezi těmito objekty jsou vyjádřeny relacemi, pro popis je volen jazyk predikátové logiky. Výhodami formálního přístupu jsou zejména jednoznačnost zápisu a možnost transformovat problémy analýzy testovatelnosti na známé a řešené problémy diskrétní matematiky a teoretické informatiky a využít známých, efektivních a ověřených postupů a algoritmů.

Práce [Růž02] rovněž pracuje s pojmy říditelnost a pozorovatelnost, nikoliv však v pojetí většiny přístupů k analýze testovatelnosti coby k mírám přístupnosti, ale jako k vyjádření pouhé vlastnosti obvodového uzlu - uzlu je v pojetí [Růž02] buď říditelný/pozorovatelný nebo není. Zvláštní role náleží při analýze testovatelnosti popsané v [Růž02] obvodovým registrům. Je to dáno tím, že [Růž02] vychází z principů tzv. strukturovaného návrhu, kdy je oddělena kombinační a sekvenční logika. Právě v registrech se při aplikaci testu ukládají diagnostická data - obdobně je tomu i v režimu normální funkce obvodu. Registry se tak stávají významnými body na *i*-cestách, přes které diagnostická data obvodem procházejí. Pokud není analýzou nalezena vhodná *i*-cesta mezi primárními vývody obvodu a bodem, kam resp. z kterého je třeba diagnostická data dopravit, hledá se vhodný registr, který může být využit při testování takového bodu tím, že zajistí říditelnost resp. pozorovatelnost *i*-cesty vedoucí z registru do tohoto bodu resp. vedoucí z tohoto bodu do registru. Tento registr je pak třeba v duchu strukturovaného návrhu upravit tak, aby byl přístupný na primárních vývodech obvodu - mechanismus zlepšení testovatelnosti v [Růž02] pak směřuje k tomu, aby právě takové registry byly začleňovány do sériového scan řetězce a aby množina takto vybraných registrů měla minimální kardinalitu.

3.3.3 Vyšší úrovně popisu

Rozsáhlost a složitost moderních obvodových návrhů vede k tomu, že jsou tyto návrhy často popisovány na vyšších úrovních abstrakce - např. popisem chování. Přestože velká většina metod pro analýzu testovatelnosti je konstruována pro nižší úrovně popisu, existují i metody schopné pracovat na úrovni popisu chování.

Výhody metod pracujících na nižších úrovních abstrakce (např. úrovni hradel či úrovni meziregistrových přenosů) jsou zřejmé - vycházejí ze strukturálního popisu, který je velmi blízký konečné implementaci na cílové platformě. Tyto metody tedy mohou poskytnout poměrně přesnou informaci o testovatelnosti daného návrhu. Jejich nevýhodou může být výpočetní nároč-

nost v případě jejich aplikace na rozsáhlé návrhové celky. S cílem snížení výpočetní náročnosti analýzy testovatelnosti rozsáhlých návrhů proto vznikají metody analýzy testovatelnosti pracující na úrovni popisu chování. Velkou nevýhodou těchto vysokoúrovňových metod je, že analýza je prováděna ještě před dokončením vysokoúrovňové syntézy, čímž těmto metodám chybí strukturální informace o daném návrhu a jejich výsledky tedy mohou poskytovat nepřesné informace o testovatelnosti návrhu.

Práce [CA90, CS02] prezentují analýzu testovatelnosti číslicového systému popsaného na úrovni chování. Metoda provádí rozklad množiny proměnných na třídu úplně říditelných proměnných (CC) a na třídu neúplně říditelných proměnných (NCC). Proměnné jsou rozděleny do tříd na základě toho, do jaké míry je možno ovládat hodnoty těchto proměnných. Je-li možno libovolně ovládat hodnoty bitů dané proměnné, je tato proměnná prvkem CC , jinak je prvkem NCC . Pro každou proměnnou N náležící NCC je definována míra účinnosti $EFF(N) = \frac{[\sum_{n_t \in NCC_t} B(n_t) - \sum_{n_m \in NCC_m(N)} B(n_m)]}{B(N)}$, kde NCC_t je původní množina NCC proměnných, $NCC_m(N)$ množina NCC proměnných poté, co byla testovatelnost proměnné N zlepšena vložením testovacího bodu a $B(N)$ je počet říditelných bitů proměnné N . Její hodnota je ukazatelem zlepšení testovatelnosti obvodu vložením testovacího bodu pro proměnnou N . Obdobným způsobem je definována i míra pro ohodnocení pozorovatelnosti, avšak ta není při volbě testovacích bodů zohledněna. To, že proměnné je možno zařadit pouze do jedné ze dvou tříd a že CC -proměnná musí splňovat přísná kritéria, vede k tomu, že většina proměnných je klasifikována jako NCC proměnné. Další nevýhodou je, že metoda nepřipouští existenci cyklů v grafu obvodu, jelikož by to mohlo vést k jejímu zacyklení.

V [SH00, SH02] je prezentována analýza testovatelnosti založená na šíření rozsahů hodnot proměnných. Výpočet říditelnosti a pozorovatelnosti (proměnných z VHDL popisu chování daného číslicového systému) je prováděn na tzv. SSA reprezentaci [SH00]. Pro popis m vážené hodnoty je zavedena notace $\{W_i[L_i; U_i; S_i], \dots\}$, kde L_i resp. U_i je dolní resp. horní hranice daného rozsahu hodnot a S_i je počet kroků, které je třeba vykonat k přechodu z hodnoty L_i na hodnotu U_i , $i = 1, 2, \dots, m$. Testovatelnost systému je dána říditelností a pozorovatelností proměnných definovaných v popisu chování tohoto systému. Říditelnost proměnné V o rozsahu hodnot $W[L : U : S]$ je pro hodnotu a z tohoto rozsahu dána vztahem $C_a = \frac{1}{P_a}$, kde P_a je pravděpodobnost nastavení proměnné V na hodnotu a . Pozorovatelnost O_V proměnné V vyjadřuje snadnost pozorování hodnoty proměnné V na primárním výstupu obvodu. Dále existují zvláštní vztahy pro výpočet říditelnosti resp. pozorovatelnosti proměnné vyskytující se v programovém cyklu, proměnné, za níž následuje větvení atd. Proměnné jsou seřazeny sestupně od proměnných s nejvyšší hodnotou říditelnosti po proměnné s nejnižší hodnotou říditelnosti. Je-li proměnná obtížně říditelná, ale snadno pozorovatelná, stává se kandidátem pro vložení testovacího bodu. Jinak je její testovatelnost zlepšena pomocí techniky částečný scan. Hlavní nevýhodou této metody je, že počet průběhů

každého cyklu musí být předem znám, aby bylo možné provést rozbalení smyček. Další nevýhodou je, že stavovému registru v popisu chování obvykle neodpovídá žádná proměnná, což znamená, že není možno ohodnotit jeho říditelnost a pozorovatelnost a zjistit tak jeho testovatelnost. Obdobnou metodu je možno nalézt v [SS03].

Systém SATAN

Systém SATAN (Systems Automatic Testability Analysis) [MKB01] byl vytvořen pro analýzu testovatelnosti a generování funkčních testů a je založen na grafovém modelu přenosu informací obvody. Uzly grafu reprezentují vývody a moduly obvodu, hrany grafu reprezentují různé režimy přenosu informací mezi uzly. Tok informací modelem spočívá v informačních cestách ze vstupů obvodu na jeho výstupy, přičemž během toku jsou informace transformovány moduly vyskytujícími se na těchto cestách. Graf přenosu informací vzniká v systému z popisu chování obvodu vyjádřeného soustavou rovnic sestavených z elementárních operací.

V systému je zaveden pojem tok, vyjadřující informační cestu ze vstupů obvodu na jeho výstupy a pojem matice pokrytí, v níž je pro každý modul zaznamenáno, zda je daným tokem využíván. Moduly se stejnými množinami toků jsou nazvány nerozlišitelné a tvoří tzv. množinu nerozlišitelnosti.

Analýza testovatelnosti v systému SATAN pak sestává z těchto kroků:

- definování toků,
- ohodnocení testovatelnosti komponent,
- strukturování testu do sekvence toků podle zvolené strategie testování a
- ohodnocení kvality zvolené sekvence z pohledu diagnostiky.

Cílem systému je nalézt všechny funkční aktivity²⁶ na úrovni obvodu z pohledu jednotlivých komponent. Konkrétní funkční aktivace obvodu je charakterizována jistým tokem - ten určuje: vstupy obvodu, které je třeba nastavovat pro ovládání cest potřebných pro požadovanou funkci, výstupy obvodu, na nichž je možné pozorovat výsledky provedení dané funkce a komponenty aktivované během provádění funkce.

Celková testovatelnost obvodu je pak vypočtena na základě odhadu hodnot říditelnosti a pozorovatelnosti komponent v různých tocích. Odhady říditelnosti resp. pozorovatelnosti jsou provedeny podle množství informací přístupných na vstupech komponenty z primárních vstupů toků resp. podle množství informací z výstupů komponenty pozorovatelných na primárních výstupech toků.

²⁶ funkční aktivací je myšleno provádění konkrétní dílčí funkce, kterou obvod vykonává

Nástroj FACTOR

V článku [VA02] je prezentován princip generování funkčního hierarchického testu pomocí nástroje FACTOR (FUnctional Constraint extractOR). Metoda implementovaná v nástroji FACTOR je založena na rozčlenění obvodu na moduly (komponenty), přičemž pro každý modul, který má být testován, je analyzováno jeho okolí, jsou nalezeny zdroje a cíle signálů pro jeho testování a cesty pro šíření hodnot těchto signálů. Zejména se jedná o získání tzv. *def – use* nebo *use – def* řetězců, které obsahují příkazy, umožňující snadný průchod hodnot signálů obvodovou strukturou. Dále jsou získány údaje informující o funkčních omezeních tohoto modulu a popisují zjednodušený pohled na modul z hlediska generátoru testu. Po této etapě obvykle následuje analýza funkčních omezení dílčích modulů a zahájení generování testu.

Před vlastním generováním testu je možno získaných informací - zejména informací o signálech, kterým přísluší prázdný *def – use* nebo *use – def* řetězec - využít k odhadu obtížně testovatelných signálů a následně i odhadu testovatelnosti příslušných částí obvodu. Návrhář tak může být informován o případných nedostatcích v testovatelnosti, tyto nedostatky napravit a zvýšit tímto krokem efektivitu generovaného testu.

Jelikož za vyšší úroveň popisu je často označován algoritmičtý popis činnosti obvodu, mají metody pracující na těchto vysokých úrovních popisu velmi blízko k metodám analýzy testovatelnosti "softwarových algoritmů" publikovaným např. v pracích [TBN02, TBN03, Qua04, Gao04]. Přístupy řešící problematiku analýzy testovatelnosti softwaru často vycházejí z principů metod zabývajících se analýzou testovatelnosti obvodů - jako příklad uveďme metodu [TBN02] inspirovanou metodou tvořící jádro výše zmíněného systému SATAN.

3.3.4 Shrnutí

Přes rozdílnost jednotlivých konkrétních přístupů k analýze testovatelnosti lze existující přístupy souhrnně charakterizovat následujícími společnými znaky:

- v přístupu je použit nějaký popis, v ideálním případě formální model, daného systému. Model umožňuje popsat jak strukturu daného systému, tak i vlastnosti dílčích prvků tohoto systému včetně jejich rozhraní, modelu vybraných vlastností a chování. Obvykle se jedná o jistý model datových popř. také řídicích cest systému a o model přenosu diagnostických dat po těchto cestách. Algoritmus analýzy testovatelnosti je pak popsán s využitím prostředků tohoto modelu,
- algoritmus analýzy testovatelnosti obvykle provádí ohodnocení zvolených diagnostických vlastností. Na základě tohoto ohodnocení lze rozpoznat obtížně testovatelné části v návrhu a zdroje této obtížné testovatelnosti. Nevýhodou je, že pojmy z této oblasti nejsou standardizovány. Vede to

k rozdílným definicím testovatelnosti a jejích složek a tím i k ohodnocování testovatelnosti s přihlédnutím k různým ovlivňujícím faktorům a pomocí odlišných měr,

- obecnou snahou přístupů je, aby výpočetní složitost algoritmu analýzy testovatelnosti byla mnohem menší než výpočetní složitost nejefektivnější metody generování testu pro tutéž třídu obvodů a tutéž úroveň popisu [ABF90]. Jelikož algoritmus analýzy testovatelnosti je vždy úzce spjat s předpokládaným způsobem generování testu [Uba04], pak by návrh algoritmu nesplňujícího tuto podmínku nedával smysl. V takovém případě by k témuž účelu bylo možno využít generátor testu. Ten poskytuje přesnou informaci o testovatelnosti návrhu pro daný typ testu. Lze tedy konstatovat, že úkolem analýzy testovatelnosti je poskytnout rychlý, avšak co nejméně zkreslený odhad testovatelnosti daného návrhu vzhledem k předpokládanému typu generování testu.

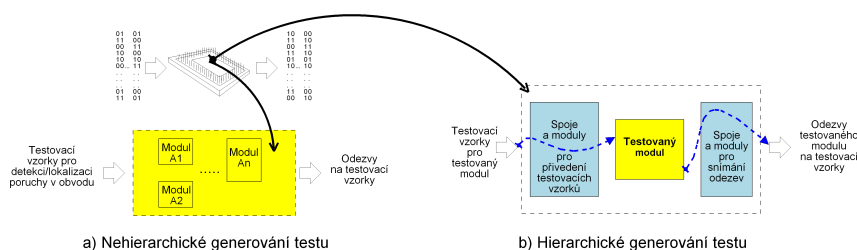
3.4 Hierarchický test

Dříve než se budeme blíže věnovat kapitolám zahajujícím prezentaci vlastního přínosu této práce v dané oblasti výzkumu, vrátíme se na chvíli k problematice generování testu číslicových obvodů, přičemž se zaměříme zejména na pojmy a myšlenky související s tzv. *hierarchickým testem* - prostor jim není věnován zbytečně a z dalšího textu vyplyne, jakým způsobem souvisejí s vlastním jádrem této disertační práce.

Bylo dokázáno [Fuj85], že generování testu pro obecný obvod je NP-úplný problém, proto stále existují snahy o vylepšení stávajících metod resp. snahy o návrh nových efektivnějších metod. Poměrně velké úsilí bylo věnováno zejména nízkourovňovým metodám generování testu pro číslicové obvody. Tyto metody jsou však prakticky limitovány na "menší" kombinační, popř. "jednoduché" sekvenční obvody. Na druhou stranu však tyto metody (díky detailní znalosti nízkourovňové struktury obvodu) umožňují generovat velmi kvalitní testy, obecně se vyznačující vysokým pokrytím poruch dosažitelným relativně malým počtem testovacích vektorů. Omezení praktické použitelnosti těchto metod je pak dáno jednak NP-úplností problému generování testu pro obecný obvod a jednak poměrně nízkou úrovní popisu, na které tyto metody pracují.

Uvědomíme-li si, že obecně jsou číslicové systémy v dnešní době mnohem složitější než např. v době vzniku D-algoritmu a že jsou v návrhových nástrojích obvykle navrhovány na vyšších úrovních popisu, než je úroveň hradel (viz např. podkapitoly 2.1 a 2.2), pak tušíme, že generování testu pro obvodový celek popsany na úrovni hradel (s popisem odpovídajícím danému vysokoúrovňovému popisu - viz obrázek 17a) bude obecně složitější, než generování testu pro jednotlivé podčásti (např. funkční jednotky, registry, multiplexory atd. nebo větší obvodové celky - aritmeticko-logické jednotky, bloky

60 3 Blízká témata



Obrázek 17. Ilustrace rozdílu mezi hierarchickým a nehierarchickým generováním testu

pamětí RAM, (mikro)řadiče, (mikro)procesory atp. - viz obrázek 17b), na které je možné obvod rozčlenit.

Právě na základě této myšlenky vznikla koncem 80. let oblast zabývající se tzv. *hierarchickým generováním testu* - např. [CB89, MH90, PVS93, LP97, YMV99, MO02, VA02, VMVT03], od níž se očekával zejména návrh metod vedoucích k výpočetně méně náročnému generování testu pro rozsáhlé (kombinační i sekvenční) obvody při současném dosažení obdobných kvalit testu (pokrytí poruch, počet testovacích vzorků atd.) jako u metod generování testu pracujících na nízkých úrovních popisu. Princip hierarchického generování testu je možné vyjádřit jako posloupnost následujících činností:

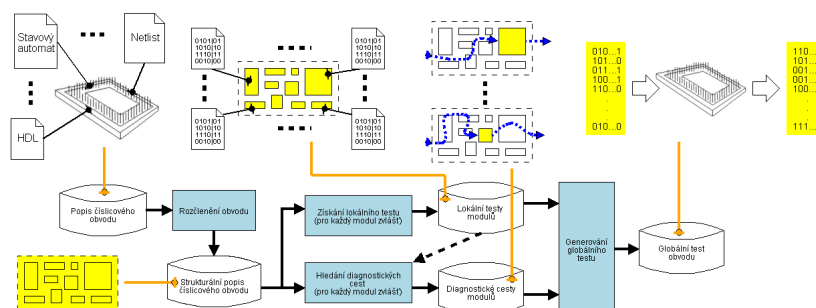
- rozčlenění obvodu na podčásti (moduly),
- generování/získání (tzv. *lokálního*) testu pro jednotlivé moduly,
- hledání diagnostických cest pro jednotlivé moduly,
- generování (tzv. *globálního*) testu pro obvod jako celek.

Rozčlenění obvodu na moduly

Rozčlenění obvodu na moduly (viz obrázek 18) bývá, zejména v případě strukturovaného návrhu obvodu, z velké části určeno již samotným stylem návrhu obvodu a/nebo, zejména v případě návrhu obvodu popisem jeho chování, bývá provedeno (či usnadněno) nějakým nástrojem vhodným pro tento účel, např. s využitím nástroje pro vysokoúrovňovou syntézu (viz odstavec 2.2.2, strana 11). Cílem je dosáhnout rozčlenění jeho struktury na vhodně složité vzájemně propojené moduly, což v podstatě znamená získat jistý strukturální popis daného obvodu (viz odstavec viz odstavec 2.1.2, strana 8).

Získání lokálních testů

Při hierarchickém generování testu se předpokládá, že každý modul, u kterého se požaduje testování, je vybaven diagnostickými daty, tj. testovacími vektory a odpovídajícími správnými odezvy. Tato data (tvořící tzv. *lokální test* modulu) mohou být získána (viz obrázek 18) např. spolu popisem daného



Obrázek 18. Ilustrace k principu hierarchického testu

modulu od dodavatele nebo je lze individuálně pro každý modul získat např. některou z metod pro generování testu. Podstatné je, že lokální test je testem pro samostatný (tj. do nadřazeného systému nezapojený) modul.

Hledání diagnostických cest

Je-li daný modul zapojen do systému, pak je nutné vyřešit, jakým způsobem (bude-li to pro každý vzorek lokálního testu modulu, vzhledem k zapojení modulu do systému a ke struktuře systému, vůbec možné) bude prováděn přenos diagnostických dat mezi vývody systému a vývody modulu. Poté, co jsou známy lokální testy jednotlivých modulů, je třeba pro každý modul analyzovat strukturu obvodu jednak za účelem nalezení cest pro přivedení testovacích vzorků²⁷ na vstupy modulu a jednak za účelem nalezení cest pro odvedení odezev z výstupů modulu na primární výstupy obvodu²⁸. Zkráceně pak hovoříme o hledání *diagnostických cest*²⁹ daného modulu (viz obrázek 18). Umožňuje-li to struktura daného obvodu, pak je pro každý modul po skončení této analýzy známo, jakou posloupností hodnot generovanou na primárních vstupech obvodu lze zajistit výskyt požadovaných dat (testovacích vzorků) na vstupech modulu resp. jakou posloupností lze zajistit pozorování dat (odezev)

²⁷ tj. je třeba nalézt způsob, jakým bude umožněno nastavení testovacích vzorků z primárních vstupů obvodu na vstupy modulu

²⁸ tj. je třeba nalézt způsob, jakým bude umožněno sledování odezev na výstupech modulu na primárních výstupech obvodu

²⁹ v obrázku 17 resp. 19 jsou všechny diagnostické cesty pro daný testovaný modul souhrnně reprezentovány dvojicí přerušovaných křivek, z nichž každá je zakončena šipkou ve tvaru vyplněného trojúhelníku; směr šipky zobrazuje směr toku diagnostických dat - ta procházejí diagnostickými cestami ve směru z primárních vstupů obvodu na primární výstupy obvodu. Křivka vlevo resp. vpravo od testovaného modulu symbolizuje diagnostické cesty pro nastavení testovacích vzorků na vstupy testovaného modulu resp. diagnostické cesty pro pozorování odezev z výstupů testovaného modulu

z výstupů modulu. Současně jsou známy diagnostické cesty, které mohou být pro přenos těchto hodnot využívány.

Generování globálního testu

Poznamenejme, že obecně jsou testy jednotlivých (vzájemně různých) modulů disjunktní; to však již obecně neplatí o diagnostických cestách modulů, kdy dílčí úseky diagnostických cest bývají často společné několika modulům. Proto mohou existovat data, jejichž výskyt na diagnostické cestě po nějakou dobu znemožní výskyt jiných dat a naopak usnadní výskyt dat dalších. Pro účely generování efektivního testu je tedy třeba důsledně zvážit posloupnost aplikace testovacích vzorků na jednotlivé moduly (resp. posloupnost hodnot na primárních vstupech obvodu). Poslední etapa hierarchického generování testu, jejímž výsledkem je tzv. *globální test* obvodu (viz obrázek 18) tedy spočívá v nalezení takové posloupnosti testovacích vzorků, která (bude-li v daném pořadí aplikována na primární vstupy obvodu) v ideálním případě umožní v co nejkratším čase přivést testovací vzorky na vstupy všech modulů a zachová rozlišitelnost správnosti odezev modulů na primárních výstupech obvodu.

Realizace hierarchického generování testu je však prakticky vždy spojena s jistou (oproti nižší úrovni popisu ještě větší) abstrakcí od cílové obvodové struktury; proto současné metody pro hierarchické generování testů kvalitativně nedosahují výsledků nízkourovňových metod, čímž je omezena zejména jejich praktická použitelnost při řešení problematiky efektivního generování testu a tím i jejich konkurenceschopnost ve vztahu k ostatním metodám generování testu. Příčinou těchto nedostatků a omezení je zejména tzv. *koncepte transparentnosti*, z níž tyto metody obvykle vycházejí a která znamená vnesení dalších abstrakcí. Přestože byla tato koncepte podrobně studována a postupně rozpracována pro účely generování testu, v praxi je z důvodů výpočetní náročnosti použitelná pouze její malá část. Na příčiny nevýhod spojených s její použitelností upozorňují např. práce [PVA93, IGJ96b, YMV99, MO02] a zároveň navrhují řešení, jak tyto nevýhody odstranit.

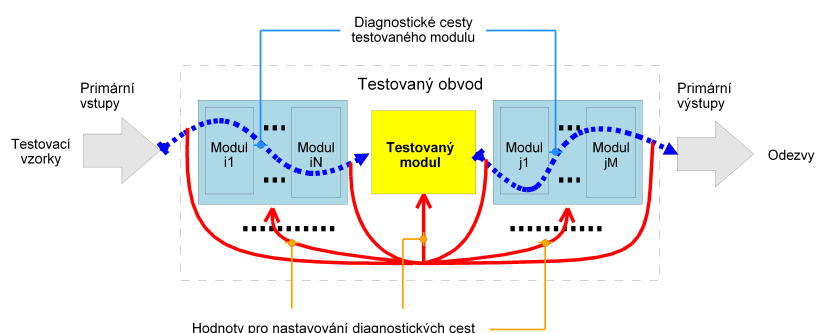
3.4.1 Přehled základních pojmů koncepte transparentnosti

Jelikož pro jádro této disertační práce je z oblasti hierarchického generování testu nejvíce podstatná výše zmíněná koncepte transparentnosti, nebudeme se v dalším textu více zabývat přehledem ani principy vybraných metod hierarchického generování testu (bližší informace je možno nalézt např. v [CB89, MH90, PVS93, LP97, YMV99, MO02, VA02, VMVT03]), ale zaměříme se především na přiblížení základních principů, z nichž koncepte transparentnosti vychází.

Nastavování diagnostických cest

Nyní předpokládejme, že rozčlenění číslicového obvodu na moduly již je nějakým způsobem vyřešeno a že pro každý modul (který má být testován) je

k dispozici jeho lokální test. Pro úspěšné dokončení hierarchického generování testu pak zbývá pro každý modul nalézt jemu příslušející diagnostické cesty a na jejich základě vygenerovat globální test obvodového celku. Již v obrázku 17b, bylo naznačeno, že diagnostické cesty daného modulu jsou tvořeny jednak moduly, vyskytujícími se na datových cestách mezi vývody testovaného modulu a vývody obvodu a jednak spoji, určujícími, které vývody těchto modulů a jakým způsobem jsou vzájemně propojeny.



Obrázek 19. Bližší pohled na hierarchické testování

Tedy, chceme-li za výše uvedeného předpokladu dosáhnout toho, aby na vstupy konkrétního testovaného modulu (viz obrázek 19) byl přiveden daný testovací vzorek resp. aby bylo možné vyhodnotit správnost odezvy testovaného modulu, obecně musíme nejdříve vyřešit, jak bude tento vzorek přenesen přes moduly a spoje, vyskytující se na diagnostických cestách testovaného modulu. To znamená nalézt alespoň jednu posloupnost hodnot³⁰ (existuje-li taková), jejichž postupným generováním na primárních vstupech obvodu bude zajištěno postupné nastavování částí diagnostických cest, umožňujících přenos diagnostických dat a po generování poslední hodnoty bude zaručen výskyt daného vzorku na vstupech testovaného modulu resp. výskyt nezkrácené informace o správnosti odezvy na primárních výstupech obvodu.

Jsou-li spoje propojující rozhraní modulů neměnně ustaveny, pak smyslem této posloupnosti hodnot je ovlivňovat pouze činnost konkrétních modulů vyskytujících se v diagnostických cestách testovaného modulu a zajistit tak (vhodným směřováním toku dat, a/nebo vhodnou transformací dat) nastavení

³⁰ hodnoty pro nastavení diagnostických cest jsou závislé na hodnotách, které se vyskytují v datových cestách obvodu - proto obecně platí, že diagnostické cesty pro daný testovaný modul a obecný obvod nemohou zůstat nastaveny po celou dobu přenosu všech diagnostických dat mezi vývody testovaného modulu a obvodu. Pro přenos diagnostických dat je však postačující, aby bylo možné nastavovat diagnostické cesty "po částech"

konkrétní části dané diagnostické cesty testovaného modulu za účelem přenosu dílčích dat a výsledného přenosu diagnostických dat mezi vývody testovaného modulu a vývodu obvodu.

Pokud předpokládáme, že existence neporušeného fyzického spoje mezi dvěma uzly zaručuje, že hodnota vyskytující se na zdrojovém uzlu se bude (s jistým zanedbatelným zpožděním) vyskytovat i na cílovém uzlu, pak lze z hlediska přenosu diagnostických dat očekávat výskyt možných problémů pouze při přenosu dat přes moduly, které se vyskytují na diagnostických cestách testovaného modulu.

Transparentnost modulů

Odhlédneme-li od konkrétní funkce, kterou každý modul plní, pak lze moduly z hlediska přenosu diagnostických dat obvodovou a jejich vnitřní strukturou rozčlenit na moduly

- neumožňující přenos diagnostických dat,
- umožňující částečný/úplný přenos diagnostických dat,
- umožňující generování diagnostických dat.

Moduly mohou pracovat obecně v několika *režimech činnosti*³¹, a proto nebude nijak výjimečným, nalezneme-li modul, který současně patří do více než do jedné z výše uvedených skupin. Ze všech skupin se však zaměříme zejména na skupinu modulů umožňujících částečný/úplný přenos diagnostických dat. Pro tyto moduly lze nalézt podmínky, za kterých jsou tyto moduly částečně či zcela ”průhledné”³² (transparentními) pro datový tok a umožňují tak přenos požadovaných dat přes svou strukturu.

Jedna z prvních prací [AB85] zabývajících se myšlenkou přenosu diagnostických dat přes moduly se snažila pro každý modul M nalézt takový režim činnosti (M_W), ve kterém by daný modul byl schopen přenést data $a \in \{0, 1\}^m$ z některého svého (m -bitového) vstupu x_{iM} na některý (m -bitový) výstup y_{jM} . Znamená to, ze všech režimů činnosti modulu M , nalézt alespoň jeden režim M_{W_k} , kdy existuje bijekce $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$ taková, že $g(y_{jM}) = f((g(x_{iM})))$, kde g je zobrazení přiřazující m -bitovému vývodu modulu M m -bitovou hodnotu, která se na něm vyskytuje.

Existuje-li taková bijekce f , pak režim M_{W_k} bývá označován pojmem *T-režim*³³ [AB85], jelikož umožňuje aby ze vstupu x_{iM} byla na výstup y_{jM} (po jisté transformaci) přenesena konkrétní data.

Je-li f navíc identickým zobrazením na množině $\{0, 1\}^m$ (tj. platí-li navíc $\forall a \in \{0, 1\}^m: f(a) = a$), pak režim M_{W_k} bývá označován [AB85] pojmem

³¹ tento pojem bude upřesněn až v dalším textu (viz definice 13, strana 86). Nyní jej budeme chápat jako stav, ve kterém modul aktuálně pracuje a který je závislý na datech vyskutujících se na jistých vstupech modulu

³² z angl. transparent

³³ z angl. transfer

*I-režim*³⁴, jelikož umožňuje aby data ze vstupu x_{iM} byla přenesena (tj. beze změn, kdy platí $g(y_{jM}) = g(x_{iM})$) na výstup y_{jM} . Je zřejmé, že I-režim je speciálním případem T-režimu (viz např. obrázek 20c).

Jelikož f je bijekce, pak lze pomocí libovolného z těchto režimů zajistit, aby se na daném výstupu vyskytla konkrétní data. Proto, existuje-li pro modul M režim M_{W_k} , který je T-režimem, pak pro přenos diagnostických dat modulem není nutné vyžadovat, aby režim M_{W_k} byl také I-režimem. Nicméně lze nalézt skupiny modulů, u nichž je existence I-režimu a/nebo T-režimu dána již samotným principem činnosti těchto modulů. Jako zástupce modulů s T-režimem uveďme sčítačky, odčítačky, některé násobičky, bloky pro negaci, moduly pro rotace a další. Z modulů vybavených I-režimem zmiňme např. multiplexery/multiplexory, registry, sčítačky a odčítačky.

Příklad 3:

Na obrázku 20 jsou uvedeny příklady několika modulů a jejich vybraných T-režimů resp. I-režimů. Vstupy resp. výstupy modulů jsou umístěny vlevo resp. vpravo od schematické značky modulu, příslušný režim M_{W_k} je na obrázku zapsán jako množina $\{(x, y) \mid x \text{ je vstup modulu } M \text{ nutný pro nastavení režimu } M_{W_k}, y \in \{0, 1\}^+ \text{ je hodnota, která musí být pro nastavení a držení režimu } M_{W_k} \text{ po jistou dobu přítomna na vstupu } x\}$.

□

Na koncepci [AB85] navazují další práce - zmiňme zde pouze

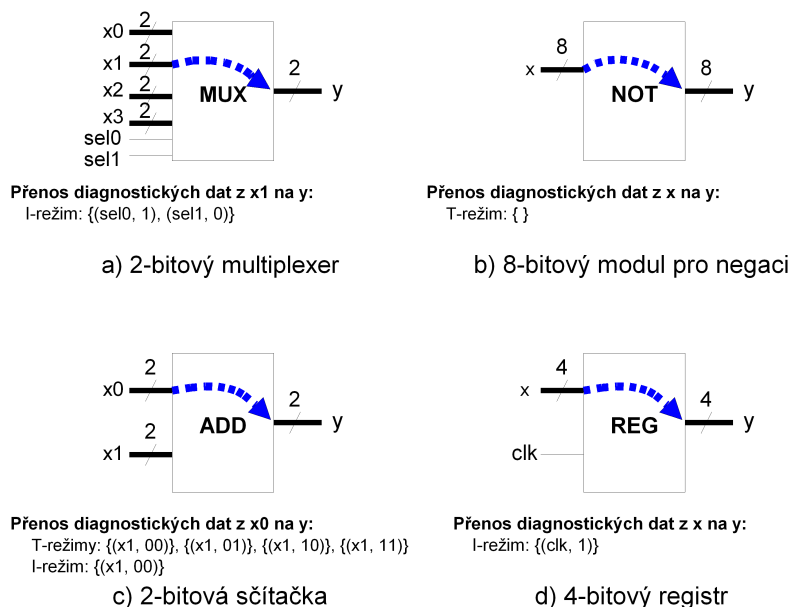
- *nejednoznačné množiny*, [MH91] - vzhledem k velké složitosti je tento přístup použitelný pouze na malé obvody,
- *transparentní režimy*, [PVA92] - automatická extrakce informací o transparentnosti z behaviorálního popisu činnosti obvodových prvků kombinovaného se strukturálním popisem jejich propojení,
- *transparentní vlastnosti*, [YM99, MO02] - popis transparentnosti prvků pomocí jednoduchých symbolických operací.

Přestože z teoretického hlediska je možné všechny výše uvedené a také další - např. [MO02, YMV99, MH90, PVS93, LP97, IGJ98, RSTA99] - přístupy využít pro generování hierarchického testu, prakticky jsou, díky jejich velké výpočetní náročnosti, použitelné jen jisté podmnožiny těchto koncepcí. I ty jsou však v praxi často degradovány pouze na úroveň koncepce I-cest resp. T-cest.

Výše uvedené přístupy rozšiřují myšlenky koncepce I-režimů a T-režimů o další prvky a v podstatě se snaží o její zobecnění, aby bylo možné při hierarchickém generování testu využít pro přenos diagnostických dat i těch prvků (příklad viz obrázek 21), které sice nejsou z principu své činnosti schopny pracovat v I-režimu resp. T-režimu, avšak které jsou alespoň částečně využitelné

³⁴ z angl. identity

66 3 Blízká témata



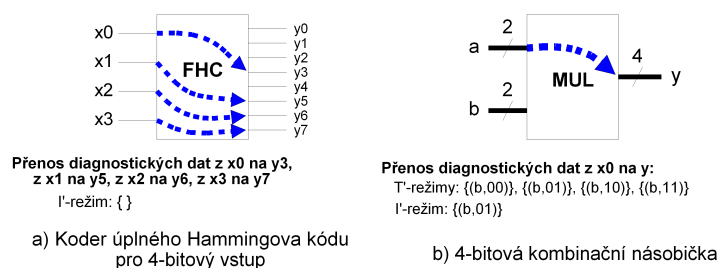
Obrázek 20. Ilustrace k T-režimu a I-režimu

pro přenos diagnostických dat. Pro takové moduly obvykle bývá typickým, že sice mnohdy jsou schopny zajistit přenos jistých konkrétních n -bitových dat z některého svého vstupu na některý ze svých m -bitových výstupů, $n \leq m$, avšak již nejsou schopny poskytnout na svých výstupech libovolnou kombinaci hodnot, ale pouze jistou podmnožinu. Příčinou jsou zejména datové závislosti mezi výstupními hodnotami resp. jejich podčástmi - viz příklad modulů na obrázku 21 (kodér není schopen na svém výstupu zajistit např. hodnotu 00001111, násobička není na svém výstupu např. schopna zajistit každé prvočíslo zapsatelné na 4 bitech). Důsledkem toho je, že není-li to zajištěno jinak (např. modifikací modulu či aplikací některé z DFT technik), nelze zaručit úplnou říditelnost datového toku z takových výstupů, čímž je jistě obecně zhoršena transparentnost takových modulů a říditelnost uzlů následujících za výstupy těchto modulů.

Není-li modul M schopen pracovat v I-režimu resp. T-režimu, je obecnou snahou metod, navazujících na [AB85], nalézt režim činnosti M_{W_k} , množiny M_X resp. M_Y podčástí vstupů resp. výstupů modulu M tak, aby $g''(y') = f'(g'(x'))$, kde

- x' je virtuální vstup³⁵ vytvořený z prvků množiny M_X ,

³⁵ je běžné, že z důvodu přehlednosti schématu i návrhu jsou jisté spolu související jednobitové vývody modulu seskupeny (obvykle návrhářem) do jednoho vícebitového vývodu, který je "zapouzdřuje". Na vícebitový vývod se pak dále



Obrázek 21. Příklad prvků nemajících I-režim resp. T-režim

- y' je virtuální výstup vytvořený z prvků množiny M_Y ,
- g' resp. g'' je zobrazení, přiřazující virtuálnímu vývodu modulu hodnotu, která se na něm vyskytuje,
- f' je zobrazení $f' : \{0, 1\}^{W_{X'}} \rightarrow \{0, 1\}^{W_{Y'}}$, kde $W_{X'}$ resp. $W_{Y'}$ je bitová šířka virtuálního vstupu x' resp. výstupu y' .

Příklad 4:

Pro koder na obrázku 21a, je režim činnosti $M_{W_k} = \{\}$, množiny $M_X = \{x_0, x_1, x_2, x_3\}$, $M_Y = \{y_3, y_5, y_6, y_7\}$, virtuální vstup x' resp. výstup y' je uspořádaná čtveřice, v níž se každý prvek z M_X resp. z M_Y vyskytuje právě jednou a zobrazení f' je bijekce definovaná předpisem $g''(y') = f'(g'(x'))$.

□

V podstatě se jedná o nalezení dílčích³⁶ I-režimů resp. T-režimů daného modulu (na obrázku 21 je díleč I-režim resp. T-režim označen jako I-režim resp. T'-režim), přičemž je důležité, že tyto obecně nemusí být omezeny (jako v [AB85]) pouze na jeden vstup (tj. zdroj dat) a jeden výstup (tj. cíl dat), ani nemusí být omezeny na vývody stejné bitové šířky či na plné pokrytí šířky daných vývodů.

Lze snadno nahlédnout, že pro účely přenosu diagnostických dat nemůže být zobrazení f' libovolné a také, že není nutné, aby f' bylo bijektivním zobrazením. Budeme-li zkoumat požadavky kladené na přenos částí diagnostických dat, tj. na přenos testovacích vzorků a odezev, pak snadno zjistíme, že pro přenos testovacího vzorku postačuje, aby f' bylo surjektivní zobrazení a obdobně, že pro přenos odezvy postačuje, aby f' bylo injektivní zobrazení. Z tohoto pohledu lze režimy činnosti daného prvku rozdělit na:

pohlíží jako na jeden (logický) vývod. Rozhraní modulu se pak často nevyobrazuje detailně pomocí jednobitových vývodů, ale obvykle postačuje stručnější a přehlednější vyobrazení pomocí logických vývodů. Virtuálním vývodem budeme rozumět logický vývod, který je tvořen vybranými jednobitovými vývody modulu, a to bez ohledu na již utvořené logické vývody tvořící rozhraní modulu. Virtuální vývod tedy obecně sestává z vybraných podčástí jistých logických vývodů. Konkrétně pak hovoříme o virtuálním vstupu resp. výstupu

³⁶ ve smyslu podmnožin příslušných zobrazení

- režimy nevhodné pro přenos diagnostických dat,
- režimy vhodné pro přenos testovacích vzorků,
- režimy vhodné pro přenos odezev.

U některých modulů mohou také existovat režimy (pro které je f' zároveň injekce i surjekce, tzn. f' je bijekce), které jsou vhodné jak pro přenos testovacích vzorků, tak odezev.

Transparentní diagnostické cesty

Již bylo uvedeno, že spoje nepředstavují pro šíření diagnostických dat problém, jelikož každý spoj je pro data transparentním. Dále bylo ilustrováno, že od jednotlivých modulů nelze tuto vlastnost spojů obecně očekávat. Zřejmě, čím detailnější informaci o transparentnosti modulů je možno získat, tím je možno očekávat kvalitnější informaci o diagnostických cestách daného obvodu a efektivnější globální test.

Z pohledu hierarchického generování testu je tedy třeba pro každý modul, který má být testován, analyzovat transparentnost modulů, vyskytujících se v datových cestách mezi vývody testovaného modulu a vývody obvodu. Cílem této analýzy je nalézt diagnostické cesty jednak pro přenos testovacích vzorků ze vstupů obvodu na vstupy testovaného modulu a jednak pro přenos odezev z výstupů testovaného modulu na výstupy obvodu (viz např. obrázek 19, strana 63).

Příklad 5:

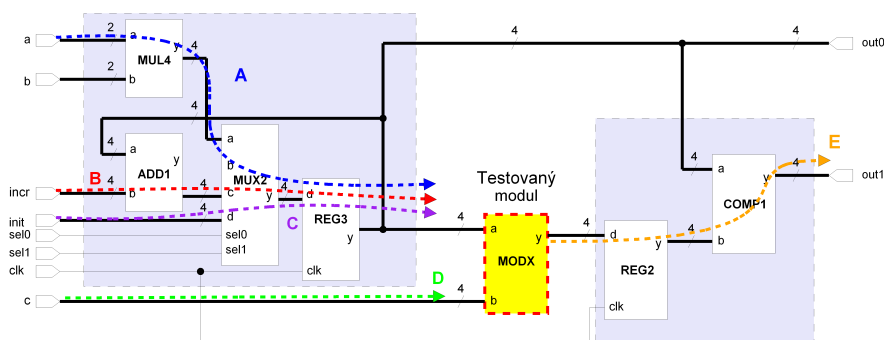
Na obrázku 22 je výsek obvodu, umožňující provádění několika základních operací (násobení, součet, rozdíl, od/čítání s volitelným krokem a inicializací či porovnání mezivýsledků). Může se např. jednat o část aritmeticko-logické jednotky. Testovaným modulem byl v tomto příkladu zvolen modul označený *MODX*; funkce tohoto modulu není pro tento příklad podstatná.

Byly vybrány a vlevo od odčítačky jsou vyobrazeny tři různé diagnostické cesty (značeny písmeny *A, B, C*) pro přenos testovacích vzorků ze vstupů obvodu na vstup *a* modulu *MODX* a jediná diagnostická cesta (značena písmenem *D*) pro přenos testovacích vzorků ze vstupu *c* obvodu na vstup *b* modulu *MODX*, vpravo je vyobrazena jediná diagnostická cesta (značena písmenem *E*) pro přenos odezev z výstupu *y* modulu *MODX* na výstup *out1* obvodu. Následuje komentář k jednotlivým diagnostickým cestám.

- cesta *A*: na výstupu *y* násobičky *MUL4* je možno generovat pouze podmnožinu 4-bitových dat. Pouze pomocí *MUL4* tedy nelze zajistit libovolný 4-bitový vzorek pro vstup *a* modulu *MODX*. Výstup *MUL4* však může zajistit podstatnou část testovacích vzorků - ty pak mohou být dále modifikovány sčítačkou *ADD1*, popř. inkrementovány či dekrementovány s daným krokem (v takovém případě by se však již nejednalo pouze o cestu *A*). Výstup *MUL4* tak může např. postačovat k nastavení transparentního režimu činnosti modulu *MODX* pro přenos diagnostických dat cestou *D*

- cesta *B*: nastavení cesty musí předcházet inicializace hodnoty v *REG3* (např. na hodnotu 0000); poté je možno testovací vzorek z obvodového vstupu *incr* přenést (přes vstup *c* multiplexeru *MUX2* a registr *REG3*) na vstup *a* modulu *MODX*
- cesta *C*: tato cesta slouží k inicializaci registru *REG3* na hodnotu přímo představující daný testovací vzorek pro vstup *a* modulu *MODX*
- cesta *D*: vstup *b* modulu *MODX* je bezprostředně říditelný z obvodového vstupu *c*, cesta *D* je plně transparentní (stejně jako cesty *B* a *C*) a existence cesty *D* ani jejích vlastností nezávisí (oproti cestám *B* a *C*) na hodnotách vyskytujících se na vstupech obvodu
- cesta *E*: odezvu modulu *MODX* je možné sledovat na výstupu *out1* obvodu za předpokladu, že byl generován hodinový pulz na vstupu *clk* registru *REG2* a následně aktivován vhodný režim činnosti modulu *COMP1* přiložením daného vzorku na jeho vstup *a*.

□



Obrázek 22. Ilustrace k transparentním diagnostickým cestám

I pomocí tohoto konkrétního a jednoduchého příkladu lze ilustrovat a shrnout následující obecné závěry:

- každá diagnostická cesta je podmnožinou datových cest daného obvodu a sestává jednak ze spojů a jednak z modulů, vyskytujících se na této cestě. Přitom se předpokládá, že tato cesta je nepřerušená, tj. že spoje a moduly na sebe vzájemně navazují. Diagnostická cesta obsahuje alespoň jeden spoj a nemusí obsahovat žádný modul,
- obsahuje-li diagnostická cesta alespoň jeden modul, pak je nutné (v konkrétním časovém okamžiku) zajistit, aby tento modul pracoval v transparentním režimu, umožňujícím existenci dílčího transparentního úseku

této cesty - režim modulu obecně závisí na hodnotách, které se vyskylují na jeho virtuálních vstupech. Jelikož spoje (a také některé moduly - viz např. modul *NOT* na obrázku 20, strana 65) jsou transparentní vždy, pak množina vstupů nutných pro řízení jejich režimu je prázdná³⁷. Takové moduly jsou zřejmě podmnožinou kombinačních obvodů,

- každou diagnostickou cestu tedy charakterizuje jednak část určená pro přenos diagnostických dat a jednak část určená pro řízení tohoto přenosu. Zatímco datová část diagnostické cesty je tvořena posloupností spojů a modulů, tak řídicí část diagnostické cesty je tvořena posloupností jim příslušejících režimů činnosti. Přenos diagnostických dat danou diagnostickou cestou je pak zajištěn postupnou aktivací režimů tvořících řídicí část této cesty. Za předpokladu, že v době aktivace prvního režimu jsou diagnostická data na začátku této cesty, pak aktivace posledního režimu zajistí, že diagnostická data budou k dispozici na konci této cesty.

Příklad 6:

Příklad datových a řídicích částí vybraných diagnostických cest obvodu z obrázku 22 (na vývod x modulu M se budeme odkazovat zápisem $M.x$, obvod nechť je pojmenován ALU a úsek cesty z vývodu x na vývod y zápisem $x - y$):

Datová část cesty A je posloupnost: $ALU.a - MUL4.a, MUL4.a - MUL4.y, MUL4.y - MUX2.a, MUX2.a - MUX2.y, MUX2.y - REG3.d, REG3.d - REG3.y, REG3.y - MODX.a$, řídicí část cesty A je posloupnost: $\emptyset, \{(MUL4.b, 01)\}, \emptyset, \{(MUX2.sel0, 0), (MUX2.sel1, 0)\}, \emptyset, \{(REG.clk, 1)\}, \emptyset$,

datová část cesty C je posloupnost: $ALU.init - MUX2.d, MUX2.d - MUX2.y, MUX2.y - REG3.d, REG3.d - REG3.y, REG3.y - MODX.a$, řídicí část cesty C je posloupnost: $\emptyset, \{(MUX2.sel0, 1), (MUX2.sel1, 1)\}, \emptyset, \{(REG.clk, 1)\}, \emptyset$,

datová část cesty D je jednočlenná posloupnost: $ALU.c - MODX.b$, řídicí část cesty D je jednočlenná posloupnost: \emptyset .

□

Je možné konstatovat, že obecně je pro daný modul několik diagnostických cest a jejich dílčí části lze za účelem efektivnějšího přenosu diagnostických dat vzájemně kombinovat. Proto je nutno vybrat takovou kombinaci, která povede k co nejefektivnějšímu přenosu co největšího objemu diagnostických dat daného modulu. Není-li možno přenést všechna diagnostická data daného modulu, pak je nezbytné zvážit možná řešení tohoto problému, jejich dopad a cenu. Kromě toho je při volbě konkrétní kombinace (dílčích částí) diagnostických cest pro daný modul nutné analyzovat, jak tyto cesty ovlivní existenci či vlastnosti diagnostických cest jiných modulů a zda a jak lze tyto cesty využít pro testování dalších modulů. Cílem je nalézt takový plán aktivací

³⁷ přesněji daný režim je prázdná množina; v tomto případě modul provádí z hlediska jeho vstupně-výstupního chování pouze jedinou činnost, která je neměnná a nezávisí na okolních podnětech

transparentních režimů činnosti obvodových modulů, aby jím bylo dosaženo co nejeфекtivnějšího přenosu diagnostických dat pro všechny testované moduly. Tato analýza je jednou z nejnáročnějších částí hierarchického generování testu, je obvykle součástí poslední etapy hierarchického generování testu (tj. generování globálního testu) a její výsledky mají podstatný vliv na kvalitu výsledného testu.

3.5 Shrnutí

Hlavním cílem této kapitoly bylo představit témata úzce související s problematikou řešenou v rámci této práce za účelem usnadnění jejího zařazení do kontextu témat z oblastí diagnostiky číslicových obvodů. První podkapitola se blíže věnovala návrhu pro snadnou testovatelnost s využitím techniky scan; byly v ní představeny vybrané varianty tzv. scan buněk a základní typy scan technik - zejména technika částečný sériový scan. Technice scan byl věnován dodatečný prostor ne proto, že se jedná o techniku poměrně populární, ale zejména proto, že pro zlepšení testovatelnosti daného návrhu bylo na základě výsledků navržené metody pro analýzu testovatelnosti využito právě této techniky. Proto bylo důležité neodbyť tuto techniku povrchní zmínkou, ale věnovat se jí o něco blíže, protože při její aplikaci v souvislosti s touto prací byl podstatný nejen výběr vhodné množiny scan registrů, ale také volba vhodného rozmístění těchto registrů do scan řetězců.

Demonstraci náročnosti výběru vhodné množiny scan registrů analýzou obvodové struktury se věnovala druhá podkapitola, která představila tento problém z matematického pohledu. Metody zabývající se řešením tohoto problému využívají skutečnosti [SH01], že složitost generování testu pro sekvenční obvod závisí exponenciálně na délkách cyklů v obvodu a lineárně na sekvenční hloubce obvodu; metody pak hledají co nejlepší způsob, jak rozbít zpětné vazby tak, aby cena za toto rozbítí byla co nejmenší a současně, aby jím bylo dosaženo co nejlepších diagnostických vlastností obvodu při splnění dalších návrhových omezení. Nevýhodou těchto metod však je, že pro obvody popsané na vyšších úrovních popisu bývá (s ohledem na jednoduchost modelu) tento problém řešen na zjednodušeném grafovém modelu datových cest daného obvodu a nebývá přitom brána v úvahu funkce, kterou obvod resp. jeho prvky plní, i když to je důležitým faktorem pro co nejpřesnější zjištění testovatelnosti obvodu. Abstrakce od některých vlastností obvodových prvků pak může vést např. k redundantnímu výběru scan registrů. Tento fakt vede např. k tomu, že vznikají metody obohacující klasické MFSP přístupy o informace získané pomocí generátoru testu či o vhodné modely chování obvodových prvků tvořících strukturu obvodu - u takových metod je pak přístup typu "všechny cykly jsou si rovné" je nahrazen přístupem typu "pokryj jen ty cykly, u nichž je to nutné".

V další podkapitole byl představen přehled existujících metod pro analýzu testovatelnosti, počínaje metodami pracujícími na úrovni hradel, přes metody

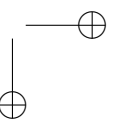
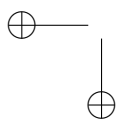
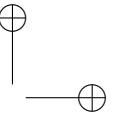
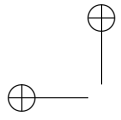
pracující na úrovni meziregistrových přenosů a konče u metod pracujících na vyšších úrovních popisu. Hlavní snahou bylo ukázat, že v současné době neexistuje přesná definice testovatelnosti a že obecně bývá testovatelnost chápána jako charakteristika zohledňující různé náklady spojené s testováním číslicového obvodu, a to zejména jako ukazatel efektivity tvorby a aplikace testu. Jelikož jednotná definice testovatelnosti v současnosti neexistuje, liší se dosavadní přístupy k její analýze jak svými cíli, tak úrovněmi abstrakce popisu obvodu, na nichž jsou použitelné. Nicméně i přes odlišnost existujících metod lze použité přístupy charakterizovat několika společnými znaky jako např. použití jistého modelu číslicového systému, snaha o nalezení obtížně testovatelných částí v rámci daného systému na základě měř řiditelnosti a pozorovatelnosti či snaha poskytnout rychlý, avšak co nejméně zkreslený odhad testovatelnosti daného číslicového systému. Jak je z této podkapitoly patrné, metody analýzy testovatelnosti na úrovni meziregistrových přenosů lze rozdělit do dvou skupin. Metody z první skupiny jsou založeny na jistém pravděpodobnostním modelu chování obvodových prvků a obvykle slouží jako pomůcka při efektivní implementaci BIST technik. Druhá, podstatně menší skupina, vychází z jistého (nepravděpodobnostního) modelu chování a vybraných vlastností obvodových prvků a struktury obvodu a jejím cílem je analýza struktury obvodových datových cest. Nevýhodou metod patřících do první skupiny je otázka využití jejich výsledků pro účely generování deterministického testu, nevýhodou metod patřících do druhé skupiny je buď jejich spjatost s konkrétní DFT technikou (obvykle scan technikou) či jejich velká výpočetní složitost. Jednou z výchozích motivací této práce bylo přispět k odstranění nevýhod metod druhé skupiny, a to tím, že v rámci této práce bude navržena metoda založená na obecnějším modelu datových cest, předpokládající hierarchické generování testu³⁸, jejíž princip ani výsledky nebudou spjaty s konkrétní DFT technikou a která bude pracovat s časovou složitostí blízkou lineární.

Poslední podkapitola je kromě základního přehledu motivací, základních pojmů a principů z oblasti tzv. hierarchického testu věnována zejména pohledu na tzv. transparentní režimy a cesty tak, jak jsou chápány v následujících kapitolách. Za nejvýznamnější část této podkapitoly lze, v souvislosti s tématem řešeným v této práci, považovat část týkající se transparentnosti modulů. V ní je ukázáno, že pro přenos diagnostických dat obvodovým prvkem není třeba u tohoto prvku vyžadovat existenci režimu zajišťujícího bijekci pouze mezi daty portů téže šířky, ale že je vhodné hledat i jiné způsoby přenosu diagnostických dat strukturou obvodového prvku. Bylo ukázáno, že je výhodné rozlišovat režimy pro přenos testovacích vzorků od režimů pro přenos odezev a neomezovat přenos diagnostických dat pouze na přenos mezi porty stejných

³⁸ např. [Uba04] ukazuje, že řiditelnost a pozorovatelnost nemohou být chápány jako absolutní míry pro ohodnocení testovatelnosti obvodu použitelné bez ohledu na plánovanou metodu generování testu, jelikož jsou vždy velmi úzce spjaty s jistými principy generování testu

3.5 Shrnutí 73

šířek, ale naopak umožnit přenosy mezi porty různých šířek a mezi několika porty současně. Popisu těchto skutečností však muselo nutně předcházet rozšíření výchozího modelu [Růž02], o němž pojednává následující kapitola.



4

Model obvodu popsaného na úrovni meziregistrových přenosů

Aby bylo možné přesně, přehledně a nesporně popsat skutečnosti, jimiž jsem se během svého výzkumu zabýval, rozhodl jsem se pro tento účel vycházet z jednoho z již existujících modelů struktur číslicového obvodu. Tento model byl postupně budován a publikován s cílem jeho použití v oblasti diagnostiky číslicových obvodů na úrovni meziregistrových přenosů; jeho konečnou verzi, včetně příkladů jeho využitelnosti lze nalézt v [Růž02]. Přestože lze tento model použít i pro modelování rozsáhlejších obvodových celků, popř. celých systémů, budeme v následujícím textu v souvislosti s tímto modelem pro zjednodušení hovořit pouze o číslicovém obvodu.

Protože v modelu [Růž02] nejsou zabudovány prostředky pro modelování některých skutečností souvisejících s touto prací (např. bitových složek portů a spojů, obecnějších režimů a cest pro přenos diagnostických dat apod.), bylo třeba do mnoha původních definic zasáhnout a modifikovat je, další definice bylo nutno doplnit [Str02b, JS02a, Str03b]. Vzhledem k rozsahu modifikací proto spíše než o převzetí modelu [Růž02] můžeme hovořit o jeho rozšíření či chceme-li přizpůsobení za účelem schopnosti modelovat další diagnostické vlastnosti číslicového obvodu. Poznamenejme, že jak výchozí model [Růž02] tak model prezentovaný v této práci předpokládá, že pro směrování datového toku je zvolena propojovací strategie multiplexovaných datových cest [Mär92]. Rozšířením modelu o další provozovací strategie - např. o strategii obousměrných sběrnic - se tato práce nezabývá. Mohlo by však být předmětem výzkumu navazujícího na tuto práci.

Rozšíření modelu [Růž02] bude zaveden v následující podkapitole. Aby bylo odlišeno, které definice jsou původní, které modifikované a které jsou zcela nově zavedeny, jsou čísla definic doplněna symbolem *, jde-li o definici původní (např. **Definice 1***), symbolem **, jde-li o definici modifikovanou (např. **Definice 1****) a žádným symbolem, jde-li o definici novou (např. **Definice 1**) vzhledem k modelu [Růž02].

4.1 Model struktury číslicového obvodu

Tato podkapitola je tvořena dvěma oddíly - první se zabývá modelováním rozhraní obvodových prvků a rozhraní obvodu, druhý modelováním spojů, tj. vzájemného propojení rozhraní prvků. Úroveň, na které se při popisu modelu struktury obvodu budeme pohybovat, lze - zejména vzhledem k příkladům uváděným v této práci - považovat za úroveň meziregistrových přenosů. Poznamenejme však, že pomocí níže uvedených definic lze popsat strukturu obvodu i na jiné úrovni abstrakce - např. na obecnější úrovni funkčních bloků či systémové úrovni. Nebude-li uvedeno jinak, budeme v následujícím textu předpokládat pouze datové cesty obvodu popsaného na úrovni meziregistrových přenosů a používajícího pro směrování datového toku strategii multiplexovaných datových cest. Nebude-li uvedeno jinak, pak se v následujícím textu bude předpokládat, že každá množina je konečná.

4.1.1 Model rozhraní prvků

Číslicový obvod popsaný na úrovni meziregistrových přenosů (tak jak je chápán v práci [Růž02]) je číslicový obvod, jehož struktura je tvořena propojením rozhraní tří typů prvků:

- **multiplexorů** směřujících tok dat obvodem,
- **funkčních jednotek** provádějících transformaci dat,
- **registrů** sloužících k uchování mezivýsledků operací (vnitřního stavu obvodu) a synchronizaci datového toku.

Jelikož konkrétní obvod je modelován především za účelem jeho následné analýzy, budeme se na obvod často odkazovat jako na *analyzovaný obvod*; pro tentýž účel může být také použito zkratky *CUA*¹. Budeme-li hovořit o konkrétním obvodu pojmenovaným X , pak namísto zkratky *CUA* obvykle použijeme označení X .

Definice 1:*

Bud' $E_{CUA} = MUX_{CUA} \cup FU_{CUA} \cup REG_{CUA}$, kde

- MUX_{CUA} je množina multiplexorů v *CUA*
- FU_{CUA} je množina funkčních jednotek v *CUA* a
- REG_{CUA} je množina registrů v *CUA*,

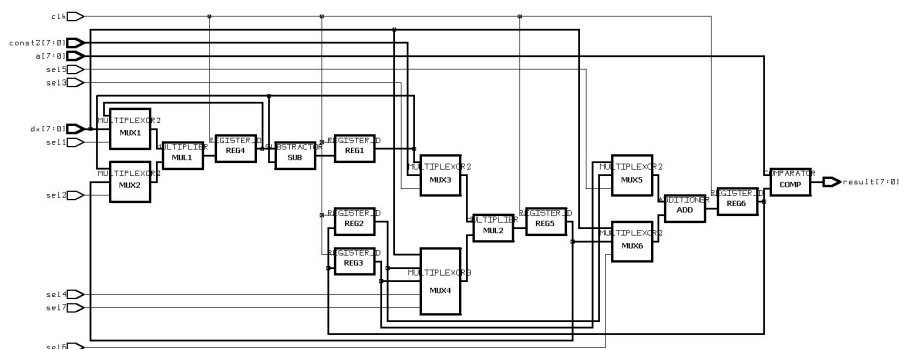
množina obvodových prvků *CUA*. Prvek $x \in E_{CUA}$ budeme nazývat *obvodovým prvkem*.

□

Z důvodu snazšího popisu některých skutečností souvisejících zejména s konstrukcí diagnostických cest a analýzou testovatelnosti byla množina obvodových prvků definovaná v modelu [Růž02] rozšířena o speciální prvek

¹ z angl. circuit under analysis

4.1 Model struktury číslicového obvodu 77



Obrázek 23. Příklad jednoduchého RTL obvodu (Diffeq) - schéma bylo vygenerováno nástrojem Synopsys Design Analyzer na základě popisu chování tohoto obvodu v jazyce VHDL

$circ_{CUA}$ označující CUA. Existence tohoto prvku bude nezbytná k modelování vybraných skutečností² pro CUA. Pomocí tohoto prvku bude možno obvodu CUA přiřadit množiny jeho vstupů a výstupů, množinu jeho vnitřních obvodových prvků, hodnotu jeho říditelnosti, pozorovatelnosti a testovatelnosti apod.

Definice 2:

Bud' $E'_{CUA} = E_{CUA} \cup \{circ_{CUA}\}$, kde $circ_{CUA}$ je speciální prvek označující CUA, rozšířená množina obvodových prvků CUA.

□

Příklad 7:

Necht' je dán číslicový obvod $Diffeq$ se strukturou na obrázku 23.

Pak $MUX_{Diffeq} = \{MUX1_{Diffeq}, MUX2_{Diffeq}, MUX3_{Diffeq}, MUX4_{Diffeq}, MUX5_{Diffeq}, MUX6_{Diffeq}\}$,

$FU_{Diffeq} = \{SUB_{Diffeq}, ADD_{Diffeq}, MUL1_{Diffeq}, MUL2_{Diffeq}, COMP_{Diffeq}\}$,

$REG_{Diffeq} = \{REG1_{Diffeq}, REG2_{Diffeq}, REG3_{Diffeq}, REG4_{Diffeq}, REG5_{Diffeq}, REG6_{Diffeq}\}$,

$E'_{Diffeq} = MUX_{Diffeq} \cup FU_{Diffeq} \cup REG_{Diffeq} \cup \{cir_{Diffeq}\}$.

□

Pokud nedojde k nejednoznačnostem, nebudeme z důvodu přehlednosti a snadnosti zápisu u obvodových prvků (či u jiných symbolů - např. množin)

² dle [Růž02] příslušejících pouze obvodovým prvkům. Patří sem např. rozhraní a diagnostické vlastnosti

78 4 Model obvodu popsaného na úrovni meziregistrových přenosů

uvádět informaci o obvodu, do jehož struktury tyto prvky patří - namísto zápisu SUB_{Diffeq} pak budeme psát pouze SUB atp.

Kromě schopnosti rozlišit od sebe jednotlivé obvodové prvky je třeba, aby měl model prostředky pro modelování rozhraní (tj. vstupů a výstupů) těchto prvků a obvodu. Rozhraní je tvořeno několika tzv. porty, z nichž každý je obecně vícebitové šířky a plní funkci vstupu resp. výstupu daného obvodového prvku nebo obvodu. Právě modelováním těchto skutečností se zabývají následující definice.

Definice 3:

Buď $BIT_{CUA} = \{x \mid x \text{ je metalický vývod}^3 \text{ některého z vnitřních nízkoúrovňových prvků}^4 \text{ v obvodu nebo metalický vývod z rozhraní obvodu}\}$ množina bran v CUA.

□

Množinu BIT_{CUA} lze v první řadě chápat jako množinu všech vývodů nízkoúrovňových prvků v CUA, z nichž však pouze některé jsou vybrány (viz např. příklad 8 a obrázek 24), aby plnily funkci rozhraní jednotlivých prvků; kromě toho, tato množina obsahuje jednobitové vývody tvořící rozhraní obvodu. Způsobem, jakým jsou jednotlivé prvky BIT_{CUA} v rámci konkrétního obvodového prvku vzájemně propojeny (a tvoří tak strukturu a funkci daného obvodového prvku) se navržený model nezabývá, jelikož to z hlediska zvoleného přístupu k řešení tématu této práce není podstatné.

Dále nepředpokládáme, že máme některou z nízkoúrovňových variant daného obvodu k dispozici a chápeme množinu BIT_{CUA} jako jedinou (nízkoúrovňovou) informaci o jednobitových nosičích, z nichž bude později tvořeno rozhraní konkrétních obvodových prvků a obvodu.

Příklad 8:

Nechť je dán prvek MUX1 z obrázku 24 a nechť $MUX1 \in E'_{CUA}$.

Pak $\{a_0, \dots, a_3, b_0, \dots, b_3, q_0, \dots, q_3, sel_0, c_0, \dots, c_{24}\} \subset BIT_{CUA}$.

□

Aby bylo možné modelovat výskyt dat (hodnoty) na bráně $x \in BIT_{CUA}$, je nutno definovat, jaké hodnoty je možné na bráně x očekávat.

Definice 4:

Buď $VAL = \{0, 1, \uparrow, \downarrow\}$, kde

- 0 reprezentuje hodnotu "logická nula",

³ předpokládáme, že na daném vývodu se může vyskytnout buď jedna z hodnot 0, 1 nebo jedna z hodnot \uparrow, \downarrow ; proto označujeme vývod resp. bránu jako jednobitovou, přestože jsou modelovány čtyři hodnoty (viz. definice 4)

⁴ např. hradla - avšak nízkoúrovňový prvek nijak blíže nespecifikujeme a mlčky se tak odkazujeme např. na existenci nějaké knihovny takových prvků

- 1 reprezentuje hodnotu "logická jednička",
- \uparrow reprezentuje nástupnou hranu,
- \downarrow reprezentuje sestupnou hranu

množina hodnot, které se mohou vyskytnout na bráně $x \in BIT_{CUA}$, přičemž budeme předpokládat, že na bráně x se může vyskytnout buď jedna z hodnot 0, 1 nebo jedna z hodnot \uparrow , \downarrow . Každou modelovanou bránu tedy chápeme jako bránu řízenou úrovní (hladinou) nebo jako bránu řízenou hranou.

Každou úrovní řízenou bránu vyjma bran tvořících adresové (výběrové) vstupy multiplexorů, a bran jakými jsou např. povolovací vstupy paměťových prvků, hradlovací vstupy či vstupy pro nulování stavu čítačů plnicích obdobnou řídicí funkci budeme nazývat *datovou branou*.

Všechny ostatní brány - tj. každou bránu, která není datová a každou bránu řízenou hranou - budeme nazývat *řídicími branami*

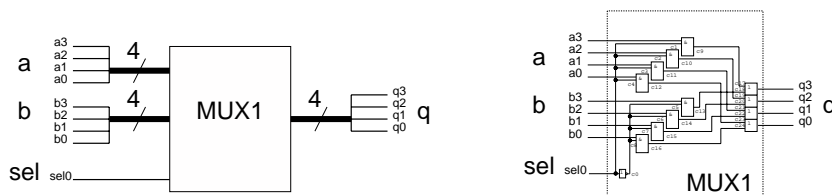
□

Zobrazení definovaná v následující definici umožní přiřadit bráně $x \in BIT_{CUA}$ jednak prvek $y \in E'_{CUA}$, jehož součástí tato brána je a jednak jedno-bitovou hodnotu, která se na bráně x vyskytuje. Jisté brány příslušející témuž prvku y budou později využity ke konstrukci a modelování rozhraní prvku y .

Definice 5:

Nechť $\beta_E : BIT_{CUA} \rightarrow E'_{CUA}$ je zobrazení přiřazující bráně $x \in BIT_{CUA}$ prvek $y \in E'_{CUA}$, jemuž tato brána náleží a nechť existuje zobrazení $\beta_V : BIT_{CUA} \rightarrow VAL$, kdy $\beta_V(x)=w$ znamená, že na bráně $x \in BIT_{CUA}$ se vyskytuje hodnota $w \in VAL$.

□



Obrázek 24. Obvodový prvek MUX1 - vlevo schematická značka, vpravo struktura prvku

Příklad 9:

Nechť je dán prvek MUX1 z obrázku 24, nechť $MUX1 \in E'_{CUA}$ a nechť $X = \{a_0, \dots, a_3, b_0, \dots, b_3, q_0, \dots, q_3, sel_0, c_0, \dots, c_{24}\} \subset BIT_{CUA}$.

Pak $\forall x \in X: \beta_E(x) = MUX1$.

□

80 4 Model obvodu popsaného na úrovni meziregistrových přenosů

Podle toho, jakého typu jsou brány a jaký účel plní jednotlivé brány v rozhraní konkrétního prvku, můžeme prvky z množiny BIT_{CUA} začlenit do množin uvedených v následující definici. Příslušnost brány do dané množiny je pak určující pro její typ.

Definice 6:

Buďte

- $BI_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je vstupní datová brána příslušející prvku } y = \beta_E(x), y \in E_{CUA}\}$ množina vstupních datových bran obvodových prvků CUA,
- $BCI_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je vstupní řídicí brána příslušející prvku } y = \beta_E(x), y \in E_{CUA}\}$ množina vstupních řídicích bran obvodových prvků CUA,
- $BO_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je výstupní brána příslušející prvku } y = \beta_E(x), y \in E_{CUA}\}$ množina výstupních bran obvodových prvků CUA,
- $BPI_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je vstupní datová brána příslušející prvku } y = \beta_E(x), y = cir_{CUA}\}$ množina vstupních datových bran obvodu CUA,
- $BPCI_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je vstupní řídicí brána příslušející prvku } y = \beta_E(x), y = cir_{CUA}\}$ množina vstupních řídicích bran obvodu CUA,
- $BPO_{CUA} = \{x \mid x \in BIT_{CUA} \wedge x \text{ je výstupní brána příslušející prvku } y = \beta_E(x), y = cir_{CUA}\}$ množina výstupních bran obvodu CUA,
- $B_{CUA} = BI_{CUA} \cup BCI_{CUA} \cup BO_{CUA}$,
- $BP_{CUA} = BPI_{CUA} \cup BPCI_{CUA} \cup BPO_{CUA}$

množiny bran CUA.

Pro bránu obvodu CUA bude také často použito označení *primární brána* obvodu CUA, přesněji pak *primární vstupní brána* resp. *primární výstupní brána*. Obdobně brána, která není primární, bude označována za *vnitřní bránu* obvodu CUA.

Množina BI_{CUA} , BCI_{CUA} , BPI_{CUA} , $BPCI_{CUA}$, BO_{CUA} resp. BPO_{CUA} , do níž brána p náleží, určuje *typ brány*. Nepředpokládáme-li v modelu výskyt vstupně-výstupních bran, pak každá brána z množiny $B_{CUA} \cup BP_{CUA}$ náleží právě do jedné z množin BI_{CUA} , BCI_{CUA} , BPI_{CUA} , $BPCI_{CUA}$, BO_{CUA} , BPO_{CUA} .

□

Úmluva 1:

Nebude-li uvedeno jinak, budeme v dalším textu uvažovat pouze takové CUA, pro které platí:

- $|E_{CUA}| \geq 1$
- $|E'_{CUA}| \geq 2$
- $|BI_{CUA}| \geq 1$
- $|BO_{CUA}| \geq 1$

- $|BPI_{CUA}| \geq 1$
- $|BPO_{CUA}| \geq 1$

Takové CUA provádějí přenos resp. transformaci dat ve směru ze svých primárních vstupů na své primární výstupy, a proto je v jejich struktuře přítomen alespoň jeden obvodový prvek provádějící tuto činnost. Pro tyto účely musí mít CUA k dispozici alespoň jednu primární vstupní datovou bránu, alespoň jednu primární výstupní datovou bránu a alespoň jeden obvodový prvek ve struktuře CUA musí být vybaven alespoň jednou vstupní a alespoň jednou výstupní datovou branou.

□

Příklad 10:

Nechť je dán prvek MUX1 z obrázku 24 (strana 77), nechť $MUX1 \in E'_{CUA}$ a nechť $\forall x \in \{a_0, \dots, a_3, b_0, \dots, b_3, q_0, \dots, q_3, sel_0, c_0, \dots, c_{24}\} \subset BIT_{CUA}$: $\beta_E(x) = MUX1$.

$$\begin{aligned} \text{Pak } \{a_0, \dots, a_3, b_0, \dots, b_3\} &\subseteq BI_{CUA}, \\ \{sel_0\} &\subseteq BCI_{CUA}, \\ \{q_0, \dots, q_3\} &\subseteq BO_{CUA}. \end{aligned}$$

□

Nyní máme k dispozici prostředky pro modelování bran. V následujících definicích budou zavedeny prostředky, které budou v dalším textu použity pro modelování rozhraní obvodu a obvodových prvků, tzv. portů (viz níže, definice 8). V této chvíli postačí, bude-li port prvku $x \in E'_{CUA}$ chápán jako vývod tohoto prvku, tj. jako vstup resp. výstup patřící do rozhraní prvku x . Nejprve definujeme množinu $ORBIT_{CUA}$, která obsahuje m -tice bran. Některé m -tic z této množiny budou později vybrány, aby tvořily vícebitové rozhraní - tzv. porty - obvodových prvků a obvodu.

Definice 7:

Bud' $ORBIT_{CUA} = \{p \mid p = (p_{m-1}, \dots, p_0) \in BI_{CUA}^+ \cup BCI_{CUA}^+ \cup BO_{CUA}^+ \cup BPI_{CUA}^+ \cup BPCI_{CUA}^+ \cup BPO_{CUA}^+ \text{ a } \forall p_i, p_j, i \neq j: p_i \neq p_j \wedge \beta_E(p_i) = \beta_E(p_j)\}$, kde $i, j, m \in \mathbb{N}, m \geq 1; 0 \leq i, j \leq (m-1)$.

Bud' $\pi_{BIT}: ORBIT_{CUA} \times \mathbb{N} \rightarrow BIT_{CUA}$ zobrazení definované předpisem $\pi_{BIT}((p_{m-1}, \dots, p_0), i) = p_i; i, m \in \mathbb{N}, m \geq 1, 0 \leq i \leq (m-1)$.

Bud' $\pi_{LEX}: ORBIT_{CUA} \rightarrow ORBIT_{CUA}$ zobrazení, přiřazující prvku $x \in ORBIT_{CUA}$ prvek $x' \in ORBIT_{CUA}$ takový, že $\pi_{BITS}(x') = \pi_{BITS}(x)$ a brány v prvku x' jsou lexikograficky uspořádány ve směru zleva doprava.

□

Každý prvek $p \in ORBIT_{CUA}$ tedy v daném pořadí sdružuje m bran do jednoho celku, přičemž m je u dvou různých celků obecně různé. Jelikož situace, kdy port prvku $p \in E'_{CUA}$ obsahuje danou bránu vícekrát je nereálná, protože vede pouze ke složitějšímu rozhraní prvku $x = \beta_E(\pi_{BIT}(p, 0)) \in E'_{CUA}$, jsou z množiny $ORBIT_{CUA}$ vyloučeny jednak ty celky, které obsahují duplicitní

82 4 Model obvodu popsaného na úrovni meziregistrových přenosů

brány (jejich vyloučení je zajištěno podmínkou $p_i \neq p_j$ v definici 7) a jednak ty celky, které obsahují brány příslušející více než jednomu prvku (jejich vyloučení je zajištěno podmínkou $\beta_E(p_i) = \beta_E(p_j)$ v definici 7). Jisté celky z množiny $ORBIT_{CUA}$ pak mohou být vybrány, aby reprezentovaly port prvku x (viz níže, definice 8).

Podstatné také je, že v celku sdružené brány jsou téhož typu z hlediska plnění funkce datového resp. řídicího vstupu či výstupu prvku x .

K dílčím branám p_{m-1}, \dots, p_0 (prvku p množiny $ORBIT_{CUA}$) lze přistupovat pomocí zobrazení π_{BIT} tak, že zápisem $\pi_{BIT}(p, i)$ bude zpřístupněna brána p_i prvku p .

Příklad 11:

Nechť je dán prvek MUX1 z obrázku 24 (strana 77), nechť $MUX1 \in E'_{CUA}$, nechť $\forall x \in \{a_0, \dots, a_3, b_0, \dots, b_3, q_0, \dots, q_3, sel_0, c_0, \dots, c_{24}\} \subset BIT_{CUA}: \beta_E(x) = MUX1$ a nechť $A = \{a_0, \dots, a_3, b_0, \dots, b_3\} \subseteq BIC_{UA}$, $B = \{sel_0\} \subseteq BCIC_{UA}$, $C = \{q_0, \dots, q_3\} \subseteq BOC_{UA}$.

Pak $A^+ \cup B^+ \cup C^+ \subseteq ORBIT_{CUA}$.

□

Věta 1:

Nechť platí úmluva 1 ze strany 78. Pro každý $p = (p_{m-1}, \dots, p_0) \in ORBIT_{CUA}$ platí, že brány p_{m-1}, \dots, p_0 , které sdružuje, jsou vzájemně odlišné, jsou téhož typu⁵ a přísluší témuž prvku z E'_{CUA} .

□

Důkaz:

Předpokládejme, že existuje $p \in ORBIT_{CUA}$ takový, že sdružuje brány odlišných typů. Pro takový p by platilo $p \in X = (BIC_{UA} \cup BCIC_{UA} \cup BOC_{UA} \cup BPI_{CUA} \cup BPCI_{CUA} \cup BPO_{CUA})^+$. Podle definice 7 platí $p \in Y = (BI_{CUA}^+ \cup BCI_{CUA}^+ \cup BO_{CUA}^+ \cup BPI_{CUA}^+ \cup BPCI_{CUA}^+ \cup BPO_{CUA}^+)$. Pro každý prvek $p \in X$ by tedy současně muselo platit i $p \in Y$, což vzhledem k $Y \subset X$ nelze a vzniká tím spor. Vstupní předpoklad je tedy chybný.

Vzájemná odlišnost bran sdružených v p resp. příslušnost bran sdružených v p k témuž prvku z E'_{CUA} plyne z podmínky $\forall p_i, p_j, i \neq j: p_i \neq p_j \wedge \beta_E(p_i) = \beta_E(p_j)$, důkaz je tedy zřejmý.

□

⁵ podle definice 6, strana 78

*Definice 8**:*

Nechť $\psi' : E'_{CUA} \rightarrow 2^{ORBIT_{CUA}}$ je zobrazení přiřazující každému prvku $x \in E'_{CUA}$ jeho rozhraní tak, že je splněna podmínka $\forall p \in \psi'(x) : \beta_E(\pi_{BIT}(p, 0)) = x$. Uspořádanou m -tici $p = (p_{m-1}, \dots, p_0) \in \psi'(x)$ budeme nazývat m -bitový port prvku x , číslo m budeme nazývat (bitová) šířka portu p a bránu $p_i = \pi_{BIT}(p, i)$ budeme nazývat i . bit portu p .

Port p nazýváme

- vstupní (datový resp. řídicí), platí-li $\pi_{BIT}(p, 0) \in BI_{CUA} \cup BPI_{CUA} \cup BCI_{CUA} \cup BPCI_{CUA}$ ($\pi_{BIT}(p, 0) \in BI_{CUA} \cup BPI_{CUA}$ resp. $\pi_{BIT}(p, 0) \in BCI_{CUA} \cup BPCI_{CUA}$),
- výstupní, platí-li $\pi_{BIT}(p, 0) \in BO_{CUA} \cup BPO_{CUA}$.

Platí-li navíc $p \in \psi'(cir_{CUA})$, nazýváme navíc port p primárním portem (obvodu CUA). Obdobně port, který není primární, budeme nazývat portem obvodového prvku $x = \beta_E(\pi_{BIT}(p, 0))$ (obvodu CUA), popř. obecně vnitřním portem obvodu CUA.

Nedojde-li tím k nedorozumění, budeme někdy pro zřehlednění m -bitový port p prvku x zkráceně značit

- $x.p \setminus m$ nebo
- $x.p$, je-li m vztahující se k portu p známo či je v daném kontextu nepodstatné nebo pouze
- p hovoříme-li o obecném portu p nebo je-li informace o jeho prvku a šířce známa

Bránu $\pi_{BIT}(p, 0)$ m -bitového portu p budeme nazývat *nejnižším bitem portu p* nebo *nejméně významným bitem portu p* , bránu $\pi_{BIT}(p, m-1)$ budeme nazývat *nejvyšším bitem portu p* nebo *nejvíce významným bitem portu p* .

□

Příklad 12:

Nechť je dán prvek MUX1 z obrázku 24 (strana 77) a necht' $MUX1 \in E'_{CUA}$. Definujme rozhraní prvku MUX1 tak, že $\psi'(MUX1) = \{(a_3, a_2, a_1, a_0), (b_3, b_2, b_1, b_0), (sel_0), (q_3, q_2, q_1, q_0)\}$.

Pak porty $a = (a_3, a_2, a_1, a_0)$, $b = (b_3, b_2, b_1, b_0)$ jsou vstupní 4-bitové datové porty prvku MUX1, port $sel = (sel_0)$ je vstupní 1-bitový řídicí port prvku MUX1 a port $q = (q_3, q_2, q_1, q_0)$ je výstupní 4-bitový datový port prvku MUX1.

Použijeme-li zkráceného zápisu podle definice 8, budeme tyto porty prvku MUX1 značit

$MUX1.a \setminus 4$, $MUX1.b \setminus 4$, $MUX1.sel \setminus 1$, $MUX1.q \setminus 4$, resp.

$MUX1.a$, $MUX1.b$, $MUX1.sel$, $MUX1.q$, resp.

$a \setminus 4$, $b \setminus 4$, $sel \setminus 1$, $q \setminus 4$, resp.

a , b , sel , q .

□

84 4 Model obvodu popsaného na úrovni meziregistrových přenosů

Úmluva 2:

Pokud nebude uvedeno jinak, pak budeme-li v této práci reprezentovat obvod graficky, pak jej budeme vždy zakreslovat tak, že na levé straně schématu každého prvku $x \in E'_{CUA}$ budou umístěny jeho vstupní porty a na pravé straně schématu prvku budou umístěny jeho výstupní porty.

Nebude-li uvedeno jinak, tak brány jednotlivých portů budou zakreslovány tak, že brána s větším indexem bude umístěna nad branou s menším indexem. □

*Definice 9**:*

Bud'te

- $IN_{CUA} = \{x \mid x \text{ je vnitřní vstupní datový port obvodu CUA}\}$,
- $CIN_{CUA} = \{x \mid x \text{ je vnitřní vstupní řídicí port obvodu CUA}\}$,
- $OUT_{CUA} = \{x \mid x \text{ je vnitřní výstupní port obvodu CUA}\}$,
- $PIN_{CUA} = \{x \mid x \text{ je primární vstupní datový port obvodu CUA}\}$,
- $PCIN_{CUA} = \{x \mid x \text{ je primární vstupní řídicí port obvodu CUA}\}$,
- $POUT_{CUA} = \{x \mid x \text{ je primární výstupní port obvodu CUA}\}$,
- $P_{CUA} = IN_{CUA} \cup CIN_{CUA} \cup OUT_{CUA}$,
- $PORT_{CUA} = P_{CUA} \cup PIN_{CUA} \cup PCIN_{CUA} \cup POUT_{CUA}$,

množiny portů obvodu CUA.

V souvislosti s příslušností portu p do množiny IN_{CUA} , CIN_{CUA} , PIN_{CUA} , $PCIN_{CUA}$, OUT_{CUA} resp. $POUT_{CUA}$ hovoříme o *typu portu*. Nepředpokládáme-li v modelu výskyt vstupně-výstupních portů, pak každý port náleží právě do jedné z množin IN_{CUA} , CIN_{CUA} , PIN_{CUA} , $PCIN_{CUA}$, OUT_{CUA} , $POUT_{CUA}$. □

Věta 2:

Nechť platí úmluva 1 ze strany 78. Pro každý $p = (p_{m-1}, \dots, p_0) \in PORT_{CUA}$ platí, že brány p_{m-1}, \dots, p_0 , které sdružuje, jsou vzájemně odlišné, jsou téhož typu (podle definice 6, strana 78) a přísluší témuž prvku z E'_{CUA} . □

Důkaz:

Jelikož z definic 8 a 9 plyne $PORT_{CUA} \subseteq ORBIT_{CUA}$ (obvykle dokonce $PORT_{CUA} \subset ORBIT_{CUA}$), je důkaz zřejmý z důkazu platnosti věty 1, uvedeném na straně 80. □

Důsledek:

Pokud by se v popisu obvodu měl vyskytnout ”port”⁶, pro který by věta 2 neplatila, může být tento port modelován pouze tehdy, bude-li vhodně modifikován či rozčleněn na takové dílčí porty, pro které věta 2 platí

□

Nevyskytují-li se v CUA vstupně-výstupní porty, pak množiny portů IN_{CUA} , CIN_{CUA} , OUT_{CUA} , PIN_{CUA} , $PCIN_{CUA}$, $POUT_{CUA}$ zavedené v definici 9 jsou třídami rozkladu množiny $PORT_{CUA}$. Množina P_{CUA} je definována pro účely případného použití definic modelu [Růž02] - také ostatní definice jsou záměrně zapsány tak, aby byl výsledný model slučitelný s výchozím modelem [Růž02].

Příklad 13:

Nechť je dán číslicový obvod $Difreq$ se strukturou na obrázku 23, strana 75 a nechť pro rozhraní prvků množiny E'_{Difreq} (prvky množiny E'_{Difreq} viz příklad 7, strana 75) platí

$$\begin{aligned} \psi'(cir_{Difreq}) &= \{cir_{Difreq}.a, cir_{Difreq}.clk, cir_{Difreq}.const2, cir_{Difreq}.dx, \\ &cir_{Difreq}.sel1, \dots, cir_{Difreq}.sel7\}, \\ \psi'(MUX1) &= \{MUX1.a, MUX1.b, MUX1.sel, MUX1.q\}, \\ \psi'(MUX2) &= \{MUX2.a, MUX2.b, MUX2.sel, MUX2.q\}, \\ \psi'(MUX3) &= \{MUX3.a, MUX3.b, MUX3.sel, MUX3.q\}, \\ \psi'(MUX4) &= \{MUX4.a, MUX4.b, MUX4.c, MUX4.sel1, MUX4.sel2, \\ &MUX4.q\}, \\ \psi'(MUX5) &= \{MUX5.a, MUX5.b, MUX5.sel, MUX5.q\}, \\ \psi'(MUX6) &= \{MUX6.a, MUX6.b, MUX6.sel, MUX6.q\}, \\ \psi'(MUL1) &= \{MUL1.a, MUL1.b, MUL1.q\}, \\ \psi'(MUL2) &= \{MUL2.a, MUL2.b, MUL2.q\}, \\ \psi'(ADD) &= \{ADD.a, ADD.b, ADD.q\}, \\ \psi'(SUB) &= \{SUB.a, SUB.b, SUB.q\}, \\ \psi'(COMP) &= \{COMP.a, COMP.b, COMP.q\}, \\ \psi'(REG1) &= \{REG1.d, REG1.clk, REG1.q\}, \\ \psi'(REG2) &= \{REG2.d, REG2.clk, REG2.q\}, \\ \psi'(REG3) &= \{REG3.d, REG3.clk, REG3.q\}, \\ \psi'(REG4) &= \{REG4.d, REG4.clk, REG4.q\}, \\ \psi'(REG5) &= \{REG5.d, REG5.clk, REG5.q\}, \\ \psi'(REG6) &= \{REG6.d, REG6.clk, REG6.q\}, \end{aligned}$$

Pak $IN_{Difreq} = \{MUX1.a, MUX1.b, \dots, MUX6.a, MUX6.b, MUL1.a, MUL1.b, MUL2.a, MUL2.b, ADD.a, ADD.b, SUB.a, SUB.b, COMP.a, COMP.b, REG1.d, \dots, REG6.d\}$,

⁶ záměrně je psáno ”port”, protože pokud neplatí věta 2, pak se nejedná o port podle definice 8; takový ”port” nelze modelovat popisovaným modelem - leda že by jeho popis byl tomuto modelu přizpůsoben

86 4 Model obvodu popsaného na úrovni meziregistrových přenosů

$$\begin{aligned}
 CIN_{Diffeq} &= \{MUX1.sel, MUX2.sel, MUX3.sel, MUX4.sel1, \\
 &MUX4.sel2, MUX5.sel, MUX6.sel, REG1.clk, \dots, REG6.clk\}, \\
 OUT_{Diffeq} &= \{MUX1.q, \dots, MUX6.q, MUL1.q, MUL2.q, ADD.q, \\
 &SUB.q, COMP.q, REG1.q, \dots, REG6.q\}, \\
 PIN_{Diffeq} &= \{cir_{Diffeq}.a, cir_{Diffeq}.const2, cir_{Diffeq}.dx\}, \\
 PCIN_{Diffeq} &= \{cir_{Diffeq}.clk, cir_{Diffeq}.sel1, \dots, cir_{Diffeq}.sel7\}, \\
 POUT_{Diffeq} &= \{cir_{Diffeq}.result\}
 \end{aligned}$$

□

*Definice 10**:*

Bud' $D = VAL \cup \{0, 1\}^+$ množina hodnot, které se mohou vyskytnout na portu obecné šířky. Bud' $D_m = \{0, 1\}^m \cup \{\uparrow, \downarrow\}^m$ množina všech hodnot, které se mohou vyskytnout na m -bitovém portu, $m \geq 1$, $m \in \mathbb{N}$.

□

Věta 3:

Na m -bitovém portu p se může vyskytnout buď hodnota z množiny $\{0, 1\}^m$ nebo hodnota z množiny $\{\uparrow, \downarrow\}^m$.

□

Důkaz:

Podle definice 6, strana 78 je každá brána tvořící rozhraní některého prvku z E'_{CUA} umístěna dle svého typu do množin BI_{CUA} , BCI_{CUA} , BO_{CUA} , BPI_{CUA} , $BPCI_{CUA}$ resp. BPO_{CUA} . Podle předpokladu učiněného v definici 4, strana 76 se na dané bráně může vyskytovat pouze jedna z hodnot 0, 1 nebo jedna z hodnot \uparrow, \downarrow podle toho, zda se jedná o bránu datovou nebo řídicí - tj. v závislosti na typu brány.

Jelikož podle věty 1, strana 80 mj. platí, že brány, které jsou sdruženy v prvku $p \in ORBIT_{CUA}$ jsou téhož typu, tj. (podle definice 6) náležejí právě do jedné z množin BI_{CUA} , BCI_{CUA} , BO_{CUA} , BPI_{CUA} , $BPCI_{CUA}$, BPO_{CUA} , pak se na branách příslušejících prvku p mohou vyskytovat pouze hodnoty z množiny $\{0, 1\}$ nebo hodnoty z množiny $\{\uparrow, \downarrow\}$. Protože z definic 8 a 9 plyne $PORT_{CUA} \subseteq ORBIT_{CUA}$ (obvykle dokonce $PORT_{CUA} \subset ORBIT_{CUA}$), pak toto platí i pro obecný port, speciálně pak také pro m -bitový port.

□

Definice 11:

Bud' zobrazení $\pi_{BITS} : ORBIT_{CUA} \rightarrow 2^{BIT_{CUA}}$ přiřazující portu p množinu jeho bitů definované předpisem $\pi_{BITS}(p) = \{\pi_{BIT}(p, i) \mid i = 0, \dots, m-1\}$ a zobrazení $\pi_W : ORBIT_{CUA} \rightarrow \mathbb{N}$ přiřazující portu p množinu jeho šířky definované předpisem $\pi_W(p) = |\pi_{BITS}(p)|$.

□

Příklad 14:

Nechť je dán výsek *CUA* dle obrázku 25 umístěného na straně 89.

$$\begin{aligned}
 \pi_{BITS}(REG4.d) &= \{REG4.d_7, \dots, REG4.d_0\}, \\
 \pi_{BITS}(REG4.q) &= \{REG4.q_7, \dots, REG4.q_0\}, \\
 \pi_{BITS}(REG4.clk) &= \{REG4.clk_0\}, \\
 \pi_{BITS}(MUX3.a) &= \{MUX3.a_3, \dots, MUX3.a_0\}, \\
 \pi_{BITS}(MUX3.b) &= \{MUX3.b_3, \dots, MUX3.b_0\}, \\
 \pi_{BITS}(MUX3.sel) &= \{MUX3.sel_0\}, \\
 \pi_{BITS}(MUX3.q) &= \{MUX3.q_3, \dots, MUX3.q_0\}, \\
 \pi_{BITS}(SUB1.a) &= \{SUB1.a_3, \dots, SUB1.a_0\}, \\
 \pi_{BITS}(SUB1.b) &= \{SUB1.b_3, \dots, SUB1.b_0\}, \\
 \pi_{BITS}(SUB1.q) &= \{SUB1.q_3, \dots, SUB1.q_0\}, \\
 \pi_W(REG4.d) &= 8, \\
 \pi_W(REG4.clk) &= 1, \\
 \pi_W(REG4.q) &= 8, \\
 \pi_W(MUX3.a) &= 4, \\
 \pi_W(MUX3.b) &= 4, \\
 \pi_W(MUX3.sel) &= 1, \\
 \pi_W(SUB1.a) &= 4, \\
 \pi_W(SUB1.b) &= 4, \\
 \pi_W(SUB1.q) &= 4.
 \end{aligned}$$

□

*Definice 12**:*

Nechť existuje zobrazení $\nu : ORBIT_{CUA} \rightarrow D$ nazvané *ohodnocení portů* a přiřazující portu $p \setminus m$ uspořádanou m -tici jednobitových hodnot (m -bitovou hodnotu), definované předpisem $\nu(p) = (v_{m-1}, \dots, v_0) \Leftrightarrow \forall p_i : \beta_V(p_i) = v_i$, kde $p_i = \pi_{BIT}(p, i)$, $i \in \{0, \dots, m-1\}$.

□

Definice 12 umožňuje modelovat výskyt konkrétních dat na daném portu - např. n -bitové hodnoty na n -bitovém portu, nástupné resp. sestupné hrany na hodinovém portu atp. Pro výskyt dat na daném portu platí věta 3, strana 84.

Příklad 15:

Nechť je dán prvek *MUX1* z obrázku 24 (strana 77) a nechť je jeho rozhraní zvoleno dle příkladu 12, strana 81. Pak zápis $\nu(MUX1.a) = (1, 0, 0, 0)$ nebo např. zkráceně $\nu(a) = (1, 0, 0, 0)$ vyjadřuje situaci, kdy se na portu *MUX1.a* vyskytuje hodnota $(1, 0, 0, 0)$, tj. situaci, kdy $\nu(a_3, a_2, a_1, a_0) = (1, 0, 0, 0)$, což znamená, že pro složky a_3, a_2, a_1, a_0 portu *MUX1.a* současně platí $\beta_V(a_3) = 1$, $\beta_V(a_2) = 0$, $\beta_V(a_1) = 0$ a $\beta_V(a_0) = 0$.

□

Definice 13:

Nechť je dán obvodový prvek $e \in E'_{CUA}$ a množina $X_e = \{x_1, \dots, x_k\} \subseteq \bigcup_{y \in Y} \pi_{BITS}(y)$, kde $Y = \psi'(e) \setminus (OUT_{CUA} \cup POUT_{CUA})$, $k \in \mathbb{N}$. Každé zobrazení $\alpha: X_e \rightarrow D_1$ takové, že $\alpha(x_1) = \beta_V(x_1), \dots, \alpha(x_k) = \beta_V(x_k)$ nazveme *režimem činnosti prvku e* ; pak každý port $p \in \psi'(e)$, pro který $\exists i: \pi_{BIT}(p, i) \in X_e$ (každou bránu z množiny X_e) nazveme *řídícím portem (řídící branou) režimu činnosti α prvku e* .

Režim α nazveme⁷ *momentálně postačujícím pro dané chování prvku e* (resp. že režim *momentálně postačuje pro dané chování prvku e*), pokud $\neg \exists z \in Def(\alpha)$ takové, že totéž chování prvku e lze také zajistit nastavením režimem $\alpha \setminus (\{z\} \times D_1)$ prvku e .

Množina režimů činnosti prvku e je pak definována jako množina $M_e = \{\alpha \mid \alpha \text{ je režimem činnosti prvku } e \in E'_{CUA}\}$, množina režimů všech obvodových prvků CUA jako množina $M_{CUA} = \bigcup_{e \in E'_{CUA}} M_e$.

□

Množina X_e je tedy podmnožinou bran, příslušejících vstupním portům prvku $e \in E'_{CUA}$. Přítomnost dat na těchto branách způsobí, že prvek e bude mít na svých vstupech data, která obecně ovlivňují jeho činnost. To, na které ze vstupních bran prvku e je třeba nastavit jaké hodnoty, aby prvek e pracoval v jistém režimu své činnosti, je určeno jednak obsahem množiny X_e a jednak zobrazením α určujícím, jaká data se na branách z množiny X_e vyskytují. K již existujícím pojmům řídící port, řídící brána tato definice přidává pojmy řídící port režimu činnosti prvku a řídící brána režimu činnosti prvku.

Příklad 16:

Jako první příklad k definici 13 si vezmeme MUX1 z obrázku 24 (strana 77) s rozhraním podle příkladu 12, strana 81. Pro MUX1 platí $Y = \{a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0, sel_0, q_3, q_2, q_1, q_0\} \setminus \{q_3, q_2, q_1, q_0\} = \{a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0, sel_0\}$. Podle definice 13 není za režim činnosti prvku MUX1 pokládáno pouze každé nastavení řídící brány (v tomto případě brány sel_0) na danou hodnotu (tj. zobrazení α definované předpisem $sel_0 \mapsto 0$ resp. předpisem $sel_0 \mapsto 1$, kde $X_e = \{sel_0\} \subseteq Y$), ale také každé zobrazení α s nosičem $X_e \subseteq Y$ - např. zobrazení α (na množině $X_e = \{a_3, a_2, a_1, a_0\} \subseteq Y$) definované předpisem $a_3 \mapsto 0, a_2 \mapsto 1, a_1 \mapsto 1, a_0 \mapsto 0$.

Další příklady k definici 13 budou demonstrovány na prvcích z obrázku 20 (strana 65). V následujícím textu budou ke každému prvku z tohoto obrázku uvedeny příklady vybraných režimů jeho činnosti.

- MUX je prvek z obrázku 20a, strana 65. Nechť $MUX \in E'_{CUA}$, nechť pro jeho rozhraní platí $\psi'(MUX) = \{x_0, x_1, x_2, x_3, sel_0, sel_1, y\}$ a nechť $x_0 = (x_{01}, x_{00}), x_1 = (x_{11}, x_{10}), x_2 = (x_{21}, x_{20}), x_3 = (x_{31}, x_{30}), sel_0 = (sel_{00}), sel_1 = (sel_{10}), y = (y_1, y_0)$, kde $\{x_{01}, x_{00}, x_{11}, x_{10}, x_{21}, x_{20}, x_{31},$

⁷ v daném časovém okamžiku

4.1 Model struktury číslicového obvodu 89

$x_3\} \subset BICUA$, $\{sel_0, sel_1\} \subset BCICUA$, $\{y_1, y_0\} \subset BOCUA$. Pak $Y = BICUA \cup BCICUA = \{x_{0_1}, x_{0_0}, x_{1_1}, x_{1_0}, x_{2_1}, x_{2_0}, x_{3_1}, x_{3_0}, sel_0, sel_1\}$. Režimem činnosti prvku MUX je např. každé zobrazení α s nosičem $X_e = \{sel_0, sel_1\}$, tj. každé nastavení řídicích vstupů multiplexoru MUX na jistou hodnotu - touto hodnotou je určeno, ze kterého vstupu prvku MUX budou data přenesena na výstup prvku MUX - pro takovou činnost byl $MUX1$ navržen (např. zobrazení α definované předpisem $sel_0 \mapsto 0, sel_1 \mapsto 1$ zajistí takový režim činnosti, ve kterém budou data ze vstupu $MUX.x_1$ přenesena na výstup $MUX.y$)

- NOT je prvek z obrázku 20b (strana 65). Nechť $NOT \in E'_{CUA}$, nechť pro jeho rozhraní platí $\psi'(NOT) = \{x, y\}$ a nechť $x = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$, $y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0)$, kde $\{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\} \subset BICUA$, $\{y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0\} \subset BOCUA$. Pak $Y = BICUA \cup BCICUA = \{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\}$; režimem činnosti prvku NOT je každé zobrazení α s nosičem $X_e \subseteq Y$. Všimněme si však, že pro provádění činnosti, pro kterou byl prvek NOT navržen (tj. aby na výstup y byla zaslána data, která jsou negací dat ze vstupu x) není třeba nastavovat data na žádné řídicí vstupy prvku NOT (prvek NOT také žádné řídicí vstupy nemá) - stačí pouze přiložit vstupní data na vstup x a (po jistém zpoždění) odebrat jejich negaci z výstupu y . Prvek NOT je tedy příkladem prvku, který byl navržen pouze pro provádění jediné činnosti - pokud bychom chtěli tuto činnost vyjádřit režimem činnosti podle definice 13, zjistili bychom, že nosič zobrazení α musí být prázdná množina a tedy režim činnosti odpovídající této jediné činnosti bude prázdné zobrazení α .
- ADD je prvek z obrázku 20c (strana 65). Nechť $ADD \in E'_{CUA}$, nechť pro jeho rozhraní platí $\psi'(ADD) = \{x_0, x_1, y\}$ a nechť $x_0 = (x_{0_1}, x_{0_0})$, $x_1 = (x_{1_1}, x_{1_0})$, $y = (y_1, y_0)$, kde $\{x_{0_1}, x_{0_0}, x_{1_1}, x_{1_0}\} \subset BICUA$, $\{y_1, y_0\} \subset BOCUA$. Pak $Y = BICUA \cup BCICUA = \{x_{0_1}, x_{0_0}, x_{1_1}, x_{1_0}\}$. Obdobně jako předcházející prvek NOT , tak ani prvek ADD nemá žádné vstupy nutné pro řízení činnosti, pro kterou byl navržen. Pro provádění činnosti, pro kterou byl ADD navržen (tj. aby na výstup y byla zaslána hodnota, která je součtem hodnot ze vstupů x_0 a x_1) tedy stačí pouze přiložit vstupní data na vstupy x_0, x_1 a (po jistém zpoždění) odebrat jejich součet z výstupu y . Přestože však prvek ADD nemá řídicí vstupy, je jeho činnost ovlivněna nejenom tím, jaká data jsou na jednom z jeho vstupních datových portů, ale také tím, jaká data jsou na jeho druhém vstupním datovém portu. Z režimů činnosti prvku ADD tedy vyberme tři následující (každý z nich reprezentuje jiný popis původní činnosti prvku ADD):
 - režim činnosti prvku ADD je prázdné zobrazení α . Na výstup y je zaslána hodnota, která je součtem hodnot vyskytujících se na vstupech x_0, x_1 ,
 - režim činnosti prvku ADD je zobrazení α s nosičem $X_e = \{x_{0_1}, x_{0_0}\}$. Přiložením různých dat na brány z množiny X_e obecně dosáhneme toho, že na tutéž hodnotu na vstupu x_1 bude ADD reagovat různými

90 4 Model obvodu popsaného na úrovni meziregistrových přenosů

odezvami na výstupu y . Z tohoto pohledu můžeme na brány z množiny X_e pohlížet jako na řídicí,

- obdobně, režim činnosti prvku ADD je zobrazení α s nosičem $X_e = \{x_{11}, x_{10}\}$. Přiložením různých dat na brány z množiny X_e obecně dosáhneme toho, že na tutéž hodnotu na vstupu x_0 bude ADD reagovat různými odezvami na výstupu y . Z tohoto pohledu můžeme na brány z množiny X_e pohlížet jako na řídicí.
- REG je prvek z obrázku 20d (strana 65). Nechť $REG \in E'_{CUA}$, nechť pro jeho rozhraní platí $\psi'(REG) = \{x, clk, y\}$ a nechť $x = (x_3, x_2, x_1, x_0)$, $clk = (clk_0)$, $y = (y_3, y_2, y_1, y_0)$, kde $\{x_3, x_2, x_1, x_0\} \subset BI_{CUA}$, $\{clk_0\} \subset BCI_{CUA}$, $\{y_3, y_2, y_1, y_0\} \subset BO_{CUA}$. Pak $Y = BI_{CUA} \cup BCI_{CUA} = \{x_3, x_2, x_1, x_0, clk_0\}$; Režimem činnosti prvku REG je např. každé zobrazení α s nosičem $X_e = \{clk_0\}$, tj. každé nastavení vstupu clk_0 na jistou hodnotu - pro port řízený hranou je to buď hodnota \uparrow nebo hodnota \downarrow - touto hodnotou je určeno, zda data, vyskytující se na vstupu x registru REG budou uložena do paměťových prvků tohoto registru a také zda budou k dispozici na výstupu y registru REG ; tedy zobrazení, které zajistí takový režim činnosti, ve kterém budou data ze vstupu $REG.x$ uložena do registru a budou přítomna na výstupu $REG.y$ je v tomto případě zobrazení α definované předpisem $clk_0 \mapsto \uparrow$ resp. předpisem $clk_0 \mapsto \downarrow$.

Cílem tohoto příkladu nebylo uvést všechny režimy činnosti výše uvedených prvků, jelikož uvedení jejich výčtu⁸ by vedlo k nepřehlednosti textu; cílem bylo blíže demonstrovat význam režimu činnosti prvku podle definice 13 a ukázat tím, že prvek je obecně schopen pracovat ve více režimech činnosti, než pro které byl původně navržen. Dalším z cílů bylo ukázat, že režim činnosti prvku nemusí být určen pouze hodnotami na řídicích branách a řídicích portech, ale že může být určen také hodnotami na datových vstupech a výstupech daného prvku - tedy i prvek, který neobsahuje žádnou (explicitně takto označenou) řídicí bránu obecně může pracovat v několika režimech činnosti.

□

Pomocí výše uvedených definic již dokážeme zkonstruovat množinu prvků CUA a pro každý z těchto prvků modelovat jeho rozhraní, tj. vstupy a výstupy. To však pro model struktury číslicového obvodu nepostačuje, protože zatím nemáme prostředky pro modelování spojů mezi porty prvků, tj. prostředky pro modelování vzájemného vodivého propojení mezi rozhraními obvodových prvků obvodu. Tyto prostředky budou zavedeny až v následujících definicích.

⁸ celkový počet režimů činnosti daného prvku je roven počtu zobrazení α , jejichž nosiči jsou prvky potenční množiny 2^Y

4.1.2 Model spojů a datového toku

*Definice 14**:*

Nechť $C_{BIT} \subset BIT_{CUA} \times BIT_{CUA}$ je relace na množině BIT_{CUA} reprezentující spoj mezi branami definovaná takto: $(x, y) \in C_{BIT} \Leftrightarrow$ mezi x a y existuje spoj.

Dále, necht' $C \subset PORT_{CUA} \times PORT_{CUA}$ je relace na množině $PORT_{CUA}$ reprezentující spoj mezi porty definovaná takto: $(p, q) \in C \Leftrightarrow \exists x \in \pi_{BITS}(p), y \in \pi_{BITS}(q): (x, y) \in C_{BIT}$.

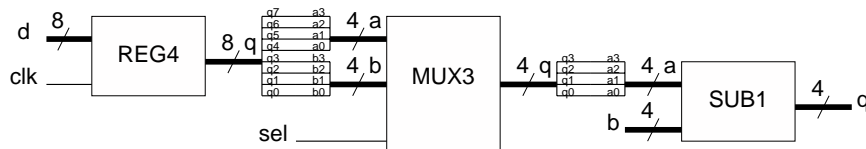
Šířka spoje mezi porty p, q je definována jako $C_W = |C_{BIT} \cap (\pi_{BITS}(p) \times \pi_{BITS}(q))|$

□

Pokud pro brány x, y platí $(x, y) \in C_{BIT}$, pak mezi branami x, y existuje vodivé spojení, tj. spoj, jehož existence je nutnou podmínkou pro přenos dat mezi těmito branami.

Obdobně, spoj mezi porty p, q existuje právě tehdy, když existuje spoj mezi branami x, y , kde x je nějaká brána příslušející portu p (tj. $x \in \pi_{BITS}(p)$) a y je nějaká brána příslušející portu q (tj. $y \in \pi_{BITS}(q)$).

Šířka spoje C_W mezi porty p, q je definována jako počet dvojic bran portů p, q , mezi nimiž existuje spoj.



Obrázek 25. Propojení rozhraní prvků

Věta 4:

Relace C_{BIT} resp. relace C je reflexivní, symetrická a tranzitivní.

□

Důkaz:

Reflexivita, symetrie a tranzitivita relace C_{BIT} resp. relace C plyne z fyzikální podstaty vodivého spoje. Necht' R označuje relaci C_{BIT} resp. relaci C . Pak 1) každý vývod je spojen sám se sebou, tj. xRx (reflexivita R), 2) je-li vývod x spojen s vývodem y (tj. xRy), pak je současně i vývod y spojen s vývodem x , tj. yRx (symetrie R), 3) je-li vývod x spojen s vývodem y (tj. xRy) a je-li současně vývod y spojen s vývodem z (tj. yRz), pak je také vývod x spojen s vývodem z , tj. xRz (tranzitivita R).

□

92 4 Model obvodu popsaného na úrovni meziregistrových přenosů

Definice 15:

Nechť $F_{BIT} \subseteq C_{BIT}$ je binární relace na množině BIT_{CUA} taková, že $(a, b) \in F_{BIT} \Leftrightarrow$ je možný tok jednobitových dat ve směru z brány a na bránu b .

Dále, nechť $F \subseteq C$ je binární relace na množině $PORT_{CUA}$ definovaná takto: $(p, q) \in F \Leftrightarrow$ množina $F_{BIT} \cap (\pi_{BITS}(p) \times \pi_{BITS}(q))$ je neprázdná.

□

F je tedy relace ”směru toku dat” mezi dvěma porty p, q . Zřejmě, pokud $(p, q) \in F$, je existuje tok dat ve směru z portu $p \in PORT_{CUA}$ na port $q \in PORT_{CUA}$.

4.2 Model transparentních režimů a datových cest

V následujícím textu budou, v duchu myšlenek nastíněných zejména v odstavci 3.4.1, strana 62, zavedeny pojmy týkající se transparentních režimů obvodových prvků a transparentních (diagnostických) datových cest obvodovou strukturou a tyto pojmy budou zahrnuty do budovaného modelu.

4.2.1 Model transparentních režimů

Z předchozího textu je patrné následující:

1. jisté brány obvodového prvku jsou seskupeny do jednoho vícebitového (logického) vývodu, který je ”zapouzdřuje”; takovým logickým vývodem je výše definovaný port,
2. na vícebitový vývod se pak dále pohlíží jako na jeden port,
3. rozhraní modulu se pak často nevyobrazuje detailně pomocí jednobitových vývodů, ale obvykle postačuje stručnější a přehlednější vyobrazení pomocí portů.

V následujícím textu bude zaveden další pojem - virtuální port. Virtuálním portem budeme rozumět logický, v návrhu explicitně nedefinovaný port, který je tvořen vybranými vstupně-výstupními branami modulu. Jelikož každý virtuální port je prvkem množiny $ORBIT_{CUA}$, lze jej chápat jako seskupení vybraných podčástí jistých návrhových vývodů daného modulu; konkrétně pak hovoříme o virtuálním vstupním resp. výstupním portu.

Definice 16:

Bud’ definována množina⁹ $VPORT_{CUA} = \{\pi_{LEX}(x) \mid x \in ORBIT_{CUA}\}$. Prvek $x \in VPORT_{CUA}$ nazveme *virtuálním portem* (prvku $y = \beta_E(\pi_{BIT}(x, 0))$), přesněji pak

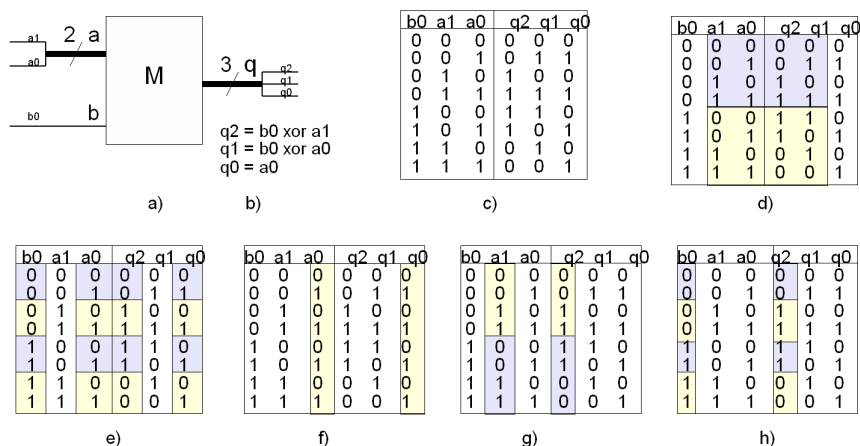
⁹ při implementaci a aplikaci v praxi bude výhodné, omezíme-li tuto množinu tak, aby $\forall x \in VPORT_{CUA}: [\exists x' \in \pi_{BITS}(x), y' \in BIT_{CUA}: (x', y') \in F_{BIT} \vee (y', x') \in F_{BIT}]$. Důležité je, že tímto omezením nedojde ke ztrátě informace o diagnostických vlastnostech obvodu. Význam M_S, M_I viz níže - definice 17, strana 93

4.2 Model transparentních režimů a datových cest 93

- virtuálním vstupním datovým portem prvku y , pokud $VPORT_{CUA} \subseteq BI_{CUA}^+ \cup BPI_{CUA}^+$,
- virtuálním vstupním řídicím portem prvku y , pokud $VPORT_{CUA} \subseteq BCI_{CUA}^+ \cup BPCI_{CUA}^+$,
- virtuálním výstupním portem prvku y , pokud $VPORT_{CUA} \subseteq BO_{CUA}^+ \cup BPO_{CUA}^+$.

Pokud $VPORT_{CUA} \subseteq BPI_{CUA}^+ \cup BPCI_{CUA}^+ \cup BPO_{CUA}^+$, pak prvek množiny $VPORT_{CUA}$ navíc nazýváme *primárním*.

□



Obrázek 26. Příklad k virtuálním portům a transparentním režimům

Důvodem pro zavádění virtuálních portů je snaha o pozdější snadné a jasné vyjádření možné existence transparentních cest (umožňujících přenos m -bitových diagnostických dat ve směru ze vstupů na výstupy daného obvodového prvku, viz zejména odstavec 3.4.1, od strany 62) a podmínek pro jejich vytvoření. Cílem je rozšířit množinu transparentních cest využitelných pro přenos diagnostických dat přes obvodové prvky z množiny cest mezi vývody stejných šířek na množinu dílčích cest o šířkách menších než je šířka nejmenšího z obou vývodů a transparentních cest mezi vývody různých šířek. Pro přenos diagnostických dat přes obvodové prvky tak bude možno co nejlépe využít režimů a cest, kterými dané prvky disponují.

V následujícím příkladu bude ukázána důležitost co nejpřesnější informace o transparentnosti daného obvodového prvku - bude-li totiž v důsledku umístění obvodového prvku do struktury obvodu vyloučena (z důvodu nemožnosti zajištění příslušného režimu činnosti daného obvodového prvku) existence některých - bitově širších - transparentních cest, bude výhodné, bude-li k dispozici informace o tom, zda a jak je možné zajistit režim, umožňující

existenci transparentní cesty, jejíž šířka je blízká požadované šířce toku diagnostických dat. Je zřejmé, že taková informace přispěje k efektivnějšímu využití původních datových cest pro účely přenosu diagnostických dat.

Příklad 17:

Na obrázku 26 je příklad obvodového prvku ($M \in E'_{CUA}$), jehož rozhraní je tvořeno vstupními porty $a = (a_1, a_0)$, $b = (b_0)$ a výstupním portem $q = (q_2, q_1, q_0)$, $a, b, q \in PORT_{CUA}$.

Vstupně-výstupní chování prvku M je popsáno rovnicemi uvedenými na obrázku 26b). Je možné ho popsat takto: 1) hodnota z brány a_0 je vždy přenesena na bránu q_0 a 2) hodnoty z bran a_1 , a_0 jsou přeneseny na brány q_2 , q_1 negované právě když $b_0 = 1$ tj. je-li $b_0 = 0$, pak $\beta_V(q_2) = \beta_V(a_1)$, $\beta_V(q_1) = \beta_V(q_0) = \beta_V(a_0)$; jinak $\beta_V(q_2) = \neg\beta_V(a_1)$, $\beta_V(q_1) = \neg\beta_V(a_0)$, $\beta_V(q_0) = \beta_V(a_0)$.

Na obrázcích 26c – h je pak uvedeno několik případů, kdy je prvek M schopen zajistit (v jistém režimu své činnosti) bijekci hodnot mezi svými jistými vstupy a výstupy.

Pro obrázek 26c zvolme z $\{b_0, a_1, a_0\}^+$ virtuální vstupní port $x_c = (b_0, a_1, a_0)$ a z $\{q_2, q_1, q_0\}^+$ virtuální výstupní port $y_c = (q_2, q_1, q_0)$. Zřejmě i pro každé jiné zvolené x_c, y_c existuje bijekce f taková, že $\nu(y_c) = f(\nu(x_c))$. Pro existenci bijekce f je režim činnosti (dle definice 13, strana 86) prázdná množina, tzn. že bijekce f existuje vždy (tj. neexistují data, jejichž přiložení na vstupy M by podmiňovalo existenci f). Prvek M je tedy schopen přenést libovolná 3-bitová data ze svých vstupů na své výstupy. To, zda je možno na virtuálním portu x_c zajistit libovolnou 3-bitovou kombinaci, však závisí na umístění prvku ve struktuře CUA. Následující případy ukáží, že za jistých okolností je možné přenést určitá data i v případě, že některé kombinace bitů není možno na vstupech M zajistit - šířka takto přenesených dat sice bude menší než 3, avšak podstatné je, že k tomuto přenosu bude možno využít původních datových cest obvodu.

Pro obrázek 26d zvolme z $\{a_1, a_0\}^+$ virtuální vstupní port $x_d = (a_1, a_0)$ a z $\{q_2, q_1\}^+$ virtuální výstupní port $y_d = (q_2, q_1)$. Zřejmě i pro každé jiné zvolené x_d, y_d existuje bijekce f taková, že $\nu(y_d) = f(\nu(x_d))$. V tomto případě však f závisí na tom, jaká hodnota se vyskytuje na bráně b_0 , a režim činnosti je jednoprvková množina $\{b_0, 0\}$ nebo $\{b_0, 1\}$. Požadavek na výskyt kombinace hodnot na vstupech M již v tomto případě není tak přísný jako v případě c. K přenosu dat - tentokrát však pouze 2-bitových - postačuje, aby bylo možno na bráně b_0 zajistit jednu z hodnot 0, 1. Volba tohoto 2-bitového přenosu řízeného hodnotou vyskytující se na bráně b_0 zřejmě bude na místě tehdy, bude-li např. ztíženo či znemožněno zajistit (nezávisle na datech na branách a_1, a_0) výskyt některé z hodnot 0, 1 na bráně b_0 . Obdobnou situaci lze sledovat v případě obrázku 26e - v tomto případě je virtuální vstupní port x_e volen z množiny $\{b_0, a_0\}^+$ a virtuální výstupní port y_e volen z množiny $\{q_2, q_0\}^+$. Existence bijekce f je v případě e podmíněna schopností nastavit režim činnosti $\{a_1, 0\}$ nebo $\{a_1, 1\}$.

Poslední tři případy vyobrazené na obrázcích 26*f* – *h* ukazují, že transparentní přenos dat ve směru ze vstupů na výstupy prvku *M* je možný i tehdy, není-li možno dostatečně či vůbec ovládat hodnoty na dvou ze tří vstupních bran. Přenos dat přes prvek *M* je pak 1-bitový a je možný ve směru ze třetí (tj. zbývající) vstupní brány na některou z výstupních bran prvku *M*. Na obrázcích *f* – *h* jsou vyznačeny vybrané bijekce mezi hodnotami na dané vstupní bráně a hodnotami na dané výstupní bráně, přičemž režim činnosti podmiňující existenci těchto bijekcí je v případě *f* prázdná množina (tj. bijekce mezi hodnotami na a_0 a hodnotami na q_0 není podmíněná a tedy existuje vždy), v případě *g* závisí bijekce mezi hodnotami na a_1 a hodnotami na q_2 na hodnotě vyskytující se na bráně b_0 , tj. režim činnosti pro každou z těchto bijekcí je $\{b_0, 0\}$ nebo $\{b_0, 1\}$ a v případě *h* závisí bijekce mezi hodnotami na b_0 a hodnotami na q_2 na hodnotě na a_1 . Na obrázcích není zvýrazněna např. bijekce hodnot mezi branami a_0 a q_1 řízená hodnotou na bráně b_0 .

□

I informace o možnosti existence transparentního datového toku o menší šířce dat než by bylo ideální, popř. informace o možnosti datového toku mezi několika v návrhu explicitně definovanými porty (tj. prvky množiny $PORT_{CUA}$), tedy může výrazně přispět k lepšímu využití datových cest v *CUA* pro přenos diagnostických dat. Díky přesnější informaci totiž dojde k minimalizaci snah o modifikaci datových cest v *CUA* za účelem lepšího přenosu diagnostických dat, což také přispěje ke snížení rozsáhlosti modifikací obvodové struktury *CUA*.

V příkladě 17 byla nastíněna problematika spojená s hledáním bijekcí mezi hodnotami na vstupech a hodnotami na výstupech daného obvodového prvku - zejména problematika hledání vnořených bijekcí a problematika hledání bijekcí umožňujících co nejširší datový tok.

Lze snadno nahlédnout, že pro přenos diagnostických dat není nutno vyžadovat nastavení režimu umožňujícího existenci bijekce mezi hodnotami na vstupech a hodnotami na výstupech daného obvodového prvku - již v odstavci 3.4.1 bylo konstatováno, že režimy činnosti obvodového prvku mohou být zařazeny do skupiny režimů vhodných pro přenos testovacích vzorků, skupiny režimů vhodných pro přenos odezev či do skupiny režimů nevhodných pro přenos diagnostických dat, přičemž mohou existovat režimy patřící současně do obou prvních skupin. Příslušnost konkrétního režimu do skupiny pak závisí na tom, zda je prvek schopen pracovat v režimu činnosti, umožňujícím existenci surjekce (pak je režim vhodný pro přenos testovacích vzorků), injekce (pak je režim vhodný pro přenos odezev) či dokonce bijekce (pak je režim vhodný pro přenos libovolných diagnostických dat) mezi daty na vstupech prvku a daty na výstupech prvku. V následujícím textu budou tyto režimy definovány pomocí prostředků zavedeného modelu.

Definice 17:

Buďte definovány množiny

96 4 Model obvodu popsaného na úrovni meziregistrových přenosů

- $M_I = \{(x, y) \mid \exists p \in E'_{CUA}: (x, y \in VPORT_{CUA} \text{ jsou virtuální porty prvku } p \wedge \exists \text{ posloupnost}^{10} \alpha_1, \dots, \alpha_n \in M_p \text{ režimů činnosti taková, že jsou-li režimy nastavovány v pořadí počínaje režimem } \alpha_1 \text{ a konče režimem } \alpha_n, \text{ pak po nastavení režimu } \alpha_n \text{ platí } \nu(y) = f(\nu(x))), \text{ kde } n \in \mathbb{N}, n \geq 1, \text{ každé ze zobrazení } \alpha_1, \dots, \alpha_n \text{ momentálně postačuje pro dané chování prvku } p, f: \{0, 1\}^{\pi w(x)} \rightarrow \{0, 1\}^{\pi w(y)} \text{ je injektivní zobrazení a data určená k přenosu jsou na } x \text{ umístěna v rámci některého z režimů } \alpha_1, \dots, \alpha_n\}$,
- $M_S = \{(x, y) \mid \exists p \in E'_{CUA}: (x, y \in VPORT_{CUA} \text{ jsou virtuální porty prvku } p \wedge \exists \text{ posloupnost}^{11} \alpha_1, \dots, \alpha_n \in M_p \text{ režimů činnosti taková, že jsou-li režimy v pořadí počínaje režimem } \alpha_1 \text{ a konče režimem } \alpha_n, \text{ pak po nastavení režimu } \alpha_n \text{ platí } \nu(y) = f(\nu(x))), \text{ kde } n \in \mathbb{N}, n \geq 1, \text{ každé ze zobrazení } \alpha_1, \dots, \alpha_n \text{ momentálně postačuje}^{12} \text{ pro dané chování prvku } p, f: \{0, 1\}^{\pi w(x)} \rightarrow \{0, 1\}^{\pi w(y)} \text{ je surjektivní zobrazení a data určená k přenosu jsou na } x \text{ umístěna v rámci některého z režimů } \alpha_1, \dots, \alpha_n\}$.

Zobrazení $f: \{0, 1\}^{\pi w(x)} \rightarrow \{0, 1\}^{\pi w(y)}$ dat z x na data y , které je podmíněno podmínkou $p_{xy}[M_I]$ resp. $p_{xy}[M_S]$, bude značeno $f[p_{xy}[M_I]]$ resp. $f[p_{xy}[M_S]]$ a bude nazýváno *přenosem dat s podmínkou $p_{xy}[M_I]$ určeným zobrazením f* resp. *přenosem dat s podmínkou $p_{xy}[M_S]$ určeným zobrazením f* . V triviálním případě, kdy neexistují brány, které by ovlivňovaly daný přenos dat, bude takový přenos dat podmíněn jednočlennou posloupností obsahující jako jediný člen prázdný režim činnosti.

□

Přítomnost prvku (uspořádaná dvojice (x, y)) v některé z těchto množin vyjadřuje, že za jisté podmínky je možno přenést diagnostická data ve směru z virtuálního vstupu x na virtuální výstup y obvodového prvku, přičemž zobrazení f pak říká, jaký je vztah mezi vstupními daty (daty na x) a výstupními daty (daty na y) tohoto přenosu. V následujícím textu budou zavedeny prostředky, které umožní vyjádřit vztah mezi uspořádanou dvojicí (x, y) , podmínkou přenosu dat ve směru z x na y a transformací dat provedenou během přenosu.

¹⁰ každou takovou posloupnost nazveme *podmínkou injektivního přenosu dat* (jistého zobrazení f dat z x na data na y) *prvkem p ve směru z x na y* (podmínka bude značena $p_{xy}[M_I]$)

¹¹ každou takovou posloupnost nazveme *podmínkou surjektivního přenosu dat* (jistého zobrazení f dat z x na data na y) *prvkem p ve směru z x na y* (podmínka bude značena $p_{xy}[M_S]$)

¹² každé ze zobrazení $\alpha_1, \dots, \alpha_n$ z posloupnosti $p_{xy}[M_I]$ resp. $p_{xy}[M_S]$ je tedy režimem činnosti neobsahujícím nadbytečné brány, jejichž ovládání nemá žádný vliv na přenos dat z x na y , tj. na zobrazení $f[p_{xy}[M_I]]$ resp. $f[p_{xy}[M_S]]$. Snahou při konstrukci $p_{xy}[M_I]$ resp. $p_{xy}[M_S]$ tedy je pro každý z režimů $\alpha_1, \dots, \alpha_n$ nalézt nejmenší množinu bran, na jejichž ovládání tento režim závisí; rozšířením takového režimu o další prvky zřejmě v daném okamžiku nedojde ke změně chování prvku, protože režim by byl rozšířen pouze o ty brány, na nichž chování prvku v daném okamžiku nezávisí - viz definice 13, strana 86

V následujícím textu budou definovány relace dávající do vztahu prvek (x, y) z některé z množin z definice 17 (tj. zdrojový (x) a cílový (y) virtuální port datového přenosu) s režimem (podmínkami) nutným pro nastavení přenosu dat z x na y a se zobrazením f , udávajícím vstupně-výstupní charakter tohoto přenosu dat strukturou daného prvku.

Definice 18:

Bud'te definovány relace $R_{HT_I} = \{(x, y, f[p_{xy}[M_I], p_{xy}[M_I]) \mid p \in E'_{CUA}, (x, y) \in M_I, \text{prvek } p \text{ se během činnosti dané aplikací } p_{xy}[M_I] \text{ započaté v libovolném vnitřním stavu prvku } p \text{ dostane do každého vnitřního stavu nejvýše jednou}\}$,

$R_{HT_S} = \{(x, y, f[p_{xy}[M_S], p_{xy}[M_S]) \mid p \in E'_{CUA}, (x, y) \in M_S, \text{prvek } p \text{ se během činnosti dané aplikací } p_{xy}[M_S] \text{ započaté v libovolném vnitřním stavu prvku } p \text{ dostane do každého vnitřního stavu nejvýše jednou}\}$

a zobrazení $R_{HT}: (R_{HT_I} \cup R_{HT_S}) \times \mathbb{N} \rightarrow M_e$ přiřazující prvku $z = (x, y, f[p_{xy}], p_{xy}) \in R_{HT_I} \cup R_{HT_S}$, kde p_{xy} je posloupnost $\alpha_1, \dots, \alpha_n$, a číslu $k \in \mathbb{N}$, $1 \leq k \leq n$, režim α_k .

□

Pro každý obvodový prvek tedy relace R_{HT_I} a R_{HT_S} poskytují dostatečně přesnou informaci jak o možnostech přenosu a transformace vstupních dat na výstupní, tak o nezbytných podmínkách, které musí být splněny, aby se tento přenos včetně transformace uskutečnil. Tato informace pak může být využita zejména k urychlení postupů, které potřebují pracovat s co nejpřesnějšími informacemi o jistých vlastnostech obvodových prvků - takové postupy pak nemusejí tyto informace složitě získávat až během provádění postupu z popisu prvků, ale mohou je jednoduše načíst např. z knihovny prvků obohacené právě o tyto informace. Příkladem oblasti, kde může být informace obsažená v relacích R_{HT_I} , R_{HT_S} úspěšně využita, je oblast zabývající se problematikou hierarchického generování testu.

Věta 5:

Nechť $p \in E'_{CUA}$. Pokud $\exists z = (x, y, f[p_{xy}], p_{xy}) \in R_{HT_I} \cup R_{HT_S}$, kde p_{xy} je posloupnost s alespoň dvěma členy, pak p je sekvenční prvek; jinak je p buď kombinační prvek nebo sekvenční prvek neumožňující pracovat v režimu činnosti zaručujícím přenos diagnostických dat

□

Pro účely analýzy testovatelnosti však tak detailní a obsáhlé informace není třeba. Řadu informací (velmi užitečných např. při hierarchickém generování testu) lze v případě analýzy testovatelnosti zanedbat, a to zejména informaci o existenci několika různých zobrazení mezi týmiž virtuálními porty řízených týmiž branami či informaci o konkrétním tvaru daných zobrazení. Z hlediska analýzy testovatelnosti totiž není podstatné, jak konkrétně přenos mezi dvěma virtuálními porty vypadá, ale zda je možné nějaký zajistit a za jakých podmínek. Co však např. je pro účely analýzy testovatelnosti třeba,

je informace o branách daného prvku, mezi nimiž je možný datový přenos, spojená do relace s informací o množině řídicích bran tohoto přenosu. Další text se bude zabývat popisem informace využitelné pro analýzu testovatelnosti *CUA*.

Definice 19:

Množinu $\bigcup_{i=1}^{\text{délka posloupnosti } p_{xy}[M_I]} \text{Def}(\alpha_i)$ resp. $\bigcup_{i=1}^{\text{délka posloupnosti } p_{xy}[M_S]} \text{Def}(\alpha_i)$, kde α_i je i . člen posloupnosti $p_{xy}[M_I]$ resp. $p_{xy}[M_S]$ nazveme *množinou řídicích bran podmínky* $p_{xy}[M_I]$ resp. $p_{xy}[M_S]$.

Buďte definovány relace $R_{TA_I} = \{(x, y, k, X) \mid (x, y, f[p_{xy}[M_I]], p_{xy}[M_I]) \in R_{HT_I} \wedge k \text{ je délka nejkratší posloupnosti } p_{xy}[M_I] \text{ s množinou } X \text{ řídicích bran podmínky } p_{xy}[M_I]\}$ a

$R_{TA_S} = \{(x, y, k, X) \mid (x, y, f[p_{xy}[M_S]], p_{xy}[M_S]) \in R_{HT_S} \wedge k \text{ je délka nejkratší posloupnosti } p_{xy}[M_S] \text{ s množinou } X \text{ řídicích bran podmínky } p_{xy}[M_S]\}$.

□

Prvky relace R_{TA_I} resp. R_{TA_S} jsou tedy uspořádané čtveřice (x, y, k, X) , kde x je zdrojový a y cílový virtuální port daného datového toku přes prvek $p = \beta_E(x) = \beta_E(y)$, k určuje nejmenší počet (délku nejkratší sekvence) režimů v podmínce přenosu a $X \subseteq \text{BIT}_{CUA}$ je množinou řídicích bran podmínky tohoto přenosu. Je-li možné přenést data prvkem p ve směru z x na y několika způsoby takovými, že množiny řídicích bran podmínek p_{xy} každého z těchto přenosů jsou si rovny, pak je zřejmé, že podle definice 19 bude v relaci R_{TA_I} resp. R_{TA_S} obsažena pouze jedna čtveřice (a to ta, která informuje o délce nejkratší sekvence režimů v příslušné podmínce), což povede k rychlejšímu zjištění, jestli mezi danými množinami bran téhož prvku může existovat požadovaný datový tok, ovládáním jakých (dalších) bran a jakou nejkratší sekvenční délkou je jeho existence podmíněna.

Z pohledu analýzy testovatelnosti tedy není nutné zjištění, jaké konkrétní zobrazení (přenos) mezy daty z x a daty z y lze zajistit, ale postačuje informace o tom, zda takový přenos je možný, zda je vhodný pro přenos testovacího vzorku či odezvy, jaká je množina řídicích bran tohoto přenosu a jaká je nejkratší sekvenční délka tohoto přenosu.

Jelikož historicky dříve zavedená koncepce *I-režimů* resp. *T-režimů* je speciálním případem v této práci použité koncepce, je možno prostředky z této práce využít také k popisu dřívějších koncepcí.

4.2.2 Model transparentních datových cest

Struktura číslicového obvodu, jehož diagnostické vlastnosti zkoumáme, je tvořena obvodovými prvky, které jsou přes svá rozhraní propojeny vodivými spoji. V předchozím textu byly zavedeny prostředky pro popis datového toku po spojích (odstavec 4.1.2) a prostředky pro popis datového toku přes strukturu každého z obvodových prvků (4.2.1); společným použitím těchto prostředků je možno - pomocí dvojice orientovaných grafů G_S , G_I definovaných níže - vyjádřit model toku diagnostických dat v obvodu *CUA*.

Definice 20:

Nechť $X = \{(x, y) \mid x, y \in VPORT_{CUA} \wedge \exists x', y': (x', y') \in F_{BIT} \cap \pi_{BITS}(x) \times \pi_{BITS}(y)\}$, $n_X = |X|$, $n_S = |R_{TAS}|$, $n_I = |R_{TAI}|$ a nechtě jsou prvky množin $|X|$, $|R_{TAS}|$ a $|R_{TAI}|$ označeny tak, že $X = \{h_1, \dots, h_{n_X}\}$, $R_{TAS} = \{h'_{n_X+1}, \dots, h'_{n_X+n_S}\}$ a $R_{TAI} = \{h''_{n_X+1}, \dots, h''_{n_X+n_I}\}$. Bud' definován

- graf datového toku testovacích vzorků $G_S = (V_S, E_S, \sigma_S)$, kde $V_S \subseteq VPORT_{CUA}$ je množina uzlů grafu G_S , $E_S = \{h_{S1}, \dots, h_{S_{n_X+n_S}}\}$ je množina hran grafu G_S a kde $\sigma_S: E_S \rightarrow (V_S \times V_S)$ je incidence grafu G_S přiřazující hraně h_{Si} , $i \in \{1, \dots, n_X + n_S\}$ uspořádanou dvojici uzlů (x, y)

$$\text{takovou, že } \begin{cases} h_i = (x, y) & \text{je-li } 1 \leq i \leq n_X \\ h'_i = (x, y, k, X) & \text{je-li } n_X + 1 \leq i \leq n_X + n_S \end{cases}$$

a kde V_S je volena největší taková, že neobsahuje izolované uzly,

- graf datového toku odezev $G_I = (V_I, E_I, \sigma_I)$, kde $V_I \subseteq VPORT_{CUA}$ je množina uzlů grafu G_I , $E_I = \{h_{I1}, \dots, h_{I_{n_X+n_I}}\}$ je množina hran grafu G_I a kde $\sigma_I: E_I \rightarrow (V_I \times V_I)$ je incidence grafu G_I přiřazující hraně h_{Ii} , $i \in \{1, \dots, n_X + n_I\}$ uspořádanou dvojici uzlů (x, y)

$$\text{takovou, že } \begin{cases} h_i = (x, y) & \text{je-li } 1 \leq i \leq n_X \\ h''_i = (x, y, k, X) & \text{je-li } n_X + 1 \leq i \leq n_X + n_I \end{cases}$$

a kde V_I je volena největší taková, že neobsahuje izolované uzly.

Dále zavedme ohodnocení

- $w_{SX}: E_S \rightarrow 2^{BIT_{CUA}}$ hran grafu G_S jako zobrazení definované předpisem

$$h_{Si} \mapsto \begin{cases} \emptyset & \text{je-li } 1 \leq i \leq n_X \\ X, \text{ kde } (x, y, k, X) = h'_i & \text{je-li } n_X + 1 \leq i \leq n_X + n_S \end{cases}$$

- $w_{IX}: E_I \rightarrow 2^{BIT_{CUA}}$ hran grafu G_I jako zobrazení definované předpisem

$$h_{Ii} \mapsto \begin{cases} \emptyset & \text{je-li } 1 \leq i \leq n_X \\ X, \text{ kde } (x, y, k, X) = h''_i & \text{je-li } n_X + 1 \leq i \leq n_X + n_I \end{cases}$$

- $w_{SSeq}: E_S \rightarrow \mathbb{N}$ hran grafu G_S jako zobrazení definované předpisem

$$h_{Si} \mapsto \begin{cases} 0 & \text{je-li } 1 \leq i \leq n_X \\ k, \text{ kde } (x, y, k, X) = h'_i & \text{je-li } n_X + 1 \leq i \leq n_X + n_S \end{cases}$$

- $w_{ISeq}: E_I \rightarrow \mathbb{N}$ hran grafu G_I jako zobrazení definované předpisem

$$h_{Ii} \mapsto \begin{cases} 0 & \text{je-li } 1 \leq i \leq n_X \\ k, \text{ kde } (x, y, k, X) = h''_i & \text{je-li } n_X + 1 \leq i \leq n_X + n_I. \end{cases}$$

100 4 Model obvodu popsaného na úrovni meziregistrových přenosů

□

Orientovanou hranou jsou v grafu G_S resp. G_I tedy spojeny ty uzly (virtuální porty obvodových prvků), mezi nimiž je možno ve směru daném orientací hrany přenášet diagnostická data, tj. testovací vzorky resp. odezvy. Redukovanou informaci využitelnou během analýzy testovatelnosti - tj. informací pouze o zdroji, cíli přenosu, řídicích branách a sekvenční délce podmínky - každého takového přenosu lze získat pomocí ohodnocení w_{SX} , w_{SSeq} resp. w_{IX} , w_{ISeq} . Další informace (využitelné např. během hierarchického generování testu), jako např. informaci o pořadí aplikace a hodnotovém řízení jednotlivých dílčích režimů dané podmínky či informaci o transformaci daných diagnostických dat prováděné během jejich přenosu mezi uzly s každou hranou incidujícími (informaci o zobrazení vstupních dat na výstupní data během daného přenosu) lze získat z relací zavedených v definici 18. V dalším textu bude představena metoda, která pro každý virtuální port v obvodu (a následně i pro port z množiny $PORT_{CUA}$) využije redukované informace k tomu, že ohodnotí možnost a snadnost jak nastavení testovacího vzorku na tento port, tak sledování hodnoty vyskytující se na tomto portu. Zejména bude analyzováno, jaké procento datové šířky každého portu je možno ovládat/sledovat a jaká bude náročnost nastavení testovacího vzorku resp. sledování odezvy.

Definice 21:

Nechť $x \in VPORT_{CUA}$ a nechť je ke každému zobrazení $\sigma: A \rightarrow (B \times B)$ zavedeno zobrazení $\bar{\sigma}: A \rightarrow (B \times B)$ takové, že $\forall x \in A: (\bar{\sigma}(x) = (b_2, b_1) \Leftrightarrow \sigma(x) = (b_1, b_2))$.

Nechť $G''_S = (V''_S, E''_S, \sigma''_S)$ je takový podgraf grafu $G'_S = (V_S, E_S, \bar{\sigma}_S)$, že G''_S je kořenovým stromem s kořenem y (kde $y \in V_S$ je virtuální port, pro který platí $\pi_{BITS}(x) \cap \pi_{BITS}(y) \neq \emptyset$), jehož každý list náleží množině BP_{CUA}^+ a $\neg \exists (u_1, u_2), (v_1, v_2) \in Im(\sigma''_S) \cap M_S^{-1}: [(u_1, u_2) \neq (v_1, v_2) \wedge \pi_{BITS}(u_1) \cap \pi_{BITS}(v_1) \neq \emptyset]$. Každý graf $\bar{G}_{S[x]} \stackrel{def}{=} (V''_S, E''_S, \bar{\sigma}''_S)$ bude nazván *systémem transparentních cest pro nastavení dat na port x* pokud $\forall z \in \{y' \mid y' \in V''_S \cap Def(M_S) \text{ není listem ani kořenem v } G''_S\}: \exists (y_1, z), \dots, (y_n, z) \in \bar{\sigma}''_S: \{z' \mid (y', z') \in F_{BIT}, y' \in \pi_{BITS}(y_i), i = 1, \dots, n\} = \pi_{BITS}(z)$. Kořenem resp. listy systému $\bar{G}_{S[x]}$ bude označován kořen resp. listy stromu G''_S .

Každý podgraf $\bar{G}_{I[x]} \stackrel{def}{=} (V'_I, E'_I, \sigma'_I)$ grafu G_I takový, že $\bar{G}_{I[x]}$ je kořenovým stromem s kořenem y (kde $y \in V_I$ je virtuální port, pro který platí $\pi_{BITS}(x) \cap \pi_{BITS}(y) \neq \emptyset$), jehož každý list náleží množině BP_{CUA}^+ , bude nazván *systémem transparentních cest pro sledování dat na portu x* pokud $[\neg \exists (u_1, u_2), (v_1, v_2) \in Im(\sigma'_I) \cap M_I: (u_1, u_2) \neq (v_1, v_2) \wedge \pi_{BITS}(u_2) \cap \pi_{BITS}(v_2) \neq \emptyset] \wedge [\forall z \in \{y' \mid y' \in V'_I \cap Def(M_I) \text{ není listem ani kořenem v } G'_I\}: \exists (y_1, z), \dots, (y_n, z) \in \sigma'_I: \{z' \mid (y', z') \in F_{BIT}, y' \in \pi_{BITS}(y_i), i = 1, \dots, n\} = \pi_{BITS}(z)]$.

□

Kořen systému $\bar{G}_{S[x]}$ resp. $\bar{G}_{I[x]}$ je virtuální port, který je v případě $\bar{G}_{S[x]}$ cílem přenosu testovacích vzorků resp. v případě $\bar{G}_{I[x]}$ zdrojem přenosu odezvy

datového toku. Listy systému $\overline{G}_{S[x]}$ resp. $\overline{G}_{I[x]}$ jsou virtuální porty, které jsou v případě $\overline{G}_{S[x]}$ zdrojem přenosu testovacích vzorků resp. v případě $\overline{G}_{I[x]}$ cílem přenosu odezev datového toku. Orientovanou hranou jsou v systému $\overline{G}_{S[x]}$ resp. $\overline{G}_{I[x]}$ spojeny ty virtuální porty, mezi nimiž je možný datový přenos (ve směru daném orientací hrany) buď vnitřní strukturou daného obvodového prvku nebo po spojích, přičemž pro každý virtuální port, který není ani kořenem ani listem systému, platí, že je koncovým uzlem právě takového počtu hran, že lze zajistit datový tok pro každou jeho bránu.

Každý systém $\overline{G}_{S[x]}$ tedy představuje cesty, pomocí nichž je možné ovládat hodnoty na bráně x , branách portu x či branách virtuálního portu x (tj. nastavit testovací vzorek či jeho část z primárních vstupů CUA na x) prvku $p = \beta_E(\pi_{BIT}(x, 0)) \in E'_{CUA}$. Obdobně, systém $\overline{G}_{I[x]}$ představuje cesty, pomocí nichž je možné sledovat data z x prvku p na primárních výstupech CUA . V obou případech je pro existenci daného systému (a tím i příslušného přenosu strukturou prvku p) nutno zajistit výskyt jistých dat (obecně posloupnosti dat) na primárních vstupech CUA . Obecně pak v konkrétním časovém okamžiku během aplikace testu neplatí, že daný systém je aktivní celý, ale je aktivní pouze jeho dílčí část momentálně sloužící k přenosu požadovaných dat. Zbýlá část systému může být neaktivní např. z důvodu současně probíhajícího přenosu diagnostických dat příslušejících jiné bráně resp. portu.

Jelikož pro každý prvek $x \in VPORT_{CUA}$ může obecně existovat více systémů (tj. způsobů přenosu diagnostických dat) $\overline{G}_{I[x]}$ resp. $\overline{G}_{S[x]}$, bude vhodné zavést jak množiny těchto systémů pro dané x , tak množiny všech těchto systémů v CUA .

Definice 22:

Pro $x \in VPORT_{CUA}$ buďte definovány množiny $\overline{G}_I[x] = \bigcup \overline{G}_{I[x]}$, $\overline{G}_S[x] = \bigcup \overline{G}_{S[x]}$ a množiny $\overline{G}_I = \bigcup_{x \in VPORT_{CUA}} \overline{G}_I[x]$, $\overline{G}_S = \bigcup_{x \in VPORT_{CUA}} \overline{G}_S[x]$. Dále buďte definována zobrazení

- $\tau_R: \overline{G}_I \cup \overline{G}_S \rightarrow 2^{BIT_{CUA}}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ množinu bran jeho kořenu podle předpisu $x \mapsto \pi_{BITS}(y)$, kde y je kořen systému x
- $\tau_L: \overline{G}_I \cup \overline{G}_S \rightarrow 2^{BIT_{CUA}}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ množinu bran jeho listů podle předpisu $x \mapsto \{y \mid y \in \pi_{BITS}(z), z \text{ je listem systému } x\}$
- $\tau_{CBits}: \overline{G}_I \cup \overline{G}_S \rightarrow 2^{BIT_{CUA}}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ množinu bran nutných k řízení režimů činností obvodových prvků v samostatném¹³ systému x podle předpisu

$$x \mapsto \begin{cases} \bigcup_{y \in E(x)} w_{SX}(y) & \text{pokud } x \in \overline{G}_S \\ \bigcup_{y \in E(x)} w_{IX}(y) & \text{pokud } x \in \overline{G}_I \\ 0 & \text{pokud } |V(x)| = 1 \end{cases}$$

¹³ samostatným systémem je myšlen systém, který je vyjmut ze struktury CUA , aby bylo možno analyzovat jej nezávisle na jeho konkrétním okolí v CUA

102 4 Model obvodu popsaného na úrovni meziregistrových přenosů

- $\tau_{Seq}: \overline{G}_I \cup \overline{G}_S \rightarrow \mathbb{N}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ *sekvenční délku samostatného systému x podle předpisu*

$$x \mapsto \begin{cases} \max_{\forall C} (\sum_{i=1}^n w_{SSeq}(h_{Ci})) & \text{pokud } x \in \overline{G}_S \\ \max_{\forall C} (\sum_{i=1}^n w_{ISeq}(h_{Ci})) & \text{pokud } x \in \overline{G}_I \\ 0 & \text{pokud } |V(x)| = 1 \end{cases}$$

kde C je orientovaná cesta v systému x a kde $\{h_{C1}, \dots, h_{Cn}\}$ je množina hran v cestě C

- $\tau_{NSTi}: \overline{G}_I \cup \overline{G}_S \rightarrow \mathbb{N}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ *odhad počtu podnětů (stimulů) bran nutných pro řízení režimů činností samostatného systému x podle předpisu*

$$x \mapsto \begin{cases} \max_{\forall C} (\sum_{i=1}^n w_{SSeq}(h_{Ci}) \cdot |w_{SX}(h_{Ci})|) & \text{pokud } x \in \overline{G}_S \\ \max_{\forall C} (\sum_{i=1}^n w_{ISeq}(h_{Ci}) \cdot |w_{IX}(h_{Ci})|) & \text{pokud } x \in \overline{G}_I \\ 0 & \text{pokud } |V(x)| = 1 \end{cases}$$

kde C je orientovaná cesta v systému x a kde $\{h_{C1}, \dots, h_{Cn}\}$ je množina hran v cestě C

- $\overline{\tau}_{CBits}: \overline{G}_I \cup \overline{G}_S \rightarrow 2^{BIT_{CUA}}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ *množinu všech bran nutných k řízení režimů činností v systému x podle předpisu $x \mapsto System_CBits(x, \{\})$, kde $System_CBits$ je funkce provádějící činnost formálně popsanou v algoritmu 1, strana 101,*
- $\overline{\tau}_{Seq}: \overline{G}_I \cup \overline{G}_S \rightarrow 2^{BIT_{CUA}}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ *jeho sekvenční délku podle předpisu $x \mapsto System_Seq(x, \{\})$, kde $System_Seq$ je funkce provádějící činnost formálně popsanou v algoritmu 2, strana 102,*
- $\overline{\tau}_{NSTi}: \overline{G}_I \cup \overline{G}_S \rightarrow \mathbb{N}$ přiřazující systému $x \in \overline{G}_I \cup \overline{G}_S$ *odhad počtu všech podnětů (stimulů) pro řízení režimů činností v systému x podle předpisu $x \mapsto System_Sti(x, 0, \{\})$, kde $System_Sti$ je funkce provádějící činnost formálně popsanou v algoritmu 3, strana 102.*

□

Pomocí zobrazení τ_R, τ_L je možno získat množinu bran kořenu resp. bran listů daného systému, tj. množinu zdrojových resp. cílových bran datového přenosu s cestami reprezentovanými daným systémem. Zobrazení τ_{CBits}, τ_{Seq} a τ_{NSTi} přiřazují danému systému množinu řídicích bran, sekvenční délku a odhad počtu podnětů tak, že není brán ohled na zapojení tohoto systému (tj. obvodových prvků, jejichž brány jsou součástí tohoto systému) do struktury CUA . Snahou těchto zobrazení je ohodnotit pouze datovou část systému (tj. část, kterou prochází přenášená data) a stanovit, jaké jsou za předpokladu, že je zanedbáno zapojení daného systému do struktury CUA (tj. za předpokladu, je-li daný systém samostatný), podmínky a náklady tohoto přenosu. Samozřejmě s touto základní informací (platnou pouze pro samostatný systém) nevystačíme, chceme-li systém ohodnotit s ohledem na jeho zapojení ve struktuře CUA - takové ohodnocení však bude vycházet z dané základní informace.

U každého systému tedy lze konstatovat, že má část datovou, která slouží k přenosu požadovaných dat a část řídicí, která slouží k řízení tohoto datového přenosu. Přitom k řízení datového přenosu jsou použity jiné systémy, které mají opět část datovou a řídicí. Z tohoto pohledu každý datový přenos obecně závisí jak na řídicích signálech, tak na datovém toku obvodem. Vhodným příkladem zde může být přenos dat z primárních vstupů *CUA* na výstup dvoustupňové sčítačky - takový přenos je např. podmíněn jednak přítomností přenášených dat na jednom datovém vstupu sčítačky a jednak existencí vhodných dat na zbylém datovém vstupu sčítačky. Vyjmutím systému ze struktury *CUA* zůstává zachována informace o jeho datové části, ale informace o řídicí části je zjednodušena tak, že brány systému neúčastníci se datového přenosu jsou považovány za primární brány *CUA*.

Pro ohodnocení systému s ohledem na jeho zapojení ve struktuře *CUA* jsou definována zobrazení $\bar{\tau}_{CBits}$, $\bar{\tau}_{Seq}$ a $\bar{\tau}_{NSti}$. Tato zobrazení, narozdíl od zobrazení uvedených výše, předpokládají takové zapojení systému, jaké plyne ze struktury *CUA* a zohledňují tak nejen datovou, ale také řídicí část přenosu. Konstrukce těchto zobrazení však není triviální, jelikož vyžaduje analýzu grafové struktury *CUA* a proto je popsána samostatnými algoritmy¹⁴ uvedenými níže.

Jako první je uveden algoritmus *System_CBits*, pomocí něhož je ilustrován význam zobrazení $\bar{\tau}_{CBits}$. Vstupem algoritmu je brána/port/virtuální port x a množina A obsahující vstupní řídicí brány. Algoritmus vrací množinu bran, účastnících se přenosu dat na x , tj. množinu bran, které je třeba pro zajištění tohoto přenosu řídit, přičemž pro každé řízení je vždy volen systém s nejkratší sekvenční délkou mající nejmenší množinu řídicích bran (4. bod algoritmu). Algoritmus ošetřuje dva speciální případy x - prvním je případ (2. bod algoritmu), kdy x je tvořen pouze primárními branami *CUA* a druhým je případ (3. bod algoritmu), kdy přenos dat na x je podmíněn předchozím řízením x , tj. dochází k výskytu datové závislosti. V prvním případě je vrácena prázdná množina, protože pro řízení primárních bran není třeba řídit žádné jiné brány; v druhém případě je zjištěna datová závislost, nemá tedy smysl touto cestou pokračovat dále a je vrácena aktuální množina řídicích bran.

Algoritmus 1

Funkce *System_CBits*(x, A): vrací podmnožinu množiny BIT_{CUA}

1. [Aktualizace množiny řídicích bran]
 $A' := A \cup \tau_{CBits}(x) \cup \tau_R(x)$
2. [Kořenem systému x je primární brána *CUA*]
Vrať prázdnou množinu **pokud** $\tau_R(x) \subseteq BP_{CUA}$
3. [Některá z řídicích bran datově závisí na x]

¹⁴ tyto algoritmy slouží pouze k formálnímu a přehlednému vyjádření významu daných zobrazení, avšak pro vlastní výpočet budou použity jiné algoritmy - viz odstavec 5.2, strana 117

104 4 Model obvodu popsaného na úrovni meziregistrových přenosů

- Vrať A' pokud $\exists y \in \tau_{CBits}(x)$:**
 $\forall z \in \overline{G_S}[y]: System_CBits(z, \{\}) \cap A' \neq \emptyset$
 4. [Všechny řídicí brány jsou datově nezávislé na x]
Vrať $A' \cup \left(\bigcup_{y \in \tau_{CBits}(x)} C_y\right)$, kde $C_y = System_CBits(z, A')$ je
 nejmenší množina při $z \in \overline{G_S}[y] \wedge System_Seq(z, \{\}) =$
 $= \min_{z' \in \overline{G_S}[y]}(System_Seq(z', \{\}))$

□

Druhým algoritmem je algoritmus *System_Seq*. Pomocí něj je ilustrován význam zobrazení $\overline{\tau}_{Seq}$. Vstupem algoritmu je opět brána/port/virtuální port x a množina A obsahující vstupní řídicí brány. Algoritmus vrací celé číslo, určující nejkratší sekvenční délku systému pro přenos dat na x (4. bod algoritmu). Algoritmus ošetřuje tytéž speciální případy x jako v případě algoritmu *System_CBits*. V prvním případě je vráceno číslo 0, protože pro řízení primárních bran není třeba žádné mezihodnoty - naopak, na primární brány lze přikládat požadovaná data přímo; v druhém případě je zjištěna datová závislost, nemá tedy smysl touto cestou pokračovat dále či zkoumat sekvenční vlastnosti této části obvodu a je vrácena hodnota ∞ .

Algoritmus 2

- Funkce *System_Seq*(x, A):** vrací číslo z množiny \mathbb{N}
1. [Aktualizace množiny řídicích bran]
 $A' := A \cup \tau_{CBits}(x) \cup \tau_R(x)$
 2. [Kořenem systému x je primární brána CUA]
Vrať číslo 0 pokud $\tau_R(x) \subseteq BPCUA$
 3. [Některá z řídicích bran datově závisí na x]
**Vrať ∞ pokud $\exists y \in \tau_{CBits}(x): \forall z \in \overline{G_S}[y]:$
 $System_CBits(z, \{\}) \cap A' \neq \emptyset$**
 4. [Všechny řídicí brány jsou datově nezávislé na x]
Vrať $\tau_{Seq}(x) + \sum_{y \in \tau_{CBits}(x)} \min_{z \in \overline{G_S}[y]}(System_Seq(z, A'))$

□

Třetím a posledním algoritmem je algoritmus *System_Sti* ilustrující význam zobrazení $\overline{\tau}_{NSti}$. Vstupem algoritmu je brána/port/virtuální port x , číslo k určující vstupní počet podnětů a množina A obsahující vstupní řídicí brány. Algoritmus vrací celé číslo, určující odhad nejmenšího počtu všech podnětů nutných pro přenos dat na x (4. bod algoritmu). Jako v předchozích algoritmech, i zde jsou ošetřeny dva speciální případy x . V prvním případě je vráceno číslo 0, protože pro řízení primárních bran není třeba žádného dalšího podnětu, než podnětu pro přiložení dat na primární bránu; v druhém případě je zjištěna datová závislost, nemá tedy smysl touto cestou pokračovat dále a je vrácen aktuálně zjištěný počet podnětů.

Algoritmus 3

- Funkce *System_Sti*(x, k, A):** vrací číslo z množiny \mathbb{N}

1. [Aktualizace počtu podnětů řídicích bran]

$$k' := k + \tau_{NSti}(x)$$

$$A' := A \cup \tau_{CBits}(x) \cup \tau_R(x)$$
2. [Kořenem systému x je primární brána CUA]
 Vrať číslo 0 **pokud** $\tau_R(x) \subseteq BP_{CUA}$
3. [Některá z řídicích bran datově závisí na x]
 Vrať k' **pokud** $\exists y \in \tau_{CBits}(x): \forall z \in \overline{G_S}[y]:$
 $System_CBits(z, \{\}) \cap A' \neq \emptyset$
4. [Všechny řídicí brány jsou datově nezávislé na x]
 Vrať $k' + \sum_{y \in \tau_{CBits}(x)} k_y$, kde $k_y = System_Sti(z, 0, A')$ je pro dané y nejmenší číslo při $z \in \overline{G_S}[y] \wedge System_Seq(z, \{\}) =$
 $= \min_{z' \in \overline{G_S}[y]} (System_Seq(z', \{\}))$

□

V dalším textu bude zavedeno zobrazení v_{CBits} , přiřazující obvodu CUA množinu bran ($\subseteq B_{CUA} \cup BP_{CUA}$), z nichž každá se účastní řízení v některém systému $x \in \overline{G_S} \cup \overline{G_I}$, dále zobrazení v_{Seq} , přiřazující obvodu CUA délku nejdelší posloupnosti nutné pro testování brány, portu či virtuálního portu v CUA a konečně zobrazení v_{NSti} , přiřazující obvodu CUA odhad největšího počtu podnětů pro testování brány/portu/virtuálního portu v CUA .

Definice 23:

Buďte definována zobrazení

- $v_{CBits}: \{cir_{CUA}\} \rightarrow 2^{BIT_{CUA}}$ přiřazující CUA množinu bran pro řízení režimů činnosti obvodových prvků obvodu CUA podle předpisu $cir_{CUA} \mapsto \bigcup_{y \in \overline{G_I} \cup \overline{G_S}} \overline{\tau}_{CBits}(y)$,
- $v_{Seq}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující CUA největší sekvenční délku pro testování brány v CUA podle předpisu $cir_{CUA} \mapsto (\max_{y \in \overline{G_I}} \overline{\tau}_{Seq}(y) + \max_{y \in \overline{G_S}} \overline{\tau}_{Seq}(y))$ a
- $v_{NSti}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující CUA odhad největšího počtu podnětů (stimulů) pro testování brány v CUA podle předpisu $cir_{CUA} \mapsto (\max_{y \in \overline{G_I}} \overline{\tau}_{NSti}(y) + \max_{y \in \overline{G_S}} \overline{\tau}_{NSti}(y))$

□

Výše uvedená zobrazení přesně vystihují informaci, která je myšlena, avšak konstrukce těchto zobrazení, založená na detailní analýze obvodové struktury CUA s použitím $\overline{\tau}_{CBits}$, $\overline{\tau}_{Seq}$ a $\overline{\tau}_{NSti}$, je poměrně složitá; při návrhu metody a algoritmu pro analýzu testovatelnosti tedy nebude požadována přímo konstrukce těchto zobrazení, ale na místo toho bude použita konstrukce takových zobrazení, která jsou jejich odhadem, nezpůsobujícím zkreslení informace o testovatelnosti obvodu CUA .

106 4 Model obvodu popsaného na úrovni meziregistrových přenosů

Definice 24:

Pro odhad zobrazení z definice 23 buďte definována zobrazení

- $\bar{v}_{CBits}: \{cir_{CUA}\} \rightarrow 2^{BIT_{CUA}}$ předpisem $cir_{CUA} \mapsto \bigcup_{(x,y,k,X) \in R_{TAS} \cup R_{TAI}} X$,
- $\bar{v}_{Seq}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ předpisem
 $cir_{CUA} \mapsto \sum_{(x,y) \in M_S \cup M_I, X \in 2^{BIT_{CUA}}} \left(\max_{(x',y',k,X) \in R_{TAS} \cup R_{TAI}, x'=x, y'=y} k \right)$
a
- $\bar{v}_{NSti}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ podle předpisu
 $cir_{CUA} \mapsto \sum_{(x,y) \in M_S \cup M_I, X \in 2^{BIT_{CUA}}} \left(\max_{(x',y',k,X) \in R_{TAS} \cup R_{TAI}, x'=x, y'=y} k \cdot |X| \right)$

□

Zobrazení v_{CBits} přiřazuje CUA pouze ty brány, které lze vzhledem ke struktuře CUA využít k řízení přenosu diagnostických dat, přičemž cesty, kterými tato data budou procházet, jsou určeny systémy transparentních cest. Odhadem tohoto zobrazení bude zobrazení \bar{v}_{CBits} , které nevychází z předchozí detailní analýzy systémů získaných z grafů G_S , G_I a které za bránu, která se účastní přenosu diagnostických dat považuje každou bránu, kterou lze použít k řízení režimu některého z obvodových prvků v CUA , přičemž se již nezkoumá, zda takový režim lze s ohledem na strukturu CUA u daného prvku nastavit či nikoliv.

Zbylá zobrazení, tj. \bar{v}_{Seq} a \bar{v}_{NSti} , také nevycházejí z detailní analýzy grafů G_S , G_I ; jejich pesimistický odhad zobrazení v_{Seq} a v_{NSti} vychází z předpokladu, že nejdelší cesta v grafu, tj. sekvenčně nejdelší cesta pro řízení resp. pozorování brány v CUA (tj. cesta v G_S resp. G_I) je cesta tvořená posloupností všech uzlů G_S resp. G_I , přičemž inciduje-li s dvěma uzly více hran řízených toutéž množinou bran, je z nich vždy vybrána hrana, která je ohodnocena největší sekvenční délkou.

4.3 Shrnutí

Cílem této kapitoly bylo rozšířit výchozí model [Růž02] založený na tzv. koncepci I-cest, o prostředky, pomocí kterých je možno popsat v rámci této práce navrženou metodu pro analýzu testovatelnosti číslicového obvodu popsaného na úrovni meziregistrových přenosů; rozšíření výchozího modelu tedy v žádném případě nebylo samoúčelným, ale bylo vyvoláno potřebou modelovat další skutečnosti týkající se struktury obvodu, jeho podčástí a vlastností. Za nejdůležitější rozšíření výchozího modelu lze považovat následující:

- **modelování bitových složek portů a spojů, rozlišení bran portů:** cílem tohoto rozšíření bylo dosáhnout stavu, umožňujícího u každého portu a každého spoje rozlišit jejich bitové složky zejména za účelem zpřesnění výsledků analýzy datového toku pro přenos diagnostických dat. Dalším důvodem pro zjemnění modelu portů níže, tj. na úroveň rozlišení jejich

bitových složek (bran), je snaha neprohlašovat celou šířku datového toku za špatně testovatelnou, je-li špatně testovatelná pouze její část, ale přesně identifikovat konkrétní z hlediska testovatelnosti problematickou část datového toku včetně bran, které jsou zdrojem špatné testovatelnosti,

- **modelování virtuálních portů:** virtuálním portem je myšlen ”port”, který není návrhářem explicitně definován jako součást rozhraní daného obvodového prvku; virtuální port může být složen z bran rozhraní daného prvku a vytvářet tak nový, avšak do návrhového rozhraní nepatřící ”port”. Důvodem pro zavedení koncepce virtuálních portů je usnadnění definice a popisu transparentních cest vhodných pro přenos diagnostických dat, tj. testovacích vzorků a odezev strukturou konkrétního obvodového prvku,
- **zobecnění modelu transparentních režimů:** s pomocí virtuálních portů je možno poměrně jednoduše a přehledně popsat režimy činnosti, během nichž je konkrétní obvodový prvek schopen přenést testovací vzorek resp. odezvu svou strukturou, tzn. pro každý obvodový prvek je možno vyjádřit, mezi kterými jeho branami, za jakých podmínek a pomocí jaké transformace je možno přenést daná diagnostická data jeho vnitřní strukturou. Podstata rozšíření oproti výchozímu modelu pak spočívá v tom, že pro přenos diagnostických dat není vyžadována přísná podmínka ve formě I-režimu¹⁵, ale že tato podmínka je rozštěpena na podmínku pro přenos testovacích vzorků¹⁶ strukturou daného obvodového prvku. Zvlášť jsou zkoumány transparentní cesty a podmínky pro přenos testovacích vzorků a zvlášť cesty a podmínky pro přenos odezev. Kromě toho, že není vyžadována bijekce, ale postačuje surjekce či injekce, lze popsat transparentní cesty nejen mezi v návrhu explicitně uvedenými porty tvořícími návrhové rozhraní daného prvku, ale zejména mezi nově zavedenými virtuálními porty. Tím je umožněno popsat mnohem více transparentních cest využitelných pro přenos diagnostických dat, přesněji analyzovat datové cesty daného obvodu s nadějí v získání přesnější informace o testovatelnosti obvodu,
- **zobecnění modelu transparentních cest:** základem pro obecnější pojetí transparentních cest (oproti koncepci I-cest) je jednak informace o struktuře obvodu a jednak informace o transparentních režimech příslušných obvodových prvků. Na základě informace o možnostech přenosu testovacích vzorků resp. odezev mezi konkrétními virtuálními porty daného obvodového prvku a informace o propojení bran z rozhraní obvodových prvků spojí je možno zkonstruovat graf datového toku testovacích vzorků a graf datového toku odezev. Analýzou struktury těchto grafů je možno nalézt systémy transparentních cest pro nastavení testovacích vzorků na

¹⁵ předpokládající přenos, při kterém existuje identická bijekce mezi daty na vstupním m -bitovém portu a daty na výstupním m -bitovém portu daného prvku

¹⁶ předpokládající existenci ”pouhé” surjekce z dat na vstupním virtuálním portu na data na výstupním virtuálním portu a na podmínku pro přenos odezev předpokládající existenci ”pouhé” injekce z dat ze vstupního virtuálního portu na data na výstupním virtuálním portu

108 4 Model obvodu popsaného na úrovni meziregistrových přenosů

virtuální port resp. pro sledování odezev z daného virtuálního portu a na základě znalostí o těchto systémech hodnotit testovatelnost daného obvodu.

5

Metoda analýzy testovatelnosti

Máme-li k dispozici model vybavený prostředky pro popis požadovaných návrhových a diagnostických vlastností číslicového obvodu na úrovni meziregistrových přenosů, můžeme těchto prostředků využít nejprve k popisu vztahů pro ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti (viz podkapitola 5.1) a posléze také k zápisu algoritmů (viz podkapitola 5.2), provádějících ohodnocení vycházející z těchto vztahů.

5.1 Vztahy pro ohodnocení říditelnosti a pozorovatelnosti

Vztahy pro ohodnocení říditelnosti a pozorovatelnosti, jak jsou pojaty v této práci, lze rozdělit do dvou skupin, a to na vztahy pro lokální ohodnocení¹ a na vztahy pro globální ohodnocení² [Str02b, JS02a, Str03b]. Podrobnější komentář k těmto vztahům i komentář ke způsobu a účelu ohodnocení bude ještě uveden v následujícím textu. Na tomto místě zatím uvedme, že za hodnotu sloužící k ohodnocení dané vlastnosti je považováno reálné číslo z uzavřeného intervalu $< 0; 1 >$, přičemž číslo 0 vyjadřuje absenci této vlastnosti³ a číslo 1 výskyt této vlastnosti v její nejlepším možném stupni⁴.

¹ tj. ohodnocení konkrétních systémů transparentních cest, dále bran, portů a virtuálních portů

² tj. ohodnocení obvodu *CUA* několika ukazateli jeho testovatelnosti s možností souhrnného ohodnocení testovatelnosti *CUA* jedinou číselnou hodnotou

³ vyjadřuje např. neříditelnost resp. nepozorovatelnost daného uzlu

⁴ vyjadřuje např. bezprostřední říditelnost resp. pozorovatelnost uzlu pomocí primárních vývodů obvodu

5.1.1 Vztahy pro lokální ohodnocení

Ohodnocení systémů transparentních cest

Aby bylo možno ohodnotit brány, porty a virtuální porty, je třeba nejprve ohodnotit systémy transparentních cest, tj. systémy, po nichž je možno mezi danými branami, porty resp. virtuálními porty a vývody obvodu přenášet diagnostická data.

Definice 25:

Definujeme zobrazení $\tau_{CO}: \overline{G}_S \cup \overline{G}_I \rightarrow \mathbb{R}_{<0,1>}$ přiřazující systému $x \in \overline{G}_S \cup \overline{G}_I$ reálné číslo vyjadřující (relativně ke strukturálním vlastnostem CUA) *snadnost konstrukce systému x podle předpisu*

$$x \mapsto \begin{cases} A \cdot B \cdot C & \text{pokud } \overline{\tau}_{Seq}(x) \neq \infty \\ 0 & \text{jinak.} \end{cases}$$

kde

$$A = \left(1 - \frac{\tau_{Seq}(x)}{v_{Seq}(cir_{CUA})+1}\right),$$

$$B = \left(1 - \frac{\tau_{NSti}(x)}{v_{NSti}(cir_{CUA})+1}\right),$$

$$C = \left(\prod_{y \in \tau_{CBits}(x)} \tau_{CO}(G_S[y])\right).$$

□

Ze vztahu pro výpočet τ_{CO} dle definice 25 je patrné, že hodnota $\tau_{CO}(x)$ je dána

- sekvenční délkou $\tau_{Seq}(x)$ samostatného systému x ,
- odhadem počtu $\tau_{NSti}(x)$ podnětů bran nutných pro řízení režimů činnosti samostatného systému x ,
- součinem $\prod_{y \in \tau_{CBits}(x)} \tau_{CO}(G_S[y])$ hodnot τ_{CO} každé z bran y nutných k řízení režimů činností obvodových prvků v samostatném systému x .

V úvodu této podkapitoly bylo zmíněno, že ohodnocením vlastnosti v této práci bude myšleno reálné číslo z intervalu $< 0; 1 >$. Následující text se bude věnovat třem speciálním případům, které mohou nastat při ohodnocení τ_{CO} .

První případ: systém x je ohodnocen hodnotou 0 pokud je splněna alespoň jedna z následujících podmínek:

- systém x obsahuje takovou bránu y , že řízení každého jí příslušejícího systému $z \in \overline{G}_S[y]$ je podmíněno předchozím řízením y . Jinými slovy je zjištěno, že existuje brána y , jejíž řízení je podmíněno předchozím nastavením této brány na jistou hodnotu. Tato situace je důsledkem existence nerozbité zpětnovazební smyčky, tj. smyčky, jejíž existence je z pohledu diagnostiky nežádoucí a jejíž testovatelnost je třeba zlepšit,

5.1 Vztahy pro ohodnocení říditelnosti a pozorovatelnosti 111

- systém x obsahuje neovladatelnou řídicí bránu y , tj. bránu, pro jejíž každý systém $z \in \overline{G}_S[y]$ platí $\tau_{CO}(z) = 0$. Neovladatelnost řídicí brány y může být způsobena buď existencí datové závislosti (viz předchozí bod) nebo nemožností ovládat data na bráně y v důsledku neexistence datového toku pro její řízení, kde tato neexistence je způsobena nedostatečnými transparentními vlastnostmi obvodových prvků vyskytujících se na cestách pro řízení y .

Druhý případ: systém x ohodnocen číslem 1, pokud současně platí:

- $\tau_{Seq}(x) = 0$, tj. je-li sekvenční délka samostatného systému x rovna 0, což platí je-li splněna některá z následujících podmínek:
 - systém x je tvořen právě jedním uzlem, který je primárním virtuálním portem CUA (pak totiž $|V(x)| = 1$ a $E(X) = \emptyset$, z čehož dle definice 22 plyne $\tau_{Seq}(x) = 0$),
 - systém x je bichromatický při takovém obarvení dvěma barvami b_1, b_2 , že každý jeho uzel, který je primárním virtuálním portem CUA lze obarvit barvou b_1 a každý zbývající uzel barvou b_2 . Pak každá hrana v systému x spojuje (bez ohledu na orientaci) uzel, který je primárním virtuálním portem CUA s uzlem, který není primárním virtuálním portem CUA . V takovém případě dle definice 20 pro každou hranu h v systému x platí $w_{SSeq}(h) = 0$ pokud $x \in \overline{G}_S$ resp. $w_{ISeq}(h) = 0$ pokud $x \in \overline{G}_I$,
- $\tau_{NSti}(x) = 0$, tj. je-li odhad počtu podnětů bran nutných pro řízení režimů činnosti samostatného systému x roven 0, což platí za podmínek analogických $\tau_{Seq}(x) = 0$,
- $\forall y \in \tau_{CBits} : \tau_{CO}(G_S[y]) = 1$, tj. - s odkazem na komentář k předchozím dvěma podmínkám - pro každou řídicí bránu b systému x platí, že b je buď primární branou CUA nebo v systému x existuje hrana, která s branou b a s některou z primárních bran CUA inciduje.

Třetí případ: není-li systém ohodnocen číslem 0 ani číslem 1, pak je zřejmě ohodnocen reálným číslem z otevřeného intervalu $(0; 1)$. Přitom platí, že čím náročnější je zajištění datového toku systémem x , tím blíže bude hodnota číslu 0 a naopak - čím snazší je zajištění datového toku systémem x , tím blíže bude hodnota číslu 1, avšak stále platí, že systém x bude ohodnocen hodnotou 0 resp. 1 pouze v dvou prvních speciálních případech.

Jsou-li ohodnoceny všechny systémy, lze přistoupit k ohodnocení virtuálních portů, bran a portů obvodových prvků. Nejprve budou definovány vztahy pro ohodnocení virtuálních portů.

Ohodnocení virtuálních portů

Předtím než budou představeny vztahy pro ohodnocení virtuálních portů, je třeba si uvědomit, že jednomu virtuálnímu portu x může příslušet 0, ale i více systémů transparentních cest, tj. obecně platí $|\overline{G}_S[x]| \geq 0$ resp. $|\overline{G}_I[x]| \geq 0$.

Platí-li $|\overline{G}_S[x]| > 1$ resp. $|\overline{G}_I[x]| > 1$, pak existuje několik způsobů přenosu (tj. několik systémů transparentních cest) testovacích vzorků z primárních vstupů CUA na x resp. odezev x na primární výstupy CUA . V takovém případě je nutno posoudit, který ze systémů z $\overline{G}_S[x]$ resp. systémů z $\overline{G}_I[x]$ představuje nejlepší způsob pro přepravu testovacích vzorků resp. odezev na resp. z x . S ohledem na zavedené ohodnocení je tedy pro každý virtuální port x hledán systém $y \in \overline{G}_S[x]$ a $\overline{G}_I[x]$ s maximální hodnotou, tj. hodnotou nejvíce se blížíci hodnotě 1. V následující definici je tedy hodnotou z intervalu $< 0; 1 >$ ohodnocena vlastnost ”snadnost ovládání hodnot na portu x hodnotami z vývodů CUA ” resp. vlastnost ”snadnost pozorování hodnot z portu x na vývodech CUA ”.

Definice 26:

Definujme zobrazení $\nu_C: VPORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující virtuálnímu portu $x \in VPORT_{CUA}$ reálné číslo vyjadřující (relativně ke strukturálním vlastnostem CUA) *snadnost ovládání hodnot na virtuálním portu x hodnotami z vývodů CUA podle předpisu $x \mapsto \max_{y \in \overline{G}_S[x]}(\tau_{CO}(y))$*

a zobrazení $\nu_O: VPORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující virtuálnímu portu $x \in VPORT_{CUA}$ reálné číslo vyjadřující (relativně ke strukturálním vlastnostem CUA) *snadnost pozorování hodnot z virtuálního portu x na vývodech CUA podle předpisu $x \mapsto \max_{y \in \overline{G}_I[x]}(\tau_{CO}(y))$.*

Každý virtuální port x , pro který platí $\nu_C(x) > 0$ resp. $\nu_O(x) > 0$ bude nazván říditelným resp. pozorovatelným virtuálním portem CUA .

□

Ohodnocení bran z rozhraní obvodových prvků

Na základě ohodnocení vlastností virtuálních portů je možno pro každou bránu x ohodnotit vlastnost ”snadnost ovládání hodnot na bráně x hodnotami z vývodů CUA ” resp. vlastnost ”snadnost pozorování hodnot z brány x na vývodech CUA ”. Jelikož obecně může být brána x součástí několika virtuálních portů, tak je třeba zjistit, pomocí kterého z těchto virtuálních portů lze nejnádhněji ovládat hodnoty na bráně x resp. sledovat hodnoty z brány x ; hodnota příslušné vlastnosti brány x je pak rovna maximální hodnotě příslušné vlastnosti virtuálního portu obsahujícího bránu x .

Definice 27:

Definujme zobrazení $\beta_C: B_{CUA} \cup BP_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující bráně $x \in B_{CUA} \cup BP_{CUA}$ reálné číslo vyjadřující (relativně ke strukturálním vlastnostem CUA) *snadnost ovládání hodnot na bráně x hodnotami z vývodů CUA podle předpisu $x \mapsto \max_{y \in VPORT_{CUA}, x \in \pi_{BITS}(y)}(\nu_C(y))$* a zobrazení $\beta_O: B_{CUA} \cup BP_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující bráně $x \in B_{CUA} \cup BP_{CUA}$

5.1 Vztahy pro ohodnocení říditelnosti a pozorovatelnosti 113

reálné číslo vyjadřující (relativně ke strukturálním vlastnostem CUA) *snadnost pozorování hodnot z brány* x na vývodech CUA podle předpisu $x \mapsto \max_{y \in VPORT_{CUA}, x \in \pi_{BITS}(y)} (\nu_O(y))$.

Každá brána x , pro kterou platí $\beta_C(x) > 0$ resp. $\beta_O(x) > 0$ bude nazvána *řiditelnou* resp. *pozorovatelnou branou* CUA .

□

Ohodnocení portů obvodových prvků

V posledním kroku je možno ohodnotit vlastnosti portů obvodových prvků. V následujících třech definicích budou postupně uvedeny vztahy pro ohodnocení říditelnosti portu, vztahy pro ohodnocení pozorovatelnosti portu a na jejich základě vybudovaný vztah pro ohodnocení testovatelnosti portu. Tyto vztahy pak budou základem pro ohodnocování diagnostických vlastností obvodu metodou analýzy testovatelnosti popsanou v podkapitole 5.2.

Dříve, než budou v následující definici zavedeny vztahy pro ohodnocení říditelnosti portů obvodových prvků a pro určení počtu říditelných bran v každém z těchto portů, bude uveden komentář, který by měl usnadnit pochopení po něm následující definice.

Ohodnocení říditelnosti portu $x \in PORT_{CUA}$ vychází z toho, že testovací vzorky mohou být na tento port či na jeho dílčí části (každou dílčí část je možno reprezentovat virtuálním portem obsahujícím alespoň jednu bránu portu x), přeneseny obecně několika způsoby, a to včetně vzájemných kombinací těchto způsobů přenosu. Každý ze způsobů přenosu lze vyjádřit pomocí konkrétního systému transparentních cest (z množiny \overline{G}_S) příslušejícího danému virtuálnímu portu. Každá z množin $Y_{Si} \in Y_S(x)$ v následující definici je množinou⁵ říditelných virtuálních portů, jejichž systémů transparentních cest lze využít k řízení bran portu x . Snahou pak je pro každý port $x \in PORT_{CUA}$ nalézt takovou množinu Y_{Si} říditelných virtuálních portů, pomocí nichž je možno dosáhnout nejlepší říditelnosti co největšího počtu bran portu x . Pro tento účel je a) stanoven největší počet bran ($y_{Smax}(x)$) portu x , které je možno pomocí systémů transparentních cest řídit a b) definován systém $Y_{Smax}(x) \subseteq Y_S(x)$ množin virtuálních portů takový, že každá množina Y_{Si} z tohoto systému obsahuje virtuální porty, jejichž systémy transparentních datových cest jsou schopny řídit právě $y_{Smax}(x)$ bran portu x . Dále jsou pro přehlednění výsledného vztahu definována pomocná zobrazení $\pi_{WBITS}(X, y)$ resp. $\pi_{NWBITS}(X, y)$, která dané množině virtuálních portů X a virtuálnímu portu y přiřazují množinu bran portu x obsažených ve virtuálních portech z X resp. součet četností bran portu x ve virtuálních portech z X . Počet $\pi_{NC}(x)$ říditelných bran portu x je pak roven hodnotě $y_{Smax}(x)$ a hodnota

⁵ každá taková množina pak obsahuje virtuální porty vybavené alespoň jednou branou portu x , jejichž množiny bran jsou buď vzájemně disjunktní nebo jejichž množiny bran omezené na množinu bran x nejsou podmnožinami a vzájemně se liší alespoň v jedné bráně portu x

114 5 Metoda analýzy testovatelnosti

$\pi_C(x)$ říditelnosti portu x je dána nejlépe ohodnocenou množinou v systému $Y_{Smax}(x)$.

Definice 28:

Nechť

- $x \in PORT_{CUA}$,
- $x \odot y$ je zkráceným zápisem pro množinu vzniklou operací $\pi_{BITS}(x) \cap \pi_{BITS}(y)$, kde $x, y \in ORBIT_{CUA}$,
- $Y_S(x)$ je systém $\{Y_{S1}, \dots, Y_{Sn}\}$ neprázdných podmnožin množiny $VPORT_{CUA}$ takový, že $\forall i \in \{1, \dots, n\}: \forall y \in Y_{Si}: y \odot x \neq \emptyset, \nu_C(y) > 0 \wedge y_1, y_2 \in Y_{Si} \Leftrightarrow y_1 \odot y_2 = \emptyset \vee [((y_1 \odot x) \setminus (y_2 \odot x)) \cup ((y_2 \odot x) \setminus (y_1 \odot x)) \neq \emptyset \wedge (y_1 \odot x \not\subseteq \pi_{BITS}(y_2)) \wedge (y_2 \odot x \not\subseteq \pi_{BITS}(y_1))]$
- $y_{Smax}(x) = \max_{i=1}^n |\bigcup_{y \in Y_{Si}} (x \odot y)|$,
- $Y_{Smax}(x) = \{Y \mid Y \in Y_S(x), |\bigcup_{y \in Y_{Si}} x \odot y| = y_{Smax}(x)\}$,
- $\pi_{WBITS}: 2^{VPORT_{CUA}} \times VPORT_{CUA} \rightarrow 2^{VPORT_{CUA}}$ zobrazení definované předpisem $\pi_{WBITS}(X, y) = \bigcup_{x \in X} x \odot y$,
- $\pi_{NWBITS}: 2^{VPORT_{CUA}} \times VPORT_{CUA} \rightarrow \mathbb{N}$ zobrazení definované předpisem $\pi_{NWBITS}(X, y) = \sum_{x \in X} |x \odot y|$.

Pak definujeme zobrazení

- $\pi_{NC}: PORT_{CUA} \rightarrow \mathbb{N}$ přiřazující portu $x \in PORT_{CUA}$ počet říditelných bran portu podle předpisu $x \mapsto y_{Smax}(x)$,
- $\pi_C: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující portu $x \in PORT_{CUA}$ reálné číslo (hodnotu říditelnosti portu) vyjadřující, relativně ke strukturálním vlastnostem CUA , snadnost ovládní hodnot na portu x hodnotami z primárních vstupů obvodu podle předpisu $x \mapsto \max_{Y \in Y_{Smax}(x)} \frac{\sum_{y \in Y} \nu_C(y) \cdot |x \odot y|}{(\pi_{NWBITS}(Y, x) + |\pi_{BITS}(x) \setminus \pi_{WBITS}(Y, x)|)}$.

□

Je zřejmé, že při konstrukci uvedených zobrazení bude obecně časově nejnáročnější konstrukce systému množin $Y_S(x)$, která bude nutně vyžadovat analýzu všech podmnožin množiny V virtuálních portů majících s portem x společnou alespoň jednu bránu, tj. analýzu $2^{|V|}$ podmnožin množiny V . Bude tedy výhodnější - a to zejména pro "velké množiny V " - pokud výše uvedenou definici zjednodušíme a nahradíme ji následující definicí.

Definice 28:

Definujeme zobrazení

- $\pi_{NC}: PORT_{CUA} \rightarrow \mathbb{N}$ přiřazující portu $x \in PORT_{CUA}$ počet říditelných bran portu podle předpisu $x \mapsto |\{y \mid y \in \pi_{BITS}(x) \wedge \beta_C(y) > 0\}|$,

5.1 Vztahy pro ohodnocení říditelnosti a pozorovatelnosti 115

- $\pi_C: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující portu $x \in PORT_{CUA}$ reálné číslo (hodnotu říditelnosti portu) vyjadřující, relativně ke strukturálním vlastnostem CUA , snadnost ovládní hodnot na portu x hodnotami z primárních vstupů obvodu podle předpisu

$$x \mapsto \frac{\sum_{y \in \pi_{BITS}(x)} \beta_C(y)}{\pi_W(x)}.$$

□

Druhá z definic se zabývá způsobem ohodnocení pozorovatelnosti portů obvodových prvků. Analogicky k ohodnocení říditelnosti je pro port $x \in PORT_{CUA}$ stanoven systém $Y_I(x)$ množin pozorovatelných virtuálních portů, jejichž systémy transparentních cest (z množiny \overline{G}_I) lze využít pro pozorování odezev z portu x . Množina $y_{I_{max}}$, systém množin $Y_{I_{max}}(x) \subseteq Y_I(x)$ a pomocná zobrazení π_{WBITS} , π_{NWBITS} plní účel analogický účelu množiny $y_{S_{max}}$, systému množin $Y_{S_{max}}(x) \subseteq Y_S(x)$ a pomocných zobrazení π_{WBITS} , π_{NWBITS} z předchozí definice. Počet $\pi_{NO}(x)$ pozorovatelných bran portu x je roven hodnotě $y_{I_{max}}(x)$ a hodnota $\pi_O(x)$ pozorovatelnosti portu x je dána nejlépe ohodnocenou množinou v systému $Y_{I_{max}}(x)$.

Definice 29:

Nechť

- $x \in PORT_{CUA}$,
- $x \odot y$ je zkráceným zápisem pro množinu vzniklou operací $\pi_{BITS}(x) \cap \pi_{BITS}(y)$, kde $x, y \in ORBIT_{CUA}$,
- $Y_I(x)$ je systém $\{Y_{I_1}, \dots, Y_{I_m}\}$ neprázdných podmnožin množiny $VPORT_{CUA}$ takový, že $\forall i \in \{1, \dots, m\}: \forall y \in Y_{I_i}: \neg \exists y_1, \dots, y_k \in Y_{I_i}, y \neq y_j, j = 1, \dots, k: x \odot y \subseteq \bigcup_{i=1}^k \pi_{BITS}(y_i)$,
- $y_{I_{max}}(x) = \max_{i=1}^m |\bigcup_{y \in Y_{I_i}} x \odot y|$,
- $Y_{I_{max}}(x) = \{Y \mid Y \in Y_I(x), |\bigcup_{y \in Y_{I_i}} x \odot y| = y_{I_{max}}(x)\}$,
- $\pi_{WBITS}: 2^{VPORT_{CUA}} \times VPORT_{CUA} \rightarrow 2^{VPORT_{CUA}}$ zobrazení definované předpisem

$$\pi_{WBITS}(X, y) = \bigcup_{x \in X} x \odot y,$$
- $\pi_{NWBITS}: 2^{VPORT_{CUA}} \times VPORT_{CUA} \rightarrow \mathbb{N}$ zobrazení definované předpisem

$$\pi_{NWBITS}(X, y) = \sum_{x \in X} |x \odot y|.$$

Pak definujeme zobrazení

- $\pi_{NO}: PORT_{CUA} \rightarrow \mathbb{N}$ přiřazující portu $x \in PORT_{CUA}$ počet pozorovatelných bran portu podle předpisu $x \mapsto y_{I_{max}}(x)$,
- $\pi_O: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující portu $x \in PORT_{CUA}$ reálné číslo (hodnotu pozorovatelnosti portu) vyjadřující, relativně ke strukturálním vlastnostem CUA , snadnost pozorování hodnot z portu x na primárních výstupech obvodu podle předpisu

$$x \mapsto \max_{Y \in Y_{I_{max}}(x)} \frac{\sum_{y \in Y} \nu_O(y) \cdot |x \odot y|}{(\pi_{NWBITS}(Y, x) + |\pi_{BITS}(x) \setminus \pi_{WBITS}(Y, x)|)}.$$

□

Obdobně jako u definice zobrazení pro ohodnocení říditelnosti portů je i u výše uvedené definice problematická konstrukce systému množin - v tomto případě $Y_I(x)$. Také tuto definici je tedy možné zjednodušit a nahradit ji následující.

Definice 29:

Definujme zobrazení

- $\pi_{NO}: PORT_{CUA} \rightarrow \mathbb{N}$ přiřazující portu $x \in PORT_{CUA}$ počet pozorovatelných bran portu podle předpisu $x \mapsto |\{y \mid y \in \pi_{BITS}(x) \wedge \beta_O(y) > 0\}|$,
- $\pi_O: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující portu $x \in PORT_{CUA}$ reálné číslo (hodnotu pozorovatelnosti portu) vyjadřující, relativně ke strukturálním vlastnostem CUA , snadnost pozorování hodnot z portu x na primárních výstupech obvodu podle předpisu
$$x \mapsto \frac{\sum_{y \in \pi_{BITS}(x)} \beta_O(y)}{\pi_W(x)}.$$

□

Pomocí výše uvedených definic jsou tedy zavedeny prostředky, umožňující pro každý port $x \in PORT_{CUA}$ zjistit jak počet a množinu říditelných resp. pozorovatelných bran, tak číselné ohodnocení obtížnosti jejich řízení resp. pozorování. Aby bylo možno jednoduše porovnávat testovatelnost konkrétních portů, je výhodné, bude-li zavedena funkce ($\pi_T: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$) přiřazující každému portu x jediné číslo vyjadřující snadnost testovatelnosti daného portu. Je zřejmé, že pro tento účel nemůže být použita libovolná funkce, ale pouze funkce, která splňuje zejména následující požadavky:

- za předpokladu, že port x je testovatelný právě když je současně říditelný (tj. pokud je možno zajistit na jeho branách výskyt požadovaného testovacího vzorku) i pozorovatelný (tj. pokud je možno sledovat odezvu vyskytující se na jeho branách) je požadováno, aby port x byl považován za netestovatelný⁶ (tj. $\pi_T(x) = 0$), je-li splněna alespoň jedna z následujících podmínek:
 - neexistuje brána portu x , která je říditelná (tj. na port x není možno nastavit testovací vzorek ani jeho dílčí část),
 - neexistuje brána portu x , která je pozorovatelná (tj. není možno sledovat odezvu z žádné z bran portu x),

⁶ port je tedy považován za testovatelný, pokud obsahuje alespoň jednu bránu, která je říditelná a alespoň jednu bránu, která je pozorovatelná. V případě, že se jedná o port složený z několika bran je zřejmé, že řízení resp. pozorování pouhé podmnožiny bran daného portu k jeho testování nepostačí - protože však je říditelná a pozorovatelná jistá část portu, je tato situace chápána jako "částečná testovatelnost" portu a nikoliv jako jeho netestovatelnost

5.1 Vztahy pro ohodnocení říditelnosti a pozorovatelnosti 117

- testovatelný port x_1 , pro jehož testování (tj. jeho řízení a pozorování) je třeba generovat řídicí posloupnost o délce k_1 , musí být ohodnocen⁷ jako lépe testovatelný (tj. $\pi_T(x_1) > \pi_T(x_2)$) než port x_2 , pro jehož testování je třeba generovat řídicí posloupnost o délce $k_2 > k_1$ je-li poměr počtu říditelných a pozorovatelných bran ku celkovému počtu bran u obou portů přibližně stejný,
- testovatelný port x_1 , pro jehož testování je třeba generovat řídicí posloupnost o délce k s celkovým počtem l_1 podnětů řídicích bran musí být ohodnocen jako lépe testovatelný než port x_2 , pro jehož testování je třeba generovat řídicí posloupnost o délce k s celkovým počtem $l_2 > l_1$ podnětů řídicích bran, je-li poměr počtu říditelných a pozorovatelných bran ku celkovému počtu bran u obou portů přibližně stejný,
- testovatelný port x_1 , pro jehož testování je třeba generovat řídicí posloupnost o délce k s celkovým počtem l podnětů řídicích bran musí být ohodnocen jako lépe testovatelný než port x_2 , pro jehož testování je třeba generovat řídicí posloupnost o délce k s celkovým počtem l podnětů řídicích bran, pokud poměr počtu říditelných a pozorovatelných bran ku celkovému počtu bran je u portu x_1 větší než u portu x_2 ,
- testovatelný port x_1 s poměrem počtu říditelných a pozorovatelných bran ku celkovému počtu bran rovným k musí být ohodnocen jako lépe testovatelný než port x_2 s tímž poměrem, pokud je řízení i pozorování bran portu x_1 jednodušší (ve smyslu kratší řídicí posloupnosti a/nebo menšího celkového počtu podnětů) než u portu x_2 .

Příklad funkce, splňující výše uvedené požadavky, je uveden v následující definici. Její snahou je shrnout číselná ohodnocení říditelnosti ($\pi_C(x)$, $\pi_{NC}(x)$) a pozorovatelnosti ($\pi_O(x)$, $\pi_{NO}(x)$) portu x , tj. reálná čísla z intervalu $< 0; 1 >$ a vyjádřit testovatelnost portu x jediným reálným číslem $\pi_T(x)$ z tohoto intervalu jako funkci hodnot $\pi_C(x)$, $\pi_{NC}(x)$, $\pi_O(x)$ a $\pi_{NO}(x)$.

Definice 30:

Nechť je definováno zobrazení $\pi_T: PORT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující portu $x \in PORT_{CUA}$ reálné číslo (*hodnotu testovatelnosti portu*) podle předpisu

$$\pi_T(x) = \frac{\pi_{NC}(x)}{\pi_W(x)} \cdot (1 + \pi_C(x)) \cdot \frac{\pi_{NO}(x)}{\pi_W(x)} \cdot (1 + \pi_O(x))/4.$$

□

Touto definicí končí část týkající se zavádění prostředků pro ohodnocení testovatelnosti virtuálních portů, bran a portů obvodových prvků, tj. část, kterou je možno chápat jako část zabývající se lokálním ohodnocením testovatelnosti *CUA* - lokálním proto, že se týkala ohodnocení částí obvodu *CUA*. V následující definici jsou zavedeny prostředky pro ohodnocení testovatelnosti *CUA* jako celku, tj. prostředky pro globální ohodnocení testovatelnosti *CUA*. Problematickým je opět shrnutí dílčích ohodnocení testovatelnosti ($v_{NC}, v_{RC}, v_C, v_{NO}, v_{RO}, v_O$) do jednoho čísla tak, aby toto číslo co nejlépe

⁷ při řádově stejném počtu řídicích podnětů

odráželo testovatelnost obvodu jako celku, tj. aby svou hodnotou co nejlépe vyjadřovalo snadnost (hodnota blíží se 1) či značnou obtížnost/nemožnost (hodnota blíží se 0) testování⁸ daného obvodu. Požadované číslo je obvodu CUA přiřazeno zobrazením ν_T , jehož nejdůležitější vlastnosti lze shrnout do následujících bodů:

- $\nu_T(cir_{CUA}) = 0 \Leftrightarrow \nu_{NC}(cir_{CUA}) = 0 \vee \nu_{NO}(cir_{CUA}) = 0$, tj. pokud v CUA neexistuje brána, která by byla říditelná nebo pokud v CUA neexistuje brána, která by byla pozorovatelná (neexistuje brána, kterou je možno otestovat)⁹,
- $\nu_T(cir_{CUA}) = 1 \Leftrightarrow (\nu_{NC}(cir_{CUA}) = \nu_{NO}(cir_{CUA}) = |B_{CUA} \cup BP_{CUA}|) \wedge (\nu_C(cir_{CUA}) = \nu_O(cir_{CUA}) = 1)$, tj. pro každou bránu platí, že je říditelná i pozorovatelná bez nutnosti řízení jakékoliv další brány¹⁰,
- na základě předchozích bodů je tedy možno konstatovat, že naprostá většina návrhů CUA bude ohodnocena reálným číslem z intervalu $(0; 1)$, jehož hodnota je určena ohodnoceními¹¹ portů obvodových prvků v CUA . Přitom platí, že čím větší je počet říditelných a pozorovatelných bran v portech obvodových prvků a čím lepší (tj. bližší hodnotě 1) je říditelnost a pozorovatelnost těchto bran, tím lepší bude celková testovatelnost obvodu a naopak, čím menší je počet říditelných a pozorovatelných bran v portech obvodových prvků a čím horší (tj. bližší hodnotě 0) je říditelnost a pozorovatelnost těchto bran, tím horší bude celková testovatelnost obvodu.

5.1.2 Vztahy pro globální ohodnocení

Definice 31:

Bud'te definována zobrazení

- $\nu_{NC}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující CUA počet říditelných bran obvodu podle předpisu

$$cir_{CUA} \mapsto \sum_{x \in PORT_{CUA}} \pi_{NC}(x)$$

⁸ generování a aplikaci testu

⁹ jelikož u CUA se mlčky předpokládá alespoň jeden primární vstup - ten je vždy říditelný - a/nebo alespoň jeden primární výstup - ten je vždy pozorovatelný - pak by tuto podmínku splňoval pouze CUA postrádající jakékoliv rozhraní; výskyt takového CUA v praktickém návrhu je však velmi nepravděpodobný

¹⁰ dle zavedených definic by CUA splňující tuto podmínku nesměl obsahovat vnitřní prvek nezbytný k přenosu diagnostických dat a rozhraní takového CUA by muselo být propojeno galvanickými spoji tak, že ke každé výstupní bráně CUA by musela existovat vstupní brána CUA , se kterou by byla tato výstupní brána spojena; výskyt CUA s těmito vlastnostmi je však v praxi velmi diskutabilní či velmi speciální

¹¹ komentář ke způsobu ohodnocení testovatelnosti portů viz předchozí definice a požadavky kladené na funkci π_T , strana 114

- $v_{RC}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující *CUA podíl říditelných bran obvodu* podle předpisu

$$cir_{CUA} \mapsto \frac{v_{NC}(cir_{CUA})}{|B_{CUA} \cup BP_{CUA}|}$$
- $v_{NO}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující *CUA počet pozorovatelných bran obvodu* podle předpisu

$$cir_{CUA} \mapsto \sum_{x \in PORT_{CUA}} \pi_{NO}(x)$$
- $v_{RO}: \{cir_{CUA}\} \rightarrow \mathbb{N}$ přiřazující *CUA podíl pozorovatelných bran obvodu* podle předpisu

$$cir_{CUA} \mapsto \frac{v_{NO}(cir_{CUA})}{|B_{CUA} \cup BP_{CUA}|}$$
- $v_C: \{cir_{CUA}\} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující *CUA průměrnou hodnotu říditelnosti bran obvodu* podle předpisu

$$x \mapsto \frac{1}{|B_{CUA} \cup BP_{CUA}|} \sum_{x \in PORT_{CUA}} \pi_C(x)$$
- $v_O: \{cir_{CUA}\} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující *CUA průměrnou hodnotu pozorovatelnosti bran obvodu* podle předpisu

$$x \mapsto \frac{1}{|B_{CUA} \cup BP_{CUA}|} \sum_{x \in PORT_{CUA}} \pi_O(x)$$
- $v_T: \{cir_{CUA}\} \rightarrow \mathbb{R}_{<0,1>}$ přiřazující *CUA reálné číslo vyjadřující hodnotu testovatelnosti obvodu* podle předpisu

$$x \mapsto v_{RC} \cdot (1 + v_C) \cdot v_{RO} \cdot (1 + v_O)/4$$

□

V této podkapitole byla definována zobrazení, pomocí nichž je možno ohodnotit lokální a globální testovatelnost daného *CUA*. Tato zobrazení představují formální zápis způsobu ohodnocení testovatelnosti. Avšak ani ona - stejně jako prostředky zavedené v definicích předchozí kapitoly - neudávají postup, pomocí něhož je možné je zkonstruovat. V následující kapitole tedy bude uveden algoritmus, jehož cílem je popsat konstrukce těch zobrazení z této podkapitoly, které se zdají být obtížnými. Vzhledem k činnosti, kterou tento algoritmus provádí, je možno jej označit za algoritmus analýzy testovatelnosti číslicového obvodu na úrovni popisu dané výše popsaným modelem.

5.2 Algoritmus analýzy testovatelnosti

Navržený algoritmus analýzy testovatelnosti, kterému je věnován prostor v této podkapitole, je založen na prohledávání grafu (přesněji na prohledávání dvou grafů: grafu G_S při analýze říditelnosti v *CUA* a prohledávání grafu G_I při analýze pozorovatelnosti v *CUA*) [ZK01d, Str02b, JS02a, Str03b]. Jelikož současně s prohledáváním grafu G_S (G_I) je prováděno ohodnocování jeho uzlů, nejedná se o klasický prohledávací algoritmus. Dříve než bude vlastní algoritmus analýzy testovatelnosti představen, budou zmíněny některé informace k prohledávání grafů.

5.2.1 Komentář k prohledávání grafů

Základním účelem prohledávání grafů je zjišťování dostupnosti, tj. zjišťování, do kterých vrcholů grafu vedou cesty z daného výchozího vrcholu; k dalším cílům může patřit hledání těchto cest či vykonání nějaké akce v každém dostupném vrcholu. [Dem02] uvádí tři základní způsoby prohledávání grafů. První z nich (značkování vrcholů) je značně obecný a nesmírně jednoduchý. Další dva způsoby, prohledávání do šířky a prohledávání do hloubky, lze pokládat za jeho speciální případy. Během značkování vrcholů jsou vrcholům grafu přiřazovány značky tak, že má-li vrchol značku, znamená to, že do něj vede nějaká cesta z daného výchozího vrcholu r . Algoritmus značkování je velmi prostý [Dem02]:

1. [Inicializace] Označujeme vrchol r , ostatní vrcholy jsou beze značek
2. [Výběr hrany] Vybereme libovolnou hranu e , jejíž počáteční vrchol má značku a koncový vrchol nikoli; pokračujeme podle kroku 3. Jestliže taková hrana neexistuje, výpočet končí
3. [Značkování] Označujeme koncový vrchol hrany e , následuje skok na krok 2

Ze speciálních případů značkování vrcholů pouze stručně zmiňme principy jejich činnosti [Dem02]:

- prohledávání grafu do šířky - tento způsob prohledávání lze charakterizovat takto: v prvním kroku je označován (výchozí) vrchol r , v druhém kroku všichni jeho následníci, v třetím kroku všichni neoznačovaní následníci následníků z druhého kroku atd. Lze si to představit i tak, že graf současně prozkoumává "velký počet průzkumníků", kteří "zaplavují" postupně "po hladinách" z vrcholu r dostupnou část grafu,
- prohledávání grafu do hloubky - tento způsob prohledávání si lze představit jako průzkum grafu cestovatelem, který cestuje po hranách grafu a vrací se důsledně cestou, po které přišel.

Jak bude ukázáno níže, snahou navrženého algoritmu analýzy testovatelnosti je kromě zjištění dostupnosti konkrétních uzlů zjistit také cenu této dostupnosti, tj. ohodnocení vzdálenosti uzlů od primárních vstupů obvodu v případě analýzy říditelnosti a ohodnocení vzdálenosti uzlů od primárních výstupů obvodu v případě analýzy pozorovatelnosti. I když je obtížné zařadit navržený algoritmus do některé z výše uvedených základních kategorií, je možno konstatovat, že principiálně vychází z prohledávání grafu do šířky.

5.2.2 Komentář k principu navržené metody

Vzhledem k definici grafů G_S , G_I je třeba si uvědomit přesný význam hrany v každém z těchto grafů: tou je buď modelována existence galvanického propojení spojujícího jisté brány dvou virtuálních portů s touto hranou incidujících nebo je jí modelována schopnost (ovšem podmíněná schopností ovládat příslušné řídicí brány) přenosu diagnostických dat mezi dvěma

virtuálními porty s ní incidujícími, přičemž směr přenosu je v obou případech dán orientací této hrany. Z významu hrany v grafech G_S , G_I tedy vyplývá, že zatímco značka z bran počátečního uzlu (virtuálního portu) hrany představující existenci galvanického spoje může být bezpodmínečně přenesena na ty brány koncového uzlu hrany, se kterými jsou brány z počátečního uzlu galvanicky propojeny, tak v případě hrany modelující podmíněnou existenci datového přenosu tomu tak již obecně není; vzhledem k tomu, že taková hrana je podmíněna řízením jistých řídicích bran, je i přenos značky přes tuto hranu podmíněn schopností řídit tyto brány - z pohledu značkování tedy musí být každá z bran podmiňujících přenos značky přes tuto hranu označována.

Princip prohledávání grafu G_S prováděného algoritmem analýzy testovatelnosti je možné vyjádřit následujícím algoritmem¹²:

1. [Inicializace] Označujeme uzly z G_S , které jsou vstupními virtuálními porty CUA a jejich brány, ostatní ponecháme beze značek
2. [Výběr hran] Vybereme hrany G_S , jejichž počáteční vrchol má značku a koncový vrchol buď nikoli nebo má horší značku a jejichž řídicí brány již jsou označeny; pokračujeme dle kroku 3. Jestliže taková hrana neexistuje, výpočet končí
3. [Značkování] Označujeme koncový vrchol a příslušné brány každé z vybraných hran a provedeme skok na krok 2

Po tomto značkování následuje prohledávání grafu G_I . To je analogické prohledávání grafu G_S : začíná se značkováním primárních výstupů a přenos značek postupuje proti směru orientace hran směrem k primárním vstupům dokud existuje hrana, umožňující přenos značky. Prohledávání G_I musí následovat až po dokončení prohledávání G_S , jelikož nezbytně potřebuje informaci o značkách říditelnosti z G_S . Na základě výsledků obou prohledávacích algoritmů, tj. na základě výsledných značek pro říditelnost a pozorovatelnost virtuálních portů a bran CUA je číselně ohodnocena říditelnost, pozorovatelnost a testovatelnost portů obvodových prvků CUA a následně (globální) testovatelnost CUA jako celku.

V následujících odstavcích bude popsán a stručně komentován navržený algoritmus analýzy testovatelnosti a jeho dílčí části. Pro co nejpráhlednější popis algoritmu bylo rozhodnuto jej rozčlenit na několik bloků provádějících klíčové činnosti a využít těchto bloků k popisu algoritmu metodou "shora dolů".

5.2.3 Hlavní blok algoritmu

V tomto odstavci bude popsán princip činnosti a struktura hlavního bloku algoritmu analýzy testovatelnosti. Dílčím podblokům pak bude věnován pros-

¹² značky používané při analýze testovatelnosti mj. obsahují číselné ohodnocení dané vlastnosti (např. říditelnosti, pozorovatelnosti) uzlu/brány; jednoduchým porovnáním číselných ohodnocení dvou značek tedy lze stanovit, zda zkoumaná vlastnost jedné značky je horší, lepší či stejná jako u druhé značky

122 5 Metoda analýzy testovatelnosti

tor v následujících odstavcích. Snahou je popsat algoritmus metodou ”shora dolů”, tj. tak, že nejprve bude popsán nadřazený celek a až poté bude věnován prostor jeho dílčím částem. Nejprve tedy bude popsán algoritmus pro výpočet testovatelnosti (strana 121) - ten bude předpokládat, že již byla provedena analýza říditelnosti a pozorovatelnosti. Poté bude uveden algoritmus pro ohodnocení říditelnosti (strana 122) a nakonec algoritmus pro ohodnocení pozorovatelnosti (strana 126). Hlavní blok algoritmu analýzy testovatelnosti je tvořen posloupností pěti dílčích kroků:

1. v prvním kroku jsou zavedena pomocná zobrazení (v podstatě se jedná o zavedení pomocných proměnných),
2. v druhém kroku je provedena inicializace ohodnocení primárních virtuálních portů (uspořádaná pětice hodnot (virtuální port, sekvenční délka, počet podnětů, součin hodnot říditelnosti jeho řídicích bran, hodnota jeho říditelnosti) a jejich bran (uspořádaná dvojice (brána, hodnota její říditelnosti)), inicializace ohodnocení ostatních virtuálních portů a jejich bran a inicializace zobrazení pro ohodnocení říditelnosti a pozorovatelnosti portů a testovatelnosti obvodu,
3. ve třetím kroku je prohledáván graf G_S za účelem značkování říditelnosti virtuálních portů a bran,
4. ve čtvrtém kroku je prohledáván graf G_I za účelem značkování pozorovatelnosti virtuálních portů a bran,
5. v posledním, pátém, kroku je na základě značek vyjadřujících hodnotu říditelnosti a pozorovatelnosti virtuálních portů a bran ohodnocena říditelnost, pozorovatelnost a testovatelnost portů obvodových prvků a (globální) testovatelnost CUA . Ukončením tohoto kroku algoritmus analýzy testovatelnosti končí.

Algoritmus 4

Procedura *Analýza_Testovatelnosti*(CUA, G_S, G_I)

1. [Zavedení pomocných zobrazení]
Zaved' $\nu_C, \nu_O: VPORT_{CUA} \rightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times \mathbb{R}$, $\beta_C, \beta_O: BIT_{CUA} \rightarrow \mathbb{R}_{<0,1>}$, $\pi_{NC}, \pi_{NO}: PORT_{CUA} \rightarrow \mathbb{N}$, $\pi_C, \pi_O, \pi_T: PORT_{CUA} \rightarrow \mathbb{R}$, $v_T: cir_{CUA} \rightarrow \mathbb{R}$
2. [Inicializace hodnot pro primární virtuální porty]
Přihrad' $\nu_C := \{(x, 0, 0, k, k) | x \in VPORT_{CUA}, \text{ kde } k = 1 \text{ pokud } \pi_{BITS}(x) \subseteq BPI_{CUA} \cup BPCI_{CUA}, \text{ jinak } k = 0\}$,
 $\nu_O := \{(x, 0, 0, k, k) | x \in VPORT_{CUA}, \text{ kde } k = 1 \text{ pokud } \pi_{BITS}(x) \subseteq BPO_{CUA}, \text{ jinak } k = 0\}$,
 $\beta_C := \{(x, 1) | x \in BPI_{CUA} \cup BPCI_{CUA}\} \cup \{(x, 0) | x \notin BPI_{CUA} \cup BPCI_{CUA}\}$
 $\beta_O := \{(x, 1) | x \in BPO\} \cup \{(x, 0) | x \notin BPO_{CUA}\}$
 $\pi_{NC} := \emptyset, \pi_{NO} := \emptyset, \pi_C := \emptyset, \pi_O := \emptyset, \pi_T := \emptyset, v_T := \emptyset$
3. Ohodnocení říditelnosti($CUA, G_S, \nu_C, \beta_C, \pi_{NC}, \pi_C$)

4. *Ohodnocení_pozorovatelnosti*($CUA, G_I, \beta_C, \nu_O, \beta_O, \pi_{NO}, \pi_O$)
5. *Ohodnocení_testovatelnosti*($CUA, \pi_{NC}, \pi_C, \pi_{NO}, \pi_O, \pi_T, \nu_T$)

□

5.2.4 Blok pro ohodnocení testovatelnosti

Prostor tohoto prvního odstavce bude věnován bloku pro ohodnocení testovatelnosti portů obvodových prvků a testovatelnosti CUA ; ten odpovídá pátému kroku algoritmu 4. Vstupem tohoto bloku je model CUA a zobrazení $\pi_{NC}, \pi_C, \pi_{NO}, \pi_O$, představující ohodnocení dílčích složek testovatelnosti portů obvodových prvků. Výstupem je a) zobrazení π_T ohodnocující testovatelnost každého portu v CUA dle definice 30, strana 115 a b) zobrazení ν_T ohodnocující globální testovatelnost CUA dle definice 31, strana 116. Algoritmus zřejmě pracuje v čase $O(|PORT_{CUA}|)$.

Algoritmus 5

Procedura *Ohodnocení_testovatelnosti*($CUA, \pi_{NC}, \pi_C, \pi_{NO}, \pi_O$,
var : π_T, ν_T)

1. [Zavedení a inicializace pomocných proměnných]

Zaved' $\nu_C, \nu_O, \nu_{NC}, \nu_{NO}, \nu_{RC}, \nu_{RO} \in \mathbb{R}$

2. [Výpočet globálních ukazatelů testovatelnosti CUA]

Přiřad' $\nu_{NC} := \sum_{x \in PORT_{CUA}} \pi_{NC}(x)$,

$$\nu_{RC} := \frac{\nu_{NC}(circ_{CUA})}{|B_{CUA} \cup BP_{CUA}|},$$

$$\nu_C := \frac{1}{|B_{CUA} \cup BP_{CUA}|} \sum_{x \in PORT_{CUA}} \pi_C(x),$$

$$\nu_{NO} := \sum_{x \in PORT_{CUA}} \pi_{NO}(x),$$

$$\nu_{RO} := \frac{\nu_{NO}(circ_{CUA})}{|B_{CUA} \cup BP_{CUA}|},$$

$$\nu_O := \frac{1}{|B_{CUA} \cup BP_{CUA}|} \sum_{x \in PORT_{CUA}} \pi_O(x)$$

3. [Ohodnocení testovatelnosti portů]

Pro každý $x \in PORT_{CUA}$ **proved'**

$$\pi_T := \pi_T \cup \{(x, \frac{\pi_{NC}(x)}{\pi_W(x)} \cdot (1 + \pi_C(x)) \cdot \frac{\pi_{NO}(x)}{\pi_W(x)} \cdot (1 + \pi_O(x)) / 4)\}$$

4. [Ohodnocení testovatelnosti CUA]

Přiřad' $\nu_T := \nu_{RC} \cdot (1 + \nu_C) \cdot \nu_{RO} \cdot (1 + \nu_O) / 4$

□

5.2.5 Blok pro ohodnocení říditelnosti

V tomto odstavci budou popsány bloky sloužící k ohodnocení říditelnosti virtuálních portů, bran a portů obvodových prvků v CUA ; ty odpovídají třetímu kroku algoritmu 4. Vstupem hlavního bloku pro ohodnocení říditelnosti je model CUA , graf G_S datového toku testovacích vzorků a inicializovaná

zobrazení pro ohodnocení říditelnosti virtuálních portů, bran a portů obvodových prvků v CUA . Výstupem jsou zobrazení s ohodnocením říditelnosti virtuálních portů, bran a portů obvodových prvků v CUA .

Algoritmus je tvořen čtyřmi kroky, se smyčkou mezi kroky 2 a 3. V prvním kroku jsou deklarovány pomocné množiny sloužící ke značkování dostupnosti bran (B_{RDY}) resp. virtuálních portů (V_{RDY}) z primárních vstupů CUA , množiny obsahující nově označované brány (B_{NEW}) resp. virtuální porty (V_{NEW}) a proměnná *nastala_změna* oznamující, zda došlo ke změně značení, tj. zda přibyla nová značka. V kroku 2 jsou nejprve značky z nově označených virtuálních portů resp. jejich bran přeneseny (krok 2.2) po hranách grafu G_S představujících spoje v CUA ; pokud neexistuje značka, kterou by bylo možno přenést, tj. pokud není k dispozici žádný nově označený virtuální port resp. brána, značkování je ukončeno a algoritmus pokračuje ohodnocením říditelnosti portů obvodových prvků v kroku 4. V opačném případě, tj. existuje-li nově označený virtuální port resp. brána, pak jsou v grafu G_S hledány (krok 2.3) hrany, reprezentující tok testovacích vzorků příslušnými obvodovými prvky CUA , přes které je možno přenést (přenos je podmíněn označením všech řídicích bran každé z těchto hran) značky dále a tento přenos je proveden. Nebyla-li přenesena ani jedna značka, algoritmus pokračuje krokem 4; v opačném případě o nově označené brány resp. virtuální porty obohaceny příslušné množiny (V_{NEW} , B_{NEW} , V_{RDY}) a algoritmus se vrací na krok 2.

V následujícím textu budou představeny tři podbloky algoritmu pro ohodnocení říditelnosti. První podblok popisuje způsob aktualizace ohodnocení říditelnosti nově označených bran, druhý podblok popisuje způsob přenosu značek přes spoje a třetí podblok způsob přenosu značek přes obvodové prvky.

Algoritmus 6

Procedura *Ohodnocení říditelnosti*(CUA, G_S , var: $\nu_C, \beta_C, \pi_{NC}, \pi_C$)

1. [Zavedení a inicializace pomocných proměnných a množin]

Zaved' $B_{RDY}, B_{NEW} \subset BIT_{CUA}$, $nastala_změna \in \{false, true\}$, $V_{NEW}, V_{RDY} \subseteq VPORT_{CUA}$

Přiřad' $B_{RDY} := \emptyset$, $B_{NEW} := BPI_{CUA} \cup BPCI_{CUA}$,

$V_{RDY} := \{x \mid x \in VPORT_{CUA}, \pi_{BITS}(x) \subseteq B_{NEW}\}$,

2. [Smyčka pro šíření hodnot říditelnosti (začátek)]

2.1. **Přiřad'** $nastala_změna := false$

2.2. [Šíření hodnot říditelnosti po spojích]

Proved' Přenos_přes_spoje($B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu_C$)

Pokud $|B_{NEW}| = \emptyset$ **jdi na bod 4**

2.3. [Šíření hodnot říditelnosti přes obvodové prvky]

Proved' *Aktualizace_ohodnocení_bran*(B_{NEW}, ν_C, β_C)

Přenos_přes_prvky($G_S, \beta_C, B_{NEW}, V_{NEW}$,

B_{RDY}, V_{RDY}, ν_C)

Aktualizace_ohodnocení_bran(B_{NEW}, ν_C, β_C)

Přiřaď *nastala_změna* := ($B_{NEW} \neq \emptyset$)

3. [*Smyčka pro šíření hodnot říditelnosti (podmíněný skok na začátek)*]

Pokud *nastala_změna* **jdi na** bod 2

4. [*Ohodnocení říditelnosti portů*]

Pro každý $x \in PORT_{CUA}$ **proved'**

$$\pi_{NC} := \pi_{NC} \cup \{(x, |\{y | y \in \pi_{BITS}(x) \wedge \beta_C(y) > 0\}|)\}$$

$$\pi_C := \pi_C \cup \{(x, \frac{\sum_{y \in \pi_{BITS}(x)} \beta_C(y)}{\pi_W(x)})\}$$

□

Úkolem níže popsaného podbloku je - na základě ohodnocení říditelnosti virtuálních portů (zobrazení ν) - aktualizovat informaci o hodnotě říditelnosti nově označených bran (tj. bran z množiny B_{NEW}). Z množiny β je nejprve vyjmuta stará informace a následně je do ní vložena aktualizovaná informace o říditelnosti bran.

Algoritmus 7

Procedura *Aktualizace_ohodnocení_bran*(B_{NEW}, ν , **var:** β)

1. **Přiřaď** $\beta := (\beta \setminus \{(x, k) | x \in B_{NEW}\}) \cup \{(x, k) | x \in B_{NEW}, k = \max_{(y, y_1, y_2, y_3, y_4) \in \nu} y_4\}$

□

Druhým podblokem je podblok pro přenos značek přes spoje. Značka z každé nově označené brány (z množiny B_{NEW}) je (ve směru datového toku určeného relací F_{CUA} souhlasném s orientací hran v G_S) přenesena na všechny její cílové brány (dané relací F'_{CUA}), je vytvořena množina V_{NEW} nově označených virtuálních portů a je aktualizována informace o říditelnosti každého z těchto portů.

Algoritmus 8

Procedura *Přenos_přes_spoje*(**var:** $B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu$)

1. **Zaved'** $F'_{BIT} \subset F_{BIT}$

2. **Přiřaď** $F'_{BIT} := \{(x, y) | (x, y) \in F_{BIT}, x \in B_{NEW}\}$,

$$B_{RDY} := B_{RDY} \cup B_{NEW},$$

$$B_{NEW} := Im(F'_{BIT}),$$

$$V_{NEW} := \{x | x \in V_{PORT_{CUA}}, \pi_{BITS}(x) \subseteq B_{RDY} \wedge$$

$$\pi_{BITS}(x) \cap B_{NEW} \neq \emptyset\}$$

$$\nu := (\nu \setminus \{(y, y_1, y_2, y_3, y_4) | y \in V_{NEW}\}) \cup \{(y, y_{seq}, y_{sti}, y_{mul}, y_{CO}) | y \in V_{NEW},$$

$$y_{seq} = \max_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} z_{seq},$$

$$y_{sti} = \max_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} z_{sti},$$

$$y_{mul} = \frac{1}{|\pi_{BITS}(y)|} \prod_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} (z_{mul} \cdot |\pi_{BITS}(y) \cap \pi_{BITS}(z)|),$$

126 5 Metoda analýzy testovatelnosti

$$y_{CO} = \frac{1}{|\pi_{BITS}(y)|} \prod_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} (z_{CO} \cdot |\pi_{BITS}(y) \cap \pi_{BITS}(z)|),$$

$$(z, z_{seq}, z_{sti}, z_{mul}, z_{CO}) \in \nu\},$$

$$V_{RDY} := V_{RDY} \cup V_{NEW}$$

□

Posledním podblokem je podblok pro přenos značek říditelnosti strukturou obvodových prvků. Po zavedení pomocných proměnných v kroku 1 jsou v kroku 2 inicializovány množiny nově označených bran (B_{NEW}) a virtuálních portů (V_{NEW}) a je vytvořena množina (E_{NEW}) hran grafu G_S s počátečními uzly (virtuálními porty) z množiny V_{NEW} ; prvky množiny E_{NEW} jsou hrany, přes které lze za jistých podmínek uskutečnit přenos značky strukturou příslušných obvodových prvků. V kroku 3.1 je pak pro každý cílový uzel každé z těchto hran ohodnocena jeho potenciaální značka. Značkou ohodnocenou v kroku 3.1 je uzel označen v případě, je-li označen horší¹³ značkou (krok 3.2) nebo pokud není označen (krok 3.3). V kroku 3.4 jsou postupně vytvářeny množiny nově označených bran (B_{NEW}) a nově označených virtuálních portů (V_{NEW}). V posledním kroku (krok 4) je aktualizován obsah množiny již označených uzlů (V_{RDY}).

Algoritmus 9

Procedura *Přenos_přes_prvky*($G_S, \beta_C, \text{var: } B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu$)

1. **Zaved'** $E_{NEW} \subset V_{PORTCUA} \times V_{PORTCUA}$
došlo_ke_zlepšení $\in \{true, false\}$
2. **Přiřad'** $E_{NEW} := \{(x, y) \mid (x, y) \in Im(\sigma(G_S)), w_{SSeq}(x, y) \geq 1, x \in V_{NEW}\},$
 $B_{NEW} := \emptyset, V_{NEW} := \emptyset$
3. **Pro každé** $y \in Im(E_{NEW})$ **proved'**
 - 3.1. **Přiřad'** došlo_ke_zlepšení $:= false,$
 $y_{seq} := x_{seq} + w_{SSeq}(x, y),$
 $y_{sti} := x_{sti} + w_{SSeq}(x, y) \cdot |w_{SX}(x, y)|,$
 $y_{mul} := x_{mul} \cdot \prod_{x' \in w_{SX}(x, y)} \beta_C(x'),$
 $y_{CO} := (1 - \frac{y_{seq}}{\overline{v_{Seq}(circuitCUA)+1}}) \cdot (1 - \frac{y_{Nsti}}{\overline{v_{Nsti}(circuitCUA)+1}}).$
 $y_{mul},$ kde
 $(x, x_{seq}, x_{sti}, x_{mul}, x_{CO}) \in \nu, (x, y) \in E_{NEW},$
 $x_{CO} = \min_{(z, y) \in E_{NEW}, (z, z_{seq}, z_{sti}, z_{mul}, z_{CO}) \in \nu} z_{CO}$
 - 3.2. **Pokud** $y \in V_{RDY}$ **proved'**
 - 3.2.1. **Pokud** $\exists (y, y'_{seq}, y'_{sti}, y'_{mul}, y'_{CO}) \in \nu:$
 $y_{CO} > y'_{CO}$ **proved'**
 $\nu := (\nu \setminus (y, y'_{seq}, y'_{sti}, y'_{mul}, y'_{CO})) \cup$
 $\{(y, y_{seq}, y_{sti}, y_{mul}, y_{CO})\},$
došlo_ke_zlepšení $:= true$
 - 3.3. **Pokud** $y \notin V_{RDY}$ **proved'**

¹³ kritériem pro porovnávání značek je zde ohodnocení říditelnosti

$$\nu := \nu \cup \{(y, y_{seq}, y_{sti}, y_{mul}, y_{CO}), \\ došlo_ke_zlepšení := true$$

3.4. **Pokud** *došlo_ke_zlepšení* **proved'**

$$B_{NEW} := B_{NEW} \cup \pi_{BITS}(y),$$

$$V_{NEW} := V_{NEW} \cup \{y\}$$

4. **Přiřad'** $V_{RDY} := V_{RDY} \cup V_{NEW}$

□

5.2.6 Blok pro ohodnocení pozorovatelnosti

Po ohodnocení říditelnosti je možno přistoupit k dalšímu (čtvrtému) kroku analýzy testovatelnosti - k analýze pozorovatelnosti virtuálních portů, bran a portů obvodových prvků v *CUA*. Algoritmus této analýzy bude opět rozčleněn do několika bloků, jimž bude věnován tento odstavec.

Vstupem hlavního bloku pro ohodnocení pozorovatelnosti je model *CUA*, graf G_I datového toku odezev, zobrazení informující o říditelnosti bran a inicializovaná zobrazení pro ohodnocení říditelnosti virtuálních portů, bran a portů obvodových prvků v *CUA*. Výstupem jsou zobrazení s ohodnocením pozorovatelnosti virtuálních portů, bran a portů obvodových prvků v *CUA*.

Algoritmus je tvořen čtyřmi kroky, se smyčkou mezi kroky 2 a 3. V prvním kroku jsou deklarovány pomocné množiny sloužící ke značkování dostupnosti bran (B_{RDY}) resp. virtuálních portů (V_{RDY}) z primárních výstupů *CUA*, množiny obsahující nově označované brány (B_{NEW}) resp. virtuální porty (V_{NEW}) a proměnná *nastala_změna* oznamující, zda došlo ke změně značení, tj. zda přibyla nová značka. V kroku 2 jsou nejprve značky z nově označených virtuálních portů resp. jejich bran přeneseny (krok 2.2) po hranách grafu G_I představujících spoje v *CUA*; pokud neexistuje značka, kterou by bylo možno přenést, tj. pokud není k dispozici žádný nově označený virtuální port resp. brána, značkování je ukončeno a algoritmus pokračuje ohodnocením pozorovatelnosti portů obvodových prvků v kroku 4. V opačném případě, tj. existuje-li nově označený virtuální port resp. brána, pak jsou v grafu G_I hledány (krok 2.3) hrany reprezentující tok odezev příslušnými obvodovými prvky *CUA*, přes které je možno přenést (přenos je podmíněn označením všech řídicích bran každé z těchto hran) značky dále a tento přenos je proveden. Nebyla-li přenesena ani jedna značka, algoritmus pokračuje krokem 4; v opačném případě jsou o nově označené brány resp. virtuální porty obohaceny příslušné množiny (V_{NEW} , B_{NEW} , V_{RDY}) a algoritmus se vrací na krok 2.

V následujícím textu budou představeny podbloky algoritmu pro ohodnocení říditelnosti. První podblok popisuje způsob aktualizace ohodnocení říditelnosti nově označených bran, druhý podblok popisuje způsob přenosu značek přes spoje a třetí podblok způsob přenosu značek přes obvodové prvky. Čtvrtý podblok, používaný v prvním podbloku pro aktualizaci ohodnocení pozorovatelnosti bran, zde uveden není, jelikož je totožný s podblokem popsáním

128 5 Metoda analýzy testovatelnosti

algoritmem 7. Narozdíl od algoritmu 7, uvedeného v souvislosti s ohodnocováním říditelnosti však nejsou vstupem tohoto algoritmu zobrazení ν_C, β_C obsahující informaci o hodnotě říditelnosti bran a virtuálních portů, ale zobrazení ν_O, β_O obsahující informaci o jejich pozorovatelnosti.

Algoritmus 10

Procedura *Ohodnocení_pozorovatelnosti*($CUA, G_I, \beta_C, \mathbf{var}: \nu_O, \beta_O, \pi_{NO}, \pi_O$)

1. [Zavedení a inicializace pomocných proměnných a množin]

Zaved' $B_{RDY}, B_{NEW} \subset BIT_{CUA}$, $nastala_změna \in \{false, true\}$,

$V_{NEW}, V_{RDY} \subseteq VPORT_{CUA}$

Přiřad' $B_{RDY} := \emptyset, B_{NEW} := BPO_{CUA}$,

$V_{RDY} := \{x \mid x \in VPORT_{CUA}, \pi_{BITS}(x) \subseteq B_{NEW}\}$,
2. [Smyčka pro šíření hodnot pozorovatelnosti (začátek)]
 - 2.1. **Přiřad'** $nastala_změna := false$
 - 2.2. [Šíření hodnot pozorovatelnosti po spojích]

Proved' $Přenos_přes_spoje(B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu_O)$

Pokud $|B_{NEW}| = \emptyset$ **jdi na bod 4**
 - 2.3. [Šíření hodnot pozorovatelnosti přes obvodové prvky]

Proved' $Aktualizace_ohodnocení_bran(B_{NEW}, \nu_O, \beta_O)$

$Přenos_přes_prvky(G_I, \beta_C, B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu_O)$

$Aktualizace_ohodnocení_bran(B_{NEW}, \nu_O, \beta_O)$

Přiřad' $nastala_změna := (B_{NEW} \neq \emptyset)$
3. [Smyčka pro šíření hodnot pozorovatelnosti (podmíněný skok)]

Pokud $nastala_změna$ **jdi na bod 2**
4. [Ohodnocení pozorovatelnosti portů]

Pro každý $x \in PORT_{CUA}$ **proved'**

$\pi_{NO} := \pi_{NO} \cup \{(x, |\{y \mid y \in \pi_{BITS}(x) \wedge \beta_O(y) > 0\}|)\}$

$\pi_O := \pi_O \cup \{(x, \frac{\sum_{y \in \pi_{BITS}(x)} \beta_O(y)}{\pi_W(x)})\}$

□

Jedním z podbloků pro značkování pozorovatelných uzlů v G_I je níže uvedený podblok pro přenos (ve směru od primárních výstupů CUA k primárním vstupům CUA , tj. proti směru orientace hran v G_I) značek přes spoje. Značka je z každé nově označené brány (z množiny B_{NEW}) přenesena na všechny její cílové brány (dané relací F'_{CUA}), je vytvořena množina V_{NEW} nově označených virtuálních portů a je aktualizována informace o pozorovatelnosti každého z těchto portů.

Algoritmus 11

Procedura *Přenos_přes_spoje*($\mathbf{var}: B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu$)

1. **Zaved'** $F'_{BIT} \subset F_{BIT}$
2. **Přirad'** $F'_{BIT} := \{(x, y) \mid (x, y) \in F_{BIT}, y \in B_{NEW}\},$
 $B_{RDY} := B_{RDY} \cup B_{NEW},$
 $B_{NEW} := Def(F'_{BIT}),$
 $V_{NEW} := \{x \mid x \in VPORT_{CUA}, \pi_{BITS}(x) \subseteq B_{RDY} \wedge \pi_{BITS}(x) \cap B_{NEW} \neq \emptyset\}$
 $\nu := (\nu \setminus \{(y, y_1, y_2, y_3, y_4) \mid y \in V_{NEW}\}) \cup \{(y, y_{seq}, y_{sti}, y_{mul}, y_{CO}) \mid y \in V_{NEW},$
 $y_{seq} = \max_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} z_{seq},$
 $y_{sti} = \max_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} z_{sti},$
 $y_{mul} = \frac{1}{|\pi_{BITS}(y)|} \prod_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} (z_{mul} \cdot |\pi_{BITS}(y) \cap \pi_{BITS}(z)|),$
 $y_{CO} = \frac{1}{|\pi_{BITS}(y)|} \prod_{z \in V_{RDY}, \pi_{BITS}(z) \cap \pi_{BITS}(y) \neq \emptyset} (z_{CO} \cdot |\pi_{BITS}(y) \cap \pi_{BITS}(z)|),$
 $(z, z_{seq}, z_{sti}, z_{mul}, z_{CO}) \in \nu\},$
 $V_{RDY} := V_{RDY} \cup V_{NEW}$

□

Posledním zde zmíněným podblokem je podblok pro přenos značek pozorovatelnosti strukturou obvodových prvků. Po zavedení pomocných proměnných v kroku 1 jsou v kroku 2 inicializovány množiny nově označených bran (B_{NEW}) a virtuálních portů (V_{NEW}) a je vytvořena množina (E_{NEW}) hran grafu G_I s koncovými uzly (virtuálními porty) z množiny V_{NEW} ; prvky množiny E_{NEW} jsou hrany, přes které lze (proti směru danému jejich orientací) za jistých podmínek uskutečnit přenos značky strukturou příslušných obvodových prvků. V kroku 3.1 je pak pro každý počáteční uzel každé z těchto hran ohodnocena jeho potenciaální značka. Značkou ohodnocenou v kroku 3.1 je uzel označen v případě, je-li označen horší¹⁴ značkou (krok 3.2) nebo pokud není označen (krok 3.3). V kroku 3.4 jsou postupně vytvářeny množiny nově označených bran (B_{NEW}) a nově označených virtuálních portů (V_{NEW}). V posledním kroku (krok 4) je aktualizován obsah množiny již označených uzlů (V_{RDY}).

Algoritmus 12

Procedura *Přenos_přes_prvky*($G_I, \beta_C, \text{var: } B_{NEW}, V_{NEW}, B_{RDY}, V_{RDY}, \nu$)

1. **Zaved'** $E_{NEW} \subset VPORT_{CUA} \times VPORT_{CUA},$
 $došlo_ke_zlepšení \in \{true, false\}$
2. **Přirad'** $E_{NEW} := \{(x, y) \mid (x, y) \in Im(\sigma(G_I)), w_{ISeq}(x, y) \geq 1,$
 $y \in V_{NEW}\},$
 $B_{NEW} := \emptyset, V_{NEW} := \emptyset$
3. **Pro každé** $x \in Def(E_{NEW})$ **proved'**
 - 3.1. **Přirad'** $došlo_ke_změně := false,$

¹⁴ kritériem pro porovnávání značek je zde ohodnocení pozorovatelnosti

130 5 Metoda analýzy testovatelnosti

$$\begin{aligned}
 x_{seq} &:= y_{seq} + w_{ISeq}(x, y), \\
 x_{sti} &:= y_{sti} + w_{ISeq}(x, y) \cdot |w_{IX}(x, y)|, \\
 x_{mul} &:= y_{mul} \cdot \prod_{y' \in w_{IX}(x, y)} \beta_C(y'), \\
 x_{CO} &:= \left(1 - \frac{x_{Seq}}{\bar{v}_{Seq}(cirCUA)+1}\right) \cdot \left(1 - \frac{x_{NSti}}{\bar{v}_{NSti}(cirCUA)+1}\right) \cdot x_{mul}, \\
 &\text{kde } (y, y_{seq}, y_{sti}, y_{mul}, y_{CO}) \in \nu, (x, y) \in E_{NEW}, \\
 &y_{CO} = \min_{(z, y) \in E_{NEW}, (z, z_{seq}, z_{sti}, z_{mul}, z_{CO}) \in \nu} z_{CO}
 \end{aligned}$$

3.2. Pokud $x \in V_{RDY}$ proved'

3.2.1. Pokud $\exists(x, x'_{seq}, x'_{sti}, x'_{mul}, x'_{CO}) \in \nu$:

$x_{CO} > x'_{CO}$ proved'

$$\begin{aligned}
 \nu &:= (\nu \setminus (x, x'_{seq}, x'_{sti}, x'_{mul}, x'_{CO})) \cup \\
 &\{(x, x_{seq}, x_{sti}, x_{mul}, x_{CO})\}, \\
 došlo_ke_změně &:= true
 \end{aligned}$$

3.3. Pokud $x \notin V_{RDY}$ proved'

$$\nu := \nu \cup \{(x, x_{seq}, x_{sti}, x_{mul}, x_{CO})\},$$

$došlo_ke_změně := true$

3.4. Pokud $došlo_ke_změně$ proved'

$$B_{NEW} := B_{NEW} \cup \pi_{BITS}(x),$$

$$V_{NEW} := V_{NEW} \cup \{x\}$$

4. Přiřad' $V_{RDY} := V_{RDY} \cup V_{NEW}$

□

5.3 Informace pro zlepšení testovatelnosti

Za hlavní úkol metody analýzy testovatelnosti lze jistě považovat poskytnutí informace o testovatelnosti daného obvodu. Ta je většinou vyjádřena formou ohodnocení jistých diagnostických vlastností obvodu - v případě navržené metody analýzy testovatelnosti se jedná o číselné ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti virtuálních portů, bran, portů obvodových prvků a obvodu jako celku. Tato informace, je-li dostupná v "rozumně krátkém čase", pak může být, např. coby součást účelové funkce, s výhodou použita při řešení složitějších problémů mezi které patří např. syntéza pro snadnou testovatelnost či v další kapitole zmíněný problém výběru a rozmístění registrů do scan řetězců a problém generování benchmarkových obvodů. Z výsledků analýz velikostí prohledávacích prostorů výběru a propojení technik návrhu pro snadnou testovatelnost uvedených v odstavci 6.2.3, strana 145, z velké časové složitosti pro řešení MFSP a z požadavku nízké výpočetní složitosti analýzy testovatelnosti¹⁵ totiž plyne, že analýza testovatelnosti není vhodná pro řešení obtížných problémů, mezi které patří např. problém výběru a rozmístění minimálního počtu registrů do scan řetězců či problém výběru nejvhodnější kombinace technik návrhu pro snadnou testovatelnost a problém jejich vzájemného rozmístění a propojení v obvodové struktuře.

¹⁵ blíže viz přehled vybraných společných znaků metod analýzy testovatelnosti, strana 58, druhá odrážka shora

Hlavní vlastnosti algoritmu analýzy testovatelnosti navržené v této práci lze shrnout do následujících bodů:

- algoritmus poskytuje informaci o obtížně testovatelných částech obvodu a o zdrojích obtížné testovatelnosti a číselně ohodnocuje říditelnost, pozorovatelnost a testovatelnost vybraných částí obvodu,
- vzhledem k jeho časové složitosti¹⁶ lze navržený algoritmus považovat za ”rychlý” [Dem02].

Analýze časové složitosti algoritmu je věnován odstavec 6.1, strana 133, proto se jimi nyní zabývat nebudeme. Zaměříme se však na to, jak je možné výsledků analýzy testovatelnosti využít v postupech zabývajících se zlepšením testovatelnosti. Na základě výsledků algoritmu je možno získat zejména následující informace identifikující obtížně testovatelné části v CUA a zdroje této obtížné testovatelnosti - taková místa jsou jistě velmi vhodná např. pro aplikaci některé z technik návrhu pro snadnou testovatelnost:

- **informace o netestovatelných částech obvodu** - množina $\{x \mid x \in VPORT_{CUA} \wedge \nu_C(x) = 0\}$ neřiditelných virtuálních portů, množina $\{x \mid x \in VPORT_{CUA} \wedge \nu_O(x) = 0\}$ nepozorovatelných virtuálních portů, množina $\{x \mid x \in B_{CUA} \wedge \beta_C(x) = 0\}$ neřiditelných bran, množina $\{x \mid x \in B_{CUA} \wedge \beta_O(x) = 0\}$ nepozorovatelných bran, množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NC}(x) = 0\}$ neřiditelných portů, množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NO}(x) = 0\}$ nepozorovatelných portů a množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NC}(x) = \pi_{NO}(x) = 0\}$ netestovatelných portů,
- **informace o částečně testovatelných portech obvodu** - množina $\{x \mid x \in PORT_{CUA} \wedge 0 < \pi_{NC}(x) < \pi_W(x)\}$ částečně říditelných portů, množina $\{x \mid x \in PORT_{CUA} \wedge 0 < \pi_{NO}(x) < \pi_W(x)\}$ částečně pozorovatelných portů a množina $\{x \mid x \in PORT_{CUA} \wedge 0 < \pi_{NC}(x) < \pi_W(x) \wedge 0 < \pi_{NO} < \pi_W(x)\}$ částečně testovatelných portů,
- **informace o testovatelných částech obvodu** - množina $\{x \mid x \in VPORT_{CUA} \wedge \nu_C(x) > 0\}$ říditelných virtuálních portů, množina $\{x \mid x \in VPORT_{CUA} \wedge \nu_O(x) > 0\}$ pozorovatelných virtuálních portů, množina $\{x \mid x \in B_{CUA} \wedge \beta_C(x) > 0\}$ říditelných bran, množina $\{x \mid x \in B_{CUA} \wedge \beta_O(x) > 0\}$ pozorovatelných bran, množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NC}(x) = \pi_W(x)\}$ říditelných portů, množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NO}(x) = \pi_W(x)\}$ pozorovatelných portů a množina $\{x \mid x \in PORT_{CUA} \wedge \pi_{NC}(x) = \pi_{NO} = \pi_W(x)\}$ testovatelných portů,
- **informace o zdrojích obtížné testovatelnosti** - množina $\{x \mid x \in B_{CUA} \wedge \beta_C(x) = 0 \wedge \exists h \in E(G_S), \sigma_S(h) = (u, v): \nu_C(u) > 0 \wedge x \in w_{SX}(h)\} \cup \{x \mid x \in B_{CUA} \wedge \beta_C(x) = 0 \wedge \neg \exists h \in E(G_S), \sigma_S(h) = (u, v): x \in \pi_{BITS}(v)\}$ bran, které jsou kandidáty zdrojů špatné říditelnosti a množina $\{x \mid x \in B_{CUA} \wedge \beta_O(x) = 0 \wedge \exists h \in E(G_I), \sigma_I(h) = (u, v): \nu_O(u) > 0 \wedge x \in w_{IX}(h)\} \cup \{x \mid x \in B_{CUA} \wedge \beta_O(x) = 0 \wedge \neg \exists h \in E(G_I), \sigma_I(h) = (u, v): x \in \pi_{BITS}(u)\}$ bran, které jsou kandidáty zdrojů špatné pozorovatelnosti,

¹⁶ viz věta 7, strana 135 a její důsledek

- **informace o kandidátních registrech pro vložení do scan řetězců**
 - množina $\{x \mid x \in REG_{CUA} \wedge \exists y \in \psi'(x): 0 \leq \pi_{NC}(x) < \pi_W(x) \vee 0 \leq \pi_{NO}(x) < \pi_W(x)\}$. Pro registr x náležící této množině platí x je prvkem $FVS(G)$ vyskytujícím se v kružnici, jejíž datový tok není řídit a/nebo pozorovat¹⁷, kde $G = (V, E)$ je orientovaný graf a $V(G) = E_{CUA}$, $E(G) = \{(x, y) \mid x, y \in V(G) \wedge \exists v_1, v_2 \in VPORT_{CUA}: \beta_E(\pi_{BIT}(v_1, 0)) = x \wedge \beta_E(\pi_{BIT}(v_2, 0)) = y \wedge [(\exists h \in E(G_S) : \sigma_S(h) = (v_1, v_2)) \vee (\exists h \in E(G_I) : \sigma_I(h) = (v_1, v_2))]\}$.

5.4 Shrnutí

Text této kapitoly byl rozčleněn na tři důležité části; první dvě se týkaly navržené metody analýzy testovatelnosti předpokládající generování deterministického testu číslicového obvodu vymezeného modelem popsáním v předchozí kapitole.

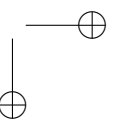
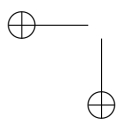
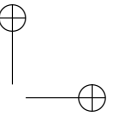
První část se věnovala jak popisu vztahů pro ohodnocení snadnosti konstrukce systémů transparentních cest v CUA , vztahů pro ohodnocení říditelnosti a pozorovatelnosti virtuálních portů a bran v CUA a vztahů pro ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti portů obvodových prvků CUA , tak popisu vztahů pro ohodnocení testovatelnosti obvodového návrhu jako celku. Vztahy patřící do první skupiny lze označit za vztahy pro lokální ohodnocení říditelnosti, pozorovatelnosti resp. pozorovatelnosti, vztahy související s ohodnocením diagnostických vlastností CUA jako celku pak za vztahy pro globální ohodnocení testovatelnosti. Všechny vztahy jsou navrženy tak, aby zkoumaná vlastnost byla ohodnocena relativně ke strukturálním vlastnostem CUA reálným číslem z intervalu $\langle 0; 1 \rangle$. Účelem relativního ohodnocení je umožnit jednoduché číselné porovnání daných vlastností různých CUA , přičemž se může jednat o ohodnocení vlastností naprosto odlišných CUA či modifikací téhož CUA . Důvodů pro používání intervalu $\langle 0; 1 \rangle$ je několik. Prvním je fakt, že tento interval je pro ohodnocení vlastností používán i některými jinými metodami analýzy testovatelnosti, druhým je snadná převoditelnost hodnot z tohoto intervalu do libovolného jiného intervalu. Třetím důvodem je vhodnost intervalu pro vyjádření relativního ohodnocení daných vlastností.

Druhá část kapitoly byla věnována popisu algoritmů popisujících konstrukci množin nutných pro ohodnocení diagnostických vlastností CUA pomocí vztahů zavedených v první části. Algoritmy jsou založeny na prohledávání grafů G_S a G_I ; jejich hlavním cílem je zjistit dostupnost uzlů (virtuálních

¹⁷ tvrzení uvedeno bez důkazu. Důkaz by vycházel z toho, že pro neřiditelný (nepozorovatelný) virtuální port y platí $\nu_C(y) = 0$ ($\nu_O(y) = 0$), že taková kružnice je složena pouze z neřiditelných a/nebo nepozorovatelných virtuálních portů a že zlepšením říditelnosti (pozorovatelnosti) některého z nich dojde ke zlepšení testovatelnosti zbývajících, tj. že dojde k odstranění zpětné vazby z grafu G z pohledu přenosu diagnostických dat

portů) a) grafu G_S z primárních vstupů CUA a ohodnotit říditelnost virtuálních portů, bran a portů obvodových prvků a b) grafu G_I na primárních výstupech obvodu a ohodnotit pozorovatelnost virtuálních portů, bran a portů obvodových prvků. Na základě výše uvedeného, tzv. lokálního ohodnocení říditelnosti a pozorovatelnosti, je ohodnocena testovatelnost portů obvodových prvků a testovatelnost obvodu CUA jako celku.

Třetí část ukázala, jaké konkrétní informace o obtížně testovatelných částech v CUA je možno na základě získaných ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti získat. Daná ohodnocení poskytují jistou informaci o vybraných diagnostických vlastnostech CUA - zejména pak informaci o říditelných a pozorovatelných virtuálních portech, branách a portech obvodových prvků (tj. informaci o tom, které části struktury CUA jsou testovatelné a které ne) a informaci o snadnosti jejich řízení a pozorování (tj. informaci o tom, jaká je cena - měřená délkou posloupnosti podnětů a četností podnětů bran - za jejich řízení a pozorování). Informaci o obtížně testovatelných částech CUA pak lze využít např. k modifikaci stávající struktury CUA za účelem zlepšení testovatelnosti CUA vhodným vložením technik návrhu pro snadnou testovatelnost do struktury CUA či při návrhu obtížně testovatelných (tzv. benchmarkových) obvodů. Příkladem výsledků a ukázkou aplikace navržené metody v oblastech návrhu pro snadnou testovatelnost a návrhu benchmarkových obvodů se zabývá následující kapitola.



6

Ověření algoritmu analýzy testovatelnosti

6.1 Důkazy vybraných vlastností

6.1.1 Správnost a časová složitost algoritmu

V tomto odstavci budou konstatována některá významná tvrzení (věty) týkající se správnosti a časové složitosti navrženého algoritmu analýzy testovatelnosti (algoritmus 4). Kromě nastínění důkazů těchto tvrzení budou představeny některé jejich důsledky. Jádrem navrženého algoritmu analýzy testovatelnosti jsou dílčí algoritmy analýzy říditelnosti (algoritmus 6) a analýzy pozorovatelnosti (algoritmus 10). Ty řeší velmi podobný problém a je tedy možno konstatovat, že výsledky analýzy jednoho algoritmu platí také pro algoritmus druhý a naopak. Proto se v této části budeme věnovat pouze analýze algoritmu 6 s tím, že výsledky této analýzy následně vztáhneme i na algoritmus 10. Před započítím analýzy algoritmů ještě poznamenejme následující:

- cíl úlohy, kterou algoritmus 6 řeší, lze formulovat jako zjištění dostupnosti uzlů (virtuálních portů CUA) grafu G_S z primárních virtuálních portů spojené s ohodnocením říditelnosti označených uzlů takové, aby $\forall x \in V(G_S)$: pro každou orientovanou cestu C z některého primárního virtuálního portu do x obsahující pouze označené uzly platí, že hodnota říditelnosti uzlu x_1 , vyskytujícího se na orientované cestě C před uzlem x_2 , je větší nebo rovna hodnotě říditelnosti uzlu x_2 ; hodnota říditelnosti neoznačených (tj. nedostupných) uzlů bude rovna 0,
- při analýze složitosti algoritmu 4 byla řešena tzv. *analýza nejhoršího případu*¹, jelikož udává časovou složitost potřebnou i pro ty nejsložitější instance dané úlohy a to i v případě, že nejsložitější instance může být velmi nepravděpodobná a "obvyklé" řešení trvá kratší dobu [MŠak01, strana 263],
- vlastnosti grafového algoritmu jsou většinou vyjadřovány pomocí počtu uzlů a hran libovolného grafu, který může být parametrem instance dané

¹ angl. worst case analysis

136 6 Ověření algoritmu analýzy testovatelnosti

úlohy [CLRS01, MŠak01, Dem02]. Je tedy obvyklé, že v případě grafových algoritmů existují dva parametry popisující rozměr vstupu, ne pouze jeden. Této zvyklosti se při analýze vlastností algoritmu 4 budeme v následujícím textu držet i my a vlastnosti grafového algoritmu pracujícího nad grafem G budeme zapisovat jako funkci $|V(G)|$ a $|E(G)|$.

Nyní můžeme představit některá významná tvrzení, jejich důkazy a důsledky.

Věta 6:

Algoritmus 6 skončí práci po konečném počtu kroků. Po zastavení jsou označovány právě ty uzly, do nichž vede z primárních virtuálních portů CUA orientovaná cesta taková, že pro každou řídicí bránu každé z hran v této cestě platí, že alespoň jeden virtuální port, kterému tato brána přísluší, je také označován.

□

Důkaz:

V kroku 2 algoritmu 6 je značka přenesena přes hranu h , $\sigma_S(h) = (x, y)$ pokud jsou splněny následující podmínky: a) značka je v uzlu x , b) uzel y buď značku nemá nebo je označován značkou s menší hodnotou říditelnosti a c) označována je každá brána z množiny $w_{SX}(h)$.

Indukcí dokážeme, že do každého označovaného uzlu vede orientovaná cesta taková, že vede pouze přes označované uzly a pro každou řídicí bránu každé z hran v této cestě platí, že alespoň jeden virtuální port, kterému tato brána přísluší, je také označován. Po provedení inicializace to triviálně platí: označován je každý primární vstupní virtuální port. Nyní předpokládejme, že toto tvrzení platí před provedením kroku 2 a že pro šíření značky byla vybrána hrana h' , $\sigma_S(h') = (x', y')$. Uzel x' tedy má značku a dle indukčního předpokladu existuje orientovaná cesta C z některého z primárních vstupních virtuálních portů do x' a řídicí brány každé z hran na cestě C jsou také označovány. Cesta C navíc prochází pouze přes označované vrcholy a ani ona, ani žádná z označovaných orientovaných cest vedoucích do řídicích bran hran cesty C neobsahuje ani hranu h' , ani vrchol y' . Díky tomu můžeme cestu C prodloužit o hranu h' a vrchol y' , čímž získáme orientovanou cestu z některého z primárních virtuálních portů, která vede přes označované vrcholy. Tvrzení tedy platí i pro označování vrcholu y' .

Obrácenou implikaci dokážeme sporem: Kdyby nebyl po skončení výpočtu označován některý uzel, do něhož vede z primárních virtuálních portů CUA orientovaná cesta taková, že pro každou řídicí bránu každé z hran v této cestě platí, že alespoň jeden virtuální port, kterému tato brána přísluší, je také označován, pak by existovala na této cestě hrana mající označované všechny řídicí brány, jejíž počáteční vrchol by byl označován a koncový buď nikoli nebo by byl označován menší hodnotou říditelností než jakou je možno získat z počátečního uzlu této hrany. To by však byl spor s krokem 2 algoritmu, neboť výpočet by v takové situaci ještě neměl být ukončen.

Při provádění algoritmu 6 zřejmě bude v nejhorším případě nutno pro každý uzel $x \in V(G_S)$ projít všechny orientované cesty v grafu G_S začínající v některém z primárních vstupních virtuálních portů a končící v uzlu x . Jelikož graf G_S je konečný, pak bude konečný i počet orientovaných cest v grafu G_S a počet kroků algoritmu.

□

Na základě důsledku následující věty bude možné určit časovou složitost navrženého algoritmu analýzy testovatelnosti. Z důvodu přehledného odvození časové složitosti algoritmu 6 převedeme řešený problém na problém nalezení nejkratších cest². Nejprve však bude třeba definovat následující zobrazení.

Definice 32:

Nechť x_{CO} je hodnota říditelnosti virtuálního portu x obsažená v 5-tici $(x, x_{seq}, x_{sti}, x_{mul}, x_{CO}) \in \nu_C$. Na množině $V(G_S)$ definujeme zobrazení $w_V: V(G_S) \rightarrow \mathbb{R}$ přiřazující každému uzlu $x \in V(G_S)$ jeho vzdálenost od primárních vstupů CUA podle předpisu

$$x \mapsto \begin{cases} \infty & \text{je-li } x_{CO} = 0 \\ \frac{1}{x_{CO}} & \text{jinak} \end{cases}$$

a zobrazení $w_E: E(G_S) \rightarrow \mathbb{R}$ přiřazující každé hraně $h \in E(G_S)$, $\sigma_S(h) = (x, y)$ její délku předpisem $h \mapsto w_V(y) - w_V(x)$.

Délku cesty C tvořené hranami h_1, \dots, h_n definujeme jako $\sum_{i=1}^n w_E(h_i)$.

□

Nyní lze vyslovit větu týkající se složitosti ohodnocení jednoho uzlu v grafu G_S .

Věta 7:

Po i -tém provedení kroku 2 (pro $i = 0, 1, 2, \dots, |V(G_S)|$) platí:

$\forall x \in V(G_S)$: $w_V(x)$ je menší nebo rovno délce nejkratší cesty z primárních virtuálních portů CUA do x , která má nejvýše i hran.

□

Důkaz:

Pro $i = 0$ tvrzení triviálně platí. Předpokládejme, že tvrzení platí pro nějaké i . Dokážeme, že bude platit také po dalším provedení kroku 2 algoritmu 6, tj. pro $i + 1$. Vezměme libovolný vrchol x , označme $d_{i+1}(x)$ délku cesty, která je nejkratší ze všech cest z primárních virtuálních portů do x , které mají nejvýše

² připomeňme však, že ohodnocení hran a uzlů grafu G_S nejsou, narozdíl od grafu G_S , jedněmi ze vstupů instance problému, ale že jsou postupně získávány během zjišťování dostupnosti uzlů grafu G_S z primárních virtuálních portů. Ohodnocení uzlů a hran grafu G_S tedy nejsou konstantní, ale mění se (k lepšímu) s každou iterací algoritmu 6

138 6 Ověření algoritmu analýzy testovatelnosti

$i + 1$ hran, a dále označme y předposlední uzel v této cestě. Počáteční úsek této cesty z primárních virtuálních portů do y má nejvýše i hran a jeho délka $d_i(y)$ je nejkratší ze všech cest z primárních virtuálních portů do y , které mají nejvýše i hran. Podle indukčního předpokladu po i -tém vykonání kroku 2 algoritmu 6 bylo $w_V(y) \leq d_i(y)$, a proto po dalším kroku 2, kdy musela být zpracována i hrana (y, x) , musí platit $w_V(x) \leq d_{i+1}(x)$.

□

Důsledek:

Krok 2 algoritmu 6 může být opakován nejvýše $|E(G_S)|$ -krát. Časová složitost algoritmu 6 je tedy $O(|V(G_S)| \cdot |E(G_S)|)$.

Po skončení algoritmu 6 platí $\forall h \in E(G_S), \sigma_S(h) = (x, y): w_V(x) \leq w_V(y)$, tj. ohodnocení uzlů hodnotami říditelnosti je provedeno v souladu s definicí 26, strana 110.

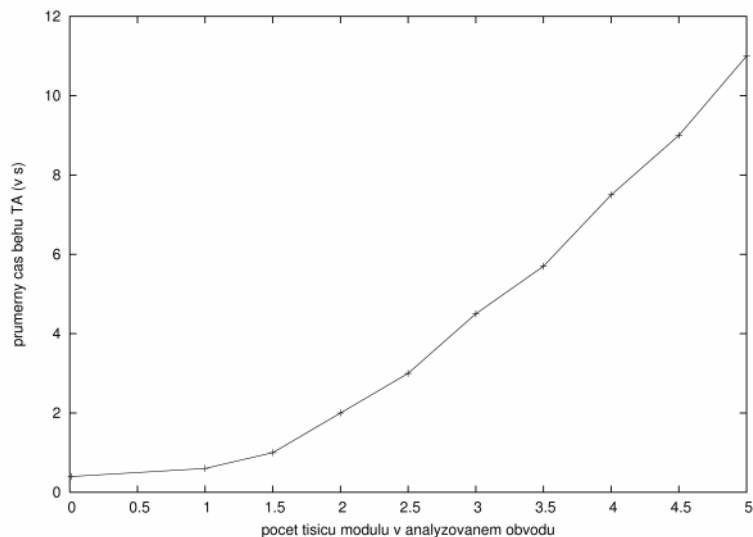
□

Časové složitosti pěti dílčích kroků navrženého algoritmu analýzy testovatelnosti (algoritmus 4) jsou: krok 1: $O(konst.)$, krok 2: $O(|V_{G_S}| + |E_{G_S}| + |V_{G_I}| + |E_{G_I}|)$, krok 3: $O(|V_{G_S}| \cdot |E_{G_S}|)$, krok 4: $O(|V_{G_I}| \cdot |E_{G_I}|)$ a krok 5 $O(|V_{G_S}| + |E_{G_S}| + |V_{G_I}| + |E_{G_I}|)$. Algoritmus 4 tedy pracuje se složitostí $O(|V_{G_S}| \cdot |E_{G_S}| + |V_{G_I}| \cdot |E_{G_I}|)$.

Obr. 27 ukazuje experimentálně zjištěnou závislost času CPU potřebného k provedení analýzy testovatelnosti na tisících modulech obsažených v analyzovaném obvodu. Z grafu je možno vidět, že závislost má lineární charakter. K analýze byly použity benchmarkové obvody ze sady [PEC07]. Závislost byla publikována v [Str06].

6.2 Experimentální výsledky a příklady aplikace metody

V této části budou představeny některé z experimentálních výsledků dosažených s využitím navržené metody analýzy testovatelnosti [ZK01d, Str02b, JS02c, JS02b, JS02a, Str03d, Str03c, Str03b, Str03a, ZK03a, ZK04b]. Text bude rozdělen do několika odstavců, přičemž každý odstavec bude věnován samostatné oblasti. První odstavec shrne současný stav a možnosti pro ověřování a vzájemné srovnávání nově navržených postupů a nástrojů z oblasti návrhu a diagnostiky elektronických systémů. Druhý odstavec bude zaměřen na experimentální výsledky dosažené metodou analýzy testovatelnosti navrženou v této práci. Ve třetím odstavci bude ukázán příklad aplikace navržené metody v návrhu pro snadnou testovatelnost s využitím techniky scan s cílem nalezení - vzhledem k zadaným návrhovým omezením - co nejvhodnějšího rozmístění registrů do scan řetězů. V posledním - čtvrtém - odstavci budou stručně shrnuty výsledky dosažené aplikací metody v oblasti generování benchmarkových obvodů.



Obrázek 27. Průměrný čas CPU potřebný pro analýzu testovatelnosti

6.2.1 Komentář k benchmarkovým obvodům

Jeden z nejobtížnějších úkolů, kterým je třeba při návrhu nových postupů, nástrojů atp. v diagnostice číslicových obvodů čelit, je jejich zhodnocení a srovnání s již existujícími postupy, nástroji atp. Zřejmě je výhodné, existují-li data, která mohou být použita jako vstup různých postupů řešících stejný nebo velmi podobný problém z téže oblasti; vychází se přitom z předpokladu, že budou-li dané postupy aplikovány na stejnou množinu dat, pak bude jejich srovnání dostatečně objektivní.

V oblastech zabývajících se návrhem a diagnostikou elektronických systémů je obvyklé, že různé postupy, nástroje atp. jsou srovnávány na základě výsledků získaných jejich aplikací na testovací (srovnávací), tzv. *benchmarkové obvody*. Tyto obvody bývají sdružovány do tzv. *sad (souborů) benchmarkových obvodů*, zkráceně do *benchmarkových sad*; každá sada pak obsahuje obvody, splňující jistá kritéria - např. obvody reprezentující specifickou oblast návrhu či diagnostiky, obvody vyznačující se předem definovanými strukturálními vlastnostmi vhodnými k zátěžovému prověření postupů či nástrojů z dané oblasti či obvody popsané na stejné úrovni abstrakce - např. pro srovnání postupů pro vysokoúrovňovou syntézu jsou vyžadovány benchmarkové obvody popsané na úrovni chování zatímco pro srovnání postupů provádějících implementaci systému je třeba použít obvody zadané nízkourovňovým fyzickým

140 6 Ověření algoritmu analýzy testovatelnosti

popisem. Příklad oblastí, pro které jsou k dispozici benchmarkové obvody, je shrnut v tabulce 1 [Brg04].

Je možno konstatovat, že benchmarkové sady představují ověřovací data, která jsou uznávána producenty nových postupů a nástrojů z oblasti návrhu a diagnostiky elektronických systémů a používána k jejich vzájemnému srovnání. Přesto, že se v praxi ukazuje, že zcela objektivního srovnání postupů a nástrojů řešících blízké problémy založeného na výsledcích dosažených pouze pomocí benchmarkových obvodů lze jen ztěžím dosáhnout, benchmarkové sady v současné době představují prakticky jedinou možnost řádného srovnání existujících postupů a nástrojů [Har00].

Tabulka 1. Přehled vybraných benchmarkových sad a jejich rozčlenění podle předpokládaných oblastí aplikace

| Oblast | Benchmarkové sady |
|-----------------------------------|--|
| Simulace | CircuitSim90 |
| Fyzická implementace | Compaction86, PRWorkshop88, Modgen89, LayoutSynth90, PDWorkshop91, LayoutSynth92, PDWorkshop93 |
| Generování testu na úrovni hradel | ISCAS85, ISCAS89 |
| Logická syntéza | LGSynth89, LGSynth91, LGSynth93, LGSynth95 |
| Vysokoúrovňová syntéza | HLSynth89, HLSynth91, HLSynth92, HLSynth95 |

K dalším nevýhodám současných benchmarkových sad patří skutečnost, že neobsahují dostatečný počet dostatečně rozsáhlých a složitých obvodů, které by zároveň bylo možno považovat za reprezentativní zástupce co největšího počtu skupin obvodových aplikací. Ideálním se může jevit vytvářet benchmarkové sady z komerčních obvodových návrhů, avšak vzhledem k ochraně vlastnických práv těchto návrhů není možné sestavit takové sady v době, kdy jsou tato práva platná a obvodový návrh dostatečně zajímavý z konkurenčního hlediska. Volné zpřístupnění komerčního návrhu tedy může trvat velmi dlouhou dobu - většinou se pak jedná o návrhy, které již nejsou komerčně zajímavé. Takové, mnohem dříve vytvořené, návrhy jsou pak často podstatně odlišné od návrhů vzniklých pomocí moderních návrhových trendů a tedy mohou být obtížně použitelné k ověřování moderních postupů a nástrojů předpokládajících obvody s jinými vlastnostmi, složitostí, popsané na jiných úrovních popisu atp.

6.2.2 Analýza testovatelnosti

V této části budou představeny výsledky dosažené při aplikaci navržené metody analýzy testovatelnosti na vybrané číslicové obvody [ZK01d, Str02b, JS02c, JS02b, JS02a, Str03d, Str03b, ZK03a, ZK04b]; pro účely experimentálního ověření byla metoda implementována v jazyce C++. Nejprve bude metoda aplikována na obvod obsahující zanořené zpětné vazby. Cílem bude na tomto jednoduchém obvodu ukázat, jak hodnota říditelnosti, pozorovatelnosti a testovatelnosti závisí na dostupnosti uzlů z primárních vstupů

a výstupů *CUA*. Poté bude metoda aplikována na vybrané benchmarkové obvody používané pro účely ověřování obdobných metod z diagnostiky číslicových obvodů.

Výsledky ohodnocení testovatelnosti každého ze zkoumaných obvodů budou shrnuty do tabulky, která bude doplněna komentářem k těmto výsledkům. Každá tabulka výsledků bude zařazena do přílohy a bude mít jednotný formát s následující strukturou: první řádek tabulky bude tvořen záhlavím, poslední dva řádky budou informovat o globální testovatelnosti daného obvodu. Každý ze zbývajících řádků pak bude příslušet konkrétnímu datovému portu z množiny $PORT_{CUA}$. Tabulka bude mít celkem čtyři sloupce. První sloupec zleva bude obsahovat jméno portu (v případě řádku příslušejícího portu) resp. sledovanou charakteristiku daného obvodu (v případě dvou posledních řádků). Další sloupce ve směru zleva doprava budou obsahovat informaci o hodnotě říditelnosti, pozorovatelnosti a testovatelnosti portů (v případě řádku příslušejícího portu) resp. informaci o hodnotě říditelnosti, pozorovatelnosti a testovatelnosti obvodu (v případě předposledního řádku) a informaci o počtu/procentu říditelných, pozorovatelných a testovatelných portů v obvodu (v případě posledního řádku).

Obvod se zanořenými zpětnými vazbami

Nejprve nechtě je navržená metoda aplikována na obvod NLoops (obrázek 15a, strana 44) se zanořenými zpětnými vazbami. Původní struktura obvodu je tvořena třemi moduly (*MOD1*, *MOD2*, *MOD3*) provádějícími nějakou funkci a třemi registry (*REG1*, *REG2*, *REG3*) sloužícími k uložení mezivýsledků a synchronizaci datového toku. Analýze obvodových zpětných vazeb jsem se věnoval v publikacích [FZ02, Str03c, Str03a].

Aplikací navržené metody analýzy testovatelnosti na původní obvodovou strukturu (obrázek 15a) bude tato struktura ohodnocena hodnotami shrnutými v tabulce 3. Obvod NLoops má celkem 21 portů, z toho 2 porty (*clk*, *In*) jsou primární vstupní, 1 port (*Out*) je primární výstupní a 18 portů tvoří rozhraní šesti obvodových prvků obvodu NLoops. Celkem je v portech obvodu NLoops obsaženo 140 bran. Z tohoto počtu je 17 portů datových, z toho 2 primární datové porty a 15 datových portů z rozhraní obvodových prvků obvodu NLoops. Celkem je v datových portech obvodu NLoops obsaženo 136 datových bran.

Ze sloupce pro říditelnost je patrné, že pouze 16 datových bran ze 136 (tj. 11.8 %) je říditelných - jedná se o brány primárního vstupního portu NLoops.In a o brány portu FU1.a spojeného s portem NLoops.In - porty, kterým tyto brány přísluší, mají nenulovou hodnotu říditelnosti. Obdobně, ze sloupce pro pozorovatelnost je patrné, že pozorovatelných je pouze 32 datových bran z celkového počtu 136 bran. Sloupec pro testovatelnost shrnuje výsledky obou sloupců, z nichž plyne, že žádný z datových portů není současně říditelný i pozorovatelný a tedy ani testovatelný; hodnota testovatelnosti každého z datových portů je tedy rovna hodnotě 0. Na základě těchto lokálních

142 6 Ověření algoritmu analýzy testovatelnosti

ohodnocení jsou ve dvou posledních řádcích vyhodnocena globální ohodnocení diagnostických vlastností datových cest obvodu. Předposlední řádek přiřazuje obvodu hodnoty říditelnosti (0.065744), pozorovatelnosti (0.141732) a testovatelnosti (0.009318). Ty, vzhledem ke své blízkosti hodnotě 0, indikují poměrně špatnou říditelnost, pozorovatelnost a testovatelnost obvodu NLoops. O špatných diagnostických vlastnostech obvodu také vypovídá již zmíněná informace z posledního řádku, tj. že pouze 16 datových bran ze 136 je říditelných, 32 je pozorovatelných a žádná není současně říditelná i pozorovatelná.

Příčina špatné testovatelnosti obvodu NLoops spočívá ve výskytu zpětných vazeb v obvodu NLoops - viz příklad 2, strana 44, kde bylo ukázáno, že zlepšení testovatelnosti obvodu NLoops lze dosáhnout několika různými způsoby, z nichž každý představuje jistou modifikaci původní obvodové struktury obvodu NLoops. V následujícím textu bude několik z modifikací obvodu NLoops představeno a ke každé z nich budou shrnuty její diagnostické vlastnosti ve formě tabulky a stručného komentáře.

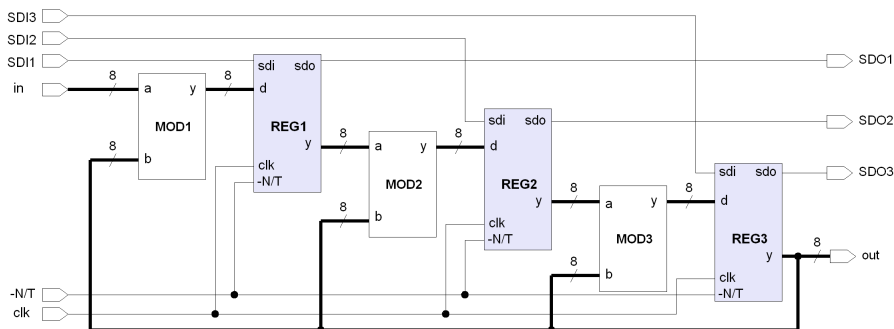
- První modifikace (NLoops1) bude spočívat v zařazení všech tří registrů obvodu NLoops do jediného scan řetězu dle obrázku 16a, strana 45. Výsledky analýzy testovatelnosti takového zapojení jsou prezentovány v tabulce 4. Oproti původní struktuře obvodu NLoops se struktura NLoops1 odlišuje zejména následujícími parametry danými modifikací registrů na scan registry a jejich zapojením do scan řetězu: větší počet portů a bran, nové datové cesty, změna dynamických vlastností obvodu, nárůst plochy a další. Počet portů narostl o 3 jednobitové porty na jeden registr (celkem o 9 portů pro všechny registry), o 2 primární vstupní jednobitové porty a 1 primární výstupní jednobitový port, celkem tedy o 12 jednobitových portů, tj. o 12 bran. Celkem je v NLoops1 obsaženo 30 portů a $136 + 12 = 148$ bran, z toho 25 datových portů sestavených ze 144 bran. Z hlediska testovatelnosti představuje struktura NLoops1 zřejmě lépe testovatelnou variantu obvodu NLoops, jelikož hodnoty říditelnosti, pozorovatelnosti a testovatelnosti obvodu NLoops1 jsou mnohem větší než u původního obvodu NLoops. Rovněž počet říditelných, pozorovatelných a testovatelných bran je u NLoops podstatně větší než u NLoops. Dokonce je touto modifikací dosaženo testovatelnosti všech datových portů obvodu. Během dalších činností jako např. při snahách o generování co nejkratšího testu, snahách o minimalizaci plochy výsledného čipu by však mohly vyplynout nevýhody, a to zejména neefektivnost (z pohledu ceny za zlepšení testovatelnosti) plynoucí z tohoto zapojení.
- Druhá modifikace (NLoops2) obvodu NLoops opět spočívá v zařazení všech tří registrů do scan řetězů; avšak tak, že každý ze scan registrů je umístěn v samostatném scan řetězu dle obrázku 28. Jelikož sériové cesty scan řetězců v NLoops2 jsou na sobě nezávislé, očekává se od této modifikace lepší ohodnocení testovatelnosti než v případě varianty NLoops1, avšak opět za cenu jistých nevýhod, z nichž některé jsou společné s předchozí

modifikací (např. přílišný nárůst plochy) a některé jsou dokonce větší (např. nárůst počtu primárních vstupů a výstupů). Současně se očekává, že touto modifikací nedojde (oproti NLoops1) ke změně počtu testovatelných datových portů, tj. že i v případě NLoops2 budou testovatelné všechny datové porty. Výsledky testovatelnosti pro NLoops2 je možno nalézt v tabulce 5; z větších hodnot při ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti vyplývá, že u modifikace NLoops2 bylo splněno očekávání lepší testovatelnosti než u předchozí modifikace (NLoops1).

- Třetí modifikace (NLoops3) obvodu využije závěru učiněného v příkladu 2, strana 44. Z něj plyne, že za nejefektivnější způsob dosažení největšího zlepšení testovatelnosti obvodu NLoops za co nejnižší ceny lze považovat takovou modifikaci struktury NLoops, která zajistí rozbití nejvíce zanořené smyčky. V příkladu 2 jsou navrženy dvě varianty rozbití smyčky - první varianta navrhuje modifikovat registr *REG3* na scan registr (obrázek 16b, strana 45). Druhá varianta navrhuje rozbití smyčku vložením multiplexoru *MUX1* (obrázek 16c). Každá z těchto možností představuje variantu modifikace struktury NLoops. K oběma bude následovat krátký komentář z pohledu jejich testovatelnosti a její ceny. Výsledky testovatelnosti pro variantu (NLoops3) se scan registrem *REG3* lze nalézt v tabulce 6, výsledky pro variantu (NLoops4) s multiplexorem *MUX1* lze nalézt v tabulce 7. Z obou tabulek je patrné, že testovatelnost variant NLoops3 a NLoops4 je velmi blízká testovatelnosti varianty NLoops2 a opět je dosaženo testovatelnosti všech datových portů. Z těchto tří variant je zřejmě díky možnosti současného nastavení několika testovacích vzorků a snímání odezev nejlépe ohodnocena varianta NLoops2, o něco hůře varianta NLoops4 umožňující "vnutit" požadovaná data do nejvíce zanořené smyčky pomocí multiplexoru a nejhůře varianta NLoops3 umožňující totéž, avšak sériovou cestou pomocí scan registru. Přestože oproti variantě NLoops4 varianta NLoops3 navíc poskytuje možnost pozorování odezev z nejvíce zanořené cyklu sériovou cestou, zřejmě není vzhledem k malé velikosti obvodu a poloze nejvíce zanořené cyklu příliš výhodné tuto cestu používat. To je indikováno tím, že hodnota pozorovatelnosti je nižší, než u variant NLoops2 a NLoops3, i přesto, že je k dispozici nový primární výstup pro sledování odezev.

V dalším textu budou komentovány výsledky analýzy testovatelnosti několika vybraných bencharkových obvodů. K tomuto účelu byly vybrány obvody, které odpovídají úrovni popisu dané modelem zavedeným v kapitole 4 a které byly použity v dříve publikovaných pracích zaměřených na testovatelnost číslicových obvodů. Přesto, že se jedná o poměrně jednoduché obvody, jsou velmi často používány k ověřování a srovnávání různých algoritmů řešících dané problémy. V dalším textu budou tyto obvody použity jak k ověření navržené metody analýzy testovatelnosti a prezentaci jejích výsledků aplikovaných na tyto obvody, tak ke srovnání jejích výsledků s dostupnými výsledky dříve publikovaných metod. Aplikacemi navržené metody analýzy

144 6 Ověření algoritmu analýzy testovatelnosti



Obrázek 28. Ilustrace k zapojení scan registrů do samostatných scan řetězců

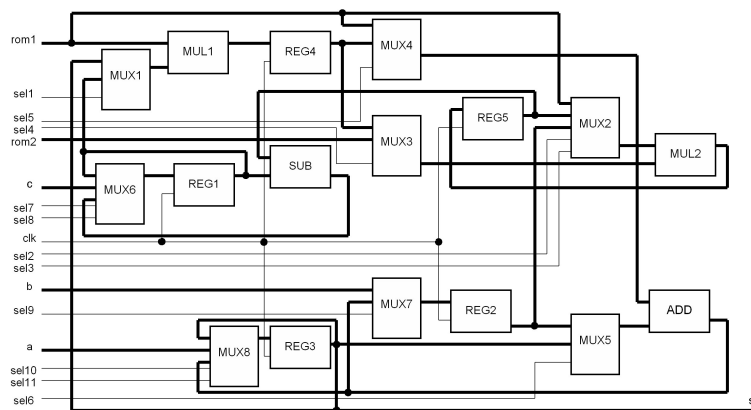
testovatelnosti se budou zabývat až odstavce 6.2.3 a 6.2.4. Výsledky výzkumu v oblasti analýzy testovatelnosti a základní principy navržené metody analýzy testovatelnosti jsem publikoval v pracích [ZK01d, ZK01b, ZK01c, ZK01a, JH01, Str02b, JS02c, JS02a, Str03b, Str03c, ZK04a]. Příkladem aplikace této metody v oblastech souvisejících s testovatelností číslicových obvodů jsem se věnoval v publikacích [ZK01d, Str02a, JS02c, JS02b, Str03d, ZK03a, Str03b, ZK04b, ZK04a].

Obvod Bert

Nejprve budou prezentovány výsledky analýzy testovatelnosti obvodu *Bert* [DBR98]. Pro ověřování metody analýzy testovatelnosti a metod pro aplikaci technik návrhu pro snadnou testovatelnost byl tento obvod použit např. v práci [Buk00]. Výsledky analýzy testovatelnosti obvodu Bert, obsahujícího celkem 440 datových bran, dosažené pomocí metody navržené v této práci jsou shrnuty v tabulce 8. Jak z této tabulky, tak z práce [Buk00] plyne, že v tomto obvodu se nevyskytuje netestovatelný datový port. To znamená, že již nelze dosáhnout většího počtu testovatelných datových portů. Lze však, za cenu nárůstu plochy, počtu primárních vstupů a výstupů atd. zlepšit snadnost jejich testování a to např. vhodným vložením technik návrhu pro snadnou testovatelnost.

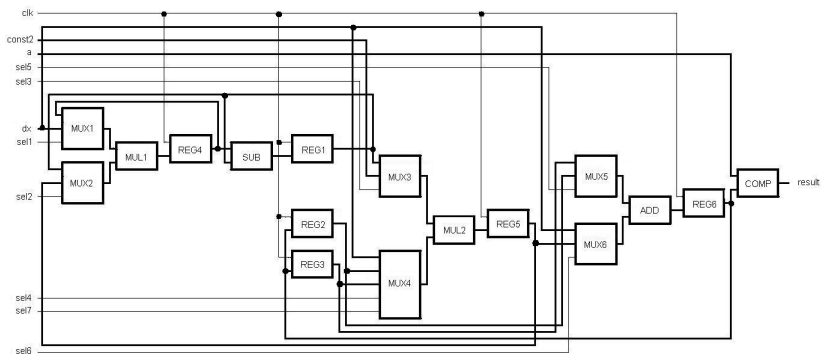
Obvod Diffeq

Druhým obvodem, jehož testovatelnost bude komentována, je obvod *Diffeq* [PGPG86]. Tento obvod je součástí benchmarkové sady HLSynth92 a pro ověřování metod analýzy testovatelnosti byl použit např. v pracích [Buk00, Růž02]. Výsledky analýzy testovatelnosti obvodu Diffeq, obsahujícího celkem 400 datových bran, dosažené pomocí metody navržené v této práci jsou shrnuty v tabulce 9. Z tabulky i z prací [Buk00, Růž02] je patrné, že obvod Diffeq je díky datovým závislostem v datových cestách obtížně testovatelným.



Obrázek 29. Schéma obvodu Bert

Z tabulky plyne, že 56 % datových bran je říditelných, 28 % datových bran pozorovatelných a ani jedna brána není testovatelná, tj. současně říditelná i pozorovatelná. Celková testovatelnost obvodu Diffeq je ohodnocena hodnotou 0.073250, která je velmi blízká hodnotě 0 představující netestovatelný obvod.



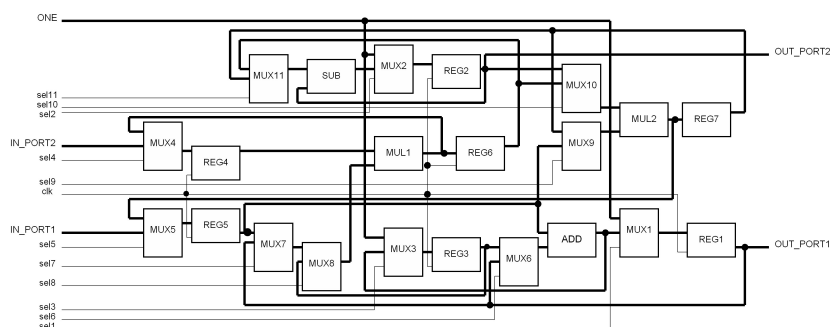
Obrázek 30. Schéma obvodu Diffeq

Obvod Paulin

Třetím z benchmarkových obvodů, použitým v této práci, je obvod *Paulin*. Ten je použit např. v práci [IGJ96a] k ověření metody návrhu pro snadnou testovatelnost založené na analýze datových cest a řadiče daného obvodu.

146 6 Ověření algoritmu analýzy testovatelnosti

Jelikož výsledky analýzy testovatelnosti nejsou v [IGJ96a] k dispozici a ani v jiné práci nebyly nalezeny, nebude možné výsledky pro obvod Paulin srovnat s výsledky jiných metod, ale pouze k prezentaci výsledků dosažených metodou navrženou v této práci. Výsledky analýzy testovatelnosti obvodu Paulin jsou shrnuty v tabulce 10. V obvodě je celkem 512 datových bran, přičemž všechny brány jsou testovatelné.



Obrázek 31. Schéma obvodu Paulin

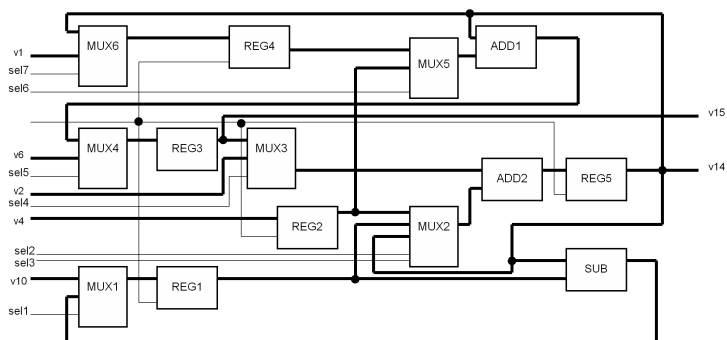
Obvod Tseng

Posledním z benchmarkových obvodů, jehož testovatelnost bude komentována, je obvod [PGPG86]. Příklady metod, které byly pomocí tohoto obvodu ověřovány, lze nalézt v [IGJ96a, Buk00, Růž02], přičemž v [Buk00, Růž02] byl obvod použit k ověření publikované metody pro analýzu testovatelnosti. Zatímco metodou publikovanou v práci [Buk00] byly ve struktuře obvodu Tseng nalezeny obtížně testovatelné části, tak metodou [Růž02] bylo zjištěno, že v obvodě Tseng se nevyskytuje datová brána, která by byla netestovatelná. Rovněž z výsledků metody navržené v této práci a shrnutých v tabulce 11 vyplývá, že všechny datové brány v obvodě Tseng jsou testovatelné.

6.2.3 Návrh pro snadnou testovatelnost pomocí techniky scan

Jednou z oblastí, ve kterých byla ověřena použitelnost výsledků navržené metody analýzy testovatelnosti, je oblast návrhu pro snadnou testovatelnost. Jelikož metody a prostředky návrhu pro snadnou testovatelnost tvoří samostatnou a velmi rozsáhlou oblast diagnostiky číslicových obvodů (přehled viz odstavec 2.3.3, strana 21), byl náš dosavadní výzkum zaměřen pouze na úzkou část této oblasti, a to na tzv. techniku sériový scan - viz odstavec 3.1, strana 35 a práce [ZK01d, JS02c, JS02b, Str02a, Str03d, Str03c, Str03b, Str03a, ZK03a, ZK04a, Str06]. Tento odstavec tedy bude věnován využití

6.2 Experimentální výsledky a příklady aplikace metody 147



Obrázek 32. Schéma obvodu Tseng

výsledků navržené metody analýzy testovatelnosti pro *výběr a rozmístění registrů do scan řetězců* s cílem dosažení co nejlepší testovatelnosti výsledného návrhu za co nejmenší cenu úměrnou velikosti zásahu do původní obvodové struktury.

Komentář k prohledávacímu prostoru a používané notaci

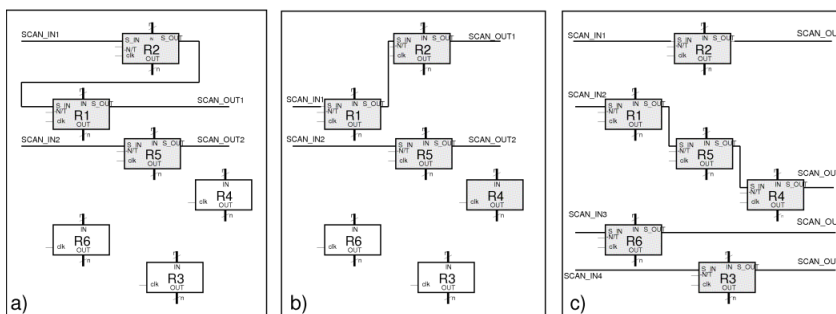
Aby bylo možno zkoumat, které z rozmístění registrů do scan řetězců představuje nejvhodnější alternativu, ukázalo se vhodným zabývat se nejprve hledáním odpovědí na následující otázky:

- **jaká je velikost prohledávacího (stavového) prostoru možných rozmístění registrů do scan řetězců?** Odpověď na tuto otázku bude mít vliv na volbu metody prohledávání stavového prostoru. Pokud bude prostor shledán "malým", pak lze předpokládat, že prohledávací algoritmus bude poměrně jednoduchý a zaručí nalezení hledaného řešení. V opačném případě bude nutno k prohledávání použít některou nebo více z optimalizačních metod obecně nezaručujících nalezení hledaného řešení,
- **jakým způsobem lze formálně popsat rozmístění daného počtu registrů ve scan řetězcích a jakou datovou reprezentaci je vhodné pro tento formální popis zvolit?** Odpověď na tyto otázky bude mít vliv nejen na přehlednost a jednoznačnost docílenou formálním zápisem, ale především na snadnost a způsob implementace prohledávacího algoritmu.

Odpovědí na tyto otázky byla věnována jistá část výzkumu k tématu této práce [JS02b, JS02c, Str03d, Str03c, ZK03a, Str06] a proto se jim bude stručně věnovat také následující text. Předtím je však třeba se seznámit s několika základními myšlenkami týkajícími se výběru a rozmístění registrů do scan řetězců. K tomuto účelu bude využito obrázku 33. Ten je složen ze tří částí, přičemž v každé z nich je do scan řetězců vybráno a různě rozmístěno k z celkem 5 registrů. Některé z registrů jsou modifikovány na scan registry

(každý z takových registrů má mj. k dispozici sériový vstup S_IN, sériový výstup S_OUT a vstup pro přepínání mezi paralelním a sériovým režimem činnosti), zbylé registry jsou ”klasické” registry plnící pouze funkci paralelního n -bitového registru.

Nyní se věnujeme principům rozmístění registrů do scan řetězců. V části a) obrázku 33 jsou do scan řetězců vybrány (tj. na scan registry modifikovány) registry R1, R2 a R5. Ty jsou dle schématu rozmístěny tak, že registry R2 a R1 tvoří, v uvedeném pořadí³, jeden scan řetězec a registr R5 tvoří druhý scan řetězec. V části b) obrázku 33 jsou do scan řetězců vybrány opět registry R1, R2 a R5, avšak tentokrát jsou rozmístěny tak, že registry R1 a R2 tvoří, v uvedeném pořadí, jeden scan řetězec a registr R5 tvoří druhý scan řetězec. Oproti schématu a) je ve schématu b) pořadí registrů R1, R2 ve scan řetězu obrácené. V části c) obrázku 33 je do scan řetězců vybráno všech pět registrů a to tak, že každý z registrů R2, R3 a R6 tvoří samostatný scan řetězec a registry R1, R5 a R4 tvoří, v uvedeném pořadí, další scan řetězec. Celkem jsou na každém z obrázků 33a, 33b dva scan řetězce a na obrázku 33c čtyři scan řetězce.



Obrázek 33. Ilustrace k výběru a rozmístění registrů do scan řetězců

Z obrázku 33 jsou patrné zejména následující skutečnosti:

- z celkového počtu n registrů obsažených v obvodu lze do scan řetězců vybrat $k \in \{1, \dots, n\}$ registrů⁴. V rámci konkrétních scan řetězců je pak třeba stanovit pořadí jednotlivých scan registrů,
- pro libovolný obvod obsahující celkem n registrů platí, že různé varianty výběru a rozmístění registrů do scan řetězců lze reprezentovat nezávisle na konkrétní struktuře daného obvodu. Velikost prohledávacího prostoru pak

³ např. práce [YBP02, SGT03] ukazují, že je nezbytné zabývat se nejen řešením problému výběru registrů do scan řetězců, ale také řešením problému jejich pořadí ve scan řetězcích

⁴ hodnotu 0 neuvažujeme, protože odpovídá případu, kdy není vybrán ani jeden registr

6.2 Experimentální výsledky a příklady aplikace metody 149

nezávisí na struktuře konkrétního obvodu, ale pouze na celkovém počtu registrů v obvodě. Analýza vlivu daného výběru a rozmístění na diagnostické a další vlastnosti obvodu, jako např. plocha, počet vstupů a výstupů či dynamické parametry, se však neobejde bez analýzy vlastností daného obvodu,

- i z této jednoduché ilustrace je patrné, že slovní popis výběru a rozmístění registrů do scan řetězců může někdy působit komplikovaně a nejednoznačně. Proto bude v následujícím textu následovat notace formálně popisující rozmístění k vybraných registrů do scan řetězců.

Pro úspěšné zodpovězení výše položených otázek necht' je dán nějaký obvod CUA popsatelný modelem z kapitoly 4 a necht' $n = |REG_{CUA}|$ je číslo vyjadřující počet registrů v obvodu CUA . Pro obrázek 33 platí $REG_{CUA} = \{R_1, R_2, R_3, R_4, R_5, R_6\}$. Zavedme množinu $SREG_{CUA} = \{R'_1, \dots, R'_k\} \subseteq REG_{CUA}$, která bude obsahovat $k \in \{1, \dots, n\}$ registrů vybraných do scan řetězců, přičemž účel a způsob výběru těchto registrů ponechme nyní stranou.

Nejprve zkoumejme, kolika způsoby je možné rozmístit registry z množiny $SREG_{CUA}$ do scan řetězců tak, že při každém rozmístění bude vždy použito všech k registrů z množiny $SREG_{CUA}$. Současně předpokládejme, že v úvahu bude brána pouze přítomnost registrů ve scan řetězcích bez ohledu na jejich vzájemné pořadí ve scan řetězcích. Za tohoto předpokladu je možné si každé takové rozmístění představit jako rozklad množiny $SREG_{CUA}$ takový, že registry x, y náležejí téže rozkladové třídě právě když jsou umístěny do téhož scan řetězce. Pro obrázky 33a a 33b pak platí $SCAN_{CUA} = \{R_1, R_2, R_3\}$ a vyobrazená rozmístění lze zapsat jako rozklad $\{\{R_1, R_2\}, \{R_5\}\}$. Pro obrázek 33c platí $SCAN_{CUA} = REG_{CUA}$ a vyobrazené rozmístění lze zapsat jako rozklad $\{\{R_2\}, \{R_1, R_5, R_4\}, \{R_6\}, \{R_3\}\}$. Za předpokladu, že na pořadí registrů ve scan řetězcích není brán ohled, je počet rozmístění obsahujících právě k registrů roven počtu rozkladů k -prvkové množiny. Ten je roven tzv. BELLOVU číslu b_k [CG96], pro které platí

$$b_k = \sum_{i=0}^{k-1} \left[\binom{k-1}{i} \times b_i \right], \text{ kde } b_0 = 1. \quad (1)$$

Vztah pro výpočet b_k lze také zapsat pomocí tzv. STIRLINGOVA čísla $S2(k, i)$ druhého řádu (druhu) vyjadřujícího počet rozkladů k -prvkové množiny na i rozkladových tříd:

$$b_k = \sum_{i=0}^k S2(k, i). \quad (2)$$

Čísla b_k a $S2(k, i)$ pro $k, i = 1, \dots, 10$ jsou uvedena v tabulce 12, strana 181. I pro takto malá k lze v tabulce vidět, že hodnota b_k (počet možných rozmístění právě k scan registrů do scan řetězců) roste velmi rychle s pomalu rostoucím k - pro $k = 1$ existuje jen jedno rozmístění, pro $k = 2$ existují dvě rozmístění, ..., pro $k = 10$ existuje 115975 rozmístění atd. Pro zjištění

150 6 Ověření algoritmu analýzy testovatelnosti

celkového počtu možností výběru a rozmístění n registrů do scan registrů pak zbývá sečíst počet rozmístění scan registrů z každé z podmnožin množiny REG_{CUA} . Jelikož z n -prvkové množiny lze vybrat celkem $\binom{n}{k}$ k -prvkových podmnožin a počet možných rozmístění právě k scan registrů do scan řetězců je roven b_k , pak celkový počet ($s1_n$) možností výběru a rozmístění n registrů do scan registrů lze vyjádřit vztahem

$$s1_n = \sum_{k=1}^n \left[\binom{n}{k} \times b_k \right]. \quad (3)$$

Pak tedy za předpokladu, že na pořadí registrů ve scan řetězcích není brán ohled, je velikost prohledávacího prostoru rovna $s1_n$.

Nyní se zabýváme otázkou velikosti prohledávacího prostoru za předpokladu, že pořadí registrů ve scan řetězcích je zohledněno. Konkrétní výběr a rozmístění registrů do scan řetězců lze opět zapsat jako rozklad množiny $SREG_{CUA}$, avšak v jednotlivých rozkladových třídách je třeba uvažovat všechny permutace jejích prvků. Počet rozkladů k -prvkové množiny je za tohoto předpokladu roven číslu

$$a_k = \sum_{i=1}^k Lah(k, i), \quad (4)$$

kde

$$Lah(k, i) = \frac{k!}{i!} \times \binom{k-1}{i-1} \quad (5)$$

je tzv. LAHovo číslo vyjadřující, kolika způsoby je možno, za výše uvedeného předpokladu, rozložit k -prvkovou množinu na i rozkladových tříd. Čísla a_k a $Lah(k, i)$ pro $k, i = 1, \dots, 10$ jsou uvedena v tabulce 13, strana 181.

Jelikož z n -prvkové množiny lze vybrat celkem $\binom{n}{k}$ k -prvkových podmnožin a počet možných rozmístění právě k scan registrů do scan řetězců je roven a_k , pak celkový počet ($s2_n$) možností výběru a rozmístění n registrů do scan registrů lze vyjádřit vztahem

$$s2_n = \sum_{k=1}^n \left[\binom{n}{k} \times a_k \right]. \quad (6)$$

Tabulka hodnot $s1_n$ (tj. počtů možných výběrů a rozmístění registrů do scan řetězců při zanedbání pořadí registrů ve scan řetězcích) a $s2_n$ (tj. počtů možných výběrů a rozmístění registrů do scan řetězců při zohlednění pořadí registrů ve scan řetězcích) pro $1 \leq n \leq 11$ je uvedena níže. I pro takto malá n lze v tabulce vidět, že hodnoty $s1_k$, $s2_k$ rostou velmi rychle s pomalu rostoucím n . Z toho plyne, že velikost prohledávacího prostoru možností výběru a rozmístění registrů do scan řetězců lze pro "dostatečně velká" n (např. pro $n > 15$) považovat za "dostatečně velké".

Tabulka 2. Tabulka hodnot $s1_n$ a $s2_n$ pro vybraná n

| n | $s1_n$ | řád | $s2_n$ | řád |
|-----|---------|--------|------------|--------|
| 1 | 1 | 10^0 | 1 | 10^0 |
| 2 | 4 | 10^0 | 5 | 10^0 |
| 3 | 14 | 10^1 | 25 | 10^1 |
| 4 | 51 | 10^1 | 147 | 10^2 |
| 5 | 202 | 10^2 | 1031 | 10^3 |
| 6 | 876 | 10^2 | 8463 | 10^3 |
| 7 | 4139 | 10^3 | 79591 | 10^4 |
| 8 | 21146 | 10^4 | 842831 | 10^5 |
| 9 | 115974 | 10^5 | 9914335 | 10^6 |
| 10 | 678569 | 10^5 | 128162463 | 10^8 |
| 11 | 4213596 | 10^6 | 1804852127 | 10^9 |

Zbývá vyřešit otázku týkající se formálního zápisu konkrétního výběru a rozmístění registrů do scan řetězců. Zápis vychází z "rozkladové" reprezentace výběru a rozmístění registrů do scan řetězců a je definován následovně [JS02c, JS02a, Str06]:

1. scan řetězec je reprezentován posloupností jemu příslušejících scan registrů,
2. speciální znak (tečka) odděluje jednotlivé scan řetězce,
3. není-li pořadí registrů ve scan řetězcích podstatné, jsou v zápisu registry patřící do téhož scan řetězce seřazeny zleva doprava podle rostoucích hodnot jejich indexů,
4. scan řetězce jsou seřazeny zleva doprava podle rostoucího indexu prvního registru v řetězci,
5. pokud $SREG_{CUA} = \emptyset$, pak notace obsahuje pouze speciální znak.

V případě, že pořadí registrů ve scan řetězcích není podstatné, lze s využitím této notace výběr a rozmístění registrů do scan řetězců z obrázků 33a a 33b vyjádřit zápisem $R_1R_2.R_5$ a výběr a rozmístění registrů do scan řetězců z obrázku 33c zápisem $R_1R_4R_5.R_2.R_3.R_6$. V případě, že pořadí registrů ve scan řetězcích je bráno v úvahu, lze výběr a rozmístění registrů do scan řetězců z obrázku 33a vyjádřit zápisem $R_2R_1.R_3$, z obrázku a 33b vyjádřit zápisem $R_1R_2.R_3$ a výběr a rozmístění registrů do scan řetězců z obrázku 33c zápisem $R_1R_5R_4.R_2.R_3.R_6$.

V souvislosti s výše uvedenou notací byly prováděny experimenty, jejichž hlavním cílem bylo ověřit či vyvrátit tvrzení, že čím více registrů je zařazeno do scan řetězců, tím větší bude nejen testovatelnost obvodu, ale také její cena vyjádřená nárůstem plochy a počtu vývodů potřebných k implementaci techniky scan.

Po provedení série experimentů bylo tvrzení potvrzeno. Experimentální výsledky dosažené pro obvod *Diffeq* jsou prezentovány v tab. 2 a obr. 33.

Tabulka 2. Výsledky prohledávání stavového prostoru techniky scan pro obvod *Diffeq*

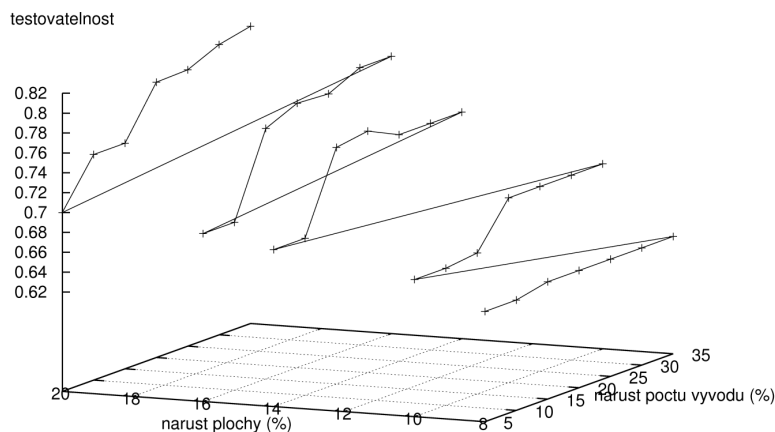
| Nárůst počtu vývodů (%) | Nárůst plochy (%) | | | | |
|-------------------------|--------------------|------------------------|----------------------------|--------------------------------|------------------------------------|
| | 8 | 10 | 14 | 16 | 20 |
| 5 | $R_6.R_1$ 0.631 | $R_6.R_1.R_3$ 0.658 | $R_6.R_1.R_3.R_4$ 0.678 | $R_6.R_1.R_4.R_2.R_3$ 0.689 | $R_6.R_1.R_2.R_4.R_3$ 0.7 |
| 10 | $R_6.R_1$ 0.631 | $R_6.R_1.R_2$ 0.658 | $R_6.R_1.R_4.R_3$ 0.678 | $R_6.R_1.R_4.R_2.R_3$ 0.689 | $R_6.R_1.R_3.R_1.R_4$ 0.747 |
| 15 | $R_1.R_6$ 0.638 | $R_1.R_3.R_6$ 0.662 | $R_1.R_4.R_6.R_3$ 0.758 | $R_1.R_2.R_6.R_3.R_4$ 0.772 | $R_1.R_4.R_2.R_6.R_5.R_3$ 0.747 |
| 20 | $R_1.R_6$ 0.638 | $R_1.R_2.R_6$ 0.706 | $R_1.R_3.R_6.R_2$ 0.763 | $R_1.R_3.R_2.R_6.R_4$ 0.786 | $R_1.R_5.R_3.R_2.R_6.R_4$ 0.797 |
| 25 | $R_1.R_6$ 0.638 | $R_1.R_3.R_6$ 0.706 | $R_1.R_2.R_3.R_6$ 0.748 | $R_1.R_4.R_2.R_3.R_6$ 0.784 | $R_1.R_4.R_2.R_5.R_3.R_6$ 0.798 |
| 30 | $R_1.R_6$ 0.638 | $R_1.R_2.R_6$ 0.706 | $R_1.R_2.R_3.R_6$ 0.748 | $R_1.R_2.R_3.R_4.R_6$ 0.799 | $R_1.R_2.R_4.R_3.R_5.R_6$ 0.812 |
| 35 | $R_1.R_6$ 0.638 | $R_1.R_2.R_6$ 0.706 | $R_1.R_2.R_3.R_6$ 0.748 | $R_1.R_2.R_3.R_4.R_6$ 0.799 | $R_1.R_2.R_3.R_4.R_5.R_6$ 0.819 |

V tab. 2 je prezentováno rozložení hodnot testovatelnosti v závislosti na rozmístění registrů do scan řetězců. Toto rozložení je pak vizualizováno na obr. 33. Je zřejmé, že v případě využití malého počtu vývodů a malé plochy (buňka v levém horním rohu tabulky) je možné zařadit méně registrů do méně scan řetězců než v případě většího prostoru pro techniku scan (buňka ve dolním pravém rohu tabulky). Tím sice vzroste cena za aplikaci techniky scan, ale také testovatelnost výsledného obvodu.

Využití výsledků analýzy testovatelnosti ke zlepšení testovatelnosti

V následujícím textu budou prezentovány výsledky experimentů, které je možno srovnat s výsledky jiných metod zabývajících se využitím výsledků analýzy testovatelnosti pro zlepšení testovatelnosti daného návrhu.

V dostupných pracích byly nalezeny pouze dvě metody [Buk00, Růž02] pracující na téže úrovni abstrakce jaká je uvažována v této práci a zabývajících se také výběrem registrů do řetězců scan za účelem zlepšení testovatelnosti daného obvodu. Z nich metoda [Buk00] byla ověřena na obvodech Bert, Diffeq a Tseng a metoda [Růž02] byla ověřena na obvodech Diffeq a Tseng. Proto i srovnání výsledků metody pro výběr registrů do řetězců scan využívající výsledků navržené analýzy testovatelnosti s výsledky dosaženými výše uvedenými metodami bude omezeno na obvody Bert, Diffeq a Tseng. Metody [Buk00, Růž02] sice pracují na téže úrovni abstrakce, ale liší se jak způsobem



Obrázek 33. Vizualizace rozložení testovatelnosti pro techniku scan a obvod *Diffeq*

analýzy datových cest⁵, tak požadavky na výsledný výběr registrů do řetězce scan⁶. Dosažené výsledky budou rozčleněny podle benchmarkových obvodů, jejichž testovatelnost byla analyzována za účelem jejího zlepšení s využitím techniky sériový scan. Při popisu výsledků metod výběru a rozmístění registrů do řetězců scan bude využita výše zavedená notace.

Metoda výběru a rozmístění registrů do scan řetězců byla nejdříve implementována pomocí genetického algoritmu [ZK01c, ZK01d, JS02c, JS02b, Str03d, Str03b, ZK03a] a později v [Her04] také pomocí dvou dalších optimalizačních technik - horolezeckého algoritmu a simulovaného žihání. Vstupem metody je struktura obvodu *CUA*, volba technik návrhu pro snadnou testovanost

⁵ metoda [Buk00] používá jistý pravděpodobnostní model, předpokládá pseudonáhodné generování testu a není svázána s konkrétní technikou návrhu pro snadnou testovatelnost, metoda [Růž02] je založena na analýze i-cest, předpokládá deterministické generování testu a je svázána s návrhem pro snadnou testovatelnost s využitím techniky scan

⁶ metoda [Buk00] se zabývá výběrem registrů do řetězců scan včetně volby pořadí registrů ve scan řetězcích, metoda [Růž02] se zabývá pouze výběrem nejmenší množiny registrů do řetězců scan, ale organizaci registrů v řetězcích již neřeší

vatelnost⁷, požadavky na testovatelnost návrhu⁸, návrhová omezení⁹, volba optimalizačního algoritmu a parametry potřebné pro běh zvoleného optimalizačního algoritmu. Výstupem metody je taková modifikace vstupní struktury *CUA*, jejíž testovatelnost se s pomocí zvolených technik návrhu pro snadnou testovatelnost co nejvíce blíží požadované testovatelnosti a která co nejvíce splňuje daná návrhová omezení. Práce [Her04] se mj. zabývala srovnáním tří výše uvedených a vzájemně principiálně odlišných optimalizačních postupů. Na testovaných obvodech z nich vycházel nejlépe horolezecký algoritmus. Výsledky tohoto srovnání je možno shrnout následovně:

- pomocí daných postupů bylo dosaženo velmi podobného a ve velkém počtu případů dokonce totožného řešení. Z toho lze vyvodit, že tyto postupy se příliš neliší v kvalitě nalezených řešení, jako spíše ve způsobu a rychlosti konvergence k hledanému řešení,
- **genetický algoritmus** - průměrný počet nově nalezených lepších řešení vztahený na jednu iteraci je u genetického algoritmu nepřímo úměrný počtu již provedených iterací. S rostoucím číslem iterace průměrný počet nově nalezených lepších řešení na iteraci klesá zhruba exponenciálně - průměrný počet iterací nutných pro nalezení nového lepšího řešení tedy velmi rychle roste s počtem již proběhnutých iterací. Ze tří zkoumaných algoritmů pro řešení tohoto problému se genetický algoritmus vyznačuje nejdelší průměrnou dobou běhu a nejmenší úspěšností nalezení hledaného řešení,
- **horolezecký algoritmus** - algoritmus velmi rychle konverguje k hledanému řešení a to často již v první iteraci. Ze zkoumaných algoritmů byl vyhodnocen jako nejrychleji konvergující k hledanému řešení, které byl schopen nalézt v každém ze zkoumaných případů,
- **simulované žíhání** - tento algoritmus se vyznačuje (ve srovnání s předchozími algoritmy) sice pomalejší konvergencí k hledanému řešení, avšak s poměrně vysokou pravděpodobností jeho nalezení; ta je ovlivněna zejména nastavením parametrů tohoto algoritmu.

Výše uvedené srovnání se týkalo pouze tří různých optimalizačních postupů, jejichž cílem bylo co nejeftivněji vyřešit otázku testovatelnosti *CUA* s pomocí techniky scan aplikované na základě výsledků analýzy testovatelnosti popsané v této práci. V následujícím textu budou tyto výsledky srovnány s výsledky dříve publikovaných metod [Buk00, Růž02] pracujících na téže úrovni abstrakce a řešících problém výběru registrů do scan řetězců.

Následující obrázek ilustruje konvergence k hledaným řešením pro obvody obsažené v benchmarkové sadě [PKS06] při použití optimalizace založené na

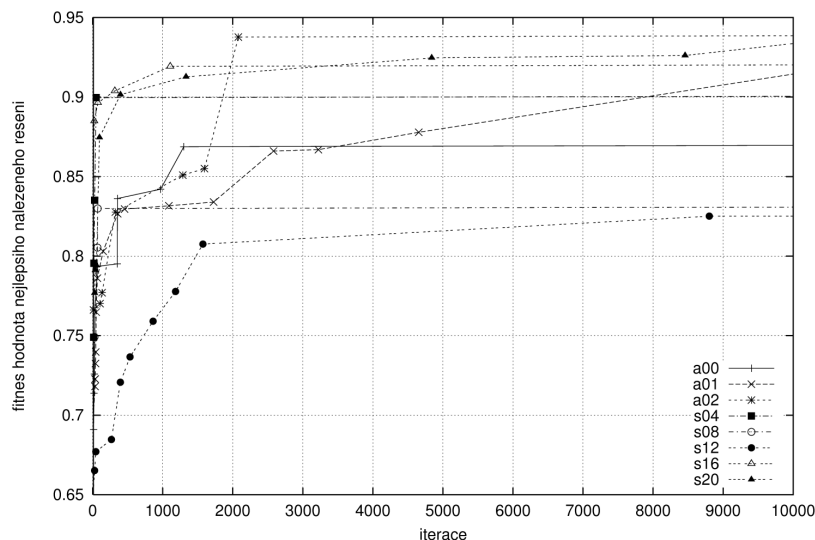
⁷ v současné době pouze technika sériový scan s možností volby (ne)zohlednění pořadí registrů ve scan řetězcích

⁸ zadané např. požadovaným počtem (procentem) testovatelných bran, portů atp. v obvodu

⁹ v současné době maximální přípustný nárůst plochy a počtu vstupů a výstupů *CUA*

6.2 Experimentální výsledky a příklady aplikace metody 155

genetickém algoritmu. Z obrázku je patrné, že zatímco průměrně přijatelná řešení jsou nalezena již po cca 2000 iteracích, tak nalezení ještě kvalitnějších řešení již trvá podstatně déle.



Obrázek 33. Evoluce nejlepších řešení při použití genetického algoritmu

Dosažené výsledky a jejich srovnání s výsledky jiných metod

Obvod Bert

Výsledky analýzy testovatelnosti obvodu Bert byly prezentovány na straně 142. Všechny brány v v obvodě Bert jsou testovatelné, tj. většího počtu testovatelných bran již nelze dosáhnout. Zatímco povolením až 20% nárůstu plochy při implementaci techniky scan bylo metodou [Buk00, strana 101] zjištěno, že výběrem a rozmístěním registrů dle notace $R_5R_4R_1R_2$ dojde k podstatnému zlepšení testovatelnosti bran v obvodu Bert, tak metodou založenou na analýze testovatelnosti publikované v této práci nebyla při stejných podmínkách experimentu doporučena žádná modifikace struktury, jelikož by nevedla k požadovanému zlepšení testovatelnosti obvodu Bert. Příčinou tohoto rozdílného výsledku může být rozdílná koncepce analýzy

datových cest při analýze testovatelnosti či heuristika používaná metodou [Buk00]. Metoda [Buk00] totiž není schopna zjistit, zda výskyt dat na daném portu je možný či nikoliv; je pouze schopna vyjádřit tuto skutečnost s jistou pravděpodobností.

Obvod Diffeq

Díky své poměrně komplikované struktuře je tento obvod velmi oblíbeným k ověřování různých metod a to zejména z oblasti diagnostiky číslicových obvodů. Výsledky analýzy testovatelnosti obvodu Diffeq byly shrnuty na straně straně 142 a označují obvod Diffeq obtížně testovatelným. Kromě metody z této práce se zlepšením testovatelnosti obvodu Diffeq pomocí techniky scan zabývaly také metody [Buk00, Růž02].

Při požadavku minimálního nárůstu plochy obvodu Diffeq po aplikaci techniky scan a požadavku testovatelnosti všech bran v obvodě bylo metodou [Buk00, strana 92] navrženo řešení $R_4R_6R_1R_5$, metodou [Růž02, strana 88] řešení¹⁰ $R_1R_2R_4$ a metodou založenou na analýze testovatelnosti z této práce řešení R_1R_6 . Na tomto obvodu je patrné, že na základě výsledků analýzy testovatelnosti navržené v této práci je možno při splnění vstupních požadavků provést cenově mnohem méně náročnější výběr registrů do scan řetězců, než jaký poskytují metody [Buk00, Růž02]. Výsledek metody [Buk00] může být ovlivněn již zmíněnou principiální odlišností způsobu analýzy datových cest obvodu či používanou heuristikou, výsledek [Růž02] analýzou založenou na tzv. koncepci *i – cest*, která je méně přesná než obecnější koncepce transparentnosti používaná v této práci. Obecně je možno konstatovat, že metoda [Růž02] sice dosahuje lepších výsledků než metoda [Buk00], ale v důsledku analýzy úžeji vymezené podmnožiny transparentních datových cest dosahuje horších výsledků než metoda založená na analýze testovatelnosti představené v této práci.

Obvod Tseng

Posledním z obvodů, který bude sloužit ke srovnání výsledků různých metod výběru registrů do řetězců scan, je obvod Tseng. Na základě výsledků analýzy testovatelnosti tohoto obvodu (strana 144) metoda [Buk00, strana 96] odhalila obtížně testovatelné části a doporučuje pro dosažení testovatelnosti všech bran v obvodu vybrat registry dle notace R_5R_1 . Oproti tomu z výsledků metody [Růž02, strana 91] a metody založené na analýze testovatelnosti prezentované v této práci plyne, že v obvodě Tseng se nevyskytuje netestovatelná brána a proto není třeba aplikovat techniku scan za účelem zlepšení testovatelnosti. Výsledek dosažený metodou [Buk00] si lze opět vysvětlit jako důsledek méně

¹⁰ metoda [Růž02] řeší pouze výběr registrů do řetězců scan; její výsledek nic neříká o rozmístění vybraných registrů do scan řetězců

přesné analýzy datových cest vlivem používaného pravděpodobnostního modelu či používanou heuristikou.

Kromě výše popsaného příkladu využití výsledků analýzy testovatelnosti pro následné zlepšení testovatelnosti daného obvodu je možno tyto výsledky využít i v řadě dalších oblastí. V dalším textu bude stručně představeno použití těchto výsledků v oblasti generování benchmarkových obvodů, což je problematika, kterou se v rámci svého výzkumu k tématu disertační práce podrobněji zabývá Ing. Tomáš Pečenka.

6.2.4 Generování benchmarkových obvodů

Vzhledem k již uvedeným¹¹ nedostatkům současně dostupných benchmarkových sad a nedostupnosti adekvátních komerčních návrhů se jako východisko z této situace jeví používání metod pro generování sad tzv. *syntetických benchmarkových obvodů* [PVC00, MDHC02, PVC02, KR03]. Obvody jsou označeny přívláskem syntetické z toho důvodu, že jsou vytvořeny ”uměle” a to pouze pro účely ověřování jistých postupů či nástrojů. Velkou výhodou syntetických benchmarkových obvodů je skutečnost, že lze zajistit plnou kontrolu nad jejich důležitými parametry jako např. funkce, plocha nebo topologické vlastnosti. Je-li známo, jaké vlastnosti musí mít obvody vhodné k ověření postupů řešících podobný problém, pak lze přímo ”na míru” vygenerovat sadu vhodných zástupců obvodů s těmito konkrétními vlastnostmi.

Obecně lze metody pro generování syntetických benchmarkových obvodů rozdělit na metody založené na *klonování* a na metody založené na *mutaci*. Obvody vzniklé pomocí klonování vycházejí z vlastností jednoho vzorového obvodu, obvody vzniklé pomocí mutace vznikají jako kombinace několika vzorových obvodů. Přehled vybraných principů generování syntetických benchmarkových obvodů lze nalézt v [PVC00].

Velkou nevýhodou sad syntetických benchmarkových obvodů je, že lze jen obtížně ověřit, zda obvody obsažené v dané sadě vhodně reprezentují obvody z požadované aplikační oblasti. Ověřující postupy tvoří samostatnou podoblast generování syntetických benchmarkových obvodů a dělí se na *přímé* a *nepřímé*. Přímé ověření je založeno na porovnání vybraných parametrů obvodů ze sady s parametry dostatečného množství skutečných obvodů z dané oblasti, nepřímé ověření na základě srovnání výsledků zprostředkovaných vhodným srovnávacím algoritmem provádějícím transformaci vstupního obvodu na předem daný výstup. V této práci se však ověřovacím postupům nebudeme blíže věnovat.

Princip navržené metody generování benchmarkových obvodů

Metoda pro generování benchmarkových obvodů, využívající pro ohodnocení testovatelnosti dílčích řešení analýzu testovatelnosti navrženou v této práci, je

¹¹ odstavce 6.2.1, strana 137

podrobněji popsána v [ZK04b, PEC07, PKS05, PKS06]. V tomto odstavci se zaměříme pouze na prezentaci základních myšlenek a principů, ze kterých tato metoda vychází.

Metoda předpokládá, že výsledky analýzy testovatelnosti prezentované v této práci věrohodně odrážejí testovatelnost daného číslicového obvodu. Pokud bychom problém návrhu pro snadnou testovatelnost označili za problém nalezení takové modifikace vstupní obvodové struktury, která se bude vyznačovat maximálním ohodnocením testovatelnosti při maximálním splnění daných návrhových omezení atp., tj. za problém nalezení maxima nějaké funkce¹² pro ohodnocení kvality daného řešení, pak problém generování benchmarkových obvodů lze označit za problém nalezení minima této funkce, tj. za problém opačný. Jinými slovy: v případě metody generování benchmarkových obvodů hledáme ta řešení, která se vyznačují velmi nízkou hodnotou testovatelnosti.

Základem metody generování benchmarkových obvodů, kterou se zabývá Ing. Tomáš Pečenka, a která je založena na využití výsledků poskytovaných algoritmem analýzy testovatelnosti, prezentovaném v této práci, je evoluční algoritmus. Vstupy tohoto algoritmu jsou následující:

- počet primárních vstupů a výstupů obvodu,
- počet a typy obvodových prvků, které mají tvořit strukturu obvodu,
- požadavky na testovatelnost obvodu,
- parametry evolučního algoritmu.

Na základě vstupních parametrů jsou evolučním algoritmem generovány obvody nejlépe splňující zadané požadavky kladené na jejich vlastnosti. Struktura výsledného obvodu je vytvořena propojením rozhraní obvodových prvků uložených v knihovně prvků. Výsledný strukturálně popsaný obvod je syntetizovatelný a je uložen do souboru ve formátu VHDL, což činí obvod prakticky ihned použitelným pro ověřování některé z metod či nástrojů.

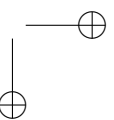
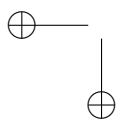
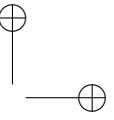
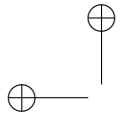
Princip metody pro generování benchmarkových obvodů spočívá ve vytvoření požadovaného počtu obvodových prvků, ohodnocení portů z jejich rozhraní unikátními čísly a v konstrukci orientovaného grafu G , jehož uzly jsou tato čísla. Problém generování benchmarkových obvodů lze z tohoto pohledu formulovat jako problém konstrukce množiny $E(G)$ tak, aby testovatelnost výsledné obvodové struktury byla co nejbližší požadované testovatelnosti.

6.3 Shrnutí

V této kapitole byly představeny některé z oblastí použití navrženého algoritmu analýzy testovatelnosti, konkrétně oblast návrhu pro snadnou testovatelnost s využitím techniky scan a oblast generování benchmarkových obvodů. Rovněž byly uvedeny a dokázány vybrané vlastnosti navrženého algoritmu. Z výsledků uvedených v této kapitole vyplývá, že metoda analýzy testovatelnosti navržená v této práci pracuje s přijatelnou časovou složitostí a díky

¹² taková funkce se obvykle nazývá účelová funkce

detailnější analýze datových cest poskytuje přesnější informaci o testovatelnosti daného obvodu a umožňuje dosáhnout lepších řešení, než jaká umožňují existující srovnatelné přístupy. Navržená metoda není svázána s žádnou z technik návrhu pro snadnou testovatelnost, a proto lze její výsledky považovat za univerzálně použitelné při řešení problémů spojených s testovatelností datových cest číslicového obvodu na úrovni meziregistrových přenosů. Jak v oblasti návrhu pro snadnou testovatelnost tak v oblasti generování benchmarkových obvodů je možné provést řadu vylepšení, která by mohla vést ke kvalitnějším řešením. V první jmenované oblasti by se mohlo jednat např. o rozšíření množiny technik návrhu pro snadnou testovatelnost a o návrh metody pro hledání přijatelné kombinace a propojení těchto technik s cílem dosažení co nejlepší testovatelnosti obvodu. V druhé jmenované oblasti by se mohlo jednat např. o rozšíření počtu ovládaných parametrů generovaných obvodů.



7

Závěr

Tato poslední kapitola je věnována shrnutí hlavních výsledků dosažených v této práci a jejich srovnání s výzkumnými cíli stanovenými na jejím začátku. V závěru této kapitoly bude prezentován jednak přínos této práce - tak, jak je viděn jejím autorem - a jednak možné směry výzkumu navazujícího na tuto práci.

7.1 Shrnutí výsledků práce

Tato práce se zabývá analýzou testovatelnosti číslicového obvodu popsaného na úrovni meziregistrových přenosů a využitím jejích výsledků ve vybraných oblastech souvisejících s diagnostikou číslicových obvodů. Práce se zaměřuje pouze na problematiku související s testovatelností obvodových datových cest, ale řadičem ovládaným tok dat těmito cestami se blíže nezabývá. Předpokládá se, že pro směrování toku dat je zvolena propojovací strategie multiplexovaných datových cest. V části zabývající se přehledem současného stavu v oblasti analýzy testovatelnosti práce shrnuje hlavní nedostatky existujících metod analýzy testovatelnosti. V práci je ukázáno, že většinu ze zmíněných nedostatků lze odstranit, je-li každý prvek uložený v knihovně obvodových prvků kromě informací týkajících se návrhového popisu jeho rozhraní, činnosti atp. vybaven také informacemi usnadňujícími jak jeho diagnostiku, tak také diagnostiku systému, jehož je tento prvek součástí. Tato informace je poté popsána pomocí prostředků zavedeného matematického modelu, který je rozšířením výchozího modelu [Růž02] o nové prostředky nezbytné pro popis a modelování obecnější koncepce tzv. transparentních cest, než jakými jsou příliš přísné a často používané koncepce, obvykle založené na tzv. koncepci I-cest. Výhodou rozšířeného modelu je možnost modelování a analýzy většího počtu obvodových datových cest, tj. i těch cest, které jsou z pohledu přísnějších koncepcí pro účely přenosu diagnostických dat nepoužitelné, přestože ve skutečnosti použitelné jsou. Hlavním cílem rozšířeného modelu je poskytnout prostředky potřebné pro formální

popis vztahů a dílčích algoritmů pro analýzu testovatelnosti. Navržená metoda analýzy testovatelnosti je založena na číselném ohodnocení říditelnosti, pozorovatelnosti a testovatelnosti vnitřních částí obvodu a obvodu jako celku a jelikož není svázána s žádnou z technik návrhu pro snadnou testovatelnost, lze její výsledky považovat za univerzálně použitelné při řešení problémů souvisejících s testovatelností číslicových obvodů popsaných na úrovni meziregistrových přenosů. Metoda není popsána pomocí v současné době nejednotných pojmů z oblasti diagnostiky, ale pomocí modelem přesně definovaných prostředků. V závěru se práce věnuje důkazům významných vlastností navržené metody a příkladu jejího použití v oblasti návrhu pro snadnou testovatelnost s využitím techniky scan a oblasti generování benchmarkových obvodů. Výsledky dosažené touto metodou, založenou na rozšířené koncepci transparentních datových cest, jsou shrnuty a srovnány s výsledky existujících metod. Z výsledků vyplývá, že díky detailnější analýze obvodových datových cest poskytuje tato metoda přesnější informaci o testovatelnosti daného obvodu, než jakou poskytují metody řešící stejný problém na téže úrovni abstrakce. To je patrné zejména při řešení problému výběru a rozmístění registrů do scan řetězců - navržená metoda byla schopna ve všech zkoumaných případech vybrat do scan řetězců méně registrů než metody, s nimiž byla srovnávána.

7.2 Přínos práce

Hlavních přínosů této práce je možno vysledovat několik. K významným přínosům jistě patří snaha práce odstranit hlavní nedostatky existujících metod analýzy testovatelnosti datových cest se směřováním toku dat pomocí multiplexovaných datových cest na úrovni meziregistrových přenosů předpokládajících deterministické generování testu. Jedno z možných řešení nabízí tato práce, která ukazuje, že je-li každý prvek uložený v knihovně obvodových prvků kromě informací týkajících se návrhového popisu jeho rozhraní, činnosti atp. vybaven také vhodnými diagnostickými informacemi, pak je možné docílit velmi detailní analýzy obvodových datových cest, a to algoritmem s přijatelnou časovou složitostí, který není svázán s žádnou z technik návrhu pro snadnou testovatelnost, ale pouze s předpokladem deterministického generování testu. Ze skutečnosti, že navržený algoritmus analýzy testovatelnosti není svázán s žádnou z technik návrhu pro snadnou testovatelnost vyplývá, že jeho výsledky jsou univerzálně použitelné při řešení problémů souvisejících s testovatelností číslicových obvodů popsaných na úrovni meziregistrových přenosů.

Za další přínos práce je možno považovat popis dodatečné diagnostické informace pomocí prostředků zavedeného matematického modelu. Tento model je rozšířením výchozího modelu [Růž02] o nové prostředky nezbytné pro popis a modelování obecnější koncepce tzv. transparentních cest než jakými jsou příliš přísné, avšak přesto často používané koncepce obvykle založené

na tzv. koncepci I-cest. Výhodou rozšířeného modelu je možnost modelování a analýzy většího počtu obvodových datových cest, tj. i těch cest, které jsou z pohledu přísnějších koncepcí pro účely přenosu diagnostických dat nepoužitelné. Rozšíření modelu [Růž02] není samoučelné. Jeho hlavním účelem je poskytnout prostředky potřebné pro formální popis vztahů a dílčích algoritmů pro analýzu testovatelnosti. Metoda analýzy testovatelnosti pak není popsána pomocí nejednotných pojmů diagnostiky, ale pomocí modelem přesně definovaných prostředků. Vzhledem k připravovanému standardu IEEE P1522 lze konstatovat, že snaha o popis problémů souvisejících s testovatelností odpovídá současnému trendu v oblasti návrhu a diagnostiky číslicových obvodů.

Jako poslední významný přínos této práce uvedeme transformaci problému analýzy testovatelnosti na matematický problém. Tím je v této práci problém prohledávání dvou orientovaných grafů zkonstruovaných na základě modelu daného obvodu, a to grafu datového toku testovacích vzorků a grafu datového toku odezev. Výhody transformace řešeného problému na problém matematický jsou zřejmé - matematicky popsáný problém je jednoznačně definován, k jeho řešení je možné využít řadu již existujících principů a algoritmů a v neposlední řadě je možno poměrně snadno dokázat významné vlastnosti algoritmu řešícího tento problém. Důkazy významných vlastností navrženého algoritmu, a to zejména důkaz správnosti ohodnocování diagnostických vlastností obvodu ve smyslu zavedených definic a důkaz časové složitosti, je možno také považovat za přínos této práce.

7.3 Možné směry navazujícího výzkumu

Kromě praktického směru, jakým je snaha aplikovat navrženou metodu v dalších oblastech návrhu a diagnostiky číslicových obvodů, lze za jeden ze směrů výzkumu navazujícího na tuto práci jistě považovat snahu o obohacení metody návrhu pro snadnou testovatelnost o další techniky návrhu pro snadnou testovatelnost s cílem hledání přijatelné kombinace technik za účelem maximálního zlepšení testovatelnosti daného obvodu [Str08] či snahu o sjednocení definic z této práce s konečnou verzí standardu IEEE P1522.

Další možné směry lze vidět např. v návrhu metody extrakce požadované diagnostické informace z návrhového popisu prvků [Str07] či v návrhu metody pro hierarchické generování testu založené na využití prostředků rozšířeného modelu.

Jelikož jak výchozí model tak model prezentovaný v této práci předpokládá, že pro směrování toku dat je zvolena propojovací strategie multiplexovaných datových cest a jelikož rozšířením modelu o další propojovací strategie - např. o strategii obousměrných sběrnic - se tato práce nezabývá, mohla by být k výše uvedeným možným směrům výzkumu připojena také snaha o toto rozšíření.

Další výzkum může být také zaměřen na oblasti, které sice přímo nesouvisejí s tématem řešeným v této práci, ale kterými se autor v rámci své

164 7 Závěr

výzkumné činnosti již okrajově zabýval - jedná se např. o návrh řadiče testu [DM02, RS07] či o oblast plánování testu [ZK03b, StD06].

Reference

- [AB85] M. S. Abadir and M. A. Breuer. A Knowledge-Based System for Designing Testable VLSI Chips. *IEEE Design & Test of Computers*, 2(3):56–68, 1985.
- [ABF90] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. IEEE Press, Piscataway, NJ, 1990.
- [Ait95] R. C. Aitken. An Overview of Test Synthesis Tools. *IEEE Design & Test of Computers*, 1(1):8–15, 1995.
- [ARME02] Z. Peng A. R. Mohamed and P. Eles. BIST Synthesis: An Approach to Resources Optimization under Test Time Constraints. In *Proceedings of 5th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, s. 346–351, Brno, 2002. FIT VUT.
- [ARME03] Z. Peng A. R. Mohamed and P. Eles. A Wiring-Aware Approach to Minimizing Built-In Self-Test Overhead. In *Proceedings of 4th Workshop on RTL and High Level Testing*, (poster presentation only, not published in the proceedings), Piscataway, 2003. IEEE Computer Society.
- [Ben84] R. G. Bennetts. *Design of Testable Logic Circuits*. Addison-Wesley Publishing Company, Reading, 1984.
- [BF96] V. Boppana and W. K. Fuchs. Partial Scan Design Based on State Transition Modeling. In *Proceedings of International Test Conference*, s. 538–547, 1996.
- [BG89] M. A. Breuer and R. Gupta. BALLAST: A Methodology for Partial Scan Design. In *Proceedings of the 19th International Symposium on Fault-Tolerant Computing*, s. 118–125, 1989.
- [BG94] A. Becker and D. Geiger. Approximation Algorithm for the Loop Cutset Problem. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, s. 60–68. Morgan Kaufman, 1994.
- [BG96] A. Becker and D. Geiger. Optimization of Pearl’s Method of Conditioning and Greedy-Like Approximation Algorithms for the Vertex Feedback Set Problem. *Artificial Intelligence*, 83:167–188, 1996.
- [Brg04] F. Brglez. Computer-Aided Design Benchmarking Laboratory, 2004. <http://www.cbl.ncsu.edu/benchmarks/>.
- [Buk00] P. Bukovjan. *Allocation for Testability in High-Level Synthesis*. PhD thesis, Institute National Polytechnique de Grenoble, 2000.

166 Reference

- [CA90] K. T. Chen and V. D. Agrawal. A Partial Scan Method for Sequential Circuits With Feedback. *IEEE Transactions in Computers*, 39(1):544–548, 1990.
- [CB89] J. D. Calhoun and F. Brglez. A Framework and Method for Hierarchical Test Generation. In *Proceedings of International Test Conference*, s. 480–490, 1989.
- [CG96] J. H. Conway and R. K. Guy. *The Book of Numbers*. Springer-Verlag, New York, 1996.
- [CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, Cambridge, MA, 2001.
- [CM89] C. H. Chen and P. R. Menon. An Approach to Functional Level Testability Analysis. In *Proceedings of the International Test Conference*, s. 373–380, Washington, 1989. IEEE Computer Society.
- [Coh01] B. Cohen. *VHDL Coding Styles and Methodologies*. Kluwer Academic Publishers, Dodrecht, Netherlands, 2001.
- [Cox95] H. Cox. Synthesizing Circuits with Implicit Testability Constraints. *IEEE Design & Test of Computers*, 1(1):16–22, 1995.
- [CP90] V. Chickermane and J. H. Patel. An Optimization Based Approach to the Partial Scan Design Problem. In *Proceedings of International Test Conference*, s. 377–386, 1990.
- [CP91] V. Chickermane and J. H. Patel. A Fault Oriented Partial Scan Design Approach. In *Proceedings of International Conference on Computer Aided Design*, s. 400–403, 1991.
- [CS02] C. H. Chen and D. G. Saab. A Novel Behavioral Testability Measure. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(12):1960–1993, 2002.
- [CWS85] E. Lloyd C. Wang and M. Soffa. Feedback Vertex Sets and Cyclically Reducible Graphs. *Journal of the Association for Computing Machinery*, 32(2):296–313, 1985.
- [DBR98] M. L. Flottes D. Berthelot and B. Rouzeyre. Optibist: A Tool for Bisting Datapaths. In *Proceedings of IEEE European Test Workshop*, s. 123–127. IEEE Computer Society, 1998.
- [Dem02] J. Demel. *Grafy a jejich aplikace*. Academia, 2002.
- [DM02] J. Strnadel D. Mika, Z. Kotásek. Test Controller Design Based on VHDL Source File Analysis. In *Proceedings of 5th International Scientific Conference Electronic Computers and Informatics*, s. 135–141, Košice, 2002. TU v Košiciach.
- [FA97] D. Farren and T. Ambler. The Economics of System-Level Testing. *IEEE Design & Test of Computers*, 1(1):51–58, 1997.
- [FACW98] D. Hochbaum F. A. Chudak, M. X. Goemans and D. P. Williamson. A Primal-Dual Interpretation of Two 2-Approximation Algorithms for the Feedback Vertex Set Problem in Undirected Graphs. *Operations Research Letters*, 22:111–118, 1998.
- [FCR96] M. Rebaudengo F. Corno, P. Prinetto and M. S. Reorda. Partial Scan Flip Flop Selection for Simulation-Based Sequential ATPG. In *Proceedings of International Test Conference*, s. 558–564, 1996.
- [FCV98] M. S. Reorda F. Corno, P. Pronetto and M. Violante. Exploiting Symbolic Techniques for Partial Scan Flip Flop Selection. In *Proceedings of Design, Automation and Test in Europe*, s. 670–677, 1998.

- [Flo67] R. Floyd. Assigning Meaning to Programs. In *Proceedings of Symposium in Applied Mathematics*, volume 19, s. 19–32. American Mathematical Society, 1967.
- [FMLB97] Pires R. Flottes M. L. and Rouzeyre B. Analyzing testability from behavioral to rtl level. In *Proceedings of the European Design and Test Conference*, s. 158–165, Washington DC, 1997. IEEE Computer Society.
- [Fuj85] H. Fujiwara. *Logic Testing and Design for Testability*. MIT Press, Cambridge, MA, 1985.
- [FZ02] D. Mika J. Strnadel F. Zbořil, Z. Kotásek. The Identification of Feedback Loops in RTL Structures. In *Proceedings of 5th International Scientific Conference Electronic Computers and Informatics*, s. 142–147, Košice, 2002. TU v Košiciach.
- [Gao04] Gao J. Component Testability and Component Testing Challenges, 2003 [2004]. Dostupné z: <http://www.sei.cmu.edu/cbs/cbse2000/papers/18/18.html>.
- [Gol79] L. H. Goldstein. Controlability/Observability Analysis for Digital Circuits. *IEEE Transactions on Circuits and Systems*, 26(9):685–693, 1979.
- [Gra79] J. Grason. TMEAS - a Testability Measurement Program. In *Proceedings of IEEE/ACM Design Automation*, s. 156–161, 1979.
- [Gu96] X. Gu. *RT Level Testability Improvement by Testability Analysis and Transformations*. PhD thesis, Linkping University, 1996.
- [Gus88] D. Gusfield. A Graph Theoretic Approach to Statistical Data Security. *SIAM Journal on Computing*, 17(3):552–571, 1988.
- [Har00] J. E. Harlow. Overview of Popular Benchmark Sets. *IEEE Design and Test of Computers*, 17(3):15–17, 2000.
- [Her04] T. Herrman. *Využití optimalizačních postupů při výběru registrů do řetězce scan*. PhD thesis, FIT VUT v Brně, 2004. Diplomová práce.
- [HKTMSV88] A. R. Newton H. K. T. Ma, S. Devadas and A. Sangiovani-Vincentelli. An Incomplete Scan Design Approach to Test Generation for Sequential Machines. In *Proceedings of International Test Conference*, s. 730–734, 1988.
- [IGB98] N. K. Jha I. Ghosh and S. Bhawmik. A bist scheme for rtl controller-data paths based on symbolic testability analysis. In *Proceedings of the 35th annual conference on Design automation conference*, s. 554–559. ACM Press, 1998.
- [IGJ96a] A. Raghunathan I. Ghosh and N. K. Jha. A Design for Testability Technique for RTL Circuits Using Control/Data Flow Extraction. In *Proceedings of International Conference on Computer Aided Design*, s. 329–336. IEEE Computer Society, 1996.
- [IGJ96b] A. Raghunathan I. Ghosh and N. K. Jha. A Design for Testability Technique for RTL Circuits Using Control/Data Flow Extraction. In *Proceedings of International Conference on Computer-Aided Design*, s. 329–336, 1996.
- [IGJ98] A. Raghunathan I. Ghosh and N. K. Jha. A Design for Testability Technique for RTL Circuits Using Control/Data Flow Extraction. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 17(8):706–723, 1998.

168 Reference

- [IPS95] D. S. Ha I. Park and G. Sim. A New Method for Partial Scan Design Based on Propagation and Justification Requirements of Faults. In *Proceedings of International Test Conference*, s. 413–422, 1995.
- [JES76] J. Grason J. E. Stephenson. A Testability Measure for Register Transfer Level Digital Circuits. In *Proceedings of International Symposium on Fault Tolerant Computing*, s. 101–107, 1976.
- [JH01] R. Růžicka J. Strnadel J. Hlavička, Z. Kotásek. Interactive Tool for Behavioral Level Testability Analysis. In *Proceedings of IEEE European Test Workshop*, s. 117–119, Los Alamitos, 2001. IEEE Computer Society Press.
- [Joh75] D. B. Johnson. Finding all the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.
- [JS02a] Z. Kotásek J. Strnadel. Normalized Testability Measures at RT Level: Utilization and Reasons for Creation. In *Proceedings of 36th International Conference on Modeling and Simulation of Systems*, s. 297–304, Ostrava, 2002. MARQ.
- [JS02b] Z. Kotásek J. Strnadel. Optimising Solution of the Scan Problem at RT Level Based on a Genetic Algorithm. In *Proceedings of 5th IEEE Workshop on Design and Diagnostics of Electronics Circuits and Systems*, s. 44–51, Brno, 2002. FIT VUT v Brně.
- [JS02c] Z. Kotásek J. Strnadel. Testability Improvements Based on the Combination of Analytical and Evolutionary Approaches at RT Level. In *Proceedings of Euromicro Symposium on Digital System Design - Architectures, Methods and Tools*, s. 166–173, Los Alamitos, 2002. IEEE Computer Society Press.
- [Kar72] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, s. 85–104, New York, 1972. Plenum Press.
- [KKS95] Kimme R. K. Kim K. S. Partial Scan Flip-Flop Selection by Use of Empirical Testability. *Journal of Electronic Testing: Theory and Applications*, 7(1):47–59, 1995.
- [KP90] K. Kuchcinski and Z. Peng. Testability Analysis in a VLSI High-Level Synthesis System. *The Euromicro Journal, Microprocessing and Microprogramming*, 28(1-5):295–300, 1990.
- [KR03] P. D. Kundarewich and J. Rose. Synthetic circuit generation using clustering and iteration. In *Proceedings of the 2003 ACM/SIGDA 11th International Symposium on Field Programmable Gate Arrays*, s. 245–245. ACM, 2003.
- [KW90] A. Kunzmann and H. J. Wunderlich. An Analytical Approach to the Partial Scan Problem. *Journal of Electronic Testing: Theory and Applications*, 1:163–174, 1990.
- [Lar00] E. Larsson. *An Integrated System-Level Design for Testability Methodology*. PhD thesis, Linköping University, 2000.
- [Lee97] M. T. Lee. *High-level Test Synthesis of Digital VLSI Circuits*. Artech House, USA, 1997.
- [Lia94] Y. D. Liang. On the Feedback Vertex Set Problem in Permutation Graphs. *Information Processing Letters*, 52:123–129, 1994.
- [LL79] J. Y.-T. Leung and E. K. Lai. On Minimum Cost Recovery From System Deadlock. *IEEE Transactions in Computers*, 28(9):671–677, 1979.

- [LL88] H. Levy and L. Lowe. A Contraction Algorithm for Finding Small Cycle Cutsets. *Journal of Algorithm*, 9:470–493, 1988.
- [LP97] J. Lee and J. H. Patel. Hierarchical Test Generation Under Architectural Level Functional Constraints. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 15(9):1144–1151, 1997.
- [LR90] D. Lee and S. Reedy. On Determining Scan Flip-Flops in Partial Scan Designs. In *Proceedings of International Conference on Computer Aided Design*, s. 322–325, 1990.
- [LT97] C. L. Lu and C. Y. Tang. A Linear-Time Algorithm for the Weighted Feedback Vertex Problem on Interval Graphs. *Information Processing Letters*, 61:107–111, 1997.
- [Mär92] S. März. *High-Level Synthesis*, volume 170 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, 1992.
- [MBIR87] U. Glässer M. Bidjan-Irani and F. Ramming. Knowledge Based Tools for Testability Checking. In *Proceedings of 3rd International Conference on Fault-Tolerant Computing Systems*, s. 119–128. Springer-Verlag, 1987.
- [MDHC02] J. S. Rose M. D. Hutton and D. Corneil. Automatic Generation of Syntetic Sequential Benchmark Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(8):928–940, 2002.
- [MH90] B. T. Murray and J. P. Hayes. Hierarchical Test Generation Using Precomputed Tests for Modules. *IEEE Transactions on Computer Aided Design*, 9(6):594–603, 1990.
- [MH91] B. T. Murray and J. P. Hayes. Test Propagation Through Modules and Circuits. In *Proceedings of International Test Conference*, s. 748–757, 1991.
- [MKB01] F. Novak M. Khalil, C. Robach and A. Biasizzo. System Level Diagnostic Strategies and Tools. In *Proceedings of 4th IEEE Design and Diagnostics of Electronic Circuits and Systems*, s. 251–258, 2001.
- [MO02] Y. Makris and A. Orailoglu. Test Requirement Analysis for Low Cost Hierarchical Test Path Construction. In *Proceedings of 11th Asian Test Symposium (ATS)*, s. 134–139, 2002.
- [MS81] B. Monien and R. Schultz. Four Approximation Algorithms for the Feedback Vertex Set Problems. In *Proceedings of the 7th Conference on Graph Theoretic Concepts of Computer Science*, s. 315–326, München, 1981. Hanser-Verlag.
- [Mue01] S. Mueller. *Osobní počítač - Nejpodrobnější průvodce hardwarem PC*. Computer Press, Praha, Czech Republic, 2001.
- [MŠak01] V. Mařík, O. Štěpánková, and J. Lažanský a kol. *Umělá inteligence, Díl 3*. Academia, Praha, 2001.
- [PA95] P. S. Parikh and M. Abramovici. Testability-Based Partial Scan Analysis. *Journal of Electronic Testing*, 7:61–70, 1995.
- [PEC07] T. Pecenka. *Prostředky a metody pro automatické generování testovacích obvodů*. disertační práce, FIT VUT v Brně, Brno, 2007, 124 s.

170 Reference

- [PFR01] P. M. Pardalos P. Festa and M. G. C. Resende. Feedback Set Problems. In *Encyclopedia of Optimization*, s. 94–106, Dordrecht, 2001. Kluwer Academic Publishers.
- [PGPG86] J. P. Knight P. G. Paulin and E. F. Girczyc. Hal: A Multi-Paradigm Approach to Automatic Data Path Synthesis. In *Proceedings of the Design Automation Conference*, s. 263–270. IEEE Computer Society, 1986.
- [PKS05] T. Pečenka, Z. Kotásek, L. Sekanina and .J. Strnadel. Automatic Discovery of RTL Benchmark Circuits with Predefined Testability Properties. In *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, ISCP, Los Alamitos, 2005, s. 51–58.
- [PKS06] T. Pečenka, Z. Kotásek, L. Sekanina. FITTest_BENCH06: A New Set of Benchmark Circuits Reflecting Testability Properties. In *Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*, IEEE, 2006, Praha, s. 285–289.
- [PMP99] M. G. C. Resende P. M. Pardalos, T. Quian. A Greedy Randomized Adaptive Search Procedure for Feedback Vertex Set. *Journal of Combinatorial Optimizations*, 2:399–412, 1999.
- [PSP93] M. Abramovici P. S. Parikh. A Cost-Based Approach to Partial Scan. In *Proceedings of 30th ACM/IEEE Design Automation Conference*, s. 255–259, 1993.
- [PSP95] M. Abramovici P. S. Parikh. Testability-Based Partial Scan Analysis. *Journal of Electronic Testing: Theory and Applications*, 7(1):62–70, 1995.
- [PVA92] J. A. Abraham P. Vishakantaiah and M. S. Abadir. Automatic Test Knowledge Extraction From VHDL (ATKET). In *Proceedings of Design Automation Conference*, s. 273–278, 1992.
- [PVA93] J. A. Abraham P. Vishakantaiah, T. Thomas and M. S. Abadir. AMBIANT: Automatic Generation of Behavioral Modifications for Testability. In *Proceedings of International Conference on Computer Design*, s. 63–66, 1993.
- [PVC00] D. Stroobandt P. Verplaetse and J. V. Campenhout. On Synthetic Benchmark Generation Methods. In *Proceedings of IEEE International Symposium on Circuits and Systems*, s. 213–216. IEEE Computer Society Press, 2000.
- [PVC02] D. Stroobandt P. Verplaetse and J. V. Campenhout. Synthetic Benchmark Circuits for Timing-driven Physical Design Applications. In *Proceedings of the International Conference on VLSI*, s. 31–37. CSREA Press, 2002.
- [PVS93] J. A. Abraham P. Vishakantaiah and D. G. Saab. CHEETA: Composition of Hierarchical Sequential Tests Using ATKET. In *Proceedings of International Test Conference*, s. 606–615, 1993.
- [Qua04] Qualtech Systems. Software Testing, 2004. Dostupné z: http://www.teamqsi.com/testing/software_testing.htm.
- [RBYR98] J. Naor R. Bar-Yehuda, D. Geiger and R. M. Roth. Approximation Algorithms for the Vertex Feedback Set Problem with Applications. *SIAM Journal on Computing*, 27(4):942–959, 1998.
- [RGBR80] C. M. Maunder R. G. Bennetts and G. D. Robinson. CAMELOT: A Computer-Aided Measure for Logic Testability. In *Proceedings of*

- Intentional Conference on Computer Communication*, s. 1162–1165, 1980.
- [RS07] R. Růžička, J. Strnadel. Test Controller Synthesis Constrained by Circuit Testability Analysis. In *Proceedings of 10th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, ICSP, 2007, Los Alamitos, s. 626–633.
- [RSTA99] A. Krishnamachari R. S. Tupuri and J. A. Abraham. Test Generation for Gigahertz Processors Using an Automatic Functional Constraint Extractor. In *Proceedings of Design Automation Conference*, s. 647–652, 1999.
- [Růž02] R. Růžička. *Formální přístup k analýze testovatelnosti číslicových obvodů na úrovni RT*. PhD thesis, Vysoké učení technické v Brně, 2002.
- [SGT03] S. Basu S. Ghosh and N. A. Touba. Joint Minimization of Power and Area in Scan Testing by Scan Cell Reordering. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, s. 246–249. IEEE Computer Society, 2003.
- [SH99] S. Sharma and M. Hsiao. Partial Scan Using Multi-Hop State Reachability Analysis. In *Proceedings of VLSI Test Symposium*, s. 121–126, 1999.
- [SH00] S. Seshadri and M. S. Hsiao. Formal Value-Range and Variable Testability Technique. *Journal of Electronic Testing, Theory and Applications*, 16(1–2):131–145, 2000.
- [SH01] S. Sharma and M. S. Hsiao. Combination of Structural and State Analysis for Partial Scan. In *Proceedings of IEEE VLSI Design Conference*, s. 134–139, 2001.
- [SH02] S. Seshadri and M. S. Hsiao. Behavioral-Level DFT via Formal Operator Testability Measures. *Journal of Electronic Testing*, 18(6):595–611, 2002.
- [Sha76] A. Shamir. A Linear Time Algorithm for Finding Minimum Cutsets in Reduced Graph. *SIAM Journal on Computing*, 8(4):645–655, 1976.
- [SK04] ”J. Sheppard and M. Kaufman”. IEEE P1522 Standard for Testability and Diagnosability Characteristics and Metrics (Draft), 2000 [2004]. Informace o připravovaném standardu IEEE P1522 dostupné z: <http://grouper.ieee.org/groups/1232/pubs/>.
- [Smi97] M. J. S. Smith. *Application-Specific Integrated Circuits*. VLSI System Series. Addison-Wesley, 1997.
- [Spe89] E. Speckenmeyer. On Feedback Problems in Digraphs. In *Proceedings of 15th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 411, s. 218–231. Springer-Verlag, 1989.
- [SRJ98] G. Lakshminarayana S. Ravi and N. K. Jha. Tao: Regular expression based high-level testability analysis and optimization. In *Proceedings of International Test Conference*, s. 331–340. IEEE Press, 1998.
- [SRJ99] G. Lakshminarayana S. Ravi and N. K. Jha. A framework for testing core-based systems-on-a-chip. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, s. 385–390. IEEE Press, 1999.
- [SS03] A. H. Jahangir S. Safari, H. Esmaelzadeh. Testability Improvement During High-Level Synthesis. In *Proceedings of 12th Asian Test Symposium*, s. 505–505. IEEE Computer Society, 2003.

172 Reference

- [STCA94] A. Balakrishman S. T. Chakradhar and V. D. Agrawal. An Exact Algorithm for Selecting Partial Scan Flip-Flops. In *Proceedings of Design Automation Conference*, s. 81–86, 1994.
- [Str02a] J. Strnadel. Evaluating Cost/Quality Trade-off Solutions Proposed During a DFT Process. In *Proceedings of 8th Conference Student EEICT*, s. 506–510, Brno, 2002. VUT v Brně.
- [Str02b] J. Strnadel. Normalized Testability Measures Based on RTL Digital Circuit Graph Model Analysis. In *Proceedings of 5th International Scientific Conference Electronic Computers and Informatics*, s. 200–205, Košice, 2002. TU v Košiciach.
- [Str03a] J. Strnadel. Algebraic Analysis of Feedback Loop Dependencies in Order of Improving RTL Digital Circuit Testability. In *Proceedings of the 6th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, s. 303–304, Poznań, 2003. UNI-DRUK.
- [Str03b] J. Strnadel. Analýza a zlepšení testovatelnosti RTL číslicového obvodu. In *Sborník příspěvků ze semináře Počítačové Architektury & Diagnostika*, s. 24–29, Brno, 2003. FIT VUT v Brně.
- [Str03c] J. Strnadel. Nested Loops Degree Impact on RTL Digital Circuit Testability. In *Proceedings of Programmable Devices and Systems*, s. 202–207, Oxford, 2003. Elsevier.
- [Str03d] J. Strnadel. Scan Layout Encoding by Means of a Binary String. In *Proceedings of of 37th International Conference on Modelling and Simulation of Systems*, s. 115–122, Ostrava, 2003. MARQ.
- [Str06] J. Strnadel. Testability Analysis and Improvements of Register-Transfer Level Digital Circuits. *Computing and Informatics*, 25(5):441–464, 2006.
- [StD06] J. Strnadel and A. Dhali. Novel Optimizing Approach in the Area of STEP-Based Construction of Sessionless, Power-Constrained, TAM and Time Optimal Test Schedules. In *Proceedings of the 13th IEEE International Symposium and Workshop on the Engineering of Computer-Based Systems*, s. 360–367, IEEE, Los Alamitos, 2006.
- [Str07] J. Strnadel. On Encoding and Utilization of Diagnostic Information Extracted from Design-Data for Testability Analysis Purposes. In *Proceedings of the 6th Electronic Circuits and Systems Conference*, STUBA, 2007, Bratislava, s. 171–176.
- [Str08] J. Strnadel. Measuring Design for Testability Tool Effectiveness by Means of FITTest.BENCH06 Benchmark Circuits. *Computing and Informatics*, vyjde v roce 2008.
- [SW75] G. W. Smith and R. B. Walford. The Identification of a Minimal Feedback Vertex Set of a Directed Graph. *IEEE Transactions on Circuits and Systems*, CAS-22(1):9–14, 1975.
- [Syn04] Syntest Technologies. Tool Suite for Virtual Scan Synthesis and ATPG - VirtualScan, 2002 [2004]. Dokument dostupný z: <http://www.syntest.com/VirtualScan.htm>.
- [TB95] S. E. Tai and D. Bhattacharya. A Three-Stage Partial Scan Design Method to Ease ATPG. *Journal of Electronic Testing*, 7:95–104, 1995.
- [TBN02] C. Robach T. B. Nguyen, M. Delaunay. Testability Analysis For Software Components. In *Proceedings of the International Conference on Software Maintenance*, s. 422–429. IEEE Computer Society Digital Library, 2002.

- [TBN03] C. Robach T. B. Nguyen, M. Delaunay. Testability Analysis Applied to Embedded Data-flow Software. In *Proceedings of 3rd International Conference On Quality Software*, s. 351–359. IEEE Computer Society, 2003.
- [TIF02] A. Tamura T. Inoue, T. Miura and H. Fujiwara. A Scheduling Method in High-Level Synthesis for Acyclic Partial Scan Design. In *Proceedings of 11th Asian Test Symposium*, s. 128–133. IEEE Computer Society, 2002.
- [TOP95] Z. Kohavi T. Orensten and I. Pomeranz. An Optimal Algorithm for Cycle Breaking in Directed Graphs. *Journal of Electronic Testing*, 7:71–82, 1995.
- [TS82] N. N. Tendolkar and R. L. Swan. Automatic Diagnostic Methodology for the IBM 3081 Processor Complex. *IBM Journal on Research and Development*, 1(2):78–89, 1982.
- [Uba04] Ubar R. Functional Level Testability Analysis for Digital Circuits, 1992 [2004]. Technická zpráva LiTH-IDA-R-92-03 dostupná z: <http://the-compost-system.org/publications/cgi-bin/tr-fetch.pl?r-92-03+abstr>.
- [VA02] V. M. Vedula and J. A. Abraham. FACTOR: A Hierarchical Methodology for Functional Test Generation and Testability Analysis. In *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition*, s. 730–735. IEEE Computer Society, 2002.
- [VMVT03] J. Bhandra V. M. Vedula, J. A. Abraham and R. Tupuri. A Hierarchical Test Generation Approach Using Program Slicing Techniques on Hardware Description Languages. *Journal of Electronic Testing: Theory and Applications*, 19(1):149–160, 2003.
- [WP83] T. W. Williams and K. P. Parker. Design for Testability - a Survey. *IEEE Transactions on Computers*, 1(1):98–111, 1983.
- [XLR99] I. Pomeranz X. Lin and S. M. Reddy. Full Scan Fault Coverage With Partial Scan. In *Proceedings of Design, Automation and Test in Europe*, s. 468–472, 1999.
- [XP96] D. Xiang and J. H. Patel. A Global Algorithm for the Partial Scan Design Problem Using Circuit State Information. In *Proceedings of International Test Conference*, s. 548–557, 1996.
- [XP04] D. Xiang and J. H. Patel. Partial Scan Design Based on Circuit State Information and Functional Analysis. *IEEE Transactions on Computers*, 53(3):276–287, 2004.
- [YBP02] C. Landrault Y. Bonhomme, P. Girard and S. Pravossoudovitch. Power Driven Chaining of Flip-flops in Scan Architectures. In *Proceedings IEEE International Test Conference*, s. 796–803. IEEE Computer Society, 2002.
- [YM99] A. Orailoglu Y. Makris. Property-Based Testability Analysis for Hierarchical RTL Designs. In *Proceedings of the International Conference on Electronics Circuits and Systems (ICECS)*, s. 1089–1092, 1999.
- [YMV99] A. Orailoglu Y. Makris, J. Collins and P. Vishakantaiah. TRANS-PARENT: A System for RTL Testability Analysis, DFT Guidance and Hierarchical Test Generation. In *Proceedings of Custom Integrated Circuits Conference*, s. 159–162, 1999.

174 Reference

- [ZK99] Z. Kotásek. *Uplatnění principů říditelnosti/pozorovatelnosti při návrhu číslicových obvodů*. PhD thesis, FEI VUT v Brně, 1999. Habilitační práce.
- [ZK01a] F. Zbořil J. Strnadel Z. Kotásek, R. Růžička. Two Level Testability System. In *Proceedings of 35th International Conference on Modeling and Simulation of Systems*, s. 433–440, Ostrava, 2001. MARQ.
- [ZK01b] J. Strnadel Z. Kotásek. Analytic Approach to RTL Testability Analysis. In *Proceedings of 7th Conference Student FEI*, s. 363–367, Brno, 2001. VUT v Brně.
- [ZK01c] J. Strnadel Z. Kotásek. RTL Testability Analysis Based on Genetic Algorithm Implementation. In *Proceedings of the 1st International PhD Students’ Workshop Control & Information Technology*, s. 83–88, Ostrava, 2001. FEI VŠB.
- [ZK01d] J. Strnadel Z. Kotásek, R. Růžička. Formal and Analytical Approaches to the Testability Analysis - the Comparison. In *Proceedings of 4th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, s. 123–128, Győr, 2001. SU.
- [ZK03a] J. Strnadel Z. Kotásek, D. Mika. Methodologies of RTL Partial Scan Analysis and Their Comparison. In *Proceedings of the 6th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, s. 233–238, Poznań, 2003. UNI-DRUK.
- [ZK03b] J. Strnadel Z. Kotásek, D. Mika. Test Scheduling for Embedded Systems. In *Proceedings of Euromicro Symposium on Digital System Design - Architectures, Methods and Tools*, s. 463–467, Los Alamitos, 2003. IEEE Computer Society Press.
- [ZK04a] J. Strnadel Z. Kotásek, T. Pečenka. Improving Testability Parameters of Pipelined Circuits Through the Identification of Testable Cores. In *Proceedings of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, s. 99–104, Bratislava, 2004. SAV.
- [ZK04b] L. Sekanina J. Strnadel Z. Kotásek, T. Pečenka. Evolutionary Design of Synthetic RTL Benchmark Circuits. In *Proceedings of 9th European Test Symposium*, s. 107–108, Los Alamitos, 2004. IEEE Computer Society Press.

A

Tabulky

Tabulka 3. Ohodnocení diagnostických vlastností obvodu NLoops

| Port | Ohodnocení řid. | Ohodnocení poz. | Ohodnocení tes. |
|--------------------------|-----------------|-----------------|-----------------|
| R3.q\8 | 0.000000 | 1.000000 | 0.000000 |
| R3.d\8 | 0.000000 | 0.740132 | 0.000000 |
| R2.q\8 | 0.000000 | 0.000000 | 0.000000 |
| R2.d\8 | 0.000000 | 0.000000 | 0.000000 |
| R1.q\8 | 0.000000 | 0.000000 | 0.000000 |
| R1.d\8 | 0.000000 | 0.000000 | 0.000000 |
| FU3.q\8 | 0.000000 | 0.740132 | 0.000000 |
| FU3.b\8 | 0.000000 | 0.000000 | 0.000000 |
| FU3.a\8 | 0.000000 | 0.000000 | 0.000000 |
| FU2.q\8 | 0.000000 | 0.000000 | 0.000000 |
| FU2.b\8 | 0.000000 | 0.000000 | 0.000000 |
| FU2.a\8 | 0.000000 | 0.000000 | 0.000000 |
| FU1.q\8 | 0.000000 | 0.000000 | 0.000000 |
| FU1.b\8 | 0.000000 | 0.000000 | 0.000000 |
| FU1.a\8 | 1.000000 | 0.000000 | 0.000000 |
| NLoops.Out\8 | 0.000000 | 1.000000 | 0.000000 |
| NLoops.In\8 | 1.000000 | 0.000000 | 0.000000 |
| Ohodnocení <i>NLoops</i> | 0.065744 | 0.141732 | 0.009318 |
| (Počet/%) bran ze 136 | 16/11.8% | 32/23.5% | 0/0.0% |

176 A Tabulky

Tabulka 4. Ohodnocení diagnostických vlastností obvodu NLoops1

| Port | Ohodnocení říd. | Ohodnocení poz. | Ohodnocení tes. |
|---------------------------|-----------------|-----------------|-----------------|
| R3.SCAN_OUT\1 | 0.037073 | 1.000000 | 0.518537 |
| R3.SCAN_IN\1 | 0.342439 | 0.663415 | 0.502927 |
| R3.q\8 | 0.037073 | 1.000000 | 0.518537 |
| R3.d\8 | 0.011393 | 0.936585 | 0.473989 |
| R2.SCAN_OUT\1 | 0.342439 | 0.663415 | 0.502927 |
| R2.SCAN_IN\1 | 0.663415 | 0.342439 | 0.502927 |
| R2.q\8 | 0.342439 | 0.034722 | 0.188581 |
| R2.d\8 | 0.022135 | 0.342439 | 0.182287 |
| R1.SCAN_OUT\1 | 0.663415 | 0.342439 | 0.502927 |
| R1.SCAN_IN\1 | 1.000000 | 0.037073 | 0.518537 |
| R1.q\8 | 0.663415 | 0.012695 | 0.338055 |
| R1.d\8 | 0.033456 | 0.037073 | 0.035265 |
| FU3.q\8 | 0.011393 | 0.936585 | 0.473989 |
| FU3.b\8 | 0.037073 | 0.288651 | 0.162862 |
| FU3.a\8 | 0.342439 | 0.034722 | 0.188581 |
| FU2.q\8 | 0.022135 | 0.342439 | 0.182287 |
| FU2.b\8 | 0.037073 | 0.203879 | 0.120476 |
| FU2.a\8 | 0.663415 | 0.012695 | 0.338055 |
| FU1.q\8 | 0.033456 | 0.037073 | 0.035265 |
| FU1.b\8 | 0.037073 | 0.033171 | 0.035122 |
| FU1.a\8 | 1.000000 | 0.001374 | 0.500687 |
| NLoops1.SCAN_OUT\1 | 0.037073 | 1.000000 | 0.518537 |
| NLoops1.SCAN_IN\1 | 1.000000 | 0.037073 | 0.518537 |
| NLoops1.Out\8 | 0.037073 | 1.000000 | 0.518537 |
| NLoops1.In\8 | 1.000000 | 0.001374 | 0.500687 |
| Ohodnocení <i>NLoops1</i> | 0.634494 | 0.660173 | 0.418875 |
| (Počet/%) bran ze 144 | 144/100.0% | 144/100.0% | 144/100.0% |

Tabulka 5. Ohodnocení diagnostických vlastností obvodu NLoops2

| Port | Ohodnocení řid. | Ohodnocení poz. | Ohodnocení tes. |
|---------------------------|-----------------|-----------------|-----------------|
| R3.SCAN_OUT\1 | 0.663415 | 1.000000 | 0.831707 |
| R3.SCAN_IN\1 | 1.000000 | 0.663415 | 0.831707 |
| R3.q\8 | 0.663415 | 1.000000 | 0.831707 |
| R3.d\8 | 0.396107 | 0.936585 | 0.666346 |
| R2.SCAN_OUT\1 | 0.663415 | 1.000000 | 0.831707 |
| R2.SCAN_IN\1 | 1.000000 | 0.663415 | 0.831707 |
| R2.q\8 | 0.663415 | 0.621344 | 0.642380 |
| R2.d\8 | 0.396107 | 0.663415 | 0.529761 |
| R1.SCAN_OUT\1 | 0.663415 | 1.000000 | 0.831707 |
| R1.SCAN_IN\1 | 1.000000 | 0.663415 | 0.831707 |
| R1.q\8 | 0.663415 | 0.440119 | 0.551767 |
| R1.d\8 | 0.598691 | 0.663415 | 0.631053 |
| FU3.q\8 | 0.396107 | 0.936585 | 0.666346 |
| FU3.b\8 | 0.663415 | 0.559210 | 0.611312 |
| FU3.a\8 | 0.663415 | 0.621344 | 0.642380 |
| FU2.q\8 | 0.396107 | 0.663415 | 0.529761 |
| FU2.b\8 | 0.663415 | 0.396107 | 0.529761 |
| FU2.a\8 | 0.663415 | 0.440119 | 0.551767 |
| FU1.q\8 | 0.598691 | 0.663415 | 0.631053 |
| FU1.b\8 | 0.663415 | 0.597073 | 0.630244 |
| FU1.a\8 | 1.000000 | 0.440119 | 0.720059 |
| NLoops2.SCAN_OUT3\1 | 0.663415 | 1.000000 | 0.831707 |
| NLoops2.SCAN_IN3\1 | 1.000000 | 0.663415 | 0.831707 |
| NLoops2.SCAN_OUT2\1 | 0.663415 | 1.000000 | 0.831707 |
| NLoops2.SCAN_IN2\1 | 1.000000 | 0.663415 | 0.831707 |
| NLoops2.SCAN_OUT1\1 | 0.663415 | 1.000000 | 0.831707 |
| NLoops2.SCAN_IN1\1 | 1.000000 | 0.663415 | 0.831707 |
| NLoops2.Out\8 | 0.663415 | 1.000000 | 0.831707 |
| NLoops2.In\8 | 1.000000 | 0.440119 | 0.720059 |
| Ohodnocení <i>NLoops2</i> | 0.824327 | 0.833242 | 0.686864 |
| (Počet/%) bran ze 148 | 148/100.0% | 148/100.0% | 148/100.0% |

178 A Tabulky

Tabulka 6. Ohodnocení diagnostických vlastností obvodu NLoops3

| Port | Ohodnocení říd. | Ohodnocení poz. | Ohodnocení tes. |
|---------------------------|-----------------|-----------------|-----------------|
| R3.SCAN_OUT\1 | 0.265734 | 1.000000 | 0.632867 |
| R3.SCAN_IN\1 | 1.000000 | 0.265734 | 0.632867 |
| R3.q\8 | 0.265734 | 1.000000 | 0.632867 |
| R3.d\8 | 0.007935 | 0.885781 | 0.446858 |
| R2.q\8 | 0.033375 | 0.235382 | 0.134379 |
| R2.d\8 | 0.050810 | 0.209057 | 0.129934 |
| R1.q\8 | 0.213702 | 0.055554 | 0.134628 |
| R1.d\8 | 0.238479 | 0.048722 | 0.143601 |
| FU3.q\8 | 0.007935 | 0.885781 | 0.446858 |
| FU3.b\8 | 0.265734 | 0.026451 | 0.146093 |
| FU3.a\8 | 0.033375 | 0.235382 | 0.134379 |
| FU2.q\8 | 0.050810 | 0.209057 | 0.129934 |
| FU2.b\8 | 0.265734 | 0.039910 | 0.152822 |
| FU2.a\8 | 0.213702 | 0.055554 | 0.134628 |
| FU1.q\8 | 0.238479 | 0.048722 | 0.143601 |
| FU1.b\8 | 0.265734 | 0.043455 | 0.154595 |
| FU1.a\8 | 1.000000 | 0.012947 | 0.506474 |
| NLoops3.SCAN_OUT\1 | 0.265734 | 1.000000 | 0.632867 |
| NLoops3.SCAN_IN\1 | 1.000000 | 0.265734 | 0.632867 |
| NLoops3.Out\8 | 0.265734 | 1.000000 | 0.632867 |
| NLoops3.In\8 | 1.000000 | 0.012947 | 0.506474 |
| Ohodnocení <i>NLoops3</i> | 0.635249 | 0.652032 | 0.414203 |
| (Počet/%) bran ze 140 | 140/100.0% | 140/100.0% | 140/100.0% |

Tabulka 7. Ohodnocení diagnostických vlastností obvodu NLoops4

| Port | Ohodnocení říd. | Ohodnocení poz. | Ohodnocení tes. |
|---------------------------|-----------------|-----------------|-----------------|
| MX1.q\8 | 0.989247 | 0.741935 | 0.865591 |
| MX1.b\8 | 1.000000 | 0.733871 | 0.866935 |
| MX1.a\8 | 0.146702 | 0.733871 | 0.440286 |
| R3.q\8 | 0.733871 | 1.000000 | 0.866935 |
| R3.d\8 | 0.989247 | 0.741935 | 0.865591 |
| R2.q\8 | 0.219169 | 0.538567 | 0.378868 |
| R2.d\8 | 0.332762 | 0.355099 | 0.343930 |
| R1.q\8 | 0.497138 | 0.260597 | 0.378868 |
| R1.d\8 | 0.670742 | 0.128851 | 0.399796 |
| FU3.q\8 | 0.146702 | 0.733871 | 0.440286 |
| FU3.b\8 | 0.733871 | 0.146702 | 0.440286 |
| FU3.a\8 | 0.219169 | 0.538567 | 0.378868 |
| FU2.q\8 | 0.332762 | 0.355099 | 0.343930 |
| FU2.b\8 | 0.733871 | 0.160841 | 0.447356 |
| FU2.a\8 | 0.497138 | 0.260597 | 0.378868 |
| FU1.q\8 | 0.670742 | 0.128851 | 0.399796 |
| FU1.b\8 | 0.733871 | 0.117269 | 0.425570 |
| FU1.a\8 | 1.000000 | 0.094560 | 0.547280 |
| NLoops4.Out\8 | 0.733871 | 1.000000 | 0.866935 |
| NLoops4.dx\8 | 1.000000 | 0.733871 | 0.866935 |
| NLoops4.In\8 | 1.000000 | 0.094560 | 0.547280 |
| Ohodnocení <i>NLoops4</i> | 0.818592 | 0.728560 | 0.596393 |
| (Počet/%) bran ze 168 | 168/100.0% | 168/100.0% | 168/100.0% |

Tabulka 8. Ohodnocení diagnostických vlastností obvodu Bert

| Port | Ohodnocení říd. | Ohodnocení poz. | Ohodnocení tes. |
|--|------------------------|------------------------|------------------------|
| sub.q\8 | 0.637347 | 0.386427 | 0.511887 |
| sub.b\8 | 0.796927 | 0.308531 | 0.552729 |
| sub.a\8 | 0.824176 | 0.307954 | 0.566065 |
| mul2.q\8 | 0.959948 | 0.204858 | 0.582403 |
| mul2.b\8 | 0.992674 | 0.197497 | 0.595085 |
| mul2.a\8 | 0.996337 | 0.203357 | 0.599847 |
| mul1.q\8 | 0.797061 | 0.523286 | 0.660174 |
| mul1.b\8 | 1.000000 | 0.416462 | 0.708231 |
| mul1.a\8 | 0.821123 | 0.523286 | 0.672205 |
| add.q\8 | 0.797005 | 0.824176 | 0.810591 |
| add.b\8 | 0.996337 | 0.659139 | 0.827738 |
| add.a\8 | 0.824176 | 0.821157 | 0.822666 |
| reg5.q\8 | 0.796927 | 0.308531 | 0.552729 |
| reg5.d\8 | 0.959948 | 0.204858 | 0.582403 |
| reg4.q\8 | 0.661711 | 0.656624 | 0.659167 |
| reg4.d\8 | 0.797061 | 0.523286 | 0.660174 |
| reg3.q\8 | 0.824176 | 1.000000 | 0.912088 |
| reg3.d\8 | 0.992674 | 0.830281 | 0.911477 |
| reg2.q\8 | 0.827228 | 0.818116 | 0.822672 |
| reg2.d\8 | 0.996337 | 0.652059 | 0.824198 |
| reg1.q\8 | 0.824176 | 0.521274 | 0.672725 |
| reg1.d\8 | 0.992674 | 0.389446 | 0.691060 |
| mux8.q\8 | 0.992674 | 0.830281 | 0.911477 |
| mux8.c\8 | 0.824176 | 0.824176 | 0.824176 |
| mux8.b\8 | 1.000000 | 0.824176 | 0.912088 |
| mux8.a\8 | 0.797005 | 0.824176 | 0.810591 |
| mux7.q\8 | 0.996337 | 0.652059 | 0.824198 |
| mux7.b\8 | 1.000000 | 0.649626 | 0.824813 |
| mux7.a\8 | 0.797005 | 0.649626 | 0.723316 |
| mux6.q\8 | 0.992674 | 0.389446 | 0.691060 |
| mux6.c\8 | 0.824176 | 0.386427 | 0.605301 |
| mux6.b\8 | 1.000000 | 0.386427 | 0.693213 |
| mux6.a\8 | 0.637347 | 0.386427 | 0.511887 |
| mux5.q\8 | 0.824176 | 0.821157 | 0.822666 |
| mux5.b\8 | 0.827228 | 0.818116 | 0.822672 |
| mux5.a\8 | 0.824176 | 0.818116 | 0.821146 |
| mux4.q\8 | 0.996337 | 0.659139 | 0.827738 |
| mux4.b\8 | 1.000000 | 0.656624 | 0.828312 |
| mux4.a\8 | 0.661711 | 0.656624 | 0.659167 |
| mux3.q\8 | 0.996337 | 0.203357 | 0.599847 |
| mux3.b\8 | 0.661711 | 0.202534 | 0.432122 |
| mux3.a\8 | 1.000000 | 0.202534 | 0.601267 |
| mux2.q\8 | 0.992674 | 0.197497 | 0.595085 |
| mux2.c\8 | 1.000000 | 0.195844 | 0.597922 |
| mux2.b\8 | 0.796927 | 0.195844 | 0.496386 |
| mux2.a\8 | 0.827228 | 0.195844 | 0.511536 |
| mux1.q\8 | 0.821123 | 0.523286 | 0.672205 |
| mux1.b\8 | 0.824176 | 0.521274 | 0.672725 |
| mux1.a\8 | 0.824176 | 0.521274 | 0.672725 |
| Bert.s\8 | 0.824176 | 1.000000 | 0.912088 |
| Bert.rom2\8 | 1.000000 | 0.202534 | 0.601267 |
| Bert.rom1\8 | 1.000000 | 0.656624 | 0.828312 |
| Bert.c\8 | 1.000000 | 0.386427 | 0.693213 |
| Bert.b\8 | 1.000000 | 0.649626 | 0.824813 |
| Bert.a\8 | 1.000000 | 0.824176 | 0.912088 |
| Ohodnocení Bert (Počet/%) bran ze 440 | 0.942542 440/100.0% | 0.765381 440/100.0% | 0.721403 440/100.0% |

Tabulka 9. Ohodnocení diagnostických vlastností obvodu Diffeq

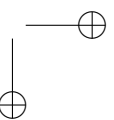
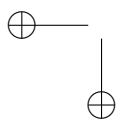
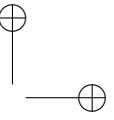
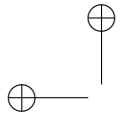
| Port | Ohodnocení fid. | Ohodnocení poz. | Ohodnocení tes. |
|-----------------------|-----------------|-----------------|-----------------|
| mux6.q\8 | 0.995935 | 0.000000 | 0.000000 |
| mux6.b\8 | 0.815614 | 0.000000 | 0.000000 |
| mux6.a\8 | 1.000000 | 0.000000 | 0.000000 |
| mux5.q\8 | 0.000000 | 0.822427 | 0.000000 |
| mux5.b\8 | 0.000000 | 0.818957 | 0.000000 |
| mux5.a\8 | 0.000000 | 0.818957 | 0.000000 |
| mux4.q\8 | 0.991870 | 0.000000 | 0.000000 |
| mux4.c\8 | 0.000000 | 0.000000 | 0.000000 |
| mux4.b\8 | 0.000000 | 0.000000 | 0.000000 |
| mux4.a\8 | 1.000000 | 0.000000 | 0.000000 |
| mux3.q\8 | 0.995935 | 0.000000 | 0.000000 |
| mux3.b\8 | 1.000000 | 0.000000 | 0.000000 |
| mux3.a\8 | 0.000000 | 0.000000 | 0.000000 |
| mux2.q\8 | 0.812158 | 0.000000 | 0.000000 |
| mux2.b\8 | 0.815614 | 0.000000 | 0.000000 |
| mux2.a\8 | 0.000000 | 0.000000 | 0.000000 |
| mul1.q\8 | 0.995935 | 0.000000 | 0.000000 |
| mul1.b\8 | 1.000000 | 0.000000 | 0.000000 |
| mul1.a\8 | 0.667837 | 0.000000 | 0.000000 |
| reg6.q\8 | 0.000000 | 0.967480 | 0.000000 |
| reg6.d\8 | 0.000000 | 0.825784 | 0.000000 |
| reg6.g\8 | 0.815614 | 0.000000 | 0.000000 |
| reg5.d\8 | 0.955582 | 0.000000 | 0.000000 |
| reg4.q\8 | 0.667837 | 0.000000 | 0.000000 |
| reg4.d\8 | 0.782445 | 0.000000 | 0.000000 |
| reg3.q\8 | 0.000000 | 0.818957 | 0.000000 |
| reg3.d\8 | 0.000000 | 0.679572 | 0.000000 |
| reg2.q\8 | 0.000000 | 0.818957 | 0.000000 |
| reg2.d\8 | 0.000000 | 0.679572 | 0.000000 |
| reg1.q\8 | 0.000000 | 0.000000 | 0.000000 |
| reg1.d\8 | 0.000000 | 0.000000 | 0.000000 |
| comp.q\8 | 0.000000 | 1.000000 | 0.000000 |
| comp.b\8 | 0.000000 | 0.967480 | 0.000000 |
| comp.a\8 | 1.000000 | 0.000000 | 0.000000 |
| mul2.q\8 | 0.955582 | 0.000000 | 0.000000 |
| mul2.b\8 | 0.991870 | 0.000000 | 0.000000 |
| mul2.a\8 | 0.995935 | 0.000000 | 0.000000 |
| mul1.q\8 | 0.782445 | 0.000000 | 0.000000 |
| mul1.b\8 | 0.812158 | 0.000000 | 0.000000 |
| mul1.a\8 | 0.995935 | 0.000000 | 0.000000 |
| sub.q\8 | 0.000000 | 0.000000 | 0.000000 |
| sub.b\8 | 0.000000 | 0.000000 | 0.000000 |
| sub.a\8 | 0.667837 | 0.000000 | 0.000000 |
| add.q\8 | 0.000000 | 0.825784 | 0.000000 |
| add.b\8 | 0.995935 | 0.000000 | 0.000000 |
| add.a\8 | 0.000000 | 0.822427 | 0.000000 |
| Diffeq.result\8 | 0.000000 | 1.000000 | 0.000000 |
| Diffeq.const2\8 | 1.000000 | 0.000000 | 0.000000 |
| Diffeq.dx\8 | 1.000000 | 0.000000 | 0.000000 |
| Diffeq.a\8 | 1.000000 | 0.000000 | 0.000000 |
| Ohodnocení Diffeq | 0.422856 | 0.173226 | 0.073250 |
| (Počet/%) bran ze 400 | 224/56.0% | 112/28.0% | 0/0.0% |

Tabulka 10. Ohodnocení diagnostických vlastností obvodu Paulin

| Port | Ohodnocení říd. | Ohodnocení poz. | Ohodnocení tes. |
|-----------------------|-----------------|-----------------|-----------------|
| sub.q\8 | 0.528018 | 0.869444 | 0.698731 |
| sub.b\8 | 0.623234 | 0.736613 | 0.679923 |
| sub.a\8 | 0.869444 | 0.541867 | 0.705656 |
| mul2.q\8 | 0.526287 | 0.625238 | 0.575762 |
| mul2.b\8 | 0.623234 | 0.527566 | 0.575400 |
| mul2.a\8 | 0.866667 | 0.389670 | 0.628168 |
| mul1.q\8 | 0.731906 | 0.627242 | 0.679574 |
| mul1.b\8 | 0.869444 | 0.527561 | 0.698503 |
| mul1.a\8 | 0.863889 | 0.545352 | 0.704621 |
| add.q\8 | 0.734259 | 0.869444 | 0.801852 |
| add.b\8 | 0.869444 | 0.734259 | 0.801852 |
| add.a\8 | 0.866667 | 0.755934 | 0.811300 |
| reg7.q\8 | 0.449619 | 0.734198 | 0.591908 |
| reg7.d\8 | 0.526287 | 0.627242 | 0.576764 |
| reg6.q\8 | 0.625291 | 0.734198 | 0.679744 |
| reg6.d\8 | 0.731906 | 0.627242 | 0.679574 |
| reg5.q\8 | 0.869444 | 0.734259 | 0.801852 |
| reg5.d\8 | 0.996825 | 0.627302 | 0.812063 |
| reg4.q\8 | 0.869444 | 0.527561 | 0.698503 |
| reg4.d\8 | 0.996825 | 0.438144 | 0.717485 |
| reg3.q\8 | 0.869444 | 0.753519 | 0.811481 |
| reg3.d\8 | 0.996825 | 0.643803 | 0.820314 |
| reg2.q\8 | 0.869444 | 1.000000 | 0.934722 |
| reg2.d\8 | 0.996825 | 0.872222 | 0.934524 |
| reg1.q\8 | 0.869444 | 1.000000 | 0.934722 |
| reg1.d\8 | 0.996825 | 0.872222 | 0.934524 |
| mux11.q\8 | 0.623234 | 0.736613 | 0.679923 |
| mux11.b\8 | 0.625291 | 0.734198 | 0.679744 |
| mux11.a\8 | 0.449619 | 0.734198 | 0.591908 |
| mux10.q\8 | 0.866667 | 0.527566 | 0.697116 |
| mux10.b\8 | 0.869444 | 0.525778 | 0.697611 |
| mux10.a\8 | 0.625291 | 0.525778 | 0.575534 |
| mux9.q\8 | 0.866667 | 0.389670 | 0.628168 |
| mux9.b\8 | 0.449619 | 0.388384 | 0.419001 |
| mux9.a\8 | 0.869444 | 0.388384 | 0.628914 |
| mux8.q\8 | 0.866667 | 0.545352 | 0.706009 |
| mux8.b\8 | 0.866667 | 0.543552 | 0.705109 |
| mux8.a\8 | 0.869444 | 0.543552 | 0.706498 |
| mux7.q\8 | 0.866667 | 0.543552 | 0.705109 |
| mux7.b\8 | 0.869444 | 0.541752 | 0.705598 |
| mux7.a\8 | 0.869444 | 0.541752 | 0.705598 |
| mux6.q\8 | 0.866667 | 0.755934 | 0.811300 |
| mux6.b\8 | 0.869444 | 0.753519 | 0.811481 |
| mux6.a\8 | 0.869444 | 0.753519 | 0.811481 |
| mux5.q\8 | 0.996825 | 0.627302 | 0.812063 |
| mux5.b\8 | 0.526287 | 0.625238 | 0.575762 |
| mux5.a\8 | 1.000000 | 0.625238 | 0.812619 |
| mux4.q\8 | 0.996825 | 0.438144 | 0.717485 |
| mux4.b\8 | 0.731906 | 0.436653 | 0.584280 |
| mux4.a\8 | 1.000000 | 0.436653 | 0.718327 |
| mux3.q\8 | 0.996825 | 0.643803 | 0.820314 |
| mux3.b\8 | 1.000000 | 0.641733 | 0.820866 |
| mux3.a\8 | 0.734259 | 0.641733 | 0.687996 |
| mux2.q\8 | 0.996825 | 0.872222 | 0.934524 |
| mux2.b\8 | 1.000000 | 0.869444 | 0.934722 |
| mux2.a\8 | 0.528018 | 0.869444 | 0.698731 |
| mux1.q\8 | 0.996825 | 0.872222 | 0.934524 |
| mux1.b\8 | 1.000000 | 0.869444 | 0.934722 |
| mux1.a\8 | 0.734259 | 0.869444 | 0.801852 |
| Paulin.OUT.PORT2\8 | 0.869444 | 1.000000 | 0.934722 |
| Paulin.OUT.PORT1\8 | 0.869444 | 1.000000 | 0.934722 |
| Paulin.ONE\8 | 1.000000 | 0.869444 | 0.934722 |
| Paulin.IN.PORT2\8 | 1.000000 | 0.436653 | 0.718327 |
| Paulin.IN.PORT1\8 | 1.000000 | 0.625238 | 0.812619 |
| Ohodnocení Paulin | 0.911514 | 0.833478 | 0.759727 |
| (Počet/%) bran ze 512 | 512/100.0% | 512/100.0% | 512/100.0% |

Tabulka 11. Ohodnocení diagnostických vlastností obvodu Tseng

| Port | Ohodnocení fid. | Ohodnocení poz. | Ohodnocení tes. |
|-----------------------|-----------------|-----------------|-----------------|
| mux6.q\8 | 0.995122 | 0.425468 | 0.710295 |
| mux6.b\8 | 0.650902 | 0.423351 | 0.537127 |
| mux6.a\8 | 1.000000 | 0.423351 | 0.711676 |
| mux5.q\8 | 0.825203 | 0.537127 | 0.681165 |
| mux5.b\8 | 0.825203 | 0.534481 | 0.679842 |
| mux5.a\8 | 0.829268 | 0.534481 | 0.681874 |
| mux4.q\8 | 0.995122 | 0.829268 | 0.912195 |
| mux4.b\8 | 0.515959 | 0.825203 | 0.670581 |
| mux4.a\8 | 1.000000 | 0.825203 | 0.912602 |
| mux3.q\8 | 0.995122 | 0.654240 | 0.824681 |
| mux3.b\8 | 0.825203 | 0.650902 | 0.738053 |
| mux3.a\8 | 1.000000 | 0.650902 | 0.825451 |
| mux2.q\8 | 0.821138 | 0.825223 | 0.823181 |
| mux2.c\8 | 0.829268 | 0.817133 | 0.823200 |
| mux2.b\8 | 0.825203 | 0.817133 | 0.821168 |
| mux2.a\8 | 0.650902 | 0.817133 | 0.734017 |
| mux1.q\8 | 0.995122 | 0.650470 | 0.822796 |
| mux1.b\8 | 1.000000 | 0.647234 | 0.823617 |
| mux1.a\8 | 0.515959 | 0.647234 | 0.581596 |
| reg5.q\8 | 0.650902 | 1.000000 | 0.825451 |
| reg5.d\8 | 0.785088 | 0.829268 | 0.807178 |
| reg4.q\8 | 0.825203 | 0.534481 | 0.679842 |
| reg4.d\8 | 0.995122 | 0.425468 | 0.710295 |
| reg3.q\8 | 0.825203 | 1.000000 | 0.912602 |
| reg3.d\8 | 0.995122 | 0.829268 | 0.912195 |
| reg2.q\8 | 0.829268 | 0.817133 | 0.823200 |
| reg2.d\8 | 1.000000 | 0.650470 | 0.825235 |
| reg1.q\8 | 0.825203 | 0.817133 | 0.821168 |
| reg1.d\8 | 0.995122 | 0.650470 | 0.822796 |
| add2.q\8 | 0.785088 | 0.829268 | 0.807178 |
| add2.b\8 | 0.995122 | 0.654240 | 0.824681 |
| add2.a\8 | 0.821138 | 0.825223 | 0.823181 |
| add1.q\8 | 0.515959 | 0.825203 | 0.670581 |
| add1.b\8 | 0.650902 | 0.654125 | 0.652513 |
| add1.a\8 | 0.825203 | 0.537127 | 0.681165 |
| sub.q\8 | 0.515959 | 0.647234 | 0.581596 |
| sub.b\8 | 0.650902 | 0.512735 | 0.581819 |
| sub.a\8 | 0.825203 | 0.421286 | 0.623245 |
| Tseng.v15\8 | 0.825203 | 1.000000 | 0.912602 |
| Tseng.v14\8 | 0.650902 | 1.000000 | 0.825451 |
| Tseng.v1\8 | 1.000000 | 0.423351 | 0.711676 |
| Tseng.v6\8 | 1.000000 | 0.825203 | 0.912602 |
| Tseng.v2\8 | 1.000000 | 0.650902 | 0.825451 |
| Tseng.v4\8 | 1.000000 | 0.650470 | 0.825235 |
| Tseng.v10\8 | 1.000000 | 0.647234 | 0.823617 |
| Ohodnocení Tseng | 0.920917 | 0.846587 | 0.779636 |
| (Počet/%) bran ze 360 | 360/100.0% | 360/100.0% | 360/100.0% |



B

Trojúhelníky čísel

Tabulka 12. Část trojúhelníku STIRLINGových čísel 2. řádu a BELLOVých čísel pro $1 \leq k \leq 10$

| | $S_2(k, i)$ | | | | | | | | | | b_k |
|-----------------|-------------|-----|------|-------|-------|-------|------|-----|----|----|--------|
| $k \setminus i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 1 | 1 | - | - | - | - | - | - | - | - | - | 1 |
| 2 | 1 | 1 | - | - | - | - | - | - | - | - | 2 |
| 3 | 1 | 3 | 1 | - | - | - | - | - | - | - | 5 |
| 4 | 1 | 7 | 6 | 1 | - | - | - | - | - | - | 15 |
| 5 | 1 | 15 | 25 | 10 | 1 | - | - | - | - | - | 52 |
| 6 | 1 | 31 | 90 | 65 | 15 | 1 | - | - | - | - | 203 |
| 7 | 1 | 63 | 301 | 350 | 140 | 21 | 1 | - | - | - | 877 |
| 8 | 1 | 127 | 966 | 1701 | 1050 | 266 | 28 | 1 | - | - | 4140 |
| 9 | 1 | 255 | 3025 | 7770 | 6951 | 2646 | 462 | 36 | 1 | - | 21147 |
| 10 | 1 | 511 | 9330 | 34105 | 42525 | 22827 | 5880 | 750 | 45 | 1 | 115975 |

186 B Trojúhelníky čísel

Tabulka 13. Část trojúhelníku LAHOVÝCH čísel a a_k čísel pro $1 \leq k \leq 10$

| | <i>Lah(k, i)</i> | | | | | | | a_k |
|-----------------|------------------|----------|----------|----------|-----|----------|-----------|----------|
| $k \setminus i$ | 1 | 2 | 3 | 4 | ... | 9 | 10 | |
| 1 | 1 | - | - | - | ... | - | - | 1 |
| 2 | 2 | 1 | - | - | ... | - | - | 3 |
| 3 | 6 | 6 | 1 | - | ... | - | - | 13 |
| 4 | 24 | 36 | 12 | 1 | ... | - | - | 73 |
| 5 | 120 | 240 | 120 | 20 | ... | - | - | 501 |
| 6 | 720 | 1800 | 1200 | 300 | ... | - | - | 4051 |
| 7 | 5040 | 15120 | 12600 | 4200 | ... | - | - | 37633 |
| 8 | 40320 | 141120 | 141120 | 58800 | ... | - | - | 394353 |
| 9 | 362880 | 1451520 | 1693440 | 846720 | ... | 1 | - | 4596553 |
| 10 | 3628800 | 16329600 | 21772800 | 12700800 | ... | 90 | 1 | 58941091 |

Název Analýza a zlepšení testovatelnosti číslicového obvodu
 na úrovni meziregistrových přenosů
Autor Ing. Josef Strnadel, Ph.D.

Vydavatel Vysoké učení technické v Brně
 Fakulta informačních technologií
Obálka Mgr. Dagmar Hejduková
Tisk MJ servis, spol. s r.o.
Vyšlo Brno, 2008
Vydání první

Tato publikace neprošla redakční ani jazykovou úpravou.