VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů

Ing. Richard Růžička

FORMÁLNÍ PŘÍSTUP K ANALÝZE TESTOVATELNOSTI ČÍSLICOVÝCH OBVODŮ NA ÚROVNI RT

FORMAL APPROACH TO THE TESTABILITY ANALYSIS OF RT LEVEL DIGITAL CIRCUITS

ZKRÁCENÁ VERZE PHD THESIS

Obor: Informační technologie

Školitel: Doc. Ing. Zdeněk Kotásek, CSc.

Oponenti: Prof. Ing. Ondřej Novák, CSc.
Doc. RNDr. Alexander Meduna, CSc.
RNDr. Elena Gramatová, CSc.

Datum obhajoby: 11. 11. 2002

KLÍČOVÁ SLOVA
Diagnostika, testovatelnost, úroveň RT, Petriho sítě, Částečný scan

KEY WORDS
Diagnostics, testability, RT level, Petri nets, Partial scan

Práce je uložena na Fakultě informačních technologií VUT v Brně.

ISBN 80-214-2342-0
ISSN 1213-4198

Obsah

Tato práce vznikla za podpory grantového úkolu GAČR 102/01/153 „Formální postupy v diagnostice číslicových obvodů – verifikace testovatelného návrhu“.

ABSTRAKT

Hlavním předmětem práce je analýza testovatelnosti číslicového obvodu v rané etapě jeho návrhu. Výsledkem budiž zjištění, nakolik je navržený obvod testovatelný, návrh mechanismu jeho testování a případná doporučení na modifikace obvodu k zajištění lepší testovatelnosti, přičemž bude přihlédnuto k jejich dopadu na cenu výsledného obvodu. Analýza směřuje k co největšímu využití datových cest, vzniklých v obvodě návrhem i pro přenos diagnostických dat. Je zvolen koncept tzv. *i cest*, tedy datových cest, které jsou pro přenášená data transparentní, nemění je.

Předmětem analýzy je datová část (datové cesty) číslicového obvodu na úrovni abstrakce odpovídající úrovni meziregistrových přenosů (Register Transfer, RT). Obvod je popsán strukturou sestávající ze vzájemně propojených bloků jako jsou např. funkční jednotky provádějící základní aritmeticko-logické operace, multiplexorů a registry. Úroveň RT je na poměrně vysoké úrovni abstrakce modelování číslicového obvodu, přitom už nese informace o jeho struktuře, přitom ještě nese jasné informace o funkci obvodu, které se neztratily v procesu syntézy, myšlenky návrháře jsou stále zřejmé, ještě úplně nezanikly v záplavě automatizovaně generovaných opakujících se a navzájem si podobných struktur jako je tomu už o úroveň níž, na úrovni hradel. Spoj v obvodě na úrovni RT přenáší najednou celé vícebitové slovo. Právě tyto skutečnosti považuji za velmi cenné. Jedním z cílů práce je využít k optimalizaci diagnostických postupů i informace o funkci obvodu. Využitím těchto informací lze nalézt v obvodě *i cest* daleko více.

Je zvolen formální přístup k celém problematice, což zahrnuje formální model obvodu, formální popis vlastností obvodu souvisejících s analýzou testovatelnosti a formální zápis navržených algoritmů. Využívá se pojmů diskrétní matematiky. Veškeré entity, které jsou předmětem analýzy (obvodové prvky, spoje atd.) jsou podle svých vlastností sdruženy do množin, další vlastnosti a vztahy mezi těmito entitami jsou vyjádřeny relacemi. Pro popis je volen jazyk predikátové logiky. Výhodami formálního přístupu jsou zejména jednoznačnost zápisu a možnost transformovat problémy analýzy testovatelnosti na již dobře známé a řešené problémy diskrétní matematiky a teoretické informatiky a využít již známých a optimalizovaných postupů a algoritmů a též možnost závěry rigorózně dokázat.

Zvláštní role náleží obvodovým registrům. Je to dáno tím, že se vychází z principů tzv. strukturovaného návrhu, kdy se kombinační a sekvenční logika navzájem striktně odděluje. Registry se stávají významnými body na *i cestách*, po kterých diagnostická data obvodem prochází. Pokud není nalezena analýzou vhodná *i cesta* z vnějšku/dovnitř obvodu, hledá se právě vhodný registr a pak *i cesta* z/do něj. Tento registr je pak třeba v duchu strukturovaného návrhu upravit tak, aby byl z vnějšku obvodu přístupný. Celý mechanismus směřuje k tomu, aby právě takové registry byly začleňovány do sériového řetězce typu *„scan“*. Prioritou je však nalézt *i cesty* ve stávající struktuře obvodu a až v případech, kdy se to nezdaří, modifikovat některý vhodný registr pro zařazení do řetězce *„scan“*. Proto je možné hovořit o *částečném „scanu“*.

1 ÚVOD

1.1 Diagnostika

Důležitým aspektem kvalitně provedeného návrhu je testovatelnost. Dnešní technologie umožňují výrobu i celých systémů na jednom čipu. Při dnešní složitosti obvodů je nutno již při návrhu zvažovat, jak bude obvod testován. Proto metody označované jako návrh či syntéza obvodu pro snadnou testovatelnost.

Při úvahách o diagnostice číslicového obvodu je třeba řešit dva okruhy problémů:

- Vytvoření souboru testovacích dat, které se budou aplikovat na vstupy prvků.
- Způsob aplikace testu – jakým způsobem dopravit testovací data na vstupy prvků, pro něž jsou určeny a jakým způsobem dopravit odezvy na ně od výstupů testovaného prvku k zařízení, které odezvy posuzuje. V praxi se také někdy hovoří o tzv. protokolu testu [Mar01].

Tato práce se zabývá právě problematikou aplikace testu.

1.2 Současný stav

V literatuře jsem nenalezl podobný přístup k analýze testovatelnosti, jako je popsán v této práci, který by byl založen čistě na analýze struktury. V obdobných přístupech vždy dojde k prohledávání stavového prostoru, přístupy se od sebe liší hlavně použitou heuristikou. Často jde o kvantitativní vyjádření míry řiditelnosti či pozorovatelnosti uzlů v obvodě nějakou číselnou hodnotou, která se pak mění v závislosti na tom, jak je obvod upravován pro zvýšení testovatelnosti. Celá metodika sestává zpravidla ze systému analytických vztahů, do nichž se dosazují konkrétní hodnoty získané analýzou struktury obvodu a pak vyčíslují.

1.3 Cíl práce

Cílem je analýza testovatelnosti číslicového obvodu v rané etapě jeho návrhu. Výsledkem bude zjištění, nakolik je navržený obvod testovatelný. Předpokládá se analýza složitých sekvenčních obvodů, obvod se tedy pro testování rozdělí na menší (vhodně volené) části. Bude užito některé metody strukturovaného návrhu. Analýza by měla být schopna určit, která část obvodu je bez problémů testovatelná a u které je v navrženém obvodě testování obtížné. Dále bude předmětem práce návrh mechanismu testování obvodu, tj. jakým způsobem aplikovat vygenerované testovací vektory a snímat odezvy na tyto testovací vektory. Pro každou testovanou část obvodu bude určena vhodná cesta testovacích vektorů i odezev a jednoznačně určeno co je třeba nastavit, aby tato cesta byla průchozí. Případná doporučení na modifikace obvodu by měla směřovat k co nejmenšímu dopadu na cenu výsledného obvodu. Veškeré postupy a mechanismy, použité k výše uvedeným cílům by měly být zapsány formálně, čímž vlastně dojde k převedení problémů analýzy testovatelnosti na známé a již optimálně vyřešené problémy. Cílem je též ověřit vhodnost Petriho sítí jako aparátu k analýze dosažitelnosti uzlů ve struktuře číslicového obvodu.

2 ZVOLENÉ METODY ZPRACOVÁNÍ

2.1 Metody strukturovaného návrhu

Strukturu klasického číslicového obvodu lze rozdělit jednak na blok kombinační logiky, jednak na blok paměťových prvků, jejichž stav odráží stav celého obvodu. 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:

- nastavení požadované kombinace na vstupu kombinační sítě (nastavení vstupů a nastavení požadovaného stavu),
- provedení jednoho kroku obvodu (generování synchronizačních pulsů),
- kontroly výsledku testu (ověření výstupní kombinace, ověření 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 jeho ověření. Zlepšení lze dosáhnout např. tím, že do údajových vstupů klopných obvodů se zařadí přepínač, na jehož vstup *a* se přivedou výstupy kombinační sítě, na vstup *b* výstup předcházejícího klopného obvodu. V režimu běžné funkce tvoří jednotlivé klopné obvody paralelní registr, do něhož se ukládají výstupy kombinační sítě. V režimu testu jsou jednotlivé klopné obvody zapojeny jako posuvný registr. Tato úprava umožní:

- vložit před provedením kroku testu do posuvného registru požadovanou kombinaci,
- zkontrolovat na výstupu odezvu na vstupní vzorek.

Jednou z prvních prací zabývajících se popsanou problematikou je [Wil73]. Její nesporný přínos je také v tom, že se zabývá rovněž ekonomickými aspekty aplikace těchto principů.

Je zřejmé, že použitím některé z metod strukturovaného návrhu se pomyslně oddělí kombinační logika od sekvenčních prvků (odtud název „metody strukturovaného návrhu“) a problém testování kombinačních sítí, které jsou součástí složitějšího systému, se tak zjednoduší.

Výrazná pozornost byla v odborné literatuře věnována principům strukturovaného návrhu v 80. letech. Tato technika byla přijata všemi významnými světovými výrobci. Největší popularity dosáhla v systémech IBM pod názvem *LSSD* (Level-Sensitive Scan Design) [Ten82].

2.2 Číslicový obvod na úrovni RT

Pomocí automatizovaných prostředků pro syntézu obvodu na vyšší úrovni popisu se získá už strukturální model obvodu, model obvodu na úrovni meziregistrových přenosů (RT). Zde už má smysl uplatňovat úvahy o diagnostice rodícího se obvodu, respektive o analýze testovatelnosti vznikající struktury. Je třeba podotknout, že výsledkem syntézy obvodu na vyšší úrovni popisu (High-Level Synthesis, HLS) je struktura datových cest obvodu a dále pak řídicí jednotka (řadič), která ovládá

výpočet – tok dat datovými cestami. Tato práce se zabývá testovatelností datových cest, proto bude nadále zmiňována pouze datová část. V návrhovém systému pak pokračuje syntéza obvodu přes úroveň hradel, případně úroveň tranzistorů až do fáze generování rozvržení čipu.

Zobrazení popis chování → popis RT struktury není injektivní. Například RT struktura realizující nějaký algoritmus může mít mnoho podob od sekvenčního (sériového) po čistě paralelní řešení (což bude zřejmě hlavně záviset na podmínkách syntézy a požadavcích návrháře – požadavky na rychlý výpočet povedou spíše na paralelní řešení, kdežto prostorová omezení zapříčiní spíše sériové řešení). Výsledná struktura závisí též na zvolené (resp. návrhovým systémem podporované) propojovací strategii, na knihovně obvodových prvků, se kterou systém pracuje, atd.

Úroveň RT je nejabstraktnější úroveň modelování číslicového obvodu, která už nese informace o jeho struktuře a v které je současně možné nalézt dostatek transparentních cest, které jsou pro zvolený styl analýzy testovatelnosti důležité. Proto je možné začít se zabývat analýzou testovatelnosti už na této úrovni. Navíc však ještě nese další informace o jeho funkci, která už není zcela zřejmá na úrovni hradel (na níž pracuje drtivá většina dosud publikovaných prací z oboru analýzy testovatelnosti – viz [KoR00b]). Právě tuto skutečnost považuji za velmi cennou. Jedním z cílů této práce je využít k optimalizaci diagnostických postupů i informaci o funkci obvodu, jež se ve struktuře RT objevuje.

Typické pro obvod popsaný na úrovni RT jsou tyto skutečnosti[Mae92]:

- Datová část sestává z funkčních jednotek propojených vícebitovými spoji, paměťových prvků jako jsou registry či paměti a propojovacích prvků – multiplexorů, sběrnic atd. V dalším textu budou tyto elementární bloky obvodu na úrovni RT nazývány **obvodové prvky** nebo tam, kde by nemuselo dojít ke kolizi pojmů (např. s pojmem prvek množiny) též zkráceně prvky. Registry je reprezentována sekvenční část, lze říci, že registry jsou nositeli stavu obvodu, funkční jednotky a multiplexory mají charakter kombinační. Z pohledu metod strukturovaného návrhu popsaných stručně v předchozí kapitole má proto smysl věnovat registrům zvláštní pozornost, pomyslně je vydělit z celkové struktury obvodu.
- Obvodový řadič je reprezentován jako stavový automat.
- Řídicí cesty (vodiče) slouží k otevírání či adresování paměťových prvků, přepínají multiplexory či budiče sběrnice a přivádí operační kódy k aritmeticko-logickým jednotkám.

Obvod pracuje v diskrétních časových krocích, lhostejno, zda pro výslednou implementaci je zvoleno schéma s jedno- či vícefázovými hodinami. Nejjednodušší situace nastane, pokud jsou v obvodě hranově závislé registry řízené jednofázovými hodinami. Za předpokladu, že užité funkční jednotky jsou kombinačního charakteru bez vnitřního zřetězení, pak jeden krok řadiče (přechod z jednoho stavu řadiče do jiného) je vymezen jedním hodinovým cyklem.

Výsledná struktura datové části obvodu popsaná na strukturální úrovni meziregistrových přenosů (RT) se může lišit podle použité propojovací strategie.

Mezi dvě nejpoužívanější strategie patří multiplexované datové cesty a obousměrné sběrnice [Mae92]. Právě obvod na úrovni RT syntetizovaný s využitím strategie multiplexovaných datových cest je předmětem analýzy popsané v této práci.

Pro propojovací strategii multiplexovaných datových cest jsou typické tyto skutečnosti:

- aritmetické a logické operace jsou realizovány čistě kombinačními obvody,
- pro uložení dat se používají jednotlivé registry,
- přepínání spojů (postavení datové cesty) je realizováno multiplexory,
- všechny registry jsou řízeny jediným společným hodinovým signálem.

2.3 Formální model obvodu

Pro jednoznačnost a korektnost dalšího zápisu jsem zavedl (viz články [KRH00] a [KoR00c] a vyžádaný referát [KoR00d]) formální model struktury číslicového obvodu na úrovni RT využívajícího multiplexovaných datových cest. Základní pětice množin vyjadřuje staticky strukturu obvodu, k tomu se přidávají zobrazení určující chování prvků a další vztahy. Takto navržený model není samoúčelný, snahou je co nejlépe vystihnout strukturu popisu skutečných obvodů tak, jak se v praxi používá např. formou jazyků pro popis obvodů. O to jednodušší pak bude využití celého navrženého aparátu v praxi.

2.3.1 Obvod

Uspořádaná pětice ***UUA=(E, P, C , PI, PO)*** vyjadřuje model struktury číslicového obvodu na úrovni meziregistrových přenosů (RT). Význam jednotlivých množin je tento: ***E*** je množina obvodových prvků, ***P*** je množina jejich bran (vstupů a výstupů), ***C*** je množina spojů mezi branami prvků obvodu, ***PI*** je množina primárních vstupů obvodu a ***PO*** je množina primárních výstupů obvodu.

Model obvodu vychází z tradičního pohledu na strukturu číslicového obvodu na úrovni RT. Podobný pohled byl použit např. v práci [GuB95], kde se na obvod nahlíží jako na orientovaný graf s uzly tvořenými prvky a hranami tvořenými spoji. Podobný model je možné nalézt též v [Mar02].

V zásadě jsou možné dva pohledy na způsob, jakým modelovat primární brány obvodu. Jeden z přístupů je takový, že se primární brány považují za druh obvodových prvků, které mají jedinou bránu (v případě primárního vstupu výstupní a v případě primárního výstupu vstupní), z/do které ústí datová cesta vedoucí ve skutečném obvodě z/na příslušnou primární bránu. Tento přístup napomáhá ke větší „konzistentnosti“ modelu a snadnější manipulaci s ním. Při modelování obvodu se primárním branám přisuzuje chování obvodového prvku. Primární vstup se pak chová jako prvek, který na svém výstupu dává data nezávisle na stavu obvodu (jakoby je v sobě generuje, ve skutečnosti jsou ale přivedena zvnějšku), primární výstup je naopak prvkem, který obvod nijak neovlivňuje, data přivedená na něj jakoby pohltí (ve skutečnosti jsou ale odvedena dále mimo obvod).

Druhý přístup, který je naopak bližší skutečné realizaci, je takový, že primární brány se považují za výlučné entity v rámci obvodu, entity, které jsou zdrojem resp.

cílem zpracovávaných dat. Tento přístup jsem použil ve své práci. Vyjadřuje dobře hierarchii celého systému, kterého je testovaný obvod ***UUA*** součástí. Vždyť i na celý testovaný obvod ***UUA*** lze pohlížet jako na prvek nějakého většího celku, na prvek, o kterém je známo jeho chování vyjádřené v tomto případě chováním struktury jeho prvků z množiny ***E*** propojených prostřednictvím svých bran ***P*** spoji z množiny ***C***, tedy na prvek se strukturou ***(E, P, C)*** a branami ***(PI, PO)***. Stejně tak každý prvek ***E*** obvodu ***UUA*** může být pro další analýzu dekomponován na svou strukturu a na své (při pohledu na strukturu prvku primární) brány. Podobně hierarchicky je stavěn též model v [Mar02].

2.3.2 Prvky

Množinu obvodových prvků ***E*** lze rozdělit do těchto podmnožin: ***R*** je množina registrů, ***FU*** je množina funkčních jednotek a ***MX*** je množina multiplexorů**.** Toto dělení vychází opět z tradičního pohledu na dělení obvodových prvků používaných ve struktuře číslicového obvodu na úrovni RT. Podobný pohled byl použit např. v práci [GuB95]. Oddělují se prvky se sekvenčním charakterem (registry) od prvků s charakterem kombinačním (funkční jednotky a multiplexory). Multiplexory mají mezi kombinačními prvky význačnou roli spočívající v tom, že jednotlivé prvky propojují a zajišťují přenos dat mezi nimi (viz podkapitola 3.2). Podmnožiny ***R, FU*** a ***MX*** tvoří rozklad množiny ***E*** podle relace ekvivalence „být prvkem stejného druhu“, jsou tedy navzájem disjunktní a jejich sjednocení dává právě množinu ***E***.

Rozdělení množiny obvodových prvků do těchto tří tříd má význam pro analýzu testovatelnosti a odráží se v něm principy strukturovaného návrhu. Zatímco v množině registrů ***R*** jsou sdruženy prvky mající charakter paměťových prvků, které pak celému obvodu dávají sekvenční charakter (čistě kombinační obvod pak bude mít množinu ***R*** prázdnou), množiny ***FU*** a ***MX*** pokrývají zbytek obvodu, tj. kombinační sítě.

2.3.3 Brány

P je množina bran prvků, kterou lze rozdělit na tři podmnožiny: $\boldsymbol{P = (IN \cup OUT \cup CI)}$, kde: ***IN*** je množina vstupních datových bran, ***OUT*** je množina výstupních datových bran a ***CI*** je množina řídicích a synchronizačních vstupů.

Pro jednoduchost se předpokládá, že všechny brány z množin ***IN*** a ***OUT*** jsou vícebitové, stejné šířky (***m*** bitové), zatímco brány z množiny ***CI*** jsou typicky široké pouze 1 bit.

Množina ***P*** je množinou všech bran všech prvků v obvodě. Pro popis struktury obvodu je vlastně brána daleko významnějším pojmem než prvek. Tyto dva pojmy jsou spolu sice úzce svázány, ale je-li vzata do důsledku struktura obvodu, je to právě propojení bran spoji, které vytváří strukturu obvodu. Hovořit o propojení prvků je sice také možné, ale pro účely analýzy struktury jde o příliš hrubý pohled. Vždyť každý prvek má několik bran, často ne jen jednu ale více vstupních a výstupních.

Nechť existuje funkce: $\psi : E \rightarrow 2^P$, která přiřazuje množinu bran obvodovému prvku:

(1) $\psi(e) = \{p |\ p$ je brána prvku $e\}$.

(2) Funkce ψ je definována pro všechny prvky z množiny E.

(3) Musí platit, že $e_1 \neq e_2 \Leftrightarrow \psi(e_1) \cap \psi(e_2) = \emptyset$.

Funkce ψ tvoří vazbu mezi množinou prvků a množinou bran. Tato funkce určuje, která brána z množiny bran přísluší kterému prvku. Z fyzikálního významu funkce ψ plyne nutnost podmínky (2) a podmínky (3). Podmínka (2) zajišťuje, že v obvodě se nevyskytne prvek, u kterého by nebylo možné říci, zda má nějakou bránu a jakou, podmínka (3) říká, že každý prvek z množiny P je funkcí ψ vázán k jedinému (obvodovému) prvku $e \in E$. Množina P je funkcí ψ rozdělena na $|E|$ disjunktních tříd.

2.3.4 Spoje

C je množina spojů: $C \subset (PI \cup P)$ x $(PO \cup P)$.

Jde vlastně o binární relaci v množině bran P a též mezi primárními vstupy a branami prvků a branami prvků a primárními výstupy, obsahuje uspořádané dvojice bran, mezi kterými existuje v obvodě spoj. Relace C je (což plyne z jejího fyzikálního významu):

- reflexivní,
- symetrická,
- transitivní.

Je tedy vidět, že relace C je relací ekvivalence. Relace C indukuje rozklad na ekvivalenční třídy.

Prvek množiny C reprezentuje formální model galvanického obvodového spoje několika bodů v obvodě. Při analýze struktury modelovaného obvodu jde přitom zpravidla o vyšetření, zda existuje spoj mezi dvěma body v obvodě. Proto byly zvolena forma binární relace, která může jednoznačně dát na takovou otázku odpověď. Fyzicky ovšem může nastat situace (a také v běžných obvodech často nastává), kdy jsou mezi sebou navzájem propojeny více než dva body v obvodě. Pak je logické, že všechny tyto body jsou spolu (vždy po dvojicích) navzájem v relaci. Lze též říci, že v jediném okamžiku mají všechny tyto body (a též spoj) stejnou hodnotu. Množina bodů obvodu, které jsou navzájem galvanicky propojeny spojem (jsou v relaci C) se bude dále nazývat *uzel*. Množina $V \subset (P \cup PI \cup PO)$ se nazývá uzel, jestliže $\forall p_1, p_2 \in V : (p_1, p_2) \in C$.

Množina $D = \{0, 1, \uparrow, \downarrow\} \cup \{0,1\}^m$ je množina hodnot, které se na bráně mohou vyskytovat. Symboly $\uparrow$ a $\downarrow$ mají význam nástupné resp. sestupné hrany. Číslo m udává šířku datových cest obvodu, $\{0,1\}^m$ je tedy množina všech m-bitových binárních slov.

2.4 Koncepce *i cest*

Aplikace testu na obvod spočívá v přivedení testovacího vektoru na vstupy testované komponenty, získání odezvy komponenty na takový stimul a její vyhodnocení. Právě řešení problému, jak přivést testovací vektor až do bodu, kam patří (který je často někde hluboko ve struktuře obvodu, běžně zvnějšku nedostupný) a vyvedení odezvy ven resp. do bodu, kam lze připojit analyzátor odezvy, vede často na dodatečné zásahy do struktury obvodu či její komplikace. Jedním z cílů této práce je nalézt metody, které hledají ve stávající struktuře obvodu takové části, které je možné využít i v režimu testu obvodu k přenosu diagnostické informace. Takto je možné se vyhnout alespoň některým dodatečným úpravám obvodu, které mají smysl jen při testu a celý obvod prodražují, aniž by tyto úpravy vedly ke zkvalitnění funkce. V tomto směru se jeví jako velmi perspektivní využití stávajících datových cest v obvodě i pro přenos diagnostické informace. Obvod je přístupný ze svých primárních vstupů a výstupů. Skoro vždy vede nějaká datová cesta z nějakého vstupu na nějaký výstup. Zpravidla však tato cesta mívá větší hloubku a proto není otázka přístupnosti vnitřních komponent obvodu zcela jednoznačná.

Přístup, jako je tento, je sice náročnější na analýzu testovatelnosti než přístupy v současnosti převážně používané, nevýhoda složitější analýzy je však zpravidla silně převážena výslednými úsporami při výrobě obvodu. Vždyť každá obvodová struktura, kterou je nutno do výsledného obvodu přidat „jen“ pro potřeby diagnostiky, vede (násobeno počtem vyrobených kusů) k citelnému nárůstu nákladů. Právě z tohoto důvodu se otázka využití původních obvodových struktur vzniklých návrhem obvodu i při testování dostává znovu do popředí, zejména s ohledem na dnešní trend systémů na čipu, jak o tom svědčí i v současné době probíhající výzkum u renomovaných výrobců obvodů.

2.4.1 *I cesta*

Pokud je možné za určitých podmínek z bodu p_1 do bodu p_2, kde $p_1 \in (IN \cup OUT \cup PI) \wedge p_2 \in (IN \cup OUT \cup PO)$, obvodu ***UUA*** přenést data beze změny, tedy pokud za určitých podmínek platí $\forall d \in \{0, 1\}^m: v(p_1) = d \wedge v(p_2) = d$, říkáme, že mezi body p_1 a p_2 existuje *i cesta*. Množina $I = \{(p_1, p_2) | p_1 \in (IN \cup OUT \cup PI) \wedge p_2 \in (IN \cup OUT \cup PO) \wedge \forall d \in \{0, 1\}^m: v(p_1) = d \wedge v(p_2) = d\}$ je množinou všech možných *i cest* v obvodě ***UUA***.

Pokud se tyto dva body nachází na jediném spoji (jsou součástí jednoho uzlu), není většinou s transparentností problém. Problém je ovšem s *i cestami*, jejichž počáteční a koncový bod neleží ve stejném uzlu. Pak se na *i cestě* jistě objeví i nějaké obvodové prvky. U těch transparentnost není z principu zajištěna, neboť se předpokládá, že data určitým způsobem transformují, provádějí s nimi potřebné operace. U některých prvků však lze nalézt režim činnosti, kdy je prvek pro data transparentní, data přivedená na jeho vstup se objeví na jeho výstupu v nezměněné podobě. Tento režim činnosti bude dále (podle [AbB85] a též podle [KZR99]) nazýván *i režim* (i jako identita).

2.4.2 *I režim*

Pokud existuje režim činnosti prvku $e \in E$ takový, že v tomto režimu lze přenést data beze změny z některého jeho vstupu $in \in IN$ na některý jeho výstup $out \in OUT$, přičemž $in \in \psi(e) \wedge out \in \psi(e)$, tedy za určitých podmínek platí $\forall d \in \{0, 1\}^m: \nu(in) = d \wedge \nu(out) = d$, říkáme, že prvek e má *i režim* činnosti, uvnitř prvku existuje *i cesta* z jeho vstupu in na jeho výstup out. Množina $M = \{(in, out) | in \in IN \wedge out \in OUT \wedge in \in \psi(e) \wedge out \in \psi(e) \wedge \forall d \in \{0, 1\}^m: \nu(in) = d \wedge \nu(out) = d\}$ je množinou všech vnitřních *i cest* v obvodě UUA. Zřejmě $M \subseteq I$.

Zobrazení $\mu: M \rightarrow 2^{P \times D}$ je takové zobrazení, které každé *i cestě* uvnitř prvku (v *i režimu* prvku) přiřadí podmínky, za kterých tato *i cesta* (*i režim*) existuje. Pro obvodový prvek $e \in E$ s datovým vstupem $i \in \psi(e) \wedge i \in IN$ a datovým výstupem $o \in \psi(e) \wedge o \in OUT$ (obě brány jsou široké m bitů) a je-li též $(i, o) \in M$, pak je zobrazení μ definováno: $\mu(i, o) = \{(cp, x) | cp \in \psi(e) \wedge x \in D \wedge cp \neq i \wedge cp \neq o\}$.

Problém využití transparentnosti prvků je v nastavení *i režimu*. To není vždy jen otázkou nastavení nějakého řídicího vstupu na příslušnou úroveň. Jistě by se pro zjednodušení dal za *i režim* prvku pokládat pouze stav, kdy je prvek vlivem nějakého řídicího signálu „průchozí“ právě po dobu trvání tohoto signálu. Takových prvků ale je vůči celkovému počtu prvků relativně málo a pokud už se v obvodě vyskytují, nebývají příliš zřetězeny. Z toho důvodu by bylo možno nalézt pouze nedostatečné množství jen krátkých a tudíž obtížně využitelných *i cest*.

Přitom lze pojem *i režimu* snadno zobecnit a využít tak k přenosu diagnostické informace některých dalších vlastností obvodových prvků. Například je možné beze změny přenést data přes registr – registry z principu ani data měnit nemají. Jenže u registru běžně není režim, kdy by se data vložená na jeho vstup přímo objevovala na jeho výstupu. Na výstupu registru jsou data ze vstupu z jediného okamžiku příchodu hodinového impulsu. To silně připomíná transparentní chování prvku až na to, že okamžik „transparentnosti“ je velmi krátký. Protože však je tento okamžik dobře ovlivnitelný, lze podobné chování též považovat za *i režim*.

Rovněž u prvků provádějících aritmetické a logické operace je možné nalézt režim činnosti, který lze považovat za transparentní. Například sčítačka je schopna přenést data ze svého vstupu na svůj výstup v případně, že přičítá hodnotu *nula* z druhého vstupu. Podobně je tomu i s násobičkou s tím rozdílem, že transparentnost nastává v případě násobení číslem *jedna*.

Z hlediska nastavení *i režimu* lze potenciálně transparentní prvky rozdělit takto:

- Prvky s *i režimem* závislým na hladinových řídicích vstupech,
- prvky s *i režimem* závislým na hranových řídicích vstupech a
- prvky s *i režimem* závislým na datových vstupech.

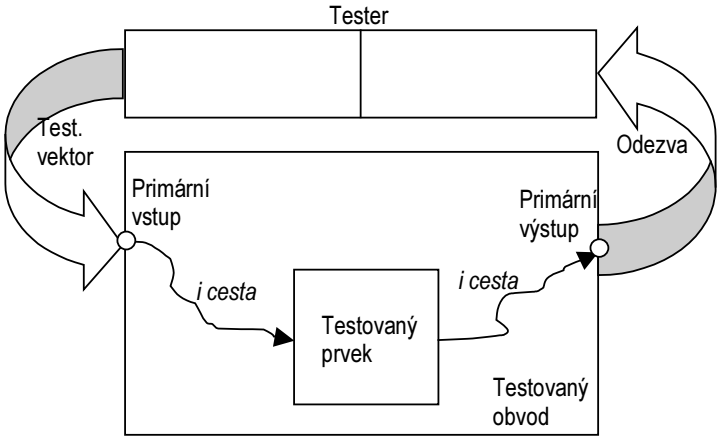
2.4.3 Nalezení *i cest*

Nechť $M \subset (PI \cup PO \cup P)$ x $(PI \cup PO \cup P)$ a nechť $C \subset (PI \cup PO \cup P)$ x $(PI \cup PO \cup P)$, pak množina všech dvojic bran v obvodě, mezi kterými existuje *i cesta*, je $I=\{(p_1, p_2) | (p_1, p_2) \in J \wedge J=(C \cup M)^* \wedge p_1 \in (IN \cup OUT \cup PI) \wedge p_2 \in$

(IN∪OUT∪PO)}. Relace, která je tranzitivním uzávěrem sjednocení relací ***C*** a ***M*** má význam ten, že dva body obvodu (brány prvků či primární brány obvodu) jsou spolu v této relaci právě tehdy, lze-li mezi těmito dvěma body přenést data beze změny. Kromě faktu, zda *i cesta* existuje, který lze vyjádřit příslušností do množiny (relací) ***I***, je pro další analýzu vhodné znát i průběh *i cesty*, tj. posloupnost bran, přes které *i cesta* vede. Toto je formálně vyjádřeno relací ρ. Pro naplnění množiny ***I*** a relace ρ byl použit upravený Dijkstrův algoritmus (v práci uvedený algoritmus 4.1).

2.5 Řiditelnost a pozorovatelnost

Řiditelnost vstupů a pozorovatelnost výstupů jsou základními podmínkami pro aplikaci testu na prvek. Otestovat prvek znamená nastavit testovací vektory na jeho vstupy a sejmout odezvu na testovací vektor z jeho výstupů. Schopnost nastavit testovací vektor na vstupy testovaného prvku (v této práci je myšleno z vnějšku obvodu) se nazývá *řiditelnost*, schopnost sejmout odezvu z výstupů testovaného prvku se nazývá *pozorovatelnost* [Cro99].



Obr. 2.1: Test prvku přes i cesty nalezené v obvodě.

Analýza *i cest* má souvislost s analýzou řiditelnosti a pozorovatelnosti. Každý prvek by měl být testovatelný buď testovacím vektorem přivedeným z testeru po *i cestě* přes primární vstup či z některého obvodového registru (tento registr pak bude třeba upravit pro zajištění jeho řiditelnosti), analogické situace vzniknou pro odezvy.

2.6 Rozdělení registrů do kategorií

V kapitole 2.1 byly prezentovány strukturované metody návrhu obvodu pro snadnou diagnostiku. Principem bylo vydělení sekvenčních prvků ze struktury obvodu a jejich zpřístupnění vně obvodu. V obvodě na úrovni RT jsou sekvenční prvky reprezentovány registry. Výstupy registru přístupného (řiditelného) zvnějšku obvodu lze považovat za významný vstupní bod do struktury obvodu, kromě primárních vstupů to bývá další možný začátek mnoha *i cest* využitelných při přenosu testovacího vektoru k testovanému prvku, vstupy registru přístupného (pozorovatelného) zvnějšku obvodu lze pak považovat za významný výstupní bod ze struktury obvodu, kromě primárních výstupů to bývá další možný konec mnoha *i*

cest využitelných při přenosu odezvy na testovací vektor od testovaného prvku. Pokud tedy testovaný prvek není řiditelný přímo po nějaké *i cestě* závislé čistě hladinově či datově, ale je svými vstupy připojen na některý z obvodových registrů, je vhodné této situace využít a aplikovat testovací vektor na vstup prvku právě z tohoto registru (rozumí se po jeho úpravě některou metodou strukturovaného návrhu).

Následující slovní definice registrů *tdr* a *trv* vychází z definice v [Kot97] a [Kot99], avšak po zkušenostech s reálnými obvody jsem je upravil do této podoby.

2.6.1 Vysílač testovacích vektorů

Registr, z něhož je možno aplikovat testovací vektor v jediném kroku hodin na vstup testovaného prvku, bude dále nazýván *vysílač testovacích vektorů*, zkráceně *tdr* (z angl. Test Driver). Při testu prvku se pak do registrů *tdr* příslušných k portům prvku vloží testovací vektory, jsou-li registry *tdr* všech portů naplněny požadovanými testovacími vektory a jsou-li otevřeny *i cesty* od registrů *tdr* na vstupy testovaného prvku, je na jeho výstupech odezva na tyto testovací vektory.

Nechť $\boldsymbol{TDR_{ip} \subset R}$ je množina vysílačů testovacích vektorů datového vstupu $\boldsymbol{ip \in IN}$ některého obvodového prvku. Formálně lze tuto množinu definovat takto: $\boldsymbol{TDR_{ip} = \{ tdr \mid tdr \in R \wedge \exists q : q \in \psi(tdr) \wedge (q, ip) \in I \wedge (\forall p_n : p_n \in \rho(q, ip) \wedge ((p_n, p_{n+1}) \in M \rightarrow \mu(p_n, p_{n+1}) = ((cp_1, x_1), \ \ldots\ , (cp_m, x_m)) \wedge \forall k \in \{1, \ldots, m\} : x_k \notin \{\uparrow, \downarrow\}))\}}$. Množinu je třeba stanovit pro všechny datové vstupy testovaných prvků $\boldsymbol{ip \in IN}$.

2.6.2 Přijímač odezev na testovací vektory

Z výstupů prvku je třeba odezvy na testovací vektory dopravit do bodu, ve kterém lze provést jejich analýzu, předpokládá se, že to je vně obvodu. Registr, do něhož je možno z výstupu testovaného prvku zapsat odezvu na testovací vektor jediným krokem hodin, bude dále nazýván *přijímačem odezev na testovací vektory*, zkráceně *trv* (z angl. Test receiver). Přijímač odezev na testovací vektory je prvním registrem na *i cestě* odezvy na testovací vektor z výstupu testovaného prvku ven z obvodu.

Nechť $\boldsymbol{TRV \subset R}$ je množina přijímačů odezev na testovací vektory datového výstupu $\boldsymbol{op \in OUT}$ některého obvodového prvku. Formálně lze tuto množinu definovat takto: $\boldsymbol{TRV_{op} = \{ trv \mid trv \in R \wedge \exists d : d \in \psi(trv) \wedge ipt((op,d)) \wedge (\forall p_n : p_n \in \rho(op, d) \wedge ((p_n, p_{n+1}) \in M \rightarrow \mu(p_n, p_{n+1}) = ((cp_1, x_1), \ \ldots\ , (cp_m, x_m)) \wedge \forall k \in \{1, \ldots, m\} : x_k \notin \{\uparrow, \downarrow\}))\}}$. Množina se stanoví pro všechny datové výstupy $\boldsymbol{op \in OUT}$.

2.6.3 Vstupní registr testu

Je snahou minimalizovat modifikace obvodu pro snadnou testovatelnost. Proto se zavádí další třídy registrů z hlediska jejich uplatnění při aplikaci testu. *Vstupním registrem testu* (zkráceně *tir* z anglického Test Input Register [Kot97]) bude nazýván registr, který je první na paralelní *i cestě* k registru *tdr* nějakého vstupu testovaného prvku, ať už tato *i cesta* začíná na primárním vstupu či uvnitř obvodu. Zatímco *tdr* se váže ke konkrétnímu vstupu prvku, *tir* se váže právě k *tdr*. Důvod

pro zavedení kategorie registrů *tir* je, že pro více registrů *tdr* může *tir* tvořit jeden obvodový registr. To je významné v situaci, kdy *i cesta*, na níž je *tir* prvním registrem, začíná někde uvnitř obvodu a prvky na ní nejsou přirozeně (paralelní *i cestou*) řiditelné. Situaci vystihuje obrázek 2.2.

Vstupní registry testu (*tir*) jsou místa v obvodě, kde začínají cesty testovacích vektorů na vstupy testovaných prvků. Do těchto míst jsou testovací vektory přivedeny zvenku. Někdy jsou přivedeny po speciální *i cestě* (řetězec *„scan“*), ale je-li to možné, přivádí se i z přirozených primárních vstupů obvodu. Z tohoto místa (*tir*) jsou pak testovací vektory dále přenášeny obvodem v režii řadiče testu.

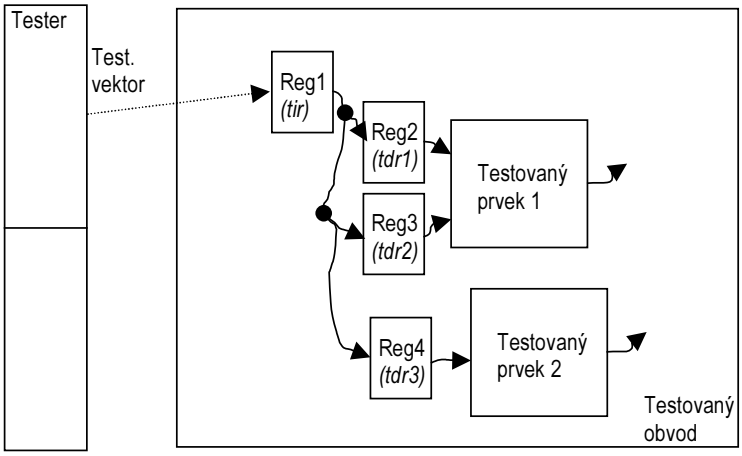
Sdílení jediného *tir* více registry *tdr* má kromě výhod (snížení počtu modifikací, …) také nevýhody, jako je např. možné prodloužením doby testu – nelze využít paralelního nastavování registrů *tdr*, které mají společný *tir*.

$\boldsymbol{TIR \subset R}$ je množina vstupních registrů testu. Formálně lze množinu vstupních registrů testu pro nějaký *tdr* stanovit takto: $\boldsymbol{TIR_{tdr} = \{ tir \mid tir \in R \wedge (\exists q_1, q_2 : q_1 \in \psi(tdr) \wedge q_2 \in \psi(tir) \wedge ipt((q_2, q_1))) \wedge (\forall (a, d_2) : ipt((a, d_2)) \wedge d_2 \in \psi(tir) \wedge (\forall p_n : p_n \in \rho((a, d_2)) \wedge (p_n, p_{n+1}) \in M \rightarrow \mu((p_n, p_{n+1})) = ((cp_1, x_1) \ldots , (cp_m, x_m)) \wedge \forall k \in \{1, \ldots, m\} : x_k \notin \{\uparrow, \downarrow\}))\}}$. Množinu registrů *tir* je třeba stanovit pro všechny $\boldsymbol{tdr \in \bigcup_{ip \in IN} (TDR_{ip})}$, tedy pro všechny registry, které byly vybrány jako *tdr* některého vstupu.

2.6.4 Výstupní registr testu

Analogickým způsobem lze optimalizovat počet modifikací na cestách sloužících při testu obvodu k dopravě odezev na testovací vektory ven z obvodu. *Výstupním registrem testu* (zkráceně *tor* z anglického Test Output Register [Kot97]) bude nazýván registr, který je posledním registrem na (paralelní) *i cestě* od registru *trv* nějakého výstupu testovaného prvku.

$\boldsymbol{TOR \subset R}$ je množina výstupních registrů testu. Formálně lze množinu výstupních registrů testu pro nějaký *trv* stanovit takto: $\boldsymbol{TOR_{trv} = \{ tor \mid tor \in R \wedge (\exists d_1, d_2 : d_1 \in \psi(tor) \wedge d_2 \in \psi(trv) \wedge ipt((d_2, d_1))) \wedge (\forall (q_2, a) : ipt((q_2, a)) \wedge q_2 \in \psi(tor) \wedge (\forall p_n : p_n \in \rho((q_2, a)) \wedge (p_n, p_{n+1}) \in M \rightarrow \mu((p_n, p_{n+1})) = ((cp_1, x_1) \ldots , (cp_m, x_m)) \wedge \forall k \in \{1, \ldots, m\} : x_k \notin \{\uparrow, \downarrow\}))\}}$. Množinu registrů *tor* je třeba stanovit pro všechny $\boldsymbol{trv \in \bigcup_{op \in OP} (TRV_{op})}$, tedy pro všechny registry, vybrané jako *trv* některého výstupu.



Obr. 2.2: Vztah registrů tir a tdr.

3 HLAVNÍ VÝSLEDKY PRÁCE

3.1 Výběr registrů pro zařazení do řetězce „scan“

Model obvodu, popsaný v předchozích kapitolách a jeho vlastností z hlediska analýzy testovatelnosti jsem využil v algoritmu pro analýzu testovatelnosti obvodu na úrovni RT. Základní myšlenkou je pro aplikaci testu v co největší míře využít možností, které nabízí obvod tak, jak je navržen. Při tom je uplatňována koncepce kategorií registrů popsaná v kapitole 3.6. Snahou je tedy nalézt co nejvíce *i cest*, po kterých by bylo možné řídit registry *tir* z primárních vstupů a co nejvíce *i cest*, které by zaručily pozorovatelnost registrů *tor* na primárních výstupech.

Je zřejmé, že není možné vždy dosáhnout situace, kdy bude v obvodě nalezena *i cesta* z vnějšku obvodu ke každému registru *tir* a od každého registru *tor* ven z obvodu. Až pro tyto případy se začne uvažovat o modifikaci příslušného registru tak, aby bylo možno jej z vnějšku řídit či pozorovat. Celá koncepce hledání registrů typu *tdr/trv* a *tir/tor* přitom navíc umožňuje řešit případné sdílení vstupního či výstupního registru testu (*tir/tor*) pro test více prvků. Je však třeba nalézt alespoň jeden registr *tdr* pro každou vstupní bránu každého testovaného prvku a podobně alespoň jeden registr *trv* pro každou výstupní bránu.

Ideální stav nastává, pokud řiditelnost, resp. pozorovatelnost registrů *tir*, resp. *tor* může být zajištěna přes (paralelní) *i cesty*, které v obvodě vznikly již díky návrhu. Registry *tir*, resp. *tor* je třeba rozdělit do dvou tříd: na registry, které jsou v původní struktuře již řiditelné, resp. pozorovatelné, tedy existuje *i cesta* z některého primárního vstupu na vstup registru *tir*, resp. existuje *i cesta* z registru *tor* na některý primární výstup a na registry, které řiditelné, resp. pozorovatelné nejsou, tedy registry, pro které taková *i cesta* ve stávajícím obvodě není. Pro tyto účely je zavedena třída ***CTR*** řiditelných registrů *tir* a třída ***NCTR*** registrů *tir*, které řiditelné nejsou. Analogicky se pro výstupní registry testu zavede třída ***OBS*** pozorovatelných registrů *tor* a třída ***NOBS*** registrů *tor*, které pozorovatelné nejsou.

Registr, který bude pro každý *tdr* vybrán, bude pak definitivně hrát roli jeho *vstupního registru testu*. Přednost budou mít samozřejmě ty registry, které jsou řiditelné z nějakého primárního vstupu obvodu, tedy registry, které jsou v množině ***CTR***.

Množina $\boldsymbol{CTDR = \{tdr \mid tdr \in ATDR \wedge \exists\, tir : (tir \in TIR_{tdr} \wedge tir \in CTR)\}}$ je množina takových registrů, které hrají pro některý vstup některého testovaného prvku roli *vysílače testovacích vektorů* (*tdr*) a mají přitom některý ze svých kandidátů na vstupní registr testu řiditelný z některého primárního vstupu obvodu.

V množině ***CTDR*** jsou tedy zatím ty registry *tdr*, které mají alespoň jeden *tir* řiditelný z primárních vstupů. Pokud by nyní platilo , že v ***CTDR*** jsou všechny registry *tdr*, byla by analýza řiditelnosti u konce. V reálných obvodech však k takovéto šťastné situaci vždy nedojde. Dále se ovšem pracuje s omezenějším systémem tříd registrů *tir*:

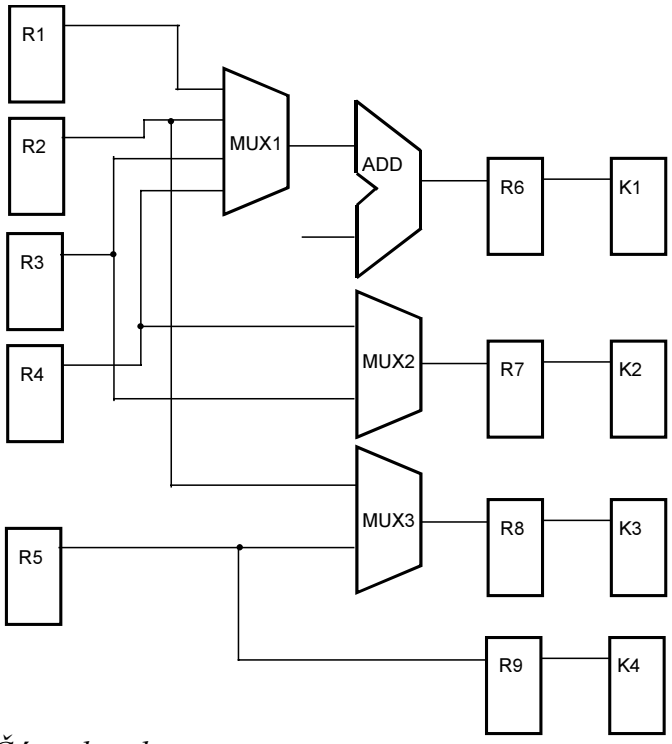
Systém množin $\mathfrak{I}_{NC} = \mathfrak{I} / \{TIR_{tdr1}, TIR_{tdr2}, ..., TIR_{tdrm}\}$, kde $\{tdr1, tdr2, ..., tdrm\} = CTDR$, je systém množin registrů *tir* pro takové registry *tdr*, které nemají zajištěnu řiditelnost žádným řiditelným registrem *tir*.

Pokud je množina $\mathfrak{I}_{NC}$ neprázdná, bude třeba některé registry upravit tak, aby byly řiditelné. Bude třeba zajistit řiditelnost alespoň jednoho registru z každé třídy $TIR_{tdr} \in \mathfrak{I}_{NC}$. V nejhorším případě by bylo třeba zajistit řiditelnost tolika registrů, kolik tříd systém množin $\mathfrak{I}_{NC}$ obsahuje. Protože však třídy z $\mathfrak{I}_{NC}$ nemusí být disjunktní, lze vybrat registry k úpravě na řiditelné právě z průniků těchto tříd a tak snížit počet registrů k úpravě.

Nechť množina ***SCAN*** obsahuje registry, které se upraví pro zajištění řiditelnosti či pozorovatelnosti. Pokud některá třída $TIR_{tdri} \in \mathfrak{I}_{NC}$ obsahuje pouze jediný registr, tedy $TIR_{tdri} = \{tir_j\}$, pak tento registr nevyhnutelně musí být upraven pro zajištění řiditelnosti – $tir_j \in SCAN$.

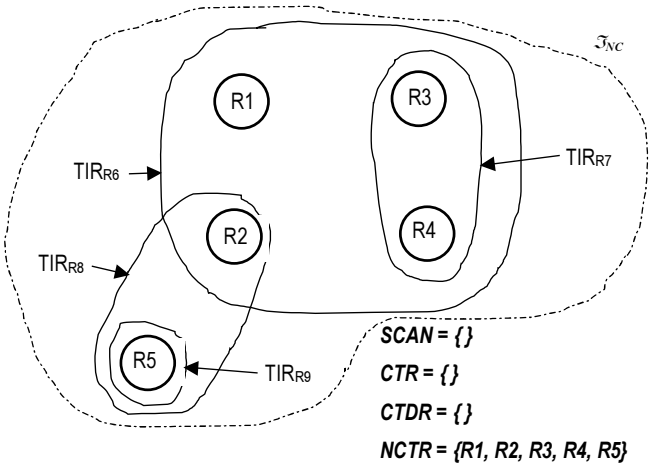
Je-li registr ***tir***, který hraje roli *vstupního registru testu* (*tir*), zařazen do množiny ***SCAN***, pak se registr ***tir*** tímto krokem dostává do množiny ***CTR*** (tedy $tir \in CTR$) a není už dále prvkem množiny ***NCTR*** (tedy $tir \notin NCTR$). Je-li registr tir_i prvkem množin TIR_{tdrj}, kde $j = \{1,... ,n\}$ a zároveň v množině ***SCAN***, pak všechny registry tdr_j pro $j = \{1,... ,n\}$ jsou v množině ***CTDR***.

Jedním z cílů celé analýzy je dosáhnout toho, aby množina ***CTDR*** obsahovala všechny *tdr*, tj. aby $CTDR = ATDR$. Jsou-li v obvodě registry *tdr*, jejichž registry *tir* nejsou zatím řiditelné, budou se tyto registry *tir* vhodným způsobem upravovat tak, aby byla zajištěna jejich řiditelnost a tím vlastně přidávat do množiny ***SCAN***.

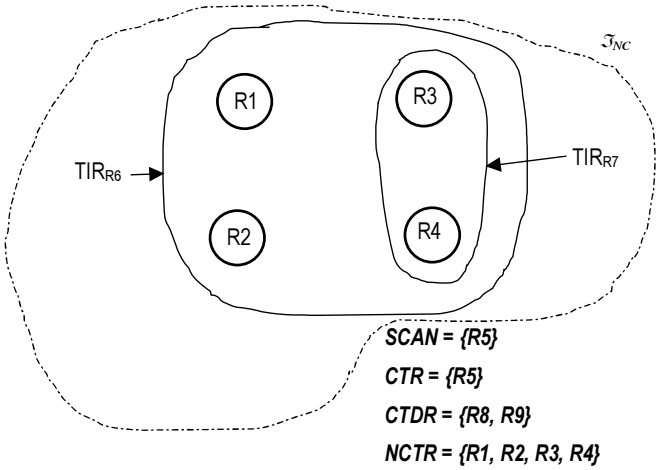


Obr. 3.1: Část obvodu.

Každým přidáním registru do množiny ***SCAN*** se zvýší počet registrů, které vyhovují definici množiny ***CTR***. S každým registrem, který bude přidán do množiny ***CTR*** přibude i jeden či více registrů, které budou v množině ***CTDR***. Přitom z každým přibyvším registrem, který vyhoví definici množiny ***CTDR*** se zmenší počet tříd v množině $\mathfrak{I}_{NC}$. Zařazením jediného registru *tir* do množiny ***SCAN*** může přibýt i více registrů *tdr* do množiny ***CTDR*** a tím se i o stejný počet tříd zmenší množina $\mathfrak{I}_{NC}$.



Obr. 3.2:Příklad tříd v množině $\mathfrak{I}_{NC}$.



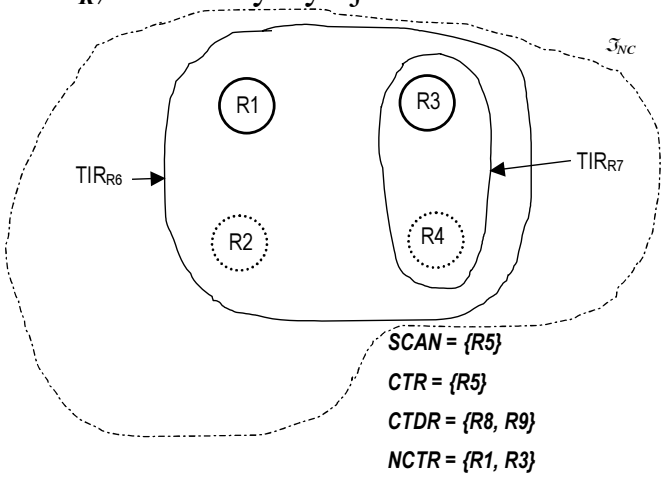
*Obr. 3.3: Třídy množiny $\mathfrak{I}_{NC}$ po úpravě zařazením **R5** do množiny **SCAN**.*

Na obrázku 3.2 jsou graficky znázorněny třídy ze systému množin $\mathfrak{I}_{NC}$ pro část obvodu na obrázku 4.1. Nevyhnutelným registrem je registr ***R5***, protože ten je jediným prvkem třídy $\boldsymbol{TIR_{R9}}$. Tento registr (***R5***) bude tedy zařazen do množiny ***SCAN***. Registr ***R5*** je nyní i prvkem množiny ***CTR*** a není již prvkem množiny ***NCTR***. Všechny registry *tdr*, které jsou řiditelné z registru ***R5*** musí být v množině ***CTDR***. Jak je vidět z obrázku 3.2, jde o registr ***R9*** (právě pro tento registr byl registr ***R5*** označen za „nevyhnutelný“), ale i o registr ***R8***, neboť registr ***R5*** patří jak do třídy $\boldsymbol{TIR_{R9}}$, tak i do $\boldsymbol{TIR_{R8}}$. Tím, že jsou nyní registry ***R8*** a ***R9*** v množině ***CTDR*** ale přestávají být třídy $\boldsymbol{TIR_{R8}}$ a $\boldsymbol{TIR_{R9}}$ v množině $\mathfrak{I}_{NC}$. Výslednou situaci zachycuje obrázek 3.3.

Jestliže $\exists\, tir_1, tir_2 : tir_1 \in NCTR \wedge tir_2 \in NCTR \wedge (\exists\, n : (n \geq 1 \wedge \forall tdr_k : (k \in \{1, ..., n\} \wedge tdr_k \in ATDR \wedge tdr_k \notin CTDR \wedge tir_1 \in TIR_{tdrk} \wedge tir_2 \in TIR_{tdrk})) \wedge \forall tdr_k : (k > n \wedge tdr_k \in ATDR \wedge tdr_k \notin CTDR \wedge tir_1 \notin TIR_{tdrk} \wedge tir_2 \notin TIR_{tdrk}))$, pak oba registry tir_1, tir_2 jsou z hlediska možnosti řídit z nich nějaký registr *tdr* ekvivalentní a je třeba posoudit podle dalších kritérií, který z nich bude vybrán jako registr *tir* pro danou množinu registrů *tdr*. Ten registr z dvojice tir_1, tir_2, který nebude vybrán jako možný registr *tir* pro danou množinu registrů *tdr*, protože se podle dodatečných kritérií jeví jako horší, bude vypuštěn z množiny ***NCTR*** a tak nebude dále předmětem analýzy.

Na obrázku 3.3 jsou graficky znázorněny třídy ze systému množin $\mathfrak{I}_{NC}$ pro část obvodu na obrázku 3.1, které zbyly po úpravě zařazením registru ***R5*** do množiny ***SCAN***.

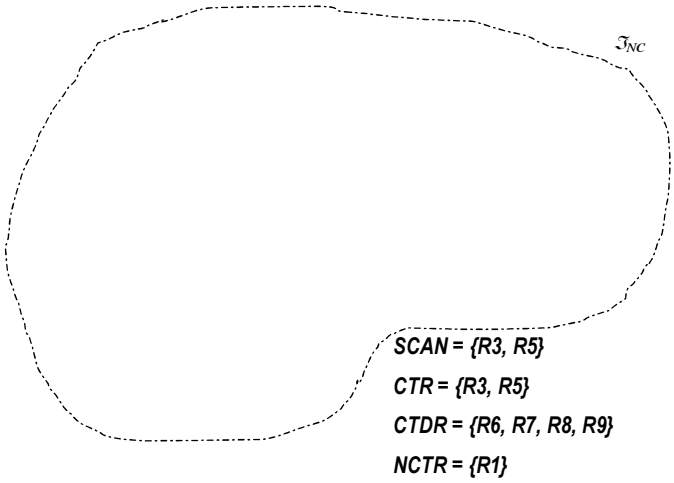
Registry ***R3*** a ***R4*** řídí oba stejnou množinu registrů *tdr*. Oba dva tyto registry se vyskytují ve stejných množinách TIR_{tdr} a to sice v množinách TIR_{R6} a TIR_{R7}, přičemž platí, že žádný z nich se nevyskytuje v množině, ve které by se nevyskytoval i ten druhý. Registry ***R3*** a ***R4*** jsou tedy z hlediska možnosti řídit z nich nějaký registr *tdr* ekvivalentní. Pro posouzení, který z nich je vhodnější, je tedy třeba použít další kritéria. Nechť je vhodnější pro modifikaci registr ***R3***. Potom registr ***R4*** bude vypuštěn z množiny ***NCTR***. Stejně tak se stane pro dvojici ***R1*** a ***R2***, kdy se oba registry vyskytují pouze spolu, ve stejné množině TIR_{R6}. Z nich nechť je podle dodatečného kritéria vhodnější např. ***R1***. Tak vznikne situace znázorněná na obrázku 3.4. Na tomto obrázku jsou registry ***R2*** a ***R4*** nakresleny čárkovaně proto, že už jsou z hlediska další analýzy nepodstatné a nebudou brány v úvahu, i když v množinách TIR_{R6} a TIR_{R7} se stále vyskytují.



*Obr. 3.4: Třídy množiny $\mathfrak{I}_{NC}$ po eliminaci role registrů **R2** a **R4**.*

Jestliže $\exists\, tir_1, tir_2 : tir_1 \in NCTR \wedge tir_2 \in NCTR \wedge (\exists\, m, n : (m,n \geq 1 \wedge m > n \wedge \forall tdr_k : (k \in \{1, ..., n\} \wedge tdr_k \in ATDR \wedge tdr_k \notin CTDR \wedge tir_1 \in TIR_{tdrk} \wedge tir_2 \in TIR_{tdrk}) \wedge \forall tdr_l : (l \in \{n+1, ..., m\} \wedge tdr_l \in ATDR \wedge tdr_l \notin CTDR \wedge tir_1 \in TIR_{tdrl} \wedge tir_2 \notin TIR_{tdrl}) \wedge \forall tdr_r : (r > m \wedge tdr_r \in ATDR \wedge tdr_r \notin CTDR \wedge tir_1 \notin TIR_{tdrr} \wedge tir_2 \notin TIR_{tdrr}))$, pak registr tir_1 je z hlediska možnosti řídit nějaký registr *tdr* vhodnější

pro úpravu pro zajištění řiditelnosti nežli registr tir_2. Registr tir_1 se přidá do množiny ***SCAN***.



*Obr. 3.5: Třídy množiny $\mathfrak{I}_{NC}$ po úpravě zařazením **R3** do množiny **SCAN**.*

Na obrázku 3.4 jsou graficky znázorněny třídy ze systému množin $\mathfrak{I}_{NC}$ pro část obvodu na obrázku 3.1. Registry ***R1*** a ***R3*** splňují tvrzení předchozího odstavce, neboť registr ***R3*** může sloužit jako *tir* pro registry ***R6*** i ***R7***, protože je zároveň v množinách TIR_{R6} i TIR_{R7}, kdežto registr ***R1*** může sloužit pouze jako *tir* registru ***R6*** (je pouze v množině TIR_{R6}). Registr ***R3*** bude přidán do množiny ***SCAN***. Registr ***R3*** je nyní rázem i prvkem množiny ***CTR*** a není již prvkem množiny ***NCTR***. Ovšem všechny registry *tdr*, které jsou řiditelné z registru ***R3*** musí být v množině ***CTDR***. Jak je vidět z obrázku 3.4, jde o registr ***R7***, ale i o registr ***R6***, neboť registr ***R3*** patří jak do třídy TIR_{R7}, tak i do TIR_{R6}. Tím, že jsou nyní registry ***R6*** a ***R7*** v množině ***CTDR*** ale přestávají být třídy TIR_{R6} a TIR_{R7} v množině $\mathfrak{I}_{NC}$. Výslednou situaci zachycuje obrázek 3.5.

Podobným způsobem se postupuje i v případě registrů *trv* a *tor*. Všechny výše popsané postupy jsou zahrnuty v algoritmu, který je v disertační práci popsán jako algoritmus 6.1. Tento algoritmus vybere registry, které je třeba modifikovat pro zajištění jejich řiditelnosti či pozorovatelnosti tak, aby byl počet modifikací co nejmenší při zachování řiditelnosti či pozorovatelnosti všech uzlů obvodu, které hrají roli při aplikaci testu.

3.2 Dosažitelnost, model aplikace testu

V kapitole 4.1 byla popsána metodika, která na základě analýzy *i cest* v obvodě ***UUA*** vytvoří množinu registrů ***SCAN***, ve které budou po jeho skončení registry, jež je třeba modifikovat tak, aby bylo možno je zařadit do řetězce *„scan“*. Tato Metodika bere v úvahu pouze samotnou existenci *i cesty*, ne však už podmínky její existence. V případě *i cesty* složené z prvků s *hladinově* či *hranově závislými i režimy* to postačí. Nastavení *i cesty* je věcí řadiče testu a analyzovaná datová část na toto nastavení nemá vliv. Jinak je tomu však v případě *i cesty*, na které se vyskytuje prvek s *datově závislým i režimem*. Nastavení takové *i cesty* často vyžaduje využít

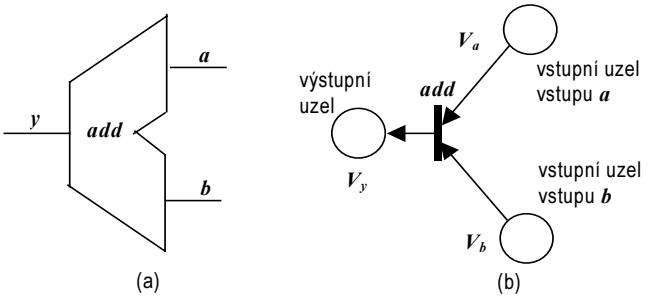
některé další (pomocné) *i cesty*. V některých případech pak tato *i cesta*, potřebná pro nastavení *i cesty* hlavní, využité pro přenos diagnostických dat, koliduje s *i cestou* pomocnou. V této kapitole je popsán způsob odhalení takových kolizí. Problém je převeden na problém analýzy dosažitelnosti v Petriho síti [Rei85]. Tento přístup byl publikován v [Ruz02].

Nechť $N_{fu} = (B, \Sigma, F)$ je C/E Petriho síť, která reprezentuje způsob aplikace testu na obvodový prvek ***fu***. Množiny z trojice N_{fu} mají následující význam: ***B*** je množina podmínek (míst) C/E sítě, v tomto případě jde o reprezentaci některých uzlů ***UUA***, Σ je množina událostí (přechodů) C/E sítě, v tomto případě odpovídá prvkům, které se podílí na testu prvku ***fu*** a ***F*** je toková relace C/E systému.

Petriho sítí se reprezentuje ta část obvodu, která se bezprostředně účastní aplikace testu, tedy testovaný prvek se všemi svými vstupy a výstupy, všechny prvky a spoje na *i cestách* vedoucích ze vstupních bodů testu (registrů *tir* či primárních bran) na vstupy testovaného prvku a všechny prvky a spoje na *i cestách* vedoucích z výstupů testovaného prvku do výstupních bodů testu (registrů *tor* či primárních bran). Na první pohled to vypadá, že až na přechod reprezentující testovaný prvek budou mít všechna místa a všechny přechody jednoprvkové presety [Ces94] a postsety [Ces94]. Není tomu ale tak. To, co narušuje linearitu modelovaných *i cest*, jsou prvky s datově závislými *i režimy*. U takových prvků se pak musí modelovat i způsob nastavení *i režimu*, tj. prakticky další *i cesta* z nějakého vstupního bodu testu k prvku s datově závislým *i režimem*.

Obvodový prvek se modeluje pomocí přechodu (události) Petriho sítě. Místa (podmínky) pak reprezentují uzly obvodu. Jak bylo řečeno v kapitole 3.3, uzlem se rozumí množina bran, které jsou navzájem svázány relací galvanického spojení ***C***. Dalo by se říci, že uzel je totéž co logický spoj, totéž co signál v jazycích pro popis obvodů. Je to soubor míst, na kterých je v jednom okamžiku vždy stejná hodnota. Tato hodnota bude v Petriho síti reprezentována značkou. Pochopitelně uzel v obvodě má nějakou hodnotu v jakémkoli okamžiku, značka v Petriho síti tedy nereprezentuje jakoukoli hodnotu, nýbrž konkrétní hodnotu důležitou z hlediska aplikace testu: testovací vektor, odezvu na něj, hodnotu pro nastavení *i režimu*. Protože každý uzel může mít v daném okamžiku jen jedinou hodnotu, je zřejmé, že půjde o C/E Petriho síť.

Obrázek 3.6 ukazuje, jak vypadá model například sčítačky. Jak bylo již řečeno, vlastní prvek je reprezentován přechodem, jeho preset obsahuje místa, která reprezentují uzly, do nichž patří jeho vstupy, jeho postset obsahuje místo, které reprezentuje uzel, do něhož patří jeho výstup. Je třeba si uvědomit, že každý uzel je vlastně sdílen s prvky, které jsou připojeny na vstupy či výstupy např. této sčítačky. Proto například vstupní uzel vstupu ***a*** sčítačky ***add*** z obrázku 3.6 může být totožný s výstupním uzlem nějakého prvku ***e***, z jehož výstupu je vstup ***a*** sčítačky ***add*** přímo (po *i cestě délky 0*) řiditelný. Lze také říci, že dva prvky e_1 a e_2 jsou spolu přímo spojeny, pokud ${}^\bullet e_1 \cap e_2{}^\bullet \neq \{\}$ nebo $e_1{}^\bullet \cap {}^\bullet e_2 \neq \{\}$.



Obr. 3.6: Sčítačka (a) a odpovídající část C/E Petriho sítě (b).

Značky v C/E Petriho síti reprezentují (jak již bylo řečeno) význačné stavy (hodnoty) uzlů, tedy význačná data z hlediska aplikace testu. Umožní to vyšetřit, zda daný způsob aplikace testu je bezkolizní či nikoliv, navíc může pomoci při paralelizaci procesu aplikace testu na obvod.

Jak již bylo popsáno výše, prvek je reprezentován přechodem. Výjimkou v tomto ohledu je však registr a multiplexor. Model registru vyžaduje kromě přechodu též místo, které by modelovalo paměťové schopnosti registru. Výjimečnost modelu multiplexoru zase spočívá v tom, že průchod přes multiplexor se nastavuje řadičem testu a ten není součástí modelu. Je proto třeba nějak transparentnost multiplexoru v libovolném okamžiku modelovat.

V disertační práci je prezentováno několik algoritmů, které pomáhají vytvořit model aplikace testu na základě formálního modelu obvodu a výsledků algoritmu, který provádí analýzu testovatelnosti a vybírá vhodné registry do řetězce *„scan“*.

Algoritmus ***Reg(N, V_d, V_q, r)*** přidá do C/E sítě ***N*** podsíť reprezentující registr ***r*** s jeho vstupem připojeným na uzel V_d a jeho výstupem připojeným na uzel V_q.

Algoritmus ***SReg(N, V_d, V_q, r)*** přidá do C/E sítě ***N*** podsíť reprezentující registr ***r***, který je upraven pro zařazení do řetězce *„scan“*, s jeho vstupem připojeným na uzel V_d a jeho výstupem připojeným na uzel V_q, jeho sériový vstup je připojen na přechod, který je právě registrován proměnnou ***lastscan*** a jeho sériový výstup je připojen na přechod ***scout***, který je vytvořen a vložen též do proměnné ***lastscan*** jako aktuální konec řetězce *„scan“*.

Algoritmus ***Mux(N_{fu}, V_a, V_b, V_y, mux)*** přidá do C/E sítě ***N*** podsíť reprezentující dvouvstupový multiplexor ***mux*** s jeho vstupy připojenými na uzly V_a a V_b jeho výstupem připojeným na uzel V_q.

Rekurzivní algoritmus ***Net(N, a, b)*** rozšiřuje C/E síť ***N*** o podsíť, která reprezentuje *i cestu* z nějakého výstupu ***a*** nějakého prvku na nějaký vstup ***b*** jiného prvku.

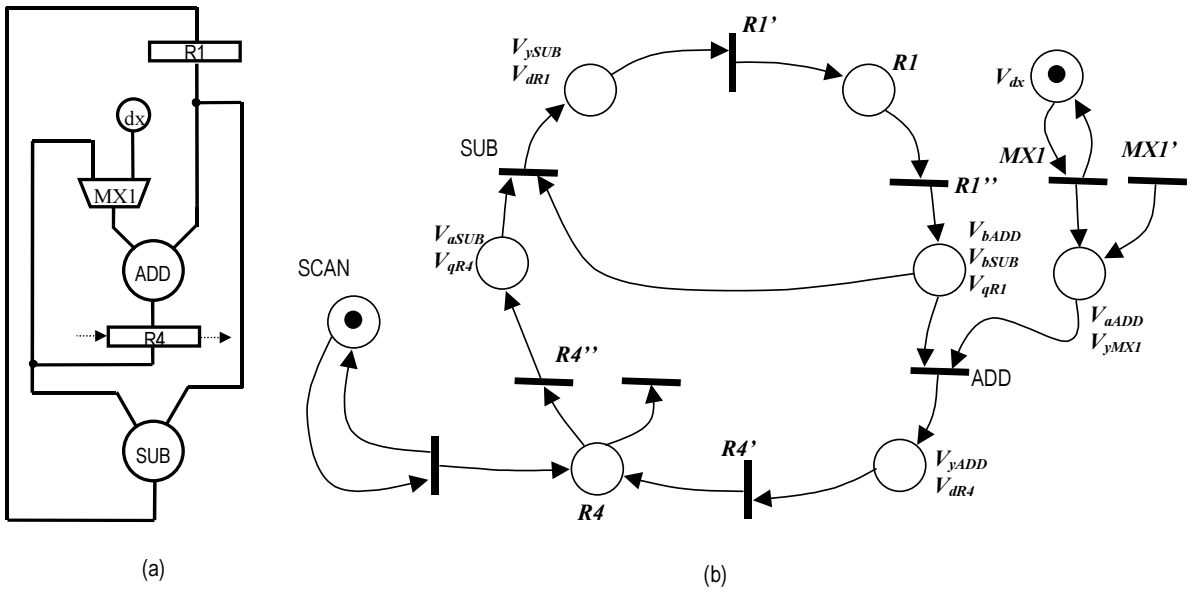
V disertační práci je dále (jako algoritmus 7.5) popsán algoritmus, který vytvoří model aplikace testu na základě formálního modelu obvodu a výsledků algoritmu, který provádí analýzu testovatelnosti a vybírá vhodné registry do řetězce *„scan“*.

Počátečním značením každé sítě N_{fu} pro všechny prvky $fu \in E$ budiž značení $M_0 = \{sdi, V_{pi1}, ..., V_{pin}\}$, kde V_{pi1} ... V_{pin} jsou všechny uzly odpovídající primárním vstupům, které se v síti vyskytují.

Je-li v množině dosažitelných značení Petriho sítě N_{fu}, která modeluje způsob aplikace testu na prvek ***fu***, značení $\mathcal{M}_i$ dosažitelné z počátečního značení $\mathcal{M}_0$, přičemž platí, že $\forall V_p$: $p \in \psi(fu) \wedge p \in IN \wedge V_p \in \mathcal{M}_i$, tedy že je dosažitelné takové značení, kdy všechny vstupní uzly prvku ***fu*** zároveň obsahují značku, pak je možné v obvodě ***UUA*** tak, jak je, aplikovat testovací vektory na prvek ***fu*** bez dalších úprav.

Je-li v množině dosažitelných značení Petriho sítě N_{fu}, která modeluje způsob aplikace testu na prvek ***fu*** ze značení $\mathcal{M}_k$ dosažitelné značení $\mathcal{M}_0$, přičemž platí, že $\mathcal{M}_k = \{V_p | \; p \in \psi(fu) \wedge p \in OUT \wedge p \in V_p\}$, tedy že je ze značení, kdy všechny výstupní uzly prvku ***fu*** zároveň obsahují značku, dosažitelné počáteční značení, pak je možné v obvodě ***UUA*** tak, jak je, snímat odezvy na testovací vektory prvku ***fu*** bez dalších úprav.

Na obrázku 3.7 (a) je vidět část číslicového obvodu Diffeq. Pro prvek ***ADD*** z tohoto obrázku byla vytvořena Petriho síť N_{ADD}, jejíž grafické znázornění je na obrázku 3.7 (b). Už z obrázku 3.7 (a) je jasně zřejmé, že vstup ***a*** prvku ***ADD*** je řiditelný z primárního vstupu ***dx***. Není tedy třeba hledat žádný registr, který by pro tento vstup hrál roli vysílače testovacích vektorů. Pro řízení vstupu ***b*** prvku ***ADD*** bude použit registr ***R1***. Jako přijímač odezev na testovací vektory byl vybrán registr ***R4***. Registr ***R4*** bude zároveň výstupním registrem testu, protože je součástí řetězce *„scan“*. Nyní zbývá určit, který registr bude vstupním registrem testu pro registr ***R1***. Platí, že $(q_{R4}, d_{R1}) \in I$, tedy že existuje *i cesta* z výstupu ***q*** registru ***R4*** na vstup ***d*** registru ***R1***, protože jistě platí, že $(a_{SUB}, y_{SUB}) \in M$. Registr ***R4*** je v řetězci *„scan“* (a je tedy snadno řiditelný), byl vybrán předchozí analýzou za vstupní registr testu registru ***R1***. Petriho síť na obrázku 3.7 (b) tuto situaci modeluje, v tabulce 3.1 jsou pak vypsána všechna dosažitelná značení sítě z počátečního značení $\mathcal{M}_0 = \{SCAN, V_{dx}\}$. Množina dosažitelných značení byla vypočtena systémem PESIM [CeSk93].



Obr. 3.7: Část obvodu (a) a Petriho síť pro test prvku ADD (b).

	1	2	3	4	5	6	7	8	9
	V_{ySUB} V_{dR1}	***R1***	V_{bADD} V_{bSUB} V_{qR1}	V_{aADD} V_{qMX1}	V_{yADD} V_{dR4}	V_{aSUB} V_{qR4}	V_{dx}	***SCAN***	***R4***
M_0							1	1	
M_1				1			1	1	
M_2				1			1	1	1
M_3				1		1	1	1	
M_4				1		1	1	1	1
M_5							1	1	1
M_6						1	1	1	
M_7						1	1	1	1

Tab. 3.1:Dosažitelná značení sítě z obrázku 4.7(b).

Je zřejmé, že žádné ze značení dosažitelných z počátečního značení $M_0 = \{SCAN, V_{dx}\}$ neodpovídá podmínce $\forall V_p: p \in \psi(fu) \wedge p \in IN \wedge V_p \in M_i$. Je to vidět na první pohled. V této situaci není brána ***b*** prvku ***ADD*** vůbec řiditelná – ve sloupci, který odpovídá místu simulujícímu uzel s branou ***b*** (sloupec 3) není ani v jednom případě jednička – do tohoto místa se diagnostická data za stávající konfigurace nemohou dostat. Proč tomu tak je? Vždyť předchozí analýza určila za vstupní registr testu registru ***R1*** (který je vysílačem testovacích vektorů vstupu ***b***) registr ***R4***. Ano, sice platí, že $(q_{R4}, d_{R1}) \in I$, ale $\rho(q_{R4}, d_{R1}) = (q_{R4}, a_{SUB}, y_{SUB}, d_{R1})$ a $\mu(a_{SUB}, y_{SUB}) = (b_{SUB}, „00000000“)$. A právě zde je kámen úrazu, protože nyní je třeba řídit i vstup ***b*** odečítačky ***SUB***. A jeho vysílač testovacích vektorů je ***R1***. Ovšem ten je řiditelný právě z ***R4***! Analýza Petriho sítě N_{ADD} odhalila konflikt.

3.3 Experimenty

Jako příklad pro demonstraci výsledků aplikace navržených metodik a postupů na číslicové obvody byly vybrány „benchmarkové“ obvody [PKG86], které se pro podobné účely často využívají. Tato volba umožnila též srovnat dosažené výsledky s výsledky publikovanými při prezentaci metodik a systémů s obdobným zaměřením. V disertační práci jsou představeny výsledky analýzy pro obvod „Diffeq“ a obvod „Tseng“. Kromě výsledků v podobě vybraných registrů s důležitou rolí při aplikaci testu na obvody je prezentováno rovněž několik zajímavých modelů aplikace testu (Petriho sítí) pro prvky, u kterých dojde při stavu úprav obvodu po aplikaci algoritmu pro výběr registrů do řetězce *„scan“* ke kolizím v cestách, po kterých jsou transportována obvodem diagnostická data. K modelům jsou vždy uvedeny i výsledky analýzy systémem PESIM. Pokud je odhalena kolize, je navrženo i řešení (spočívající v dodatečné modifikaci některého registru).

Pro získání experimentálních výsledků byla využita implementace popsaného aparátu, která byla provedena v rámci diplomové práce [Pro02]. Tato práce rovněž využívá výsledků prací [Mrk00] a [Zbo00]. Protože téměř veškerý formální aparát je popsán jazykem predikátové logiky, byl pro implementaci zvolen jazyk Prolog.

4 ZÁVĚR

4.1 Zhodnocení

Přínosem této práce je čistě formálně popsaná metodika analýzy testovatelnosti, převádějící problémy analýzy testovatelnosti na problémy matematiky a teoretické informatiky, na problémy, které jsou známé a mají ověřená řešení. Při srovnání výsledků popsané metodiky se systémem IDAT [Buk00], tedy systémem, který též vybírá registry pro zařazení do řetězce *„scan“* ve struktuře datových cest obvodů na úrovni RT (je ale univerzálnější – dovoluje aplikovat i vestavěnou diagnostiku a vkládat testovací body), se ukazuje, že metodika, popsaná v této práci, dává pro stejné obvody poněkud lepší výsledky (zařadí do řetězce *„scan“* méně registrů, nežli systém IDAT). Tyto lepší výsledky lze zdůvodnit tím, že heuristika, užitá v systému IDAT, vždy nemusí nalézt řešení optimální, může zůstat v lokálním extrému, kdežto popsaná metodika heuristiku vůbec nevyužívá. Další příčina úspěchu metodiky může být spatřována v lepší práci s datovými cestami – koncepce *i cesty* často realizuje transport diagnostických dat i tam, kde jiné metodiky toto nepřipouští. To je ovšem vykoupeno složitější analýzou a o něco komplikovanějším řadičem testu.

Hlavními výsledky práce jsou:

- Formální model struktury datových cest číslicového obvodu na úrovni RT,
- formální model vlastností obvodu, vycházející z koncepce *i cest* a tříd registrů dle jejich role při aplikaci testu, algoritmus hledání *i cest* a jejich vlastností,
- algoritmus analýzy testovatelnosti vybírající registry k zařazení do *„scanu“*,
- formální model aplikace testu a jeho další analýza, jíž je možné určit, zda zvolený způsob aplikace testu je prakticky realizovatelný či nikoliv a algoritmy, které model generují z modelu obvodu a výsledků analýzy.

Ukázalo se, že Petriho sítě jsou vhodným aparátem k analýze dosažitelnosti uzlů ve struktuře číslicového obvodu na úrovni RT a jejich potenciál umožní navázáním a rozvinutím postupů popsaných v této práci v budoucnu řešit i další problémy spojené s testováním číslicových obvodů, např. problém rozvržení pořadí testu a řešení možností paralelního testování více obvodových prvků.

4.2 Možná rozšíření a další práce

Otevřeným problémem zůstává automatizovaný způsob řešení kolizních situací a uváznutí, pokud se vyskytnou při analýze modelu aplikace testu. Prozatím byly tyto situace řešeny „ručně“, zásahem operátora, který po zhodnocení kolizní situace navrhl nejvhodnější řešení (zpravidla přidání dalšího registru do množiny ***SCAN***).

Další rozšíření systému by mohlo spočívat v hledání postupů, jak na základě formálního modelu obvodu a navržených úprav a po provedení kontroly, resp. dalších úpravách na modelu aplikace testu generovat konečný automat modelující řadič testu, z nějž by pak bylo možno generovat rovnou popis řadiče testu. To by předpokládalo postupy pro rozvržení pořadí testu obvodových prvků, s čímž souvisí úvahy o vhodném pořadí testu prvků a možném paralelním testování více prvků.

LITERATURA

[AbB85] Abadir, M. S., Breuer, M. A.: A Knowledge Based System for Designing Testable VLSI Chips. Časopis IEEE Design & Test of Computers, srpen 1985, str. 56 – 68

[Buk00] Bukovjan, P.: Allocation for Testability in High-Level Synthesis [disertační práce]. Institut National Polytechnique de Grenoble, Francie, září 2000, 124 stran

[Ces94] Češka, M.: Petriho sítě. Akademické nakladatelství CERM, Brno, 1994, 94 stran

[CeSk93] Češka, M., Skácel, M.: Petri Net Tool PESIM. Sborník 5th International Workshop on Petri Nets and Performance Models. IEEE Computer Society Press, Praha, 1993, str. 1 – 10

[Cro99] Crouch, A. L.: Design for Test for Digital IC's and Embedded Core Systems. Prentice Hall, New Jersey, 1999, 350 stran

[GuB95] Gupta, R., Breuer, M. A.: Partial Scan Design of Register-Transfer Level Circuits. Časopis Journal of Electronic Testing: Theory and Applications (JETTA) svazek 7, č. 1 a 2, 1995, str. 25 – 46

[KoR00b] Kotásek, Z., Růžička, R.: Partial Scan Methodologies – a Survey. Sborník PDS2000, Elsevier Science Ltd. Oxford, Ostrava, ČR, 2000, str. 133 – 137

[KoR00c] Kotásek, Z., Růžička, R.: The Implementation of RTL Testability Analysis Algorithms trough the Discrete Mathematics Concepts. Sborník Fourth International Scientific Conference on Electronic Computers and Informatics, VIENALA Press, Košice–Herľany, 2000, str. 177 – 182

[KoR00d] Kotásek, Z., Růžička, R.: Testability Analysis Based on Discrete Mathematics Concepts. Sborník 9th International Colloquium on Numerical Analysis and Computer Science with Applications, TU of Plovdiv, Plovdiv, 2000, str. 113

[Kot97] Kotásek, Z.: RT Level Element Classification. Design and Diagnostics of Electronic Circuits and Systems (DDECS) 1997, Sczyrk, Polsko, 1997, str. 41 – 46

[Kot99] Kotásek, Z.: Uplatnění principů řiditelnosti/pozorovatelnosti při návrhu číslicových obvodů [habilitační práce]. FEI VUT v Brně, 1999, 80 stran

[KRH00] Kotásek, Z., Růžička, R., Hlavička, J.: Formal Approach to RTL Testability Analysis. Sborník IEEE LATW 2000, IEEE, Rio de Janeiro, 2000, str. 98 – 103

[KZR99] Kotásek, Z., Zbořil, F., Růžička, R.: Partial Scan Methodology in VHDL Environment. Sborník Third International Scientific Conference on Electronic Computers and Informatics (CEI'99), VIENALA Press, Košice, Herľany, SR, 1999, str 146 – 151

[Mae92] Maerz, S.: High-Level Synthesis. In: The Synthesis Approach to Digital System Design (Michel, P., Lauther, U., Duzy, P., eds.), Kluwer Academic Publishers, 1992, str. 115 – 220

[Mar01] Marinissen, E., J.: Philips' Approach to Core-Based System Chip Testing. Sborník 4th IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS) 2001, Gyor, Hungary, duben 2001, str. 15 – 24

[Mar02] Marinissen, E. J.: The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. Časopis Journal of Electronic Testing: Theory and Applications (JETTA) svazek 18, č. 4, přijato k publikaci, vyjde v srpnu 2002

[Mrk00] Mrkus, L.: Analýza popisu chování komponent číslicového obvodu na úrovni RT [Ročníkový projekt]. UIVT FEI VUT v Brně, 2000, 25 stran

[PKG86] Paulin, P. G., Knight, J.P., Girczyc, E.F.: HAL: A multi–paradigm approach to automatic data path synthesis, sborník 23rd ACM/IEEE Design Automation Conference (DAC), Association for Computing Machinery, Inc., 1986, str. 263 – 270

[Pro02] Procházka, P.: Analýza testovatelnosti datových cest číslicového obvodu na úrovni RT [diplomová práce]. FIT VUT v Brně, 2002, 61 stran

[Rei85] Reisig, W.: Petri Nets, An Introduction. Springer Verlag, Berlin Heidelberg, 1985, 160 stran

[Ruz02] Růžička, R.: The Formal Approach to the RTL Test Application Problem Using Petri Nets. Sborník 5th IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS) 2002, Brno, duben 2002, str. 78 – 86

[Zbo00] Zbořil, F.: VHDL RT Level Parser/Analyser of a Source Code. Sborník Fourth International Scientific Conference on Electronic Computers and Informatics, VIENALA Press, Košice–Herľany, 2000, str. 150 – 155

AUTOROVO CV

- Narozen 1975,
- „Ing.“ na UIVT FEI VUT v Brně, 1999,
- interní doktorand na UIVT FEI VUT v Brně / FIT VUT v Brně 1999–2002,
- 25.6.2001 Státní doktorská zkouška,
- „Ph.D.“ na FIT VUT v Brně, 2002.

Publikace

[1] Kotásek, Z., Zbořil, F., Růžička, R.: Partial Scan Methodology in VHDL Environment. Sborník CEI'99, Košice – Herľany, SR, 1999, str. 146 – 151.

[2] Růžička, R.: Využití metod Partial scan v diagnostice číslicových obvodů. *Diplomová práce*. UIVT FEI VUT, Brno, 1999, 81 stran.

[3] Růžička, R.: Využití metod Částečný Scan v diagnostice číslicových obvodů. Sborník prací studentů a doktorandů, Brno, 1999, str. 101 – 102.

[4] Kotásek, Z., Růžička, R., Sochůrek, M.: Behavioral Analysis for Testability on VHDL Source File. Sborník IEEE DDECS 2000, Smolenice, SR, 2000, str. 209 – 212.

[5] Sekanina, L., Růžička, R.: Design of the Special Fast Reconfigurable Chip Using Common FPGA. Sborník IEEE DDECS 2000, Smolenice, SR, 2000, str. 161 – 168.

[6] Kotásek, Z., Růžička, R.: Partial Scan Methodologies – a Survey. Sborník PDS2000, Ostrava, ČR, 2000, Elsevier Science Ltd. Oxford, str. 133 – 137.

[7] Kotásek, Z., Růžička, R.: The Implementation of RTL Testability Analysis Algorithms trough the Discrete Mathematics Concepts. Sborník 4th ECI 2000, Košice – Herľany, SR, 2000, str. 177 – 182.

[8] Růžička, R.: PCB for Teaching Purposes to Present the ISA Bus Adapter Design. Sborník 4th ECI 2000, Košice – Herľany, 2000, str. 213 – 218.

[9] Kotásek, Z., Růžička, R.: Testability Analysis Based on Discrete Mathematics Concepts. *Vyžádaný referát.* Sborník 9th International Colloquium on Numerical Analysis and Computer Science with Applications, Plovdiv, 2000, TU of Plovdiv, str. 113.

[10] Kotásek, Z., Růžička, R., Hlavička, J.: Formal Approach to RTL Testability Analysis. Sborník IEEE LATW 2000, Rio de Janeiro, 2000, str. 98 – 103.

[11] Růžička, R., Sekanina, L.: The Role of Simulation During Design of Evolvable Systems. Sborník 22nd International Colloquium Advanced Simulation of Systems 2000, Sv. Hostýn, 2000, str. 85 – 90.

[12] Růžička, R.: Data Dependent I Path and their Utilisation in DFT. Sborník prací studentů a doktorandů FEI VUT, Brno, 2000, str. 228 – 230.

[13] Kotásek, Z, Růžička, R. Strnadel, J.: Formal and Analytical Approach to the Testability Analysis – the Comparison. Sborník 4th IEEE DDECS 2001, Györ, Maďarsko, 2001, str. 123 – 130.

[14] Hlavička, J., Kotásek, Z., Růžička, R., Strnadel, J.: Interactive Tool for Behavioral Level Testability Analysis. Sborník IEEE ETW 2001, Stockholm, Švédsko, 2001, str. 117 – 119
[15] Kotásek, Z., Růžička, R., Strnadel, J., Zbořil, F.: Two Level Testability System. Sborník 35th Spring International Conference MOSIS'01, Ostrava, CZ, str. 433 – 440.
[16] Růžička, R.: The Formal Approach to the RTL Test Application Problem Using Petri Nets. Sborník 5th IEEE DDECS 2002, Brno, CZ, str. 78 – 86.
[17] Růžička, R.: VHDL Circuit Description Transparency Analysis. Sborník ECI 2002, Košice-Herľany, SR, 2002, str. 194 – 199.

Citace

Bukovjan, P.: Allocation for Testability in High-Level Synthesis. Institut National Polytechnique de Grenoble, Francie, září 2000.
Torresen, J.: Reconfigurable Logic Applied for Designing Adaptive Hardware Systems. SSGRR 2002W, January 2002, L`Aquila, Itálie
Torresen, J., Vinger, K. A.: High Performance Computing by Context Switching Reconfigurable Logic. 16th ESM2002, Darmstadt, Německo.
Baláž, M., Pikula, T., Trebatický, P., Gramatová, E.: Memory Self-Testing Using a Non-Linear Cellular Automaton in the Circuit for Data Encryption. IEEE DDECS 2002, Brno.

Ostatní aktivity

- Účast na řešení grantového úkolu GAČR 102/98/1463
- Návrh a výroba desky pro podporu návrhu desek a programového vybavení adaptérů sběrnice ISA. Deska byla vystavena na veletrhu INVEX 2000 v Brně a prezentována na konferenci ECI 2000 v Herľanech, Slovensko.
- Účast na řešení grantového úkolu FRVŠ 1565 T.O. H Experimentální laboratoř pro zaměření ECBS.
- Organizace mezinárodní konference IEEE Design and Diagnostics of Electronic Circuits and Systems 2002, konané ve dnech 17. – 19. dubna 2002,
- editorství sborníku IEEE DDECS 2002.
- Účast na řešení grantového úkolu GAČR 102/01/1531 Formální přístupy k diagnostice číslicových obvodů – verifikace testovatelného návrhu.
- Řešení grantového úkolu GAČR 102/03/P176 Formální přístup k plánování testu číslicových obvodů.

Ocenění

Cena Siemens 2001 – Stipendijní podpora pro doktorandy
Cena rektora VUT v Brně 2002

ABSTRACT

The goal of the thesis was to develop a methodology for a digital circuit testability analysis to be performed in the early stage of the design process with the following outputs: testability factors, test application protocol and the recommendation for structure modifications to gain better testability. Such modifications are limited by the price of the final design. The goals of the methodology are reached through the analysis of data paths existing in the circuit as the result of the design process to utilize them for diagnostic data (test vectors and responses to them) transfers, the *i path* concept is used for these purposes.

The methodology operates on data paths of the digital circuit at the register-transfer level of abstraction (RT). The circuit under analysis is described as the structure consisting of mutually interconnected blocks such as functional units, multiplexers and registers. The RT level is fairly high abstraction level of a digital circuit model but the information on the structure of the circuit in such granularity is available on this design level, which is sufficient for the testability analysis based on the *i path* concept. The advantage of such approach is that RT level structure carries the information on the circuit function which can be easily identified in the structure. The interconnections between elements have typically a width of more than one bit. These facts are very important for the analysis. One of the aspects of the methodology is that the information on circuit function is utilized, in this way higher number of *i paths* can be identified in the circuit.

To develop the methodology, formal tools were used. It allows to create a formal model of a circuit, to describe its diagnostic and testability properties and describe testability analysis algorithms, all of them formally. The concepts utilized in discrete mathematics and computer science are used. The entities in the circuit (elements, interconnections, etc.) are subdivided into sets, other features and dependencies are expressed by relations. As the description language, the language of predicate logic is used. The exactness of the description and ability to transform problems of the testability analysis to well-known and solved problems of discrete mathematics and theoretical computer science are the main advantages of the formal approach.

In the approach described in the thesis, registers play a special role. This is due to the fact that the concept of the structured design for testability is used in which combinational and sequential logic are strictly separated. Registers become important points of *i paths*, which are used to transfer diagnostic data. If no *i path* from outside the circuit to the proper node or in opposite direction are identified, a suitable register and also an *i path* from/to this register is searched. The register must be then modified to guarantee its controllability and/or observability, the modification must be done in compliance with the structured design rules. The modification lies in the register inclusion into the scan chain. As the priority is to find an *i path* from/to outside the circuit and only if it fails, modify a register to be included into the *scan* chain, it is possible to declare that the proposed methodology corresponds to the partial scan methodology.