Analýza VHDL

Luboš Lorenc

lorenc@fit.vutbr.cz

zpráva projektu
Virtuální laboratoř aplikace mikroprocesorové techniky
MŠMT 2C06008

http://www.vlam.cz
UNIS spol. s r.o, R&D; ÚTIA AV ČR; VUT Brno, FIT, ÚIFS

Analýza syntaxe a sémantiky VHDL

Jedním z úkolů, na kterých v tomto roce probíhaly práce v projektu VLAM, byl i výzkum v oblasti zpracování jazyka VHDL. VHDL je jazyk určený pro popis struktury a chování číslicových elektronických obvodů. Původně byl tento jazyk vytvořen pro sjednocení popisu chování různých číslicových obvodů. VHDL je zkratka pro VHSIC Hardware Description Language, kde VHSIC je zkratka pro Very High Speed Integrated Circuit. Původní určení tohoto jazyka nepředpokládalo, že by někdy v budoucnu měl být analyzován či jinak zpracováván automatickými nástroji. Postupně však byl takovýto druh popisu číslicových obvodů shledán dostatečným k vytvoření nástrojů pro simulaci. Tyto nástroje tedy začaly VHDL používat jako zdrojový kód pro svoji práci. Jazyk VHDL tak byl rozšířen o podporu simulace obvodů a následně i o podporu jejich syntézy tak, aby na základě popisu struktury a chování obvodů bylo možné jejich automatizované vytvoření v různých typech programovatelných obvodů. Postupným vývojem a doplňováním různých vlastností ovšem vzrostla i jeho složitost.

V současné době každá oblast využití jazyka VHDL využívá svoji vlastní podmnožinu jazykových konstrukcí s jistým společným jádrem a speciální metodologii zápisu zdrojového kódu. Tím se složitost jazyka VHDL dostala na takovou úroveň, že jeho analýza je velmi obtížná a jednotlivé nástroje proto bývají obvykle tvořeny jako vysoce specializované pouze pro konkrétní oblast zpracování. Obecně lze dostupné nástroje rozdělit na nástroje pro simulaci a nástroje pro syntézu obvodů. Užší zaměření nástroje umožňuje omezit analýzu vstupního VHDL kódu na jistou dobře specifikovanou podmnožinu jazyka VHDL a vytvořit tak efektivní analyzátor. V podstatě lze říci, že v současné době neexistuje žádný analyzátor, který by byl schopen analyzovat celý jazyk VHDL.

Při podrobnější analýze jazyka VHDL bylo zjištěno, že pomocí dostupných prostředků je tento jazyk velmi obtížně analyzovatelný. V průběhu vývoje VHDL byly vytvořeny různé konstrukce, které jsou podmíněny vlastnostmi popisovaných komponent. Ve standardu se tak často vyskytují shodné syntaktické konstrukce, ale tyto jsou podmíněny sémantickým významem. Pokud tedy je úkolem vytvořit čistě syntaktický analyzátor VHDL, je výchozí gramatika vytvořená podle standardu v mnoha syntaktických kategoriích nejednoznačná.

Těžiště výzkumu v oblasti syntaktické analýzy bylo tudíž soustředěno na návrh vhodného modelu obecného syntaktického analyzátoru kompletního jazyka VHDL 2002 a experimentální ověření jeho realizovatelnosti. Pro implementaci byla vybrána metoda LR analýzy, přičemž vlastní analyzátor je vytvářen generátorem syntaktických analyzátorů Bison.

Popis řešení

Z pohledu možností analýzy je jazyk VHDL poměrně rozsáhlý a ne příliš dobře specifikovaný. Během dlouhého vývoje prošel řadou změn, během kterých byl různě modifikován a doplňován o nové vlastnosti. Při postupném doplňování byl zřejmě kladen důraz především na kompatibilitu se staršími verzemi a mnohem méně pak na možnost analýzy či dalšího zpracování tohoto jazyka. Jazyk obsahuje řadu různých konstrukcí s velmi podobnou či dokonce stejnou syntaxí, ale jiným sémantickým významem. Pomineme-li řadu chyb, které standard IEEE 1076 z roku 2002 obsahuje (viz seznam na www stránkách tohoto standardu), narazíme na řadu z pohledu analýzy poněkud nelogických syntaktických

konstrukcí. Řada syntaktických kategorií umožňuje derivovat syntakticky naprosto shodné podstromy, ale s jiným sémantickým významem. Tento sémantický význam autoři VHDL začlenili do syntaktického popisu jazyka. Vlastní gramatika VHDL je tak ve standardu definována pomocí jisté modifikace EBNF (rozšířená Backus-Naurova forma). Tato modifikace spočívá v obohacení sémantickými podmínkami, které jsou následně popisovány v okolním textu. Jazyk popsaný takovýmto způsobem je poté jen obtížně možné analyzovat, i když se omezíme jen na syntaktickou analýzu. V podstatě se nabízejí dvě možnosti. Buď sémantické informace z gramatiky zcela vypustit a nebo řešit alespoň částečnou sémantickou analýzu a s využitím získaných sémantických informací provádět syntaktickou analýzu. V projektu byly řešeny oba způsoby. Při vypuštění sémantických informací z gramatiky VHDL vznikne gramatika v čisté EBNF. Podle gramatiky v EBNF je možné za jistých podmínek vytvořit parser buď přímým zápisem podle pravidel gramatiky a nebo pomocí specializovaných nástrojů, jako je například generátor syntaktických analyzátorů Bison. Tento generátor byl použit i při práci na syntaktickém analyzátoru VHDL v tomto projektu. Bison zpracovává vstupní gramatiku ve formátu BNF, původní EBNF byla tudíž rozepsána na odpovídající tvar v BNF – podmíněné části a iterace EBNF byly přepsány pomocí nových pravidel na základní tvar BNF. Gramatika zapsaná v BNF je podmínkou vytvoření parseru, nikoliv však dostačující podmínkou. Nutnou podmínkou pro vytvoření deterministického LR parseru je, aby gramatika byla typu LALR. Určení, zda daná gramatika je, či není, LALR jen na základě znalosti pravidel je poměrně obtížné. Obecně lze říci, že bezkontextová gramatika je LALR gramatika, pokud podle ní vytvořený parser dokáže v každém okamžiku parsingu na základě znalosti právě jednoho příštího tokenu rozhodnout, podle kterého pravidla se v příštím kroku bude pokračovat. Z výše uvedeného je zřejmé, že gramatika VHDL tuto nutnou podmínku nesplňuje. Existuje-li v gramatice více větví, kterými může být derivován stejný derivační strom, nelze po vytvoření takovéhoto stromu při parsingu jednoznačně rozhodnout, kterou cestou pokračovat v budování syntaktického stromu. Bison dokáže tento problém v některých případech řešit použitím takzvaného zobecněného LR parsingu, během kterého postupně zkouší pokračovat ve vytváření syntaktického stromu všemi možnými způsoby a nakonec vybere ten, kterým je možné bezchybně vytvořit syntaktický strom. Jak je vidět z popisu, je tento způsob práce velmi neefektivní, protože vlastně dochází k simulaci nedeterministického chování a postupnému prohledávání potenciálně velmi velkého stavového prostoru. Problém nastává v okamžiku, kdy takovýchto cest existuje více a nelze vybrat jedinou správnou. Jak bylo zjištěno v průběhu výzkumu, má gramatika VHDL zbavená sémantických rozšíření právě tu vlastnost, že i k velmi jednoduchým VHDL programům lze při parsingu nalézt několik zcela správně vytvořených ale odlišných syntaktických stromů. Pro VHDL tedy selhává i jinak velmi mocný nástroj, jakým je GLR parser generovaný Bisonem.

Tvorba jednoznačné gramatiky

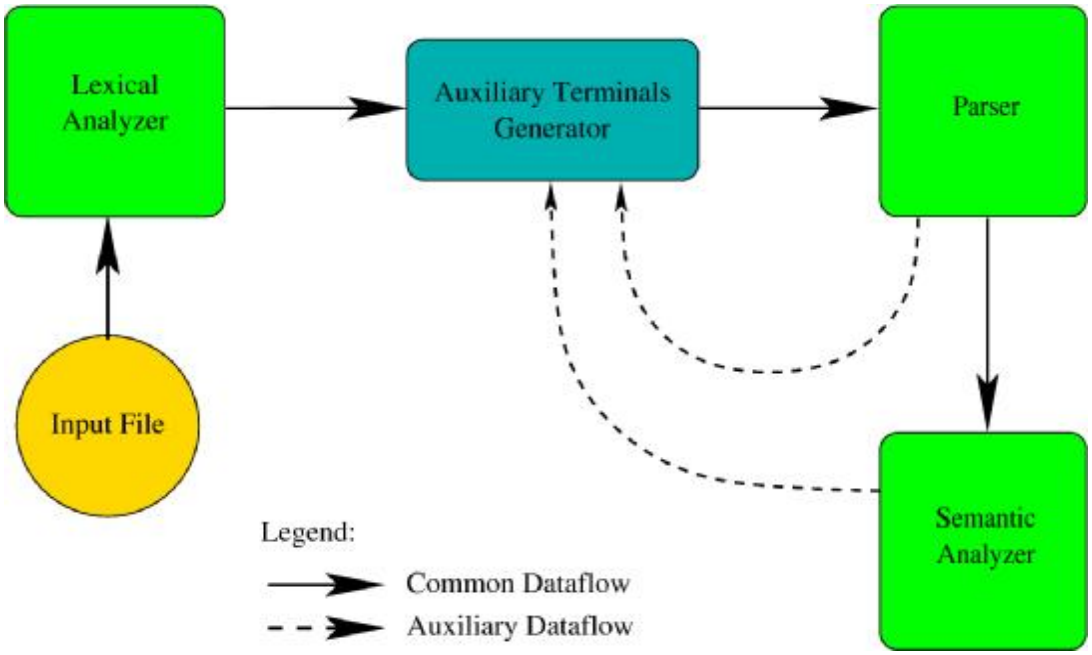
Po zjištění skutečného rozsahu nejednoznačnosti gramatiky VHDL byl učiněn pokus o její přepis na ekvivalentní ale jednoznačnou gramatiku. Takováto gramatika by umožňovala jednoznačné generování derivačního stromu pro všechny korektní VHDL programy. Díky tomu by bylo možné vytvořit deterministický parser VHDL.
Jednoznačná gramatika byla postupně vytvářena úpravami původní gramatiky. Nová gramatika se však velmi brzy dostala do stavu, kdy již bylo jasné, že případný syntaktický strom vytvořený analyzátorem pracujícím podle této gramatiky nebude použitelný jako zdroj dat pro sémantickou analýzu. V gramatice totiž bylo nutné slučovat konfliktní větve a

vypouštět různé konstrukce, které sice ve standardu měly nějaký sémantický význam, ale ze syntaktického hlediska byly nepodstatné. Každá nejednoznačnost v gramatice totiž způsobuje jeden či více konfliktů pravidel ve vytvářeném automatu parseru. Konflikty mohou být dvojího druhu. První typ konfliktu vzniká v okamžiku, kdy je v daném konfliktním místě možné provést dvě různé redukce – tedy redukovat vytvořený podstrom na více nonterminálních symbolů vyšší úrovně. V tom případě hovoříme o R/R (reduce/reduce) konfliktu. Druhým typem konfliktu je možnost současného provedení vložení na zásobník a pokračování v konstrukci rozpracovaného podstromu a možnost redukování tohoto podstromu na nonterminální symbol stojící v gramatice na vyšší úrovni. Takovýto typ konfliktu nazýváme S/R (shift/reduce). V původní gramatice VHDL zbavené veškerých sémantických doplňků bylo 579 R/R konfliktů a 78 S/R konfliktů. V současné době je část konfliktů vyřešena a zbývá vyřešit ještě 76 nevyřešených R/R konfliktů. Tato gramatika je však už díky různým úpravám natolik vzdálena původní gramatice VHDL a díky mnohonásobným rozsáhlým modifikacím pravděpodobně poškozená zavlečenými chybami, že byly práce na jejím dokončení zastaveny.

Podmíněná tvorba syntaktického stromu

Tato metoda využívá druhé možné cesty, která byla uvedena v úvodu. Tedy využití sémantických informací při konstrukci syntaktického stromu. Metoda vychází ze sémantických informací, které jsou ve standardu uváděny jako součást gramatiky – sémantických rozšíření EBNF diskutovaných v úvodu. Po přepisu gramatiky VHDL do BNF a upravení pro zpracování Bisonem byla na základě analýzy gramatiky VHDL a informací ze zpracování gramatiky Bisonem vybrána vhodná místa, na kterých docházelo ke konfliktům. Na těchto místech bylo následně provedeno blokování konfliktů přidáním nových pomocných terminálů do gramatiky. Tyto terminály se vždy vztahují k sémantickému významu části jazyka, jejíž syntaktický podstrom je tímto terminálem při překladu blokován. Pomocí této jednoduché úpravy gramatiky VHDL se podařilo eliminovat většinu R/R konfliktů původní gramatiky a zbylé byly eliminovány drobnými změnami gramatiky, které by neměly ovlivnit možnost zpracování celého jazyka VHDL podle standardu.

Doplněním nových terminálů se však změnil původní jazyk generovaný touto gramatikou. Aby mohl být analyzátorem nadále přijímám původní jazyk, bylo nutné vymyslet a prakticky ověřit vhodný způsob, jak ve vhodný okamžik do zdrojového kódu VHDL doplnit příslušné pomocné terminály. Po zvážení všech možností byl vybrán způsob provádění velmi zjednodušené sémantické analýzy zpracovávaného kódu a ukládání získaných informací do tabulky symbolů. Vzhledem k tomu, že většina konfliktů nějakým způsobem souvisí se zpracováním jmen různých prvků jazyka, byla sémantická analýza zaměřena především na zjišťování příslušnosti zpracovávaných jmen k jednotlivým sémantickým jednotkám jazyka, tak jak jsou do gramatiky VHDL vloženy ve standardu. Po návrhu a experimentálním odzkoušení funkčnosti sémantické analýzy bylo nutné vyřešit problém rozpoznání správného místa vložení pomocného terminálu do zdrojového souboru. Správná místa vložení byla nakonec určena v podstatě experimentálně. Gramatika parseru byla upravena tak, aby parser v místech vkládání procházel přes pomocná pravidla, ve kterých se rozhoduje o vložení či nevložení pomocného terminálu. Pokud se tento terminál vloží, dojde k okamžitému zablokování všech parsovacích cest, které dříve způsobovaly konflikt, protože v gramatice neobsahují tento vložený terminál. Tím se výběr parseru zúží na jedinou cestu a parsing zcela deterministicky dále pokračuje po zvolené cestě. Jak bylo uvedeno výše, rozhodnutí, zda pomocný terminál vložit, či nikoliv se provádí na základě znalosti sémantického významu



právě zpracovávané konstrukce, ve většině případů podle toho, jaký sémantický prvek popisuje právě zpracovávané jméno. Mechanismus vkládání je znázorněn na následujícím obrázku. Plnými šipkami jsou znázorněné datové toky tak, jak se běžně vyskytují v kompilátorech. Čárkovaně je pak naznačeno, jak dochází k předávání informací potřebných pro vkládání pomocných tokenů.

Po nalezení vhodných míst ke vkládání pomocných tokenů a vyřešení sémantické analýzy zbývalo vyřešení vhodného způsobu vkládání. Nejdříve byly zkoumány možnosti pomocného vkládacího prvku vloženého mezi lexikální a syntaktický analyzátor. Později byla zkoumána možnost modifikace lexikálního analyzátoru a vkládání tokenů přímo v něm. Tato možnost byla i experimentálně testována a zdála se být realizovatelná. Nakonec ale byla vybrána a implementována možnost vkládání těchto tokenů přímo v parseru. Tento přístup má oproti oběma předchozím podstatnou výhodu, že je bezpečnější. V okamžiku pokusu o vložení tokenu je možné zkontrolovat lookahead parseru a ověřit, zda v něm již není načtený jiný token, který by bránil vložení. Přítomnost jakéhokoliv tokenu v lookaheadu totiž znamená možnost, že tento token již byl nějakým způsobem použit pro výběr dalšího kroku a jeho násilná změna by vedla k nepředvídatelnému chování parseru. Odstranění této nevýhodné vlastnosti parseru generovaného Bisonem by znamenalo zásah do kódu Bisona a tím i porušení základního požadavku na využití běžně dostupných prostředků.

Jak bylo zmíněno výše, podařilo se tímto způsobem vyřešit všechny R/R konflikty původní gramatiky VHDL. Experimentálně byla ověřena funkčnost parseru na částech zdrojových kódů VHDL. Zůstává však ještě jeden poměrně zásadní problém a tím jsou S/R

konflikty, které se nepodařilo zcela eliminovat. S/R konflikty nejsou z principu tak závažné jako R/R konflikty, ale zcela neočekávaně v oblasti jejich řešení zcela fatálně selhává Bison. V současné době přesně víme, kde tyto konflikty vznikají a jak je řešit, ale Bison neposkytuje žádný způsob, jak je vyřešit. Jediný způsob, který v Bisonu existuje pro řešení S/R konfliktů, je možnost nastavení precedence operátorů. Tento způsob však pracuje pouze v rámci pravidel jednoho nonterminálního symbolu gramatiky. Pro kompletní analýzu VHDL by však bylo nutné mít k dispozici prostředek, kterým by bylo možné určit, jak se má Bison zachovat v případě libovolného S/R konfliktu, nikoliv jen v rámci pravidel jednoho nonterminálu. Současná verze (2.3) Bisonu veškeré S/R konflikty mezi pravidly různých nonterminálů zcela nemodifikovatelně řeší generováním kódu pro operaci shift. Toto je naprosto vyhovující způsob pro většinu jazyků, které se parsery generovanými Bisonem zpracovávají. Bohužel to však není případ VHDL. Jediným řešením tohoto problému je modifikace kódu Bisona tak, aby umožňoval výběr řešení libovolných S/R konfliktů. Tato modifikace však zatím nebyla řešena.

Publikace (viz příloha publikace-FIT.zip)

- Křivka Zbyněk, Lorenc Luboš, Schönecker Rudolf: A Note on the Parsing of Complete VHDL-2002, In: *Information Systems and Formal Models* (*Proceedings of 2nd International Workshop on Formal Models* (WFM'07)), Opava, CZ, SLU, 2007, s. 245-248, ISBN 978-807248-006-7.

Závěr

Syntaktická analýza jazyka VHDL je sice možná, ale neefektivní bez podpůrné sémantické analýzy a sémantických kontrol (souhrně analýza VHDL). Analýza je součástí každého sofistikovaného překladače a v případě VHDL i různých rozšíření překladačů jako simulátory či syntézní nástroje.

Analýza VHDL je tedy velmi obtížný úkol, který by si vyžádal buď opravit a doplnit existující nástroj pro generování překladačů bison, nebo napsat překladač ručně a naprosto od základů.

Rozšíření bisonu by si vyžádalo vytvoření dostatečně sofistikovaného formálního modelu a algoritmů pro řešení R/R a především S/R konfliktů. Touto výzkumnou činností se lze eventuálně zabývat v některé z vedlejších aktivit práce na projektu či v rámci ostatních výzkumných aktivit ohledně formálních jazyků a překladačů na naší fakultě. Hlavní komplikace je v nejasnosti, zda lze vůbec jazyk VHDL popsat pomocí případné LR gramatiky, která si bude schopna poradit alespoň s některými typy konfliktů.

Napsat vlastní překladač je alternativní přístup, ovšem s obrovskou mírou náročnosti na pracovní zdroje i čas. Navíc takto vytvořený překladač by byl minimálně v prvních verzích velmi neoptimalizovaný a nevhodný k náročnějším překladům. Také volba správné metody pro analýzu by si vyžádala další zkoumání (tj. rozhodnutí, zda metodu shora dolů nebo zdola nahoru, či případně konkrétnější metody jako rekurzivní sestup, prediktivní analýza a nebo ještě mnohem pravděpodobněji kombinace hned několika metod spolu s ad-hoc úpravami těchto metod přímo pro potřeby analýzy velmi špatně definovaného VHDL, kdy ani ze standardu nevyplívají jasně některé vlastnosti a dokonce i správnost či špatnost některých syntaktických/sémantických konstrukcí.