Zpráva – výzkumné a vývojové práce – CODASIP 2017
Výzkum a vývoj aplikací pro Codasip

Codasip Brno s.r.o.
FIT VUT v Brně

Řešení projektu se skládá v roce 2017 z několika etap a stav řešení jednotlivých etap je následující.

V etapě a) jsme se seznámili s problematikou paralelního programování pomocí zasílání zpráv. Na základě získaných znalostí a s využitím současných síťových technologií jsme navrhli a implementovali upravenou variantu komunikačního protokolu MSI, která umožňuje sdílení zdrojů generických simulátorů procesorů a tím provádět efektivní simulaci víceprocesorových systémů SoC (systém na čipu). Mimo testování samotného protokolu pro podporu simulace SoC jsme rozšířili některé stávající komponenty projektu tak, aby umožňovaly práci s obecným množstvím simulátorů. Byl automatizován proces instalace simulátorů a prezentační vrstva vývojového prostředí poskytuje nyní ovládání jak jednoprocesorové, tak i víceprocesorové simulace.

V etapě 2b) byl vytvořen návrh grafického specifikačního jazyka založeného na doménově specifickém jazyce UML. Nejdříve byla zvolena vhodná verze jazyka UML (na základě požadavků kladených na návrh vestavěných systémů a jejich mikroprocesorů byla zvolena verze 2.0). Pro podporu návrhu mikroprocesoru v této verzi byl využit rozšiřující mechanismus profilů, který verze 2.0 poskytuje. Dále byla zvolena množina UML diagramů, jež umožňuje nejlépe specifikovat požadavky, které jsou kladeny na navrhovaný mikroprocesor. Vybrané diagramy byly rozšířeny o nové prvky (pomocí zmíněných profilů), jež jsou typické pro oblast HW/SW co-designu.

V rámci etapy c) došlo k přehodnocení celkového konceptu simulátorů, jež byly navrženy a implementovány v první verzi. Důraz byl kladen na efektivnost vlastní simulace, které se podařilo dosáhnout použitím jiných formálních modelů pro reprezentaci instrukční sady a vlastního časování mikroarchitektury, na rozdíl od první verze. Díky aplikaci nových modelů bylo možné navrhnout efektivnější ladicí nástroje nar ozdíl od první verze, což v konečné fázi vedlo k rychlé simulaci podporující ladění simulovaného programu. Integrace protokolu navrženého v etapě a) umožní provádět víceprocesorovou simulaci systému na čipu.

Během roku byl vypracován koncept, jež řeší transformaci specifikačního jazyka mikroprocesoru (v němž je model mikroprocesoru popsán) do hardwarově popisného jazyka. Pro řešení vývojové etapy d) se s úspěchem použil princip šablon, kdy si uživatel pro funkční jednotky (jako jsou např. sčítačky, násobičky atd.) a zdrojové prvky (paměti, registry, atd.) dodá parametrizovatelné stavební bloky. Dále je možné některé stavební bloky generovat přímo na základě modelu mikroprocesoru v jazyce ISAC. Těmito stavebními bloky jsou řadič a dekodér. Při generování těchto stavebních bloků využíváme podobný princip jako u generátoru rychlých simulátorů, tedy na základě formálních modelů se generují těla těchto dvou bloků. Výhodou tohoto principu je, že simulátory simulují přesně to, co je implementováno v hardwaru (toto je stěžejní např. pro odladění časování mikroarchitektury). Princip šablon má nevýhodu v tom, že neznáme strukturu jednotlivých šablon, a tedy nemůžeme na základě modelu mikroprocesoru generovat přímo propojení všech stavebních bloků. Můžeme pouze generovat přímo propojení všech generovaných stavebních bloků. Propojení parametrizovaných stavebních bloků s generovanými prvky musí návrhář provést ručně (tedy propojení signálů na jednotlivé porty parametrizovaných stavebních bloků), pomocí signálů, které jsou generovány. Informace pro která propojení stavebních bloků mají být signály použity se nachází v komentáři u každého generovaného signálu. Dále jsou informace zachyceny v takzvaném grafu procesů. Graf procesů je orientovaný značeny graf, který zachycuje propojení všech stavebních bloků pomocí pojmenovaných signálů. Protože, je nutné ověřit správnost

propojení stavebních bloků (návrhář může zanést chybu), byl vytvořen nástroj na extrakci grafu procesů ze zkompletovaného mikroprocesoru ve VHDL jazyce. Pomocí dalšího nástroje je možné říci, zda návrhářem vytvořená propojení byla správně realizována, a za tímto účelem se provádí ověřování takzvané strukturální ekvivalence. Tedy jsme se zabývali vlastní ekvivalencí mezi jazykem pro popis architektury a hardwarovým jazykem. V následujícím výzkumu v této oblasti se zaměříme na funkční ekvivalenci, která bude založena na ověření ekvivalence konečných automatů pro přijetí instrukcí strojového jazyka a pro popis časování mikroarchitektury, které budou získány z popisného jazyka ISAC a na základě extrakce z VHDL modelu. Dále nástroj pro generování a parametrizování stavebních bloků byl navržen tak, aby zachovával informace, které návrhář ručně zanese do generovaných VHDL staveních bloků.
Po vyhodnocení etap ukončených v minulém roce bylo nutné přehodnotit zvolený koncept pro softwarové nástroje pracujících s instrukční sadou, zejména jsme se zaměřili na koncept generického nástroje assembleru, disassembleru a simulátoru. Na základě zjištěných úzkých míst se v etapě e) vytvořilo nové řešení pro nástroje assembleru a zpětného assembleru, které umožňuje pracovat s více komplexní instrukční sadou. Překlad instrukcí je realizován pomocí párových překladových konečných automatů. Vedle párových automatů bylo nutné pro zpracování kompletního jazyka assembleru použít zásobníkový automat, protože kompletní jazyk assembleru obsahuje jak jazyk direktiv, tak jazyk výrazů (pro zpracování tohoto typu jazyka nutné volit zásobníkový automat) a rovněž assemblerovský jazyk instrukční sady. V rámci etapy bylo vyřešeno propojení párového konečného překladového automatu se zásobníkovým konečným automatem.