Nové metody optimalizace intepretace scénářů portovatelných stimulů

Petr Bardonek
1. ročník doktorského studijního programu, prezenční forma studia
Ing. Marcela Zachariášová, Ph.D.

Ústav počítačových systémů
Vysoké učení technické v Brně
Brno, Česká republika
ibardonek@fit.vutbr.cz

***Abstrakt*—Neustále narůstající komplexnost hardwarových návrhů vyžaduje nejen nepřetržité zlepšování technik a postupů návrhů, ale také funkční verifikace, která se stala nedílnou součástí celého procesu. Jednou z novinek přinášející potencionální zlepšení funkční verifikace je standard pro portovatelné stimuly od organizace Accellera. Cílem dizertační práce bude podpořit využívání nového standardu pomocí různých rozšíření. Prvním krokem je vytvořit návrhové vzory, které usnadní portovatelnost mezi abstrakčními úrovněmi. Dále je cílem vytvořit propojovací systém, který umožní zaměřit verifikační scénáře z vyšších úrovní abstrakce na konkrétní oblasti (IP bloky) verifikovaného systému. Třetím krokem je automatizace verifikace pomocí algoritmu strojového učení. Posledním krokem je využití propojovacího systému pro zaměření algoritmu strojového učení na konkrétní oblast verifikovaného systému.**

***Klíčová slova*—portovatelné stimuly, funkční verifikace, procesory**

I. Úvod

Počítačové a vestavěné systémy jsou dnes již nedílnou součástí života každého člověka. U těchto systémů je potřeba zajistit jejich bezchybnou funkčnost s ohledem na specifikaci, podle které byly navrženy. V důsledku neustále narůstajících nároků na tyto systémy roste jejich komplexnost a tím i obtížnost návrhu neobsahujícího žádné chyby. Disciplína zabývající se touto problematikou se nazývá verifikace. V dizertační práci se budeme zabývat v současné době nejrozšířenějším typem verifikace v této oblasti a to funkční verifikací.

Funkční verifikace se vždy potýkala s problémy spojenými s produktivitou. Funkční verifikaci je potřeba provést na všech úrovních systému (IP bloky, subsystémy, systém jako celek), dále je také potřeba provádět verifikaci na různých platformách (simulace, emulace, FPGA prototypování). Celý tento proces je časově velice náročný a neobejde se bez chyb.

Mezi největší problémy patří míra redundance při použití pseudonáhoných verifikačních scénářů, které se řeší různými způsoby, kdy například v článku [1] vytváří propojení mezi získaným pokrytím a stimuli za běhu simulace.

Dalším problémem je pak portovatelnost mezi různými platformami a úrovněmi systému. Tímto problémem se například zabývají v článku [2], kdy se cílí na znovupoužitelnost referenčních modelů a verifikačních prostředí vytvořených v různých jazycích a to pomocí knihovny UVM-ML, která umožní využít tyto prvky napsané v různých jazycích v rámci jediného verifikačního prostředí. Cílem je tedy redukce času a usilí při verifikaci za zachování stejné kvality pomocí spojení již vytvořeného.

Poslední zde zmíněný problém je pak nemožnost použití pseudonáhodných verifikačních scénářů na systémové úrovni. Tento problém firmy řeší vytvořením generátoru testovacích programů, kdy nejrozšířenějším způsobem verifikace na této úrovni je pak spouštění velkého množství randomizovaných programů v asembleru. V článku [3] například představují nástroj pro automatické vytváření generátorů testovacích programů.

Všechny problémy spojené s produktivitou funkční verifikace se snaží řešit nový standard pro portovatelné stimuly od organizace Accellera [4]. Klíčem k portovatelným stimulům je zvýšení úrovně abstrakce pro definici verifikačních scénářů a využití jediné definice napříč celým procesem funkční verifikace (různé úrovně abstrakce, různé platformy).

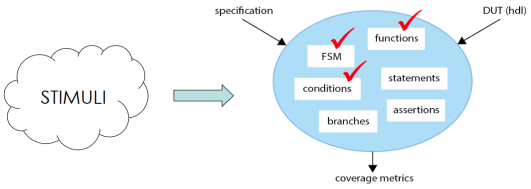
Práce se zaměřuje na podporu využití nového standardu pro portovatelné stimuly. Cílem je usnadnit portovatelnost, implementovat rozšíření podporující nový standard a také využít prvků nového standardu pro zefektivnění procesu funkční verifikace. V rámci tohoto článku budou představeny čtyři konkrétní příklady, které plánuji postupně v rámci dizertace implementovat. Vycházejí z mého ročního studia této problematiky a komunikace s autory PSS standardu a nástrojů, které ho interpretují.

II. Funkční verifikace - současný stav, problémy

Funkční verifikace je definována jako proces, kdy jsou sledovány výstupy (chování) komponenty reagující na vstupy (stimuly) s ohledem na danou specifikaci. Z těchto stimulů sestavujeme tzv. sekvence, které ve výsledku definují nějaký verifikační scénář, tedy chování komponenty při nějaké situaci vyvolané danými stimuly. Funkční verifikace je postavena na tom, že je provedena co nejvěrnější simulace reálného prostředí, ve kterém navrhovaná komponenta ve výsledku poběží a dojde tak k ověření funkční korektnosti komponenty s ohledem na její specifikaci. Simulace reálného prostředí verifikované komponenty je tvořena mnoha částmi, které zodpovídají mimo

jiné také za generování a posílání stimulů komponentě a analýze výstupů. Všechny tyto části jsou navzájem propojené a tvoří tak jeden celek, který se nazývá verifikační prostředí.

Pro sledování průběžného pokroku v procesu funkční verifikace a jeho měření je využíváno různých metrik pokrytí. Pokrytí je zpětně analyzováno a na základě této analýzy jsou provedeny případné úpravy verifikačních scénářů za účelem dosažení 100% pokrytí. Pokrytí se dělí na strukturní, kdy je sledováno pokrytí samotného kódu definujícího komponentu, přičemž se takovýmto způsobem odhalí například nevyužívaná a hlavně zbytečná část kódu (mrtvý kód). Dalším typem je pak funkční pokrytí, tento typ pokrytí je implementován verifikačním inženýrem a je zcela na jeho pochopení specifikace dané komponenty, co je potřeba sledovat. Základní princip využití různých metrik pokrytí je vidět na obrázku 1. Obrázek demonstruje generování stimulů, které postupně pokrývají všechny metriky pokrytí definované pro danou úlohu. Toto postupné pokrývání metrik je pak vyjádřeno procentuálně, kdy je vždy snahou dosáhnout 100% pokrytí, tedy pokrýt vše co bylo definováno jako podstatné.



Obrázek 1. Základní princip využití různých metrik pokrytí

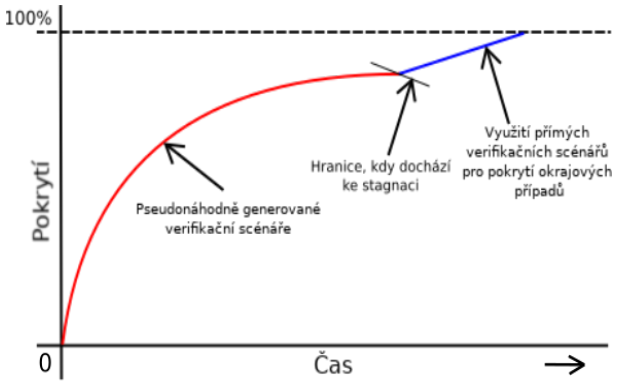
V posledních 20 letech bylo vynaloženo mnoho úsilí k zvýšení efektivity a produktivity funkční verifikace komponent s důrazem na zachování kvality. Byly vytvořeny různé verifikační metodiky, které definují jak mají vypadat verifikační prostředí.

Výsledkem je dnes běžně používaná metodika UVM (Universal Verification Methodology). Tato metodika využívá dvou způsobů pro tvorbu sekvencí stimulů definujících verifikační scénáře. Přímý, kdy verifikační inženýr danou sekvenci stimulů sestavuje ručně a tedy i zodpovídá za definování vhodného verifikačního scénáře, a pseudonáhodný, kdy je využíváno generátoru stimulů, které automaticky vytváří sekvenci definující verifikační scénář, ale s ohledem na omezení (angl. constraints), které umožňují určitou míru kontroly [5].

A. Definice problémů

Zde je definováno několik problémů, které se aktuálně vyskytují ve světě funkční verifikace:

1) Problém redundance – navzdory kontrole generátoru stimulů získané omezeními, je problémem generování velkého množství redundantních (opakujících se) kombinací stimulů, což způsobuje stagnaci při snaze dosáhnout 100% pokrytí metrik a značně tak prodlužuje dobu verifikace (největším problémem je zde zachycení okrajových případů, např. dělení nulou, dělení nekonečna nekonečnem atd.). Typický průběh tohoto procesu je zachycen na obrázku 2, kde je vyobrazeno pokrytí metrik v závislosti na čase, při využití obou způsobů pro tvorbu sekvencí stimulů definujících verifikační scénáře.
2) Problém portovatelnosti – nemožnost portovatelnosti vytvořených verifikačních scénářů při přechodu na jinou úroveň abstrakce (bloková, subsystémová a systémová úroveň) nebo při snaze verifikovat komponentu na jiné platformě (simulace, emulace, FPGA prototypování). Tento problém je způsoben tím, že verifikace se provádí na každé úrovni abstrakce a platformě různým způsobem a v jiném programátorském prostředí.
3) Problém vytváření pseudonáhodných verifikačních scénářů na systémové úrovni – absence možnosti vytváření stimulů pomocí generátoru na systémové úrovni, kdy na této úrovni již není možnost využít generátoru metodiky UVM, neboť je potřeba vytvářet verifikační scénáře ve formě programů. Tyto programy jsou posléze vykonávány na procesoru, který pak řídí ve výsledném zapojení celý systém (simulace reálného prostředí).

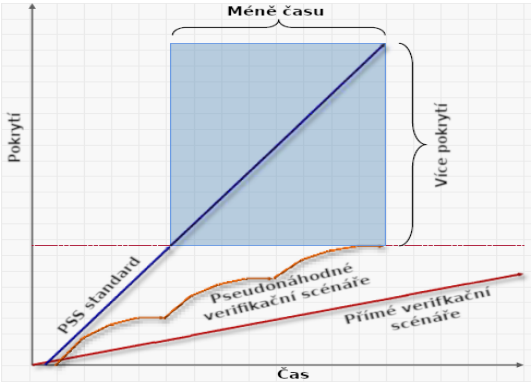


Obrázek 2. Pokrytí metrik v závislosti na čase

III. Portovatelné stimuly

Všechny tyto problémy se snaží vyřešit nový standard pro portovatelné stimuly (PSS) od organizace Accellera a to pomocí řízení generátoru vytvářejícím stimuly skrze graf, který představuje jeden nebo více verifikačních scénářů. Graf vytvořený pomocí PSS standardu je pouze modelem verifikačního scénáře, kdy chování jeho jednotlivých uzlů je implementováno verifikačním inženýrem a závisí jak na platformě tak abstrakční úrovni, kde probíhá funkční verifikace. Graf je nejdříve definován verifikačním inženýrem pomocí deklarativního jazyka PSS ve formě pravidel, která jsou posléze převedena do grafové reprezentace. Následně je vytvořena tzv. strategie pokrytí, kdy dochází určitým způsobem k mapování metrik pokrytí na vygenerovaný graf pro další směrování verifikace. Jak na pravidla definující graf, tak na namapované metriky pokrytí je možné aplikovat omezení (angl. constraints), která ještě zvyšují míru kontroly nad generovaným verifikačním scénářem. Všechny tyto řídící prvky výrazně snižují míru redundance. Na obrázku 3 je PSS standard porovnán s klasickými přístupy tvorby verifikačních scénářů, co se týká rychlosti, s jakou zvyšuje pokrytí.

Portovatelné stimuly si dávají za cíl definovat záměr verifikace (verifikačního scénáře) pouze jednou pro danou kompo-



Obrázek 3. Metriky pokrytí - PSS standard vs klasické způsoby

nentu napříč celým verifikačním procesem (různé úrovně abstrakce, různé platformy) a cílí tak na problém portovatelnosti. Toho je dosaženo tak, že je oddělena definice verifikačního scénáře, tedy záměr dané verifikace (graf), od jeho implementace, tedy konkrétní implementace chování jednotlivých uzlů grafu. V PSS je "portovatelný" pouze záměr dané verifikace (graf), portovatelnost konkrétní implementace je již ponechána na verifikačním inženýrovi. Pro praxi by portovatelnost konkrétní implementace znamenala značné usnadnění a zrychlení procesu funkční verifikace, avšak doposud nikým nebyla definována.

S příchodem standardu pro portovatelné stimuly se otevřely verifikačním inženýrům nové možnosti. Pro plné využití jeho potenciálu je ale potřeba, aby byly vytvořeny vhodné postupy, algoritmy a pomocná rozšíření/nástroje s tím spojená. V rámci verifikační komunity se řeší různé problémy se záváděním tohoto nového standardu do již zakotvených procesů funkční verifikace, využívaných v průmyslu. Probíhá vývoj různých nástrojů podporujících tento standard, kdy je potřeba, aby tyto nástroje uměly podporovat všechny funkce, se kterými tento standard přichází. Jako příklad problému lze uvést vhodnou reprezentaci grafů, vytvářených podle popisu definovaného deklarativním jazykem PSS, dále pak jsou tu různé problémy spojené s mapováním metrik pokrytí na daný graf, optimalizace algoritmů pro generování optimálních verifikačních scénářů atd. [6][7][8].

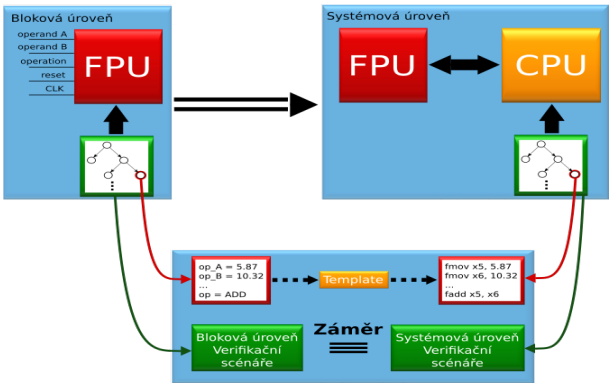
IV. Teoretické rozpracování kroků dizertace

Dizertace je zaměřena na využití PSS standardu, tedy konkrétně na ukázku vývoje návrhových vzorů (angl. templates) pro účely portovatelnosti a také na implementaci rozšíření za účelem lepšího využití PSS. Na systémové úrovni je často žádoucí vygenerovat vysoko-úrovňové stimuly, které vyvolají aktivitu na úrovni nízké. Vysoko-úrovňovými stimuly na systémové úrovni jsou obvykle instrukce vykonávané procesorem, zatímco stimuly na nízké úrovni (úroveň IP bloků) jsou vstupy jdoucí přímo do jednotlivých komponent. Jako konkrétní demonstrační příklad budou v dizertaci využity různé typy procesorových architektur, hlavní zaměření bude na procesorové architektury typu RISC. Jde totiž o vhodného reprezentanta, protože můžeme snadno rozlišit úroveň IP bloků, subsystému, případně i celého systému.

PSS jakožto nový standard přináší novinky do světa funkční verifikace a nelze se ubránit porovnávání s tím co se již v praxi používá. Je potřeba vyhodnotit klady a zápory, zjistit, zda se vyplatí na tento nový standard přecházet, protože budou vyžadovány určité změny v již zaběhnutých procesech. Dizertace je rozdělena na čtyři části, kdy každá z nich je využitelná samostatně a zároveň poskytuje základy, či další možnosti pro části nadcházející. Bude snahou vždy provést kvalifikované porovnání s klasickými přístupy při jejich aplikaci na stejný problém.

Prvním krokem dizertace bude navrhnout a odladit návrhové vzory (angl. templates) pro typické procesorové komponenty vyskytující se v různých architekturách, které by ulehčily verifikačnímu inženýrovi přechod z úrovně IP bloků na systémovou úroveň a zároveň byla zachována požadovaná stimulace chování této komponenty na systémové úrovni. Jde tedy o převedení nízkoúrovňových stimulů, které jsou podle potřeby kombinované na vstupech komponent, do formy instrukce případně bloku instrukcí posléze vykonávané procesorem. Cílem je doplnit PSS standard o možnost portovatelnosti konkrétní implementace verifikačního scénáře, tedy chování jednotlivých uzlů grafu.

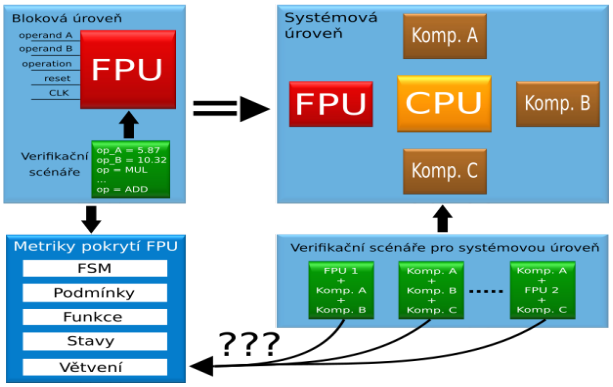
Pro odladění těchto návrhových vzorů bude využíváno metrik pokrytí z úrovně IP bloků, kdy bude porovnáváno pokrytí metrik dosaženého při funkční verifikaci na úrovni IP bloků s dosaženým pokrytím po převodu na systémovou úroveň s využitím navrhovaných vzorů (angl. templates) na stejných metrikách. Jako příklad lze uvést FPU modul, kdy nejdříve dojde k funkční verifikaci modulu samostatně na blokové úrovni a následně je verifikován jako část subsystému a systému. Je žádoucí, aby verifikační scénáře z blokové úrovně byly zachovány, neboť se jedná stále o tentýž FPU modul. Grafické znázornění s modulem FPU lze vidět na obrázku 4.



Obrázek 4. Grafické znázornění 1. kroku dizertace

Dalším krokem dizertace je zohlednění metrik pokrytí z úrovně IP bloků při vytváření/generování verifikačních scénářů na systémové úrovni. Pro tuto část dizertace bude potřeba vytvořit propojovací systém, který provede vhodnou asociaci verifikačních scénářů na systémové úrovni s metrikami pokrytí na úrovni IP bloků. Cílem je identifikovat verifikační scénáře na systémové úrovni, které s největší pravděpodobností dokáží zvýšit dosažené pokrytí na úrovni IP bloků. Díky tomu, že PSS je deklarativní jazyk, propojovací systém tohoto typu

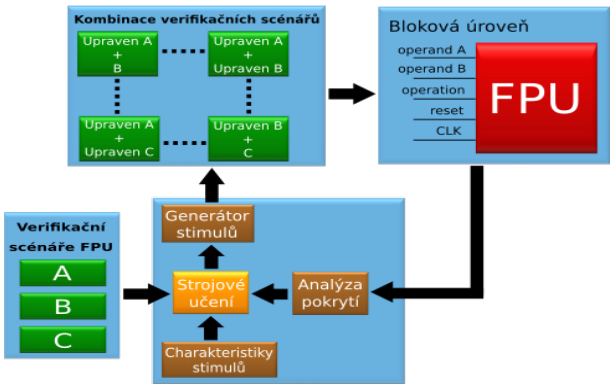
bude velice nápomocen při zaměřování verifikace na různé oblasti systému. Jako příkladu je využito FPU modulu, kdy při přesunech mezi abstrakčními úrovněmi, se jeho verifikační scénáře stávají součástí větších, použitých při verifikaci zahrnující i další komponenty, které také mohou nějakým způsobem ovlivňovat FPU modul. Je žádoucí neztratit přehled o tom, které tyto větší verifikační scénáře verifikují právě FPU, a které její konkrétní části (např. jednotlivé funkce). Grafické znázornění výše popsaného je vyobrazeno na obrázku 5.



Obrázek 5. Grafické znázornění 2. kroku dizertace

Dalším krokem dizertace bude využití algoritmů strojového učení. Dosavadní výzkum a vývoj využívající strojového učení v oblasti funkční verifikace je zaměřen na generování verifikačních scénářů za pomoci pseudonáhodného generátoru stimulů. Nicméně takovéto využití výrazně limituje možnosti algoritmů strojového učení ukázat něco užitečného, neboť stimuly vytvářené generátorem jsou charakterizovány pouze aktuálním "seedem" generátoru, kdy se jedná o velice slabou korelaci s generovanými stimuly. PSS standard přináší prvky, které pomáhají se systematickým generování stimulů a poskytuje tak mnohem více informací pro charakterizaci generovaných stimulů. PSS tak dává větší prostor algoritmům strojového učení pro adaptaci generování stimulů (a jimi tvořenými verifikačními scénáři) v závislosti na analýze výstupů reagujících na takto charakterizované stimuly (pod pojmem analýza výstupů rozumíme například analýzu použitých metrik pokrytí).

Zde se naskytuje možnost implementovat více různých algoritmů strojového učení (např. neuronové sítě, genetické algoritmy), provést jejich následné porovnání a najít tak nejvhodnější. Pro jednoduchý příklad opět využijeme FPU modul. Mějme vytvořeny tři verifikační scénáře pro tento modul, kdy každý je zaměřen na jinou jeho funkcionalitu, a můžeme je považovat za nutný základ pro pokrytí všech definovaných metrik FPU modulu. Úkolem implementovaného algoritmu bude dosažení 100% pokrytí pouze s těmito třemi verifikačními scénáři, kdy na algoritmu bude je vhodně zkombinovat/pozměnit. Změna verifikačního scénáře je do určité míry umožněna skrze omezení (angl. constraints) a některých konstrukcí jazyka standardu. Zde je tedy cílem na základě analýzy výstupů a charakterizaci stimulů, provést patřičné úpravy automaticky pomocí algoritmu strojového učení a dosáhnout tak 100% pokrytí. Výše popsaný příklad je graficky znázorněn na obrázku 6.



Obrázek 6. Grafické znázornění 3. kroku dizertace

Další část dizertace se bude zabývat kombinací dvou předchozích kroků, jenž budou do určité míry prováděny nezávisle. Po jejich dokončení se dizertace zaměří na využití vytvořeného propojovacího systému pro další adaptaci řízeného generování stimulů vybraným algoritmem strojového učení. Zatímco dosavadní práce s algoritmy strojového učení bude pracovat pouze s generickými prvky PSS standardu, v tomto kroku bude snaha využít dizertací navržené rozšíření za účelem dalšího vylepšení (lepší výsledky, rychlost, možnost zaměření určité oblasti při funkční verifikaci na systémové úrovni).

V. Závěr

Dizertace dává mnoho možností pro výzkum a vývoj a to ať už v rámci pozorování zda nové cesty, které nový PSS standard otevřel pro oblast funkční verifikace, přinášejí očekávaná vylepšení nebo v rámci zkoumání různých technik a postupů využitých při práci na dizertaci.

Práce se snaží podpořit využití nového standardu komunitou pomocí návrhových vzorů pro usnadnění přechodu mezi různými abstrakčními úrovněmi, na kterých je prováděna funkční verifikace. Dále je také snahou zlepšit možnosti využití nového standardu vytvořením různých rozšíření zaměřených na pokrytí použitých metrik napříč procesem funkční verifikace.

Lze také počítat, že mnou navrhnuté algoritmy mohou být časem integrovány do nástrojů podporujících PSS standard.

Reference

[1] A. Yehia, "Faster coverage closure: Runtime guidance of constrained random stimuli by collected," in *2013 Saudi International Electronics, Communications and Photonics Conference*, April 2013, pp. 1–6.

[2] S. Dahir, H.-M. Bluethgen, R. Zuralski, N. Luetke-Steinhorst, and C. Sauer, "Using UVM-ML library to enable reuse of TLM2.0 models in UVM test benches," *DVcon Europe*, 2018.

[3] M. Chupilko, A. Kamkin, A. Protsenko, S. Smolov, and A. Tatarnikov, "MicroTESK automated architecture validation suite generator for microprocessors," *DVcon Europe*, 2018.

[4] Accellera, "Portable test and stimulus standard," https://www.accellera.org/activities/working-groups/portable-stimulus/, Feb 2019.

[5] C. Spear and G. Tumbush, *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*, 3rd ed. Springer Publishing Company, Incorporated, 2012.

[6] T. Yang and E. Qin, "Bridge the portable test and stimulus to UVM simulation environment," *DVcon US*, 2018.

[7] P. Karppa, L. Matilainen, and M. Ballance, "Building portable stimulus into your IP-XACT flow," *DVcon US*, 2018.

[8] M. Bartley, "PSS the promises and pitfalls of early adoption," *DVcon US*, 2019.