Tento článek je původním rukopisem textu publikovaného v časopise DPS Elektronika A-Z:
J. Šťastný. Simulace číslicových obvodů na hradlové úrovni, DPS Elektronika od A do Z, pp. 8 - 11, květen/červen 2015

Simulace číslicových obvodů na hradlové úrovni

Jakub Šťastný
ASICentrum, s.r.o.
Katedra teorie obvodů FEL ČVUT Praha

1 Úvod

Předchozí díly [1,2] byly věnovány základům simulace řízené událostmi a bylo prezentováno několik užitečných triků a nebezpečných úskalí. Tento text je zaměřen na problematiku simulace na hradlové úrovni. Čtenář bude seznámen s tím, co je simulace na hradlové úrovni, s jejími výhodami a nevýhodami a s důvody pro její provádění. O hradlové simulaci je třeba hovořit v kontextu celého procesu návrhu číslicového obvodu (*design flow*), a proto bude čtenář seznámen i s dalšími nástroji, které mohou potřebu simulací na hradlové úrovni omezit (ovšem ne nahradit). Text je, přestože se hovoří o návrhových nástrojích pro FPGA obvody, vesměs nezávislý na návrhové platformě a poznatky jsou tak aplikovatelné jak při návrhu číslicových systémů na FPGA platformě, tak i při návrhu plně zákaznických číslicových obvodů (ASIC, *Application Specific Integrated Circuits*).

2 Co je simulace na hradlové úrovni

V předchozích dílech jsme hovořili o RTL úrovni. Na té simulujeme abstraktní popis obvodu bez modelování skutečného zpoždění na jednotlivých obvodových prvcích a spojích mezi nimi. Oproti tomu při simulaci na hradlové úrovni (*back-annotated gate level simulation*) je v simulátoru modelováno skutečné zapojení obvodových prvků v číslicovém obvodu včetně zpoždění jak na těchto prvcích, tak i na jejich vzájemných spojích. Navíc simulátor automaticky kontroluje dodržení základních parametrů časových průběhů signálů na vstupech jednotlivých obvodových prvků, tedy například dodržení předstihu a přesahu [3,4] a doby zotavení po resetu [5] u klopných obvodů.

Simulace na hradlové úrovni představuje zajímavý kompromis mezi simulací na RTL úrovni a simulací na tranzistorové úrovni. Při zachování jednoduchosti nastavení a práce se simulátorem skoro jako na RTL úrovni návrhář dostává možnost simulovat reálné chování obvodu téměř na tranzistorové úrovni. Oproti tranzistorové úrovni navíc získáváme příjemný bonus kontroly dodržení časových parametrů obvodových prvků a významně vyšší rychlost simulace. Na druhou stranu hradlová simulace za simulací na RTL úrovni zaostává v rychlosti a tranzistorová simulace zase věrněji modeluje fyzikální realitu.

Pro simulaci na hradlové úrovni je potřeba připravit:

- **netlist**: netlist je model skutečné obvodové implementace návrhu ve zvolené technologii. Je to textový soubor v HDL jazyce (Verilog nebo VHDL), který na strukturní úrovni popisuje obvod; jsou v něm instancovány a vzájemně propojeny jednotlivé obvodové prvky. Netlist je generován nástroji pro rozmístění a propojení návrhu (*place and route*). Hradlovou simulaci lze také rozběhnout s použitím netlistu vygenerovaného již po syntéze číslicového obvodu, ale taková simulace má jen omezené použití.
- **SDF soubor**: Spolu s netlistem simulátor potřebuje i tzv. SDF soubor (*Standard Delay File*, [6]) s informacemi o reálném zpoždění a požadovaném časování na všech částech simulovaného obvodu. SDF soubor pro návrh je také generovaný po rozmístění a propojení návrhu ze známého rozložení prvků na FPGA obvodu a parazitních kapacit a odporů jednotlivých spojů.
- **technologickou knihovnu**: Ta obsahuje modely skutečných obvodových prvků poskytnuté výrobcem FPGA obvodu, nebo oddělením podpory návrhu (*design support*) příslušné „slévárny“ (*foundry*), ve které se bude zákaznický obvod vyrábět. Ty zajišťují modelování reálného funkčního chování obvodových prvků, dále simulují zpoždění šíření signálů podle informací ze SDF souboru a nakonec kontrolují dodržování správných časových parametrů průběhů signálů na vstupech jednotlivých bloků. Nastavení časových parametrů v instancích modelů jednotlivých buněk obvodu se provádí v simulátoru po načtení netlistu a SDF souboru; proces se nazývá zpětná anotace (*back annotation*).

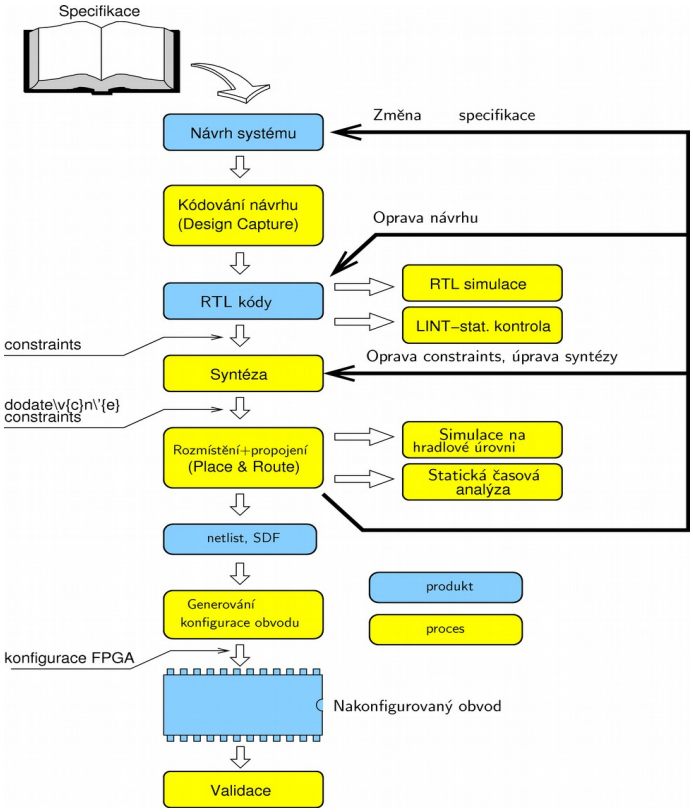
3 Statické techniky kontroly kódu

Již na RTL úrovni lze nalézt řadu problémů v návrhu pomocí tzv. statických technik kontroly návrhu. Nástroje pro statickou kontrolu mají výhodu v tom, že nevyžadují žádné stimuly a neprovádějí simulaci navrhovaného obvodu (proto statická kontrola vs. dynamická kontrola pomocí simulace). Jejich nastavení je také relativně rychlé. Nevýhodou nicméně je, že nezachytí problémy, které vznikají z dynamického běhu navrženého bloku, nemohou tedy zcela nahradit

simulace; na druhou stranu nejsou spoutány omezeními pramenícími z neúplného pokrytí obvodu simulacemi.

3.1LINT

LINT byl původně aplikací pro statickou analýzu kódu napsaného v C [7]; umožňoval odhalit některé běžné programátorské chyby, které nejsou odhalitelné kompilací zdrojového kódu a vedly by ke zhroucení aplikace později za jejího běhu. Ladění aplikace je časově náročné a tak je praktické některé typy problémů odchytit dopředu statickou kontrolou. Nejjednodušší kontrolou tohoto typu je například kompilace v simulátoru ModelSim s přepínačem *-check_synthesis* [2], nebo analýza syntézního logu a následný rozbor varování hlášených ze syntezátoru.



*Obrázek 1: Schéma procesu návrhu číslicového systému na FPGA platformě.**Soubor design_flow.eps***

3.2Statická časová analýza

Pomocí simulace na hradlové úrovni není možné vyzkoušet všechny myslitelné cesty v navrhovaném obvodu; navíc také často potřebujeme vědět na jaké maximální hodinové frekvenci je obvod ještě schopen pracovat. A ani když simulací objevíme problém s funkcí obvodu způsobený příliš pomalou cestou v obvodu, nemusí být lehké zjistit, která cesta obvodem je příliš pomalá - a už vůbec ne, zda je to jen jediný lokalizovaný problém, či zda je podobných špatných cest více. Přitom cestou obvodem zde rozumíme kombinační cestu obvodem, která typicky začíná hodinovým vstupem klopného obvodu, nebo primárním vstupem (vstupem na rozhraní navrhovaného obvodu a vnějšího světa), prochází přes kombinační obvody mezi registry a končí na vstupu D klopného obvodu či na primárním výstupu obvodu.

Proto je návrháři dostupný nástroj nazvaný statický časový analyzátor (*Static Timing Analyser*, STA). Statický časový analyzátor pracuje tak, že nalezne všechny možné cesty obvodem a posléze spočítá odhad jejich zpoždění pomocí zjednodušených modelů zpoždění. Pak porovná vypočtené odhady zpoždění s nejkratší požadovanou periodou systémových hodin a poskytne informace o cestách obvodem, které jsou příliš pomalé.

Pokud by celý proces fungoval jak je popsáno, byl by statický časový analyzátor značně pesimistický; všechny nalezené cesty obvodem totiž obsahují jak cesty, které jsou skutečně využívány, tak cesty, které v obvodě nejsou nikdy aktivovány (*false paths*, falešné cesty) a konečně cesty, po kterých se signál může šířit více hodinových cyklů (*multicycle paths*, vícecyklové cesty). Proto návrhář musí spolu s vlastním návrhem předložit také tzv. *constraints*. Ty představují model časování číslicového systému, jsou v nich definovány různé typy cest, které je třeba ignorovat a dále například hodinové frekvence jednotlivých zdrojů. Například pro vývojové prostředí firmy Xilinx lze o constraints a jejich psaní nalézt více v dokumentu [8].

Velká výhoda statické časové analýzy tkví v tom, že není třeba simulace a že jsou analyzovány všechny cesty v obvodu. Na druhou stranu je primárně určena pro synchronní systémy. Jako taková nemusí poskytovat vyčerpávající analýzu

návrhů s více asynchroními hodinovými doménami. Dále nedokáže identifikovat logicky chybně (ale syntakticky správně) napsané *constraints*; přitom statická časová analýza je jen tak dobrá, jak dobře jsou napsány *constraints*.

4 Výhody a nevýhody hradlové simulace

Úpravy verifikačního prostředí i testů pro provádění simulace na hradlové úrovni jsou časově náročné, proto se můžeme ptát, zda takové simulace návrhu vůbec dělat a co nám to přinese? Aby bylo možné tuto otázku zodpovědět, nejprve shrneme výhody simulace na hradlové úrovni (detailněji viz např. [9,10,11]):

- **Simulace reálných zpoždění na obvodových prvcích**: Zásadní výhodou hradlové simulace je, že jsou jednak simulována reálná zpoždění na obvodových prvcích a jednak je během simulace automaticky kontrolováno dodržení základních časových parametrů jednotlivých obvodových prvků. Můžeme tak odladit problémy, které na RTL úrovni nevidíme.
- **Validace výstupů statické časové analýzy**: Simulace na hradlové úrovni aktivuje ty cesty obvodem, které jsou vybuzeny připravenými stimuly a při vhodné volbě stimulů pro obvod může pomoci částečně ověřit správnost STA a zesílit v návrháři pocit, že vše funguje jak má. Tímto způsobem lze částečně validovat *constraints* pro statickou časovou analýzu a další nástroje. Hradlová simulace umožňuje při troše štěstí i nalézt některé nevhodně implementované přechody mezi jednotlivými hodinovými doménami v návrhu.
- **Verifikace správného chování návrhu po resetu/po připojení napájecího napětí**: Běžný model klopného obvodu v technologické knihovně nastavuje svůj výstup *Q* po startu simulace do stavu *X*. Stav X na výstupu klopného obvodu zůstává do doby, než je tento asynchronně resetován, nebo do okamžiku prvního zápisu do klopného obvodu (s hodinami). To za příznivých podmínek umožňuje ulehčit odhalení nečekané závislosti návrhu na počátečním stavu použitých paměťových prvků; díky agresivitě stavu X je často při problému celý návrh v simulátoru zaplaven X (*X flooding*) a návrhář má možnost odladit celý problém.
- **Generování podkladů pro odhad spotřeby elektrické energie obvodem a pro optimalizaci obvodu na spotřebu**: Dynamická složka spotřeby elektrické energie číslicovým obvodem je závislá mimo jiné na frekvenci a pravděpodobnosti přechodů vnitřních signálů mezi log. 1 a log. 0, viz např [12], vztah (1). Vhodným způsobem, jak tyto faktory změřit, je provést simulaci na hradlové úrovni a monitorovat, jak často dochází k překlápění logických hodnot na jednotlivých spojích v návrhu. Na základě této informace, známé topologie obvodu a dalších parametrů lze potom spočítat odhad spotřeby elektrické energie obvodem. Provedeme-li příslušné odhady na RTL úrovni, budou zatíženy značnou chybou, neboť na RTL úrovni jednak nejsou známy všechny detaily mikroarchitektury návrhu, dále nemáme popsány všechny spoje v obvodu a konečně nejsou simulovány hazardy v návrhu.
- **Částečná detekce hazardů na kritických signálech**: Na rozdíl od RTL popisu může reálná implementace kombinační logické funkce vykazovat hazardní chování na svých výstupech, více viz např. [3]. Hazardy v číslicovém systému jsou velmi nebezpečné, proniknou-li v důsledku chybného návrhu na hodinový signál nebo na asynchronní *set* či *reset* klopných obvodů. Nebezpečné mohou být i tehdy, pokud se objeví na primárních výstupech číslicového návrhu. Hradlová simulace umožňuje za vhodných podmínek odhalit problémy s hazardy na kombinačních signálech, nevhodně ošetřené hazardní chování se může projevit buď nefunkčností obvodu, nebo zaplavením celého návrhu stavem *X*.
- **Lepší pocit návrháře z návrhu**: Při simulaci na hradlové úrovni je – na rozdíl od všech statických analýz – jasně vidět, že „to funguje“. Návrhář tak může zrevidovat očima funkci běžícího obvodu a získat příjemný pocit, že se při návrhu neudělala hrubá chyba.

Při návrhu zákaznických obvodů má hradlová simulace ještě další výhody, umožňuje například

- **částečně ověřit správnost vložení struktur pro snadou testovatelnost**. Simulací na hradlové úrovni je možné částečně ověřit správnost implementace struktur pro snadnou testovatelnost (*scan chains* a další), které jsou často vkládány automaticky během syntézy. Hradlová simulace se také používá pro ověření správnosti generovaných vektorů pro produkční testování. Je-li třeba vektory pro produkční testování odladit, je hradlová simulace naprosto nezastupitelná. Konečně, je-li použit ad-hoc přístup k implementaci testovací logiky (více viz např. [13]), je doporučeno testovací vektory generovat také pomocí simulace na hradlové úrovni.
- **částečně ověřit správné vložení hodinového stromu**: Simulace na hradlové úrovni umožňuje částečně ověřit správnost syntézy hodinového stromu vkládaného během rozmístění a propojení.
- **částečně ověřit správnost vložení struktur pro snížení spotřeby energie obvodem**: Simulace na hradlové úrovni také umožňuje funkčně procvičit konstrukce vložené při syntéze pro dosažení menší spotřeby (*clock gating*, atd.).

Hradlová simulace má nicméně také nevýhody:

- **Pomalost**: Právě kvůli simulaci reálných časových zpoždění může být hradlová simulace až o řád pomalejší, než simulace stejného obvodu na RTL úrovni. Je to logické, neboť modelovaným obvodem se šíří významně více událostí, než v případě RTL simulace.

- **Neúplnost**: Přes to, že simulace na hradlové úrovni umožňuje rámcově ověřit, že jiné techniky poskytují správné výsledky, pokrytí návrhu pomocí simulace na hradlové úrovni není nikdy stoprocentní. Vždycky tedy potřebujeme minimálně statickou časovou analýzu. Na druhou stranu, ani dobře provedená statická časová analýza sama o sobě není postačující a je vhodné ji doplnit simulací na hradlové úrovni. Konečně, hradlová simulace je vždy prováděna jen pro konkrétní kombinaci parametrů výrobního procesu (v prvním přiblížení lze hovořit o pomalých, nebo rychlých tranzistorech), napájecího napětí a teploty okolního prostředí. Je tedy běžné, že je třeba simulací provádět více pro různé fyzikální podmínky (typicky pro maximální, minimální a např. typická zpoždění hradel v návrhu) a ani tak nejsou pokryty všechny možné kombinace.
- **Časově náročné a pracné odladění simulace**: Výhodou simulace na hradlové úrovni proti RTL simulaci je kontrola dodržení časových parametrů klopných obvodů. Při jejich porušení je pak příslušný výstup klopného obvodu nastaven do stavu X, který často „zaplaví" celý návrh a tak lze snadno zjistit, že někde dochází k problémům. Přes všechnu užitečnost tohoto modelování je třeba počítat s nezanedbatelnou pracností odladění takových problémů; nelze se tomu nicméně nijak vyhnout. Někdy ovšem může být porušení časových parametrů klopných obvodů součástí standardní funkce obvodu – typicky v resynchronizátorech asynchronních signálů, viz např [4]. V těchto případech se vyplatí dopředu simulaci připravit tak, aby ke generování X v takových registrech nedocházelo.
- **Nutnost odladit verifikační prostředí** (*testbench*)**:** Verifikační prostředí je třeba kvůli simulacím na hradlové úrovni obvykle upravovat. V prvé řadě, pracujeme-li technikou šedé či bílé skříňky a během verifikace se odkazujeme na vnitřní stavy signálů v návrhu, je třeba počítat s možností jejich přejmenování během syntézy. Může tak být potřeba připravit pro syntézu specifické nastavení a vynutit si zachování některých objektů (bloků a jejich signálů) a jejich jmen. Samostatnou kapitolou je práce se stavovými registry automatů, které jsou na RTL úrovni vyjádřeny abstraktním výčtovým typem, ale po syntéze dojde k jejich konverzi do konkrétní binární reprezentace a stavový registr automatu je konvertován na vícebitový registr.
- **Komplikované použití assertions:** Kvůli problémům zmíněným v předchozím odstavci se stává použití *assertions* v simulaci na hradlové úrovni poněkud komplikovaným. Používáte-li *assertions* pro verifikaci navrhovaného systému na RTL úrovni a *assertions* přímo vkládáte do RTL kódu, je třeba si uvědomit, že jsou během syntézy odstraněny. Zabránit tomu lze například jejich externí vazbou – tzv. *binding*. Ani použití externě umístěných assertions při simulaci na hradlové úrovni není triviální: vzájemné posuvy hodinových a datových signálů způsobené reálnými zpožděními datových signálů a hodin v hodinovém stromu mohou vyvolat obtíže při vzorkování signálů, dále např. odkazy na stavy stavových automatů nebudou fungovat, protože automaty mají syntézou konvertované stavy do jejich bitové reprezentace. Je tedy realistické počítat s tím, že bude třeba část *assertions* znovu na hradlové úrovni odladit.

Simulace na hradlové úrovni je účinný a mocný nástroj, který je nicméně třeba používat s ohledem na nároky kladené jak na simulační čas, tak i na čas návrhářů. U běžného projektu může být připravených řádově několik set testů, které jsou pouštěny na RTL úrovni. Ne všechny je ovšem nezbytné (a někdy ani možné) pouštět i na hradlové úrovni. Obecně se literatura shoduje v potřebě pustit takovou skupinu testů, aby byl návrh uspokojivě pokryt jak po funkční, tak i po strukturní stránce. Dále množství pouštěných testů závisí na tom, jak velkou důvěru vkládáme ve výsledky statické časové analýzy (tedy také na tom, jak důkladné jsou *constraints*) a jak moc je navrhovaný systém synchronní. U obvodu s jednou hodinovou doménou bude situace jednodušší než u návrhu s desítkami hodinových domén a řadou vnějších rozhraní asynchronních vůči jádru číslicového systému.

5 Závěr

Příspěvek shrnul základní vlastnosti, výhody a nevýhody simulace na hradlové úrovni. Simulace byla zasazena do kontextu návrhového cyklu a čtenář byl seznámen i s některými dalšími nástroji, které mohou potřebu simulací na hradlové úrovni zredukovat (ovšem ne zcela nahradit). Problematika simulací na hradlové úrovni je velice široká, text se díky svému omezenému rozsahu omezil jen na nejdůležitější body (v textu nebyly například zmíněny nástroje pro formální verifikaci ekvivalence, viz např. [14]). Čtenář je tak nabádán ke studiu další literatury, začít lze například s prameny uvedenými v použité literatuře.

6 Použitá literatura

[1] ŠŤASTNÝ, Jakub. Simulace číslicových obvodů: úvod. *DPS Elektronika od A do Z*, leden/únor 2015, s. 23-27

[2] ŠŤASTNÝ, Jakub. Simulace číslicových obvodů: triky i úskalí simulace. *DPS Elektronika od A do Z*, březen/duben 2015, s 20-23

[3] PINKER, Jiří a POUPA, Martin. Číslicové systémy a jazyk VHDL. Praha : BEN-technická literatura, 2006. 349 s.

[4] ŠŤASTNÝ, Jakub. *FPGA prakticky*. Praha : BEN-technická literatura, 2011. 200 s.

[5] CUMMINGS Clifford E., MILLS, Don a GOLSON, Steve. Asynchronous & Synchronous Reset Design Techniques - Part Deux. In: *SNUG 2003 Boston.* [vid. 18. února 2015] *Dostupné z* http://www.sunburst-design.com/papers/CummingsSNUG2003Boston_Resets.pdf

[6] IEEE 1497-2001 *IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process* [vid. 15. března 2015] Dostupné z http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=972829

[7] Lint (software) In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 3. 2. 2015 v 1:22. [vid. 15. března 2015]. Anglická verze. Dostupné z: http://en.wikipedia.org/wiki/Lint_%28software%29

[8] XILINX. Constraints Guide, dokumentace k návrhovému prostředí ISE12.2

[9] CADENCE. Gate-Level Simulation Methodology [online] [vid. 15. března 2015] Dostupné z http://www.cadence.com/rl/Resources/white_papers/Gate_Level_Simulation_WP.pdf

[10] *Gate Level Simulations : A Necessary Evil - Part 1,2, and 3* [online] [vid. 15. března 2015] Dostupné z http://whatisverification.blogspot.cz/2011/06/gate-level-simulations-necessary-evil.html

[11] KHANDELWAL A., GAUR A. a MAHAJAN D. Gate level simulations: verification flow and challenges. EDN network [online]. 5. března 2014 [vid. 15. března 2015]. Dostupné z http://www.edn.com/design/integrated-circuit-design/4429282/Gate-level-simulations--verification-flow-and-challenges.

[12] CHANDRASAKAN, Anantha P. a SHENG, Samuel a BRODERSEN Robert W. Low-Power CMOS Digital Design. IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, April 1992.

[13] NOVÁK, O., GRAMATOVÁ, E., UBAR, R. a kol. *Handbook of testing electronic systems*. Praha: Nakladatelství ČVUT, srpen 2005, 395 stran, ISBN 80-01-03318-X.

[14] Formal Equivalence Checking In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 18. 2. 2015 v 8:34. [vid. 13. března 2015]. Anglická verze. Dostupné z: http://en.wikipedia.org/wiki/Formal_equivalence_checking