PCS Projekt č. 2:

Optimalizace paketového filtru

1. Vzorová situace

Jako zaměstnanec korporátního giganta, firmy *Fakt Inovativní Technologie* (zkráceně *F.I.T.*), aktuálně pracujete na projektu *Páteřní Celonárodní Síť 2.0* (neboli *Projekt P.C.S. 2*). V rámci něj s kolegy vytváříte unikátní FPGA firmware a softwarové nástroje pro akcelerovanou síťovou kartu na zpracování 100 Gb/s Ethernetových dat. Potřebná funkcionalita firmwaru a nad ním pracujících softwarových vrstev byla v prvních krocích řešení rozdělena do několika nezávislých problémů jako jsou například: příjem a dekódování Ethernetových rámců z linky, analýza a extrakce hlaviček síťových protokolů nebo DMA přenosy dat přes PCIe sběrnici do počítače. Vám konkrétně byla přidělena úloha navržení a implementace filtru IP adres pro technologii FPGA. Ten na základě z paketu extrahované cílové IP adresy (klíče) a za běhu nakonfigurované sady pravidel vrátí identifikátor a parametry akce (data), kterou se paket má dále zpracovat. Případně filtr informuje o neexistenci/nenalezení vhodného pravidla v sadě.

Na základě konzultací s ostatními členy týmu a po upřesnění všech požadavků, jste definoval rozhraní (entitu) filtrační jednotky a API jeho konfiguračního software. Následně, jste zvolil použití algoritmu hešovacích tabulek jako základu pro uložení a vyhledávání pravidel ve firmwarovém filtru. Váš podřízený *Bc. Hardwired* pak vytvořil a pomocí verifikace funkčně odladil VHDL implementaci tohoto obvodu. Zatím vaše druhá podřízená *Bc. Softskills* implementovala a testovala konfigurační software pro nahrávání pravidel. Integrace obou dokončených implementací mezi sebou a též do celkového řešení právě proběhly. Z funkčního hlediska se zdá být všechno v pořádků, funkční integrační testy a simulace proběhly úspěšně. Problém se však objevil při syntéze a implementaci finálního firmware pro FPGA na akcelerované kartě. Nesplněno bylo časování a příčina byla identifikována právě někde ve vašem filtru.

Akcelerační karta je osazena FPGA čipem *Kintex-7 XC7K160T-1*. Pro zpracování síťových dat na plné rychlosti 100 Gb/s linky byla již v úvodu návrhu zvolena datová sběrnice o šířce 512 bitů taktovaná na hodinové frekvenci 200 MHz. I když filtr nepracuje přímo s datovou sběrnicí, počítá se, že bude pracovat na stejné hodinové frekvenci jako celý zbytek firmwaru. S využitím vašich inženýrských znalostí v oboru pokročilých číslicových systémů je teď na vás abyste objevený problém časování analyzoval a existující VHDL implementaci filtru dostatečně optimalizoval. Termín pro veřejné představení prvního funkčního prototypu karty vytvořené v *Projektu P.C.S. 2* se blíží a celý tým společnosti *F.I.T.* čeká jen na vás.

Dokážete popsaný problém s časováním dostatečně kvalitně a včas vyřešit?

„VHDL kódy by u mě na code review určitě neprošly, ale vyznal jsem se v nich v pohodě.“
– Anonymní člen týmu F.I.T. o kvalitě kódu kolegy Bc. Hardwired

---

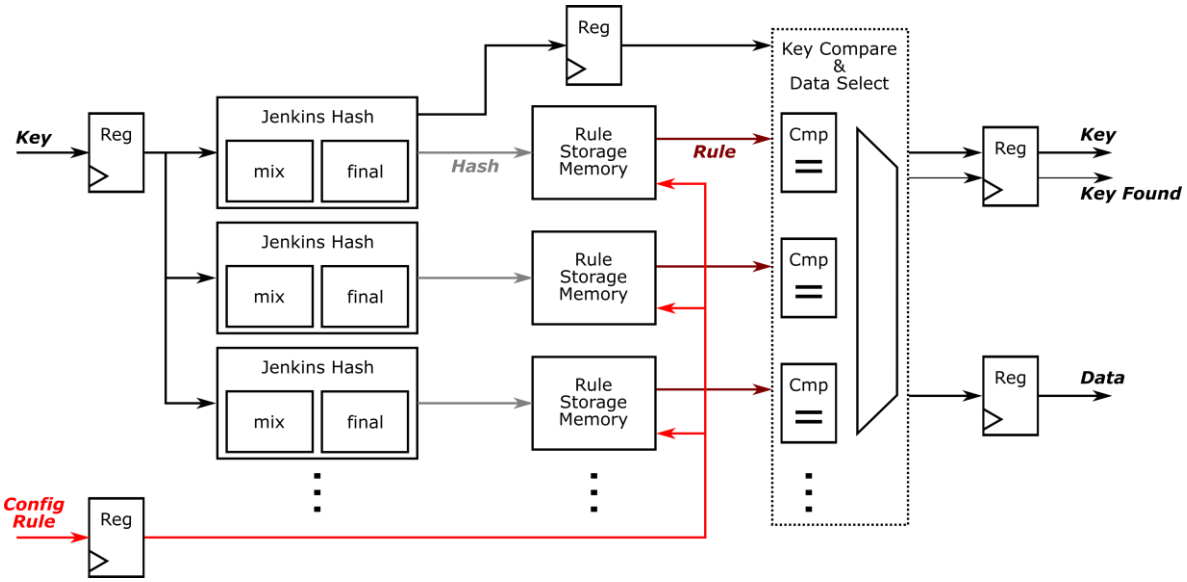
2. Architektura implementovaného filtru

Filtr umožňuje porovnávání či vyhledávání klíčů na přesnou shodu (tzv. *exact match*) a tuto funkcionalitu implementuje metodou paralelních hešovacích tabulek. Náhled na základní architekturu implementovaného filtrovacího obvodu ukazuje obrázek 1. Operace filtrování se skládá ze třech základních fází, postupně to jsou: 1) hešování klíče, 2) přístup do pamětí s nakonfigurovanými pravidly a 3) porovnání klíče s pravidly plus výběr dat. Na vstup obvodu jsou postupně přiváděny vyhledávané klíče (např. IPv6 adresy). Vstupní klíč se nejdříve zahešuje pro každou heš tabulku zvlášť použitím algoritmu *Jenkins Hash* s různými inicializačními hodnotami. Hodnota heš funkce pak v každé tabulce slouží jako adresa na vyčtení správného záznamu z paměti s uloženými pravidly. Vyčtená pravidla pro všechny hešovací tabulky jsou pak porovnána na shodu s vyhledávaným klíčem. Příznak o nalezení shody spolu s daty patřícími tomuto shodnému pravidlu jsou pak vybrány jako výstupy. V případě nenalezení shodného pravidla (*KEY_FOUND* v logické 0) je hodnota dat neplatná.

Záznamy filtračních pravidel se do pamětí filtru zapisují konfiguračním rozhraním (červená) vždy před začátkem samotného zpracování vstupních klíčů. Každé pravidlo je reprezentováno a ukládáno jako trojice hodnot: 1) klíč, 2) jemu patřící data a 3) příznak prázdného záznamu (bit *EMPTY*). Jeli příznak *EMPTY* nastaven (logická 1), hodnoty dat a klíče v záznamu nejsou platná a takovéto pravidlo nemůže být shodné se žádným klíčem.

Algoritmus *Jenkins Hash* je ve VHDL implementován podle C kódu zveřejněného přímo jeho autorem Bobem Jenkinsem. Tento kód je dostupný na https://burtleburtle.net/bob/c/lookup3.c kde jde přesně o funkci *hashword()*. Výpočetní makra *mix* a *final* použita v rámci hešovací funkce jsou definována na začátku tohoto souboru. Ve VHDL jsou pak implementovány jako samostatné výpočetní komponenty zapojené v rámci hlavní hešovací jednotky.

Implementace filtru pracuje na všech místech s hodnotami datových signálů doplněných o příznak jejich platnosti (bit *VALID*). Při neplatném příznaku (*VALID* v logické 0) mohou být hodnoty zodpovídajících datových signálů libovolné a musejí být obvodem ignorovány. Naopak platné hodnoty (*VALID* v logické 1) musejí být vždy korektně zpracována a není možné jejich zpracování nijak odmítnout. Nakonec, v průběhu aktivního resetu (*RESET* v logické 1), nesmí být žádný příznak platnosti nastaven (každý *VALID* bit musí být držen v 0).



Obrázek 1: Ilustrace zapojení filtru s více paralelními hešovacími tabulkami.

3. Zadání úlohy a postup řešení

Analyzujte existující implementaci filtru a optimalizujte jí s ohledem na dosažení co nejvyšší frekvence hodin. Pro základní splnění zadání je potřeba dostat se minimálně nad frekvenci 200 MHz. Protože informace o časování komponenty po syntéze jsou jenom odhadem, často se stává, že po jejím zapojení do většího firmware a implementaci je ve skutečnosti dosaženo nižší frekvence. Ideální je proto zvolit dostatečnou rezervu a u filtru cílit na dosaženou frekvenci po syntéze *alespoň 250 MHz*. V rámci optimalizace můžete libovolně editovat hlavní architekturu filtru (*filter.vhd*) nebo upravovat/přidávat VHDL soubory v složce *comp/*. Entitu filtru (*filter_ent.vhd*) však zachovejte původní a neměňte ani princip fungování filtru (např. změnou algoritmu nebo výměnou hešovací funkce). V případě přidání nových VHDL souborů upravte i oba níže uvedené TCL skripty tak, aby tyto soubory braly v úvahu.

Syntézu provádějte pro již dříve uvedený FPGA čip *Kintex-7 XC7K160T-1* využitím nástroje *Xilinx Vivado*. Pro pohodlnou syntézu máte již připravený TCL skript ve složce *synth/*. Ten obsahuje příkazy pro vytvoření a správné nastavení projektu v nástroji Vivado, následováno spuštěním syntézy a vygenerováním logů s dosaženým časováním (**_timing.log*) a spotřebou zdrojů na čipu (**_utilization.log*). Skript se spouští přímo z nástroje Vivado voláním příkazu `source filter.tcl`. Takovéto spuštění je možné:

- V rámci dávkového spuštění: `vivado -mode batch -source filter.tcl`.
- Při spuštění nástroje Vivado bez grafického rozhraní pomocí `vivado -mode tcl` stačí následně zadat příkaz pro spuštění skriptu přímo z příkazového řádku.
- Z grafického rozhraní nástroje, zadáním do TCL řádku (TCL Console).

Pro ověření, že vámi vykonané změny implementace neporušily správnost fungování filtru je připraveno verifikační prostředí ve složce *ver/*. Také tady je nachystán skript pro automatické spuštění. Potřebný je simulátor *ModelSim* nebo jemu obdobný nástroj. Spuštění verifikace ze skriptu se provádí příkazem `do filter.tcl`. Ten je možné zadat opět několika způsoby:

- V rámci dávkového spuštění: `vsim -c -do filter.tcl`.
- Při spuštění nástroje bez grafického rozhraní pomocí `vsim -c` stačí následně zadat příkaz pro spuštění skriptu přímo z příkazového řádku.
- Z grafického rozhraní nástroje, zadáním do TCL řádku v dolní části obrazovky.

Pro řešení je možné použít vlastní počítač s nainstalovanými nástroji Vivado a ModelSim.
Kromě toho je také možné využít libovolný počítač v CVT. Nastavení a spuštění nástrojů se zde provádí tak, jak bylo ukázáno na prvním cvičení:

```
Q:\fitkit\setup.bat
Q:\fitkit\Vivado2018\Vivado\2018.2\bin\vivado.bat
vsim
```

Nakonec je ze sítě VUT ještě dostupný také fakultní server *fitkit-build.fit.vutbr.cz*. Pamatujte však, že to není stroj s neomezeným výkonem a množstvím paměti, aby mohl spouštět syntézy vícero studentů zároveň. Nastavení a spuštění nástrojů se zde provádí:

```
. /mnt/data/tools/config
. /mnt/data/tools/config-vivado
vivado
vsim
```

4. Výstupy projektu

Výstupem projektu bude:

1. Všechny upravené nebo přidané soubory **.vhd* se zdrojovými kódy a případně též upravené skripty **.tcl*. Odevzdejte pouze ty soubory, ve kterých jste provedli nějaké změny oproti zadání. Nezapomeňte, že hlavní entita filtru (soubor *filter_ent.vhd*) je jediným VHDL souborem, který se nesmí upravovat, a tudíž ani odevzdávat.
2. Soubor *zprava.pdf* se stručnou výstupní zprávou ve formátu PDF, která bude obsahovat zejména následující informace:
   – Jméno, příjmení a login autora řešení
   – Popsaný seznam provedených změn ve zdrojových VHDL souborech. Pro každý odevzdaný soubor jen *stručně* popište, co všechno a proč jste v něm museli změnit.
   – Výsledky syntézy filtru zejména z pohledu: dosažené frekvence, latence filtrování a spotřebovaných zdrojů na čipu (logika, registry, paměti). Pro každou položku uveďte kromě absolutní hodnoty též změnu proti implementaci ze zadání.

   Rozsah zprávy by neměl překročit jednu stranu formátu A4. Ukázku formátu výstupní zprávy naleznete v příloze č. 1 na konci tohoto dokumentu.

Všechny odevzdávané soubory zabalte do ZIP archivu s názvem *<login>.zip* a zachovejte původní adresářovou strukturu zdrojových souborů ze zadání.

Před odevzdáním do informačního systému si ve vlastním zájmu tento archiv otestujte. Ideální je stažení a rozbalení čerstvého archivu se zadáním následováno rozbalením Vašeho odevzdávaného archivu do stejné složky s přepisem souborů. V takto připravené složce pak ověřte bezchybnost syntézy obvodu spuštěním skriptu *synth/fiter.tcl* a správnost fungování filtrace ve verifikaci pomocí skriptu *ver/filter.tcl*.

Otestovaný archiv odevzdejte prostřednictvím informačního systému nejpozději do data uvedeného na stránkách předmětu PCS. Pozdější odevzdání projektu nebude bráno v úvahu.

Po zkušenostech z minulých let připomínám, že podle Směrnice děkana FIT doplňující Studijní a zkušební řád VUT (Rozhodnutí děkana FIT č. 34/2010), K článku 11, odst. 4:
„Veškeré testy, projekty a další hodnocené úlohy vypracovává student samostatně, pokud projekt nebo úloha nebyly výslovně zadány pro stanovenou skupinu studentů.“

Příloha č. 1: Zpráva k Projektu P.C.S. 2 (ukázka)

Vypracoval: Jmenovič Příjmeněvič (xlogin00)

Provedené změny

Formát:

- [kde?]
  - [co?] [proč?]
  - [co?] [proč?]

Příklad:

- Jako první jsem upravil soubor *comp/block_memory.vhd*:
  - Přidána byla podpora volitelného registru na datech čtecího portu paměti. Použití registru je ovládáno generickým parametrem v entitě a samotná implementace registru byla přidána na konec souboru. Použití registru umožní dosažení lepšího časování.
- Následně jsem změnil soubor *filter.vhd*:
  - V instancích blokových pamětí byla zohledněna změna jejich entity popsána výše. Funkce obvodu filtru byla zatím ponechána původní, tedy s pamětmi bez registru. Do budoucna však již byl přichystán (zatím zakomentován) další stupeň synchronizačního registru pro klíč. Povolení popsaného čtecího registru v pamětech zvýší jejich latenci o jeden cyklus hodin, co bude potřeba kompenzovat i pro klíč.

Výsledky syntézy

Formát:

	celkově	změna
frekvence	?	± ?
latence	?	± ?
zdroje	?	± ?

Příklad:

		Celkově	**Změna**
Frekvence [MHz]		234	+ 56
Latence [cyklů]		3	+ 0
Zdroje	**LUT**	2345	- 67
	Registr	1234	- 56
	BRAM	34,5	- 4,5

Popsané změny vedly zejména na zvýšení dosažitelné frekvence přes hranici 200 MHz. Došlo též k nepatrnému snížení spotřebovaných zdrojů na čipu.