VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

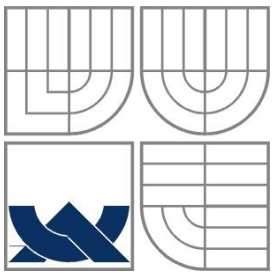
AKCELERACE ALGORITMŮ PRO POROVNÁVÁNÍ ŘETĚZCŮ NA ZÁKLADĚ PODOBNOSTI

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

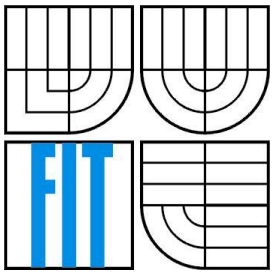
AUTOR PRÁCE
AUTHOR

JAN VOŽENÍLEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

AKCELERACE ALGORTIMŮ PRO POROVNÁVÁNÍ ŘETĚZCŮ NA ZÁKLADĚ PODOBNOSTI

ACCELERATION OF ALGORITHMS FOR APPROXIMATE STRING MATCHING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAN VOŽENÍLEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. TOMÁŠ MARTÍNEK

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Voženílek Jan**
Obor: Informační technologie
Téma: **Akcelerace algoritmů pro porovnání řetězců na základě podobnosti**
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se s technologií programovatelných hradlových polí FPGA a dostupnými nástroji pro syntézu a implementaci obvodů do FPGA hradlových polí.
2. Seznamte se s algoritmy pro hledání podobnosti dvou řetězců. Mezi tyto algoritmy patří například algoritmus Smith-Waterman nebo Needleman-Wunsch.
3. Na základě předchozích dvou bodů navrhněte vhodnou hardwarovou architekturu pro urychlení algoritmu hledání podobnosti dvou biologických sekvencí. Při návrhu uvažujte paralelizaci algoritmu s využitím několika výpočetních polí a způsob jejich zásobování daty.
4. Proveďte implementaci navrženého řešení v jazyce VHDL nebo HandelC a jeho funkci ověřte simulací. Pro různé typy aplikací ověřte celkovou propustnost systému a zrychlení oproti softwarovému řešení.
5. V závěru diskutujte vlastnosti Vaší implementace a možnosti dalšího pokračování projektu.

Literatura:

- D. Krame, M. Raymer: Fundamental Concepts of Bioinformatics, ISBN 0-8053-4722-4, 2003

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Martínek Tomáš, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

Kotásek

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jan Voženílek**
Id studenta: 79277
Bytem: Osvobození 677/52, 682 01 Vyškov
Narozen: 11. 10. 1985, Vyškov
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP:	Akcelerace algoritmů pro porovnání řetězců na základě podobnosti
Vedoucí/školitel VŠKP:	Martínek Tomáš, Ing.
Ústav:	Ústav počítačových systémů
Datum obhajoby VŠKP:	.....

VŠKP odevzdal autor nabyvateli v:

tištěné formě	počet exemplářů: 1
elektronické formě	počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Cílem této bakalářské práce je návrh a implementace architektury pro FPGA čipy akcelerující porovnávání dvou řetězců a jejich ohodnocení na podobnost. Použité postupy vycházejí z bioinformatických algoritmů, především Needleman-Wunsch a Smith-Waterman. Jednotka může díky obecnému návrhu a generickému zpracování v jazyce VHDL porovnávat libovolné sekvence znaků, což je úloha prostupující mnoha oblastmi informatiky od prohledávání databází (kde porovnání na podobnost umožňuje odhalit překlepy) po detekci nevyžádané elektronické pošty – spamu. V závislosti na specifikaci úlohy se může zrychlení oproti běžnému softwarovému řešení pohybovat v řádu stovek až tisíců.

Klíčová slova

FPGA, VHDL, akcelerace, porovnání řetězců na základě podobnosti

Abstract

The objective of this bachelor's thesis is to design and implement architecture for FPGA chips that accelerates matching of two strings and scoring them for similarity. Used processes come from bioinformatics algorithms, especially Needleman-Wunsch and Smith-Waterman. Due to general design and generic implementation in VHDL the unit is able to compare any sequences of characters, which is a task widely used in many branches of informatics from database searches (where approximate matching allows discovery of spelling errors) to spam detection. Depending on task specification the acceleration speed up against common software solution can reach orders of hundreds or even thousands.

Keywords

FPGA, VHDL, acceleration, approximate string matching

Citace

Voženílek Jan: Akcelerace algoritmů pro porovnávání řetězců na základě podobnosti. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Akcelerace algoritmů pro porovnávání řetězců na základě podobnosti

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Martínka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Voženílek
12. 5. 2008

Poděkování

Děkuji Tomáši Martínkovi za vedení práce a řešitelům projektu Liberouter za odbornou pomoc.

Obsah

Úvod

Porovnávání řetězců na základě podobnosti (angl. approximate string matching) je operace zasahující do mnoha oblastí informatiky. Její využití začíná u nejjednodušších forem v podobě internetových vyhledávačů a databází, kde umožňuje detekci překlepů či vyhledání slov s různým pravopisem, a pokračuje až k vysoce specializovaným postupům hledajícím podobné sekvence v DNA nebo bílkovinách. Právě pro tyto netriviální metody byla vyvinuta řada postupů, které při porovnávání zohledňují celou škálu aspektů vyplývající z podstaty problému. Vedle algoritmů samotných se rozvíjelo i jejich optimalizování a urychlování. Vynikajících výsledků v oblasti hardwarové akcelerace bylo dosaženo především při využití programovatelných FPGA čipů, které díky paralelizaci umožňují výrazně zkrátit celý proces výpočtu. V případě zpracování dvou řetězců lidské DNA se totiž jedná o potřebném času v řádu dnů. Popisem jedné z možných hardwarových akcelerací vybraných algoritmů pro porovnávání řetězců se zabývá tato práce.

Při návrhu akcelerační jednotky je nutné důkladně se seznámit s algoritmy, jejichž běh má být urychlován. V rámci této analýzy se objevují požadavky týkající se schopnosti detekce záměny znaku, jeho vypuštění nebo naopak vložení, porovnání dvou řetězců o výrazně rozdílných délkách a další. Rozborem algoritmů Needleman-Wunsch a Smith-Waterman, které řeší nastíněné problémy pomocí skórovacích funkcí a rozlišování globálního a lokálního zarovnání, se zabývá první kapitola.

Akcelerace těchto algoritmů má za sebou jistý vývoj. Jeho důležité body jsou shrnuty v druhé kapitole spolu s obecným konceptem hardwarového řešení porovnávání řetězců použitého při vývoji popisovaného systému.

Po nastudování existujících algoritmů a seznámení se s existujícími hardwarovými řešeními je možno přejít k návrhu vlastní akcelerační jednotky a její implementaci. Popis návrhu i výsledného řešení, a to jak z pohledu globálního, tak z hlediska komponent systému, je náplní třetí kapitoly.

S hotovým akcelerátorem byla provedena řada testů zaměřená jednak na jeho výpočetní výkonnost a také technické parametry jako je využití zdrojů FPGA čipu. Výsledky jsou shrnuty v kapitole čtvrté.

V závěru je zhodnoceno splnění vytyčených cílů a možnosti dalšího využití tohoto nástroje, které přesahuje rámec bioinformatiky, na jejichž poznatcích byl systém vyvinut a postaven.

1 Rozbor algoritmů

Řetězcem se obecně rozumí posloupnost prvků z definované množiny nazývané abeceda. Mocnost této množiny, tedy počet různých znaků, které se mohou v řetězci vyskytnout, bývá různá podle specifikace úlohy. Anglická literární abeceda rozlišuje 26 znaků, česká znaková sada pak obsahuje 42 různých písmen (dlužno podotknout, že velká a malá písmena jsou v této souvislosti považována za totožná). Z nich lze pak složit řetězec – slovo, v širším pojmu, a s rozšířením abecedy o mezeru, interpunkci a další znaky, to může být věta a veškerý text.

Kromě abeced přirozených se používají i abecedy uměle vytvořené. Ty se často snaží o zjednodušení, a proto bývá jejich mocnost malá. Příkladem za všechny může být abeceda Morseova, která si vystačí pouze se dvěma znaky – tečkou a čárkou.

V informatice se literární abeceda uchovává pomocí různých kódování, ve kterých je každému znaku přiřazena binární hodnota. Konkrétní přiřazení většinou záleží na tvůrci šifry, a proto jich existuje větší množství. Při vývoji algoritmu však použitá znaková sada nemá zásadní vliv, jelikož do systému vstupuje právě v podobě binárních kódů a zpětné převedení na řetězec je pouze otázkou reprezentace dat.

1.1 Řetězec z hlediska bioinformatického

Oba zde analyzované algoritmy se zabývají hledáním podobností v biologických řetězcích. Těmi se rozumí nejčastěji posloupnost dusíkatých bází v DNA sekvencích. Šroubovice deoxyribonukleové kyseliny obsahuje dvě vlákna, která jsou tvořena pětičlennými cukry spojenými fosfáty (zbytky kyseliny fosforečné), na něž jsou navázány nukleové báze. Právě tyto heterocyklické sloučeniny, respektive jejich posloupnost, kódují genetickou informaci a jsou předmětem hledání podobnosti. V DNA se vyskytují čtyři různé báze: od purinu odvozené adenin a guanin a z pyrimidinu vzniklé cytosin a thymin. Běžný je zápis vlákna jako sekvence počátečních písmen názvů bází – AGCT. Z chemických a fyzikálních vlastností následně vyplývá, že oba řetězce jsou spojeny právě přes nukleové báze pomocí vodíkových můstků, přičemž adenin tvoří komplementární bázi s thyminem a cytosin s guaninem. Z těchto skutečnosti lze odvodit dva důležité fakty: pro porovnání je postačující pouze jedna sekvence – druhou lze vždy odvodit pomocí komplementárních párů. Dále se objevuje požadavek na možnost ohodnocení záměny znaku v závislosti na substituentu (některé záměny jsou chemicky a fyzikálně přijatelnější než jiné).

V některých specifických případech se porovnávají sekvence RNA, které se od DNA liší přítomností pouze jednoho vlákna a výskytem nukleové báze uracilu namísto velmi podobného thyminu (abeceda se mírně pozmění na AGCU). Druhými zkoumanými řetězci jsou posloupnosti aminokyselin (existuje 20 základních), které jednoznačně identifikují bílkovinu (protein).

1.2 Zarovnání řetězců

Na každém místě v DNA řetězci může dojít ke třem základním změnám neboli mutacím. Těmi jsou záměna báze za jinou (přičemž u některých úloh můžeme chtít rozlišovat substituci tranzicí – výměnu za stejný typ báze – a transverzí – nahrazení pyrimidinové báze purinovou a naopak), inzerce (vložení jednoho či několika prvků) a delece (proces opačný – tedy vyjmutí libovolného počtu znaků). Jelikož záměna báze v DNA sekvenci je v přírodě mnohem častější než zbývající dvě možnosti (znamenající ve výsledku výskyt mezery v jednom z řetězců), pracují základní metody zarovnání s oběma sekvencemi jako s nedělitelnými.

1.2.1 Jednoduché zarovnání

V nejjednodušším případě je hledání vhodného zarovnání pouze otázkou nalezení počtu znaků z delší sekvence, které mají být vynechány před započetím samotného porovnávání. Tato metoda většinou umožňuje vyhodnocení všech možných variant zarovnání. To se provádí jednoduchým výpočtem za použití dvou konstant – „bonusu“ za každý shodný pár zarovnaných znaků a „pokuty“ za pár rozdílný – jejichž suma dává výsledné skóre. Ve schématu 1.1 je příklad ohodnocení sekvencí CAAGTCATG a CACTTC, pro bonus 1 a pokutu 0 by výsledné skóre činilo 3, 2, 0 a 2 (v pořadí zleva doprava).

```
CAAGTCATG       CAAGTCATG       CAAGTCATG       CAAGTCATG
CACTTC---       -CACTTC--       --CACTTC-       ---CACTTC
```

Schéma 1.1: Čtyři možná jednoduchá zarovnání dvou krátkých sekvencí.

1.2.2 Vkládání mezer

Zavedením možnosti vložení či vyjmutí znaku na libovolné pozici v řetězci velmi rapidně narůstá počet možných zarovnání. Jestliže pro sekvence použité v předchozím příkladu jednoduchého zarovnání existovaly čtyři možnosti jak spárovat znaky, pak zavedením tří mezer na libovolná místa v kratší sekvenci se toto číslo zvyšuje na 28. Ve schématu 1.3 jsou uvedeny pouze čtyři varianty. Zohlednit při vyhodnocování podobnosti vyjmuté či přidané znaky se provádí zavedením další „pokutující“ konstanty – tentokrát za vložení mezery. Do výsledku se započítá v případě, že v jednom z řetězců se na zkoumané pozici nachází mezera. Rozšířením dříve uvedených konstant o právě odvozenou (o hodnotě kupříkladu -1) vzniká vztah pro výpočet ohodnocení podobnosti řetězců (tzv. skórovaní funkce uvedená ve schématu 1.2), pomocí nějž vychází pro příklady ve schématu 1.3 skóre 1, 0, 1 a 1 (opět zleva doprava). Druhá varianta se tedy jeví jako nejméně pravděpodobná mutace ze zkoumaných možností při porovnání na základě uvedených kritérií.

$$\sum_{i=1}^{n} \begin{cases} \textit{pokuta za mezeru; pokud } S1_i = \text{"–"} \textit{ nebo } S2_i = \text{"–"} \\ \textit{bonus za shodu; pokud } S1_i = S2_i \textit{ (nejde o mezery)} \\ \textit{pokuta za odlišnost; pokud } S1_i \neq S2_i \textit{ (nejde o mezery)} \end{cases}$$

Schéma 1.2: Vztah pro výpočet ohodnocení podobnosti dvou sekvencí.

V úvahách o vkládání mezer lze jít ještě dál. Statisticky je totiž mnohem častější jediné vložení či vyjmutí souvislé sekvence znaků, než vícenásobné kratší mutace. Tuto skutečnost lze promítnout do hodnocení rozdělením pokuty za vložení mezery na dvě části: za počáteční mezeru (započetí nové podsekvence mezer v jednom ze zarovnávaných řetězců) a za trvající mezeru (prodloužení již existující série). Pokud použijeme tyto konstanty (o hodnotách např. -2 a -1) na náš příklad (schéma 1.3), změní se výsledné skóre na -1, -3, -1 a 0. Zavedením tohoto kritéria dosáhlo poslední zarovnání nejvyššího skóre, zatímco před rozdělením pokuty za mezeru nebylo možné z hodnot jednoznačně určit nejlepší variantu. Důvodem je právě přítomnost jediné delší mezery (chybějící sekvence znaků) namísto více kratších.

```
CAAGTCATG        CAAGTCATG        CAAGTCATG        CAAGTCATG
CAC-T--TC        C-A-C-TTC        C-A--CTTC        CA---CTTC
```

Schéma 1.3: Čtyři z variant zarovnání dvou krátkých sekvencí s možností vložení mezer.

V této práci by však uvedený přístup znamenal komplikaci, jelikož je nutné „pamatovat si“ předchozí stav. Bude tedy použita jednotná pokuta za mezeru.

1.2.3 Globální a lokální zarovnání

Kromě zkoumání, zda jsou mezery vloženy bezprostředně za sebou nebo izolovaně, se lze zajímat také o to, zda se vyskytují uvnitř nebo vně řetězce. Praktické je to zejména v případech, kdy se délky zkoumaných sekvencí podstatně liší a navíc je kratší z nich velmi podobná některému úseku té delší. Například při porovnávání CAGCTACGAC s CTACG se jako nejoptimálnější varianta jeví vložení tří mezer na začátek a dvou na konec kratšího řetězce, jelikož poté všechny jeho znaky tvoří shodné dvojice s nukleotidy delší sekvence (první příklad ve schématu 1.4). Pokutováním těchto okrajových mezer se citelně sníží výsledné skóre. U některých typů úloh tedy nejsou nepenalizovány mezery na začátku a konci řetězců. Tímto se od globálního zarovnání přechází k pologlobálnímu (též semiglobálnímu).

Ovšem ani semiglobální zarovnání nemusí být pro potřeby porovnávání postačující. Pokud bude například dána úloha nalézt v dlouhém řetězci lidské DNA libovolnou část, která je podobná některé sekvenci genomu šimpanze, pologlobální zarovnání vyprodukuje velké množství pokut za rozdílné znaky a odhalení zajímavých úseků bude velmi obtížné. Zde se používá tzv. lokální

zarovnání, které přesně odpovídá požadavkům uvedeného příkladu. Jeho cílem je nalézt nejpodobnější podsekvence z obou porovnávaných řetězců, dochází tedy vlastně k jejich oříznutí. Optimální semiglobální a lokální zarovnání dvou stejných vstupů (AATGCAGATTC a TGACCAGA) je pro ilustraci uvedeno ve schématu 1.4. Obě možnosti sice dosáhnou stejného skóre (4), ovšem pouze druhý postup odhalí naprosto totožné části obou sekvencí. Způsob, jak požadované zarovnání ovlivňuje postup výpočtu, bude rozebrán přímo u jednotlivých algoritmů.

```
CAGCTACGAC          AAT-GCAGATTC          CAGA
---CTACG--          -TGACCAGA---          CAGA
```

Schéma 1.4: Příklady semiglobálního a lokálního zarovnání.

1.3 Skórovací matice

Stejně jako lze rozdělit pokutu za vložení mezery tak, aby statisticky pravděpodobnější varianta dosáhla vyššího ohodnocení, je možno upravit i pokutu za rozdílné znaky. Zde se využívá i statistika, která nám říká, jaké záměny jsou v reálném světě častější, spolu s chemickými a fyzikálními vlastnostmi stavebních prvků řetězce. V případě DNA sekvencí lze při neshodě znaků zkoumat, zda je substituent stejného typu jako původní znak (tedy jestli je báze odvozena od purinu nebo pyrimidinu – viz substituce tranzicí versus transverzí), u řetězců aminokyselin kódujících bílkoviny jsou pak rozhodujícími aspekty především velikost a vlastnosti nových molekul a jejich vliv na funkci výsledného proteinu.

Matice identity

	A	T	C	G
A	1	0	0	0
T	0	1	0	0
C	0	0	1	0
G	0	0	0	1

Matice tranzice a transverze

	A	T	C	G
A	1	-5	-5	-1
T	-5	1	-1	-5
C	-5	-1	1	-5
G	-1	-5	-5	1

BLOSUM

	A	C	D	E
A	4	0	-2	-1
C	0	9	-3	-4
D	-2	-3	6	2
E	-1	-4	2	5

Schéma 1.5: Příklady skórovacích matic.

Skóre pro všechny existující dvojice znaků lze přehledně zapsat do tabulky – dvojrozměrné matice (schéma 1.5). Hodnoty se liší pro různé typy úloh v závislosti na jejich specifikacích. Nejjednodušší variantou je matice identity, která ohodnocuje shodu znaků kladným skóre 1, jinak vrací nulu. Jde v podstatě o hodnotící matici používanou v předcházejících příkladech. Promítnout do matice tranzici a transverzi lze provést zavedením dvou různých pokut, jak ukazuje druhý příklad. Odvozování matic se provádí mnoha různými způsoby a v některých případech (týká se to zejména

ohodnocování dvojic aminokyselin) může být i při neshodě znaků přiřazeno skóre kladné. Mezi v dnešní době nejpoužívanější skórovací matice pro proteiny patří BLOSUM, jejíž část je pro ilustraci zobrazena vedle dříve zmiňovaných variant (písmena jsou dohodnuté značky pro jednotlivé aminokyseliny).

Z pohledu této práce je skórovací matice obtížně použitelná (především pro početnější abecedy) z důvodu indexování dvojicí znaků, které by v hardwaru vyžadovalo nemalé množství zdrojů a mohlo by způsobovat další problémy. Postačující tedy bude použití dvou konstant – bonusu za shodu a pokuty za rozdílnost znaků.

1.4 Dynamické programování

V předchozím textu byla odvozena komplexní metodologie pro ohodnocování dvou řetězců. Dále je nutné nalézt jejich nejoptimálnější zarovnání, kterých může být i více. Prohledávání všech možností se stává neúnosným již při velmi krátkých řetězcích. Pro sekvence obsahující kolem stovky znaků a lišící se délkou jen o několik prvků jde o řádově desítky milionů možných zarovnání, což ani za pomoci počítače nelze zpracovat v přijatelném čase.

Řešení se nabízí v podobě tzv. dynamického programování, jehož principem je rozdělení problému na menší podproblémy, které již lze vyřešit snadněji, a následné nalezení výsledku úlohy pomocí dílčích výsledků. Tento postup při porovnávání biologických sekvencí poprvé použili Saul Needleman a Christian Wunsch v roce 1970.

1.4.1 Algoritmus Needleman-Wunsch

Při studiu tohoto algoritmu je důležité pochopit rozklad úlohy na podproblémy. Ten lze dobře ukázat na dvou krátkých sekvencích, např. GTACT a GAT. První zarovnaná dvojice může být utvořena třemi způsoby: 1) vložením mezery do prvního řetězce (zde se jeví jako nevhodné, jelikož se jedná o delší z obou sekvencí), 2) vložením mezery do druhého řetězce, nebo 3) zarovnáním znaků z obou sekvencí. V prvních dvou případech bude skóre pro první dvojici rovno pokutě za mezeru, v tom posledním pak bonusu za shodu znaků (porovnáváme dvě „G“). Zbytek skóre bude vždy záviset na tom, jak zarovnáme zbývající části řetězců.

Tato otázka (jak bude vypadat zarovnání zatím nezpracovaných znaků) by se při postupném vyšetřování různých možností objevila ještě mnohokrát. Metody dynamického programování proto uchovávají částečná skóre pro porovnání v tabulce a předchází tak opakovanému vyhodnocování stejných vstupů. Po inicializaci obsahuje tabulka v prvním řádku a sloupci násobky pokuty za mezeru (pro jednoduchost uvažujme -1) počínaje od 0 (ta je na pozici [1;1]), jak ukazuje schéma 1.6. Zarovnání dvou sekvencí je cestou z levého horního do pravého dolního rohu tabulky s tím, že horizontální pohyb (po řádku) představuje vložení mezery do řetězce u levého okraje, vertikální

pohyb (v sloupci) reprezentuje mezeru v sekvenci nad tabulkou a pohyb po diagonále je ekvivalentem zarovnání příslušných nukleotidů z obou sekvencí k sobě.

Vyplňování jednotlivých buněk tabulky vychází z dříve uvedených tří možností: 1) vložení mezery do delšího řetězce (podél levé osy) se provede přičtením pokuty za mezeru k hodnotě vlevo od vyplňovaného pole, 2) analogicky přičtení pokuty k hodnotě nad aktuální pozicí reprezentuje mezeru v sekvenci nad tabulkou, a konečně 3) lze vzít hodnotu po diagonále vlevo nahoře a podle příslušných nukleotidů na osách přidat bonus za shodu či pokutu za rozdílnost znaků. Jelikož cílem procesu je nalézt optimální zarovnání, je do buňky v tabulce zapsána vždy nejvyšší z uvedených hodnot (ve schématu 1.6 je použit bonus za shodu 1, pokuta za rozdílnost 0 a pokuta za mezeru -1). Důležitým poznatkem je, že k vyplnění buňky v tabulce je potřeba znát všechny tři uvedené hodnoty.

		G	T	A	C	T
	0	-1	-2	-3	-4	-5
G	-1					
T	-2					
C	-3					
T	-4					
A	-5					
G	-6					
T	-7					

		G	T	A	C	T
	0	-1	-2	-3	-4	-5
G	-1	1	0	-1	-2	-3
T	-2	0	2	1	0	-1
C	-3	-1	1	2	2	1
T	-4	-2	0	1	2	3
A	-5	-3	-1	1	1	2
G	-6	-4	-2	0	1	1
T	-7	-5	-3	-1	0	2

		G	T	A	C	T
	0	-1	-2	-3	-4	-5
G	-1	1	0	-1	-2	-3
T	-2	0	2	1	0	-1
C	-3	-1	1	2	2	1
T	-4	-2	0	1	2	3
A	-5	-3	-1	1	1	2
G	-6	-4	-2	0	1	1
T	-7	-5	-3	-1	0	2

Schéma 1.6: Tabulka částečných skóre po inicializaci, vyplnění a rekonstrukci zarovnání řetězců.

Výsledné skóre pro nejlépe ohodnocené zarovnání zkoumaných řetězců se nachází v pravém dolním rohu tabulky vyplněné výše uvedeným postupem. Nyní je žádoucí zpětně zjistit, jak optimální zarovnání vypadá (případně vypadají). Rekonstrukce se provádí od celkového skóre pro obě sekvence, a to tím způsobem, že z aktuální buňky lze provést krok zpět na takovou, ze které mohlo vzejít skóre pro momentální pozici. Názorně je to patrné z příkladu: výsledné skóre (buňka vpravo dole [6;8]) mohlo vzniknout jedině přičtením bonusu za shodu k hodnotě na diagonále (na příslušných osách se nacházejí totožné znaky). Stejným způsobem skončí rozhodování i u dvou následujících buněk. Hodnoty 0 na pozici [3;5] už lze dosáhnout jak opět po diagonále (červená šipka), tak přičtením pokuty za mezeru ke skóre v buňce nad ní (modrá šipka). Opakováním tohoto postupu vznikají dvě různé cesty do levého horního rohu tabulky.

Pro převod sekvence přechodů na zarovnání řetězců je potřeba vrátit se k úvaze o cestě tabulkou. Rekonstrukce „modré“ varianty začíná opět od konce, a to zarovnáním posledních tří nukleotidů z obou sekvencí (zarovnání je reprezentováno diagonálním pohybem). Následuje vložení dvou mezer do kratšího řetězce (a jejich spárování se dvěma znaky z toho delšího), což v tabulce

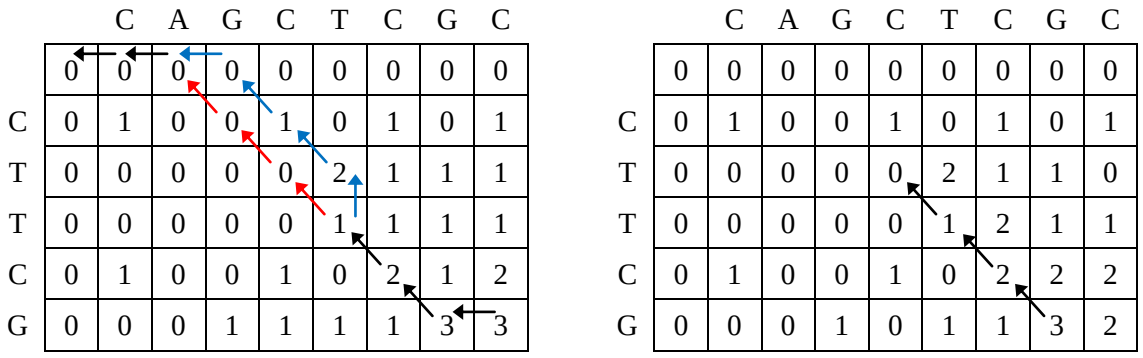
znázorňují dvě šipky nahoru. Zbývající cesta je opět diagonální. Tímto způsobem lze zjistit všechna (v tomto případě dvě uvedená ve schématu 1.7) optimální zarovnáním zkoumaných sekvencí, která dosahují nejvyššího skóre pro zvolené konstanty (pokuty).

```
GTCTAGT        GTCTAGT
G--TACT        GT--ACT
```

Schéma 1.7: Výsledná optimální zarovnání pro červenou (vlevo) a modrou variantu.

Za použití metod dynamického programování je tedy možno vypočítat jak skóre pro optimální zarovnání řetězců, tak zjistit všechna zarovnání, která tohoto ohodnocení dosahují, a to bez nutnosti zkoumat všechny možnosti.

Uvedený algoritmus pracuje s jednotnou pokutou za mezeru, může využívat skórovací matici a vyšetřuje globální zarovnání. Upravit jej pro potřeby pologlobálního zarovnání není obtížné. Technicky je vložení mezery reprezentováno horizontálním či vertikálním pohybem v tabulce (podle toho, do kterého řetězce mezeru vkládáme). Pro zrušení penalizace mezery na začátku a konci sekvence stačí pozměnit dva kroky algoritmu: 1) při inicializaci není první řádek a sloupec vyplněn násobky pokut za mezeru, ale nulami (tedy žádná pokuta) a podobně 2) v posledním řádku a sloupci není započítána pokuta za mezeru při horizontálním respektive vertikálním pohybu. Touto úpravou vzniká algoritmus Needleman-Wunsch, který hledá semiglobální zarovnání dvou sekvencí. Ve schématu 1.8 je použito konstant -1, 0 a 1 jako ohodnocení mezery, rozdílnosti a shody znaků v uvedeném pořadí.




```
CAGCTCGC        CAGCT-CGC        TCG
--CTTCG-        ---CTTCG-        TCG
```

Schéma 1.8: Tabulka a zarovnání sekvencí pro semiglobální (vlevo) a lokální zarovnání.

1.4.2 Algoritmus Smith-Waterman

Dalším logickým krokem je úprava prezentovaného postupu tak, aby vyšetřoval podobnost řetězců na lokální zarovnání. To se provede rozšířením algoritmu pro nalezení globálního zarovnání o specifikum zarovnání lokálního – zanedbání rozdílných znaků před a za podobným úsekem sekvencí. Toho lze dosáhnout přidáním čtvrté možnosti při vyplňování buňky v tabulce. Pokud je výsledek všech tří stávajících alternativ (skóre shora nebo zleva zvětšené o pokutu za mezeru a diagonální hodnota s bonusem za shodu či pokutou za rozdílnost znaků) záporný, není vybrána žádná z nich, nýbrž je zapsána nula. To se týká i inicializačních hodnot – nebudou to tedy násobky pokuty za vložení mezery, ale samé nuly (podobně jako u semiglobálního zarovnání).

Při lokálním zarovnání se k sobě přiřazují pouze části sekvencí, je proto potřeba upravit i zpětnou rekonstrukci zarovnání. Cesta nebude vycházet dogmaticky z pravého dolního rohu, ale od nejvyššího dosaženého částečného skóre v celé tabulce. Z této pozice pokračuje již známým způsobem tak dlouho, dokud není dosaženo buňky s hodnotou nula (oproti předešlému opakování až do levého horního rohu). Zde proces končí a lze sestrojit optimální zarovnání dvou subsekvencí porovnávaných řetězců (ve schématu 1.8 jsou opět použity pokuty -1 za mezeru, 0 za rozdílnost znaků a bonus 1 za jejich shodu). Výsledný algoritmus, který zaznamenává velké úspěchy při práci s velmi dlouhými řetězci a stal se jedním z pilířů bioinformatiky, představili v roce 1981 Temple Smith a Michael Waterman.

2 Hardwarové řešení

Z hlediska počítačového programu spočívá vyplňování tabulky parciálních skóre (ať již pro potřeby globálního, semiglobálního či lokálního zarovnání) v postupném výpočtu všech jejích buněk. Běžně rozšířené a používané jednoprocesorové počítače mohou v jednom kroku „vyplnit“ právě jedno pole tabulky, která může nabývat velkých rozměrů (víceprocesorové sestavy pochopitelně mohou vyhodnocovat souběžně větší množství vstupů, je ale nutné přizpůsobit program jejich architektuře – přepsáním na vlákna apod.).

Zde se nabízí použití specializovaného hardwaru s jeho obrovskou výhodou – paralelizmem. Díky zřetězenému zpracování lze podstatně zkrátit dobu výpočtu. Jelikož k určení jednoho parciálního skóre D je zapotřebí znát tři hodnoty (vlevo, nahoře a nazpět po hlavní diagonále od aktuální pozice) A, B a C, tak výpočty hodnot v jednotlivých sloupcích tabulky se mohou spouštět postupně od prvního. V jednom kroku lze tedy vyhodnotit celou vedlejší diagonálu tabulky počínaje od levého horního rohu, což naznačují červené linie ve schématu 2.1.

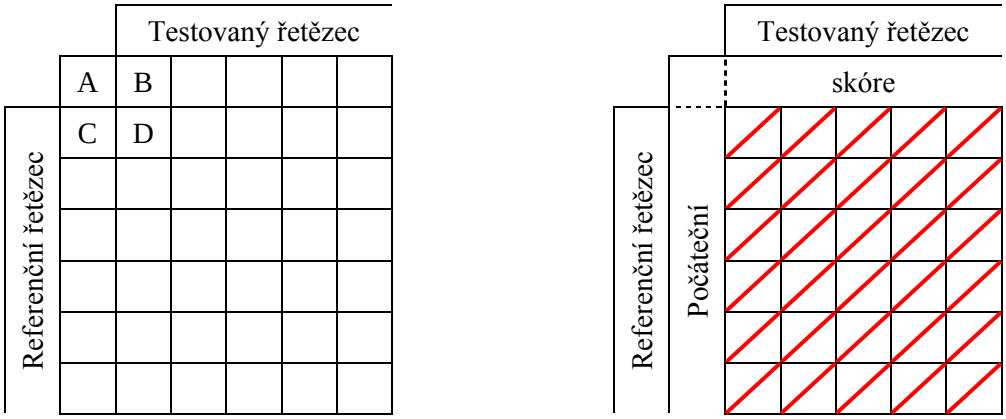


Schéma 2.1: Postup výpočtu skrz tabulku dílčích skóre.

Pro každý sloupec tabulky je tedy zapotřebí jednoho výpočetního článku, který postupně určí všechny jeho hodnot. Jelikož článků bude stejně jako znaků testovaného řetězce, lze pomocí vztahu známého z teorie zřetězených zpracování odvodit, že pro dvojici sekvencí o sto znacích je v jednoprocesorovém počítači potřeba deset tisíc kroků, zatímco v hardwaru postačuje 199 paralelních operací (což je padesátina). Obecným konceptem akcelerace je tedy využití řetězu propojených výpočetních elementů pro postupné určení parciálních skóre až po výsledné ohodnocení podobnosti zkoumaných sekvencí.

2.1 Akcelerace algoritmu Smith-Waterman

Ovšem ani prostá implementace nastíněné architektury nemusí být postačující. Uvedené zrychlení oproti softwarovému výpočtu je ideální hodnotou, které lze dosáhnout jen za určitých okolností. Mnoho projektů akcelerujících bioinformatické algoritmy se proto snaží o maximální zjednodušení všech částí systému.

Přesně takový přístup byl představen i na konferenci FPL v roce 2003 [1]. Je zde jako v mnoha dalších použit zjednodušený vztah samotného algoritmu Smith-Waterman prezentovaný Richardem J. Liptonem a Danielem P. Loprestim v roce 1985. Díky němu je výstup procesního elementu zredukován na šířku jednoho bitu (přičemž znaky jsou ukládány na dvou). Následně byl proveden návrh komponenty, její optimalizace na úrovni primitiv FPGA a implementace ve VHDL pro platformu Pilchard s SDRAM rozhraním namísto klasického PCI a čipem Virtex firmy Xilinx, do nějž bylo možno umístit až 4 032 procesních elementů. Vznikl tím jeden z nejúčinnějších akcelerátorů algoritmu Smith-Waterman, který zajišťoval i zpětnou rekonstrukci zarovnání obou DNA sekvencí, a to bez nutnosti rekonfigurace FPGA (na rozdíl od některých jiných implementací).

Toto řešení je ovšem silně fixní. Neumožňuje totiž změnu pokut za mezeru a rozdílnost znaků (použity jsou typické hodnoty 1 a 2), hledání semiglobálního či globálního zarovnání (přechod k algoritmu Needleman-Wunsch), ani porovnávání jiných než dvoubitových vstupů (čtyřčlenná abeceda – DNA sekvence).

2.2 Obecnější architektury

V návaznosti na specializované architektury se objevila i řešení, pokrývající širší spektrum úloh. Jedno z nich bylo prezentováno v roce 2004 na konferenci ASAP [2]. Ani zde se však nejedná o obecný systém, nýbrž o sadu (či jak autoři uvádí „rodinu“) komponent, přičemž variabilita je dána jejich různými kombinacemi. V jazyce VHDL byly implementovány porovnávací buňky pro oba základní algoritmy (Needleman-Wunsch a Smith-Waterman), stejně jako jednotky porovnávající znaky sekvencí dle předepsaných pravidel (např. skórovacích matic). Výsledná architektura byla vysyntetizována pro FPGA Virtex-II Pro, který mohl podle specifik řešeného problému vykonávat funkci až 126 výpočetních buněk.

Uvedený systém už pokrývá většinu nejpoužívanějších bioinformatických postupů (včetně porovnávání proteinů), jeho rozšíření však není otázkou zobecnění existujících, nýbrž návrhu a implementace nových komponent.

Cílem této práce je tudíž navrhnout a implementovat jednotku akcelerující porovnávání řetězců na základě algoritmů pro hledání podobnosti, které byly vyvinuty primárně pro biologické sekvence. Místo optimalizací pro konkrétní typy úloh a cílových zařízení je však kladen důraz na co možná nejvyšší obecnost, a to především využitím generických parametrů. Systém bude moci používat

libovolné hodnoty pro pokuty při porovnání znaků, přičemž velikost vstupní abecedy není nijak omezena. Díky tomu bude možno porovnávat i textové řetězce v libovolném kódování, což v dnešní době napomáhá např. při detekci nevyžádaných e-mailů – spamů.

3 Akcelerační jednotka

V předchozím textu byl pro článek zřetězeného zpracování použit pojem „procesní element“. Kromě údajů potřebných pro logickou funkci jednotky je totiž zapotřebí synchronizovat (řídit) celý proces porovnání. Procesní element v sobě obsahuje buňku pro porovnání znaků a výpočet příslušného parciálního skóre (hodnoty D z A, B a C) stejně jako kontrolní signály. Právě na implementaci porovnávací komponenty závisí výsledná funkce (metoda), kterou systém akceleruje.

Celý řetěz je spojen do pole procesních elementů, běžně označovaného jako „systolické pole“. K němu je nutno přidat řídicí jednotku, která zajistí správný průběh porovnání a výpočtu celkového skóre. Závislosti jednotlivých prvků systému spolu s jejich účelem jsou naznačeny ve schématu 3.1.

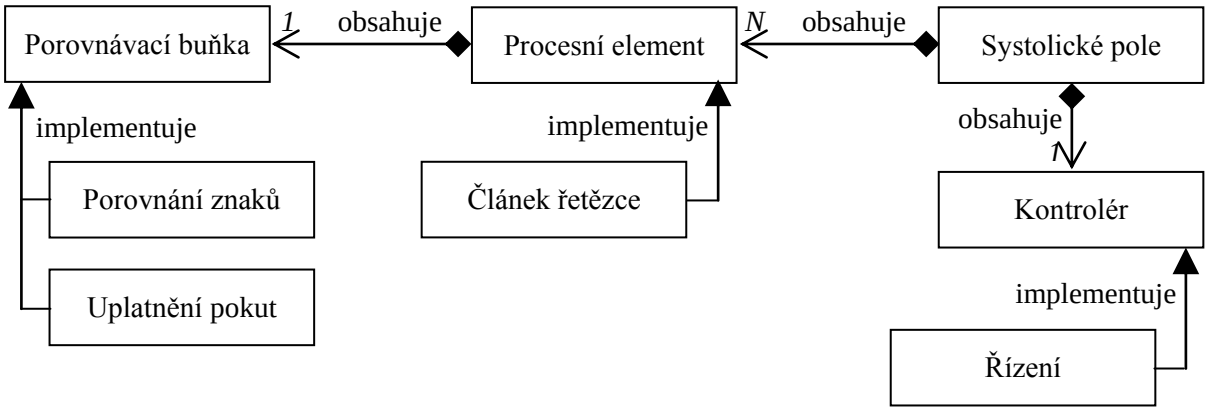


Schéma 3.1: Abstraktní model systému.

3.1 Cílová technologie

Nástrojem zvoleným pro realizaci akcelerační jednotky je FPGA (z angl. field-programmable gate array), což je čip obsahující pole programovatelných logických bloků a propojů. Jeho hlavní výhodou je rekonfigurovatelnost, tedy možnost kompletní změny vykonávané funkce. K jejímu zápisu se používají speciální programovací jazyky pro popis hardwaru (angl. hardware description language – HDL), mezi něž patří v USA rozšířený Verilog a převážně evropský VHDL („V“ je první písmeno anglické zkratky VHSIC – velmi rychlé integrované obvody), který je použit i v této práci. Největším světovým distributorem je firma Xilinx, která také poskytuje software pro syntézu, čili převod kódu v programovacím jazyce na přesný popis konfigurace konkrétního čipu. Vysyntetizovaný design, který bývá většinou uložený v paměti typu EEPROM či FLASH, je nahráván do FPGA při každém jeho startu přes sériové rozhraní.

Jelikož samotný čip k realizaci nestačí, nabízí je výrobce i ve formě development kitů, tedy vývojových karet, které jsou připraveny na nahrání architektury. Při jejich použití není nutno řešit

otázky zavádění designu či komunikace s hostitelským počítačem přes sběrnici (ať už USB, PCI, PCI-E apod.).

Nad takovými kartami vznikají i komplexní soubory ovladačů, nástrojů a dalších nezbytných prvků, které zcela odstiňují jejího uživatele od hardwaru. Takovou platformou je NetCOPE vyvinutý na projektu Liberouter. S jeho pomocí lze vytvořenou architekturu spustit na libovolné kartě, nad kterou NetCOPE funguje (v současné době několik karet projektu Liberouter a ML555 firmy Xilinx s čipem Virtex5). Právě takovou aplikací je i akcelerační jednotka pro porovnávání řetězců.

3.1.1 Protokol FrameLink a formát dat

Vnitřní komunikace v aplikaci platformy NetCOPE probíhá pomocí protokolu FrameLink, který zasílá data v rámcích (angl. frame) obsahujících jeden či více paketů. Ohraničení těchto jednotek je prováděno pomocí kontrolních signálů (začátek a konec rámce, paketu). Pro synchronizaci vysílající a přijímající jednotky pak slouží signály SRC_RDY (zdroj připraven – data jsou platná) a DST_RDY (cíl připraven – schopný přijmout data). Všechny kontrolní signály jsou použity v negované formě, tedy aktivní v logické nule.

Pro potřeby akcelerační jednotky byl stanoven následující formát vstupních dat: každý rámec obsahuje právě dva pakety, přičemž první má pevnou velikost 64 bitů a obsahuje délku sekvence ve znacích a počet jejích segmentů (každý údaj je uložen na 32 bitech), druhý pak vlastní data řetězce.

V designu jsou oba řetězce rozlišovány jako „referenční“ a „testovaný“. Pro zjednodušení návrhu a vzhledem k faktu, že softwarová aplikace používající jednotku může obě sekvence před zasláním k porovnání jednoduše vyměnit, je při rozdílných délkách očekáván jako delší referenční řetězec.

3.2 Popis komponent

Jelikož se má jednat o aplikaci pro platformu NetCOPE, je nutno akceptovat její komunikační protokol pro přenos dat, kterým je FrameLink (modifikace LocalLinku od Xilinxu). Z globálního pohledu tedy jde o komponentu, do které vstupují dva toky (referenční a testovaný řetězec v dohodnutém formátu dat) a vystupuje jeden, který obsahuje výsledné skóre pro podobnost sekvencí (jedinou hodnotu v celém rámci a paketu). Těchto jednotek lze na čip umístit více – záleží na zabraných zdrojích a jejich dostupnosti u konkrétního čipu.

Celá jednotka je pro lepší přehlednost a udržovatelnost složena z více komponent. Některé jsou velmi jednoduché, např. obálky, které pouze mění rozhraní funkčního kódu, jiné plní logické funkce potřebné při porovnávání řetězců. Kromě popisu vstupů a výstupu jednotlivých složek systému jsou důležitými parametry i generika, kterými lze jednoduše a efektivně přizpůsobit architekturu ke konkrétním potřebám úlohy.

3.2.1 Obálka systolického pole

Jak již název napovídá, jedná se o hierarchicky nejvyšší komponentu, samotnou porovnávací jednotku. Vstupem jsou dva a výstupem jeden FrameLinkový tok. Uvnitř komponenty je instancováno systolické pole s obecnějším rozhraním, jehož vstupy jsou částečně řízeny jednoduchým konečným automatem. Je totiž nezbytné zajistit synchronizaci obou řetězců na vstupu tak, aby nejdříve byly k dispozici délky a počet segmentů obou sekvencí (ty jsou ukládány v rámci obálky) a následně vstupovala do pole vždy dvojice znaků (z každého řetězce právě jeden). Jakmile jsou zaslány oba řetězce, vyčká jednotka na platné skóre na výstupu pole a celý proces se opakuje.

Pro tuto jednotku lze genericky nastavit jak vstupní (pro oba toky stejná), tak výstupní datovou šířku a dále počet bitů, do kterých jsou kódovány znaky na vstupu a skóre na výstupu. Zatímco šířka FrameLinkových toků se očekává jako mocnina dvou (tedy 1, 2, 4, 8, 16, 32, 64 či 128 bitů), tak znaky sekvencí, stejně jako ohodnocení jejich podobnosti, mohou být uloženy na libovolném počtu bitů, který je menší nebo roven celkové šířce dat. Komponenta provede oříznutí o případné nadbytečné bity a do systolického pole posílá samotné znaky. Na jeho výstupu naopak dle potřeby přidává k výstupu nuly až do odpovídající datové šířky.

Kromě spíše technických parametrů je možné nastavit i generika související s logickou funkcí jednotky. Prvním je počet procesních elementů instancovaných uvnitř systolického pole. Z praktického hlediska se jedná o maximální počet znaků testovaného řetězce, který je jednotka schopna zpracovat – touto hodnotou je omezen nejvyšší možný počet sloupců vyplňované tabulky. Aby mohl systém pracovat dle očekávání, je potřeba zadat i pokuty za vložení mezery (pro každý řetězec zvlášť) a neshodu znaků. Uvedené čtyři konstanty nejsou nastavovány pomocí generik, ale byly vyčleněny do „balíčku“ (angl. package) především proto, že porovnávací buňka (která je kromě generátorů skóre jedinou komponentou využívající pokut) je hierarchicky nejníže a bylo by nutné předávat generika skrz celou strukturu.

3.2.2 Systolické pole a generátory počátečního skóre

Samotné pole procesních elementů je ovládáno sadou řídících signálů, jejichž správné nastavení musí zajistit nadřazený prvek systému (v našem případě obálka, která převádí FrameLink na požadované vstupy). Jsou jimi délky obou sekvencí a počet segmentů testovaného řetězce. Jejich platnost se komponentě sděluje signálem REQ – požadavkem na porovnání. Následně jednotka vystaví signál BUSY jako potvrzení požadavku a očekává na datových vstupech vždy dvojici znaků (z každého řetězce jeden). Na výstup pak vystaví ohodnocení podobnosti obou sekvencí a jeho platnost potvrdí signálem SCORE_DV.

Posledním vstupním signálem je PIPE_EN, který povoluje celé jednotce provádět výpočet a během něj prakticky říká, že na datových vstupech jsou správná data. Možnost pozastavit celý proces porovnávání v kterékoli jeho fázi je velmi důležitá vlastnost, která umožňuje použití jednotky

prakticky v jakémkoli prostředí, jelikož není nutné zajistit pravidelné zásobování komponenty vstupními daty.

V systolickém poli se uplatňuje princip zřetězeného zpracování, díky kterému dochází k akceleraci výpočtu. Některé signály prostupují klasickým způsobem „rourou“ (obecně se jedná především o signály související s vertikálním – referenčním – řetězcem), zatímco jiné jsou přiváděny paralelně ke všem procesním elementům (analogicky data a skóre z horizontální, čili testované sekvence) jak lze vidět ve schématu 3.2.

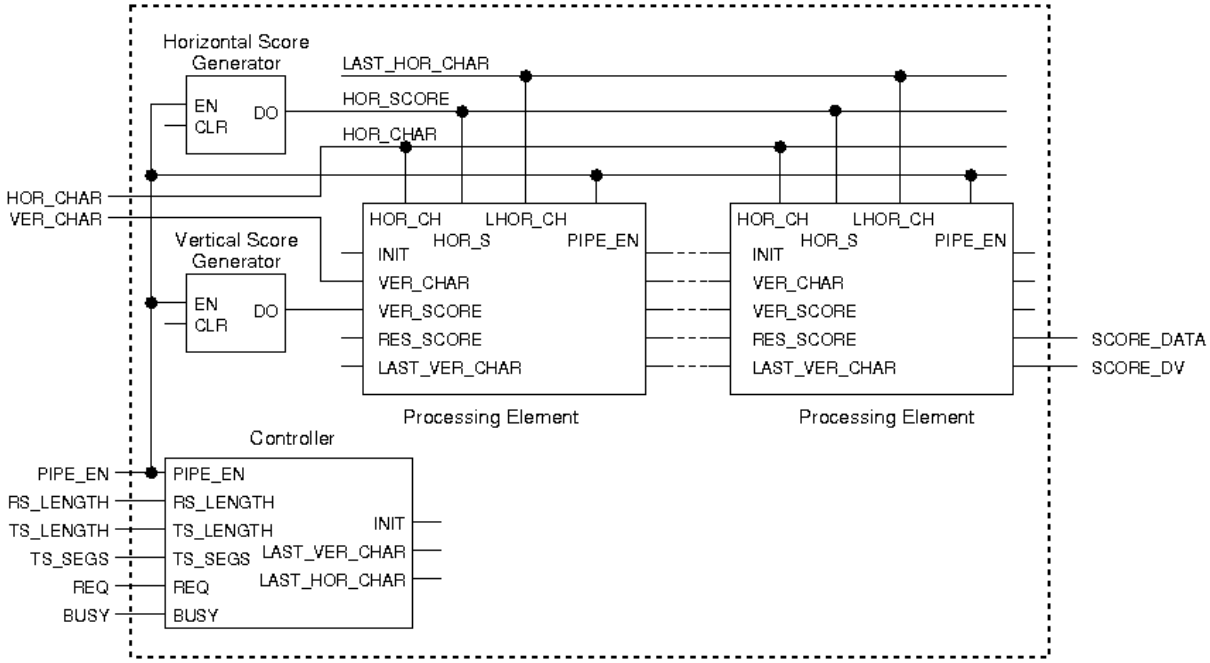


Schéma 3.2: Architektura systolického pole.

Genericky lze nastavit datovou šířku vstupních znaků a výstupního skóre. Tyto parametry byly již popsány v rámci globální obálky. Komponenta obsahuje kromě propojeného pole (lépe řečeno řetězce) procesních elementů ještě kontrolér systolického pole pro jeho řízení a dva generátory počátečního skóre pro první řádek a sloupec tabulky. Ty jednoduše vrací postupně násobky pokuty za vložení mezery do příslušného řetězce.

3.2.3 Kontrolér systolického pole

Pole procesních elementů je řízeno pomocí konečného automatu, jehož popis je součástí právě tohoto kontroléru. V něm je obsloužen příchozí požadavek na porovnání řetězců, a to načtením délek sekvencí (a počtu segmentů testovaného řetězce) do registrů a následným nastavením BUSY signálu. Ve stejném taktu je prvnímu procesnímu elementu zaslán signál INIT, jehož funkce bude popsána později. Registry s délkami obou sekvencí jsou poté dekrementovány s každým taktem, ve kterém je nastavený signál PIPE_EN (tj. na vstupu jsou platné znaky) až do chvíle, kdy skončí testovaný

řetězec. V tu chvíli se zašle všem výpočetním prvkům signál LAST_HOR_CHAR, který společně se signálem INIT určuje, ze kterého elementu vzejde výsledné skóre. S posledním znakem referenčního řetězce je do zřetězeného zpracování vyslán signál LAST_VER_CHAR, jenž na jeho konci označí ohodnocení podobnosti řetězců (ze systolického pole jako SCORE_DV).

Přesnější popis funkce uvedených signálů je uveden v části věnované procesnímu elementu. Pro potřeby řízení výpočtu není zapotřebí žádných generických parametrů.

3.2.4 Procesní element a porovnávací buňka

Před technickým popisem procesního elementu, který je v hierarchii systému pod systolickým polem, je vhodné poupravit algoritmus, jakým je určováno parciální skóre pro jednu buňku tabulky. Zásadní obměnou je nepoužívání záporných hodnot, které zbytečně komplikuje výpočet v hardwaru a může vést k nepříjemnostem. Vše tedy bude řešeno pomocí pokut, které však budou vždy kladné (bude se jednat o „trestné body“). Bonus za shodu znaků nebude v této modifikaci přítomen, nahradí ho „nulová pokuta“. Nejpodobnější řetězce pak systém ohodnotí nejnižším skóre.

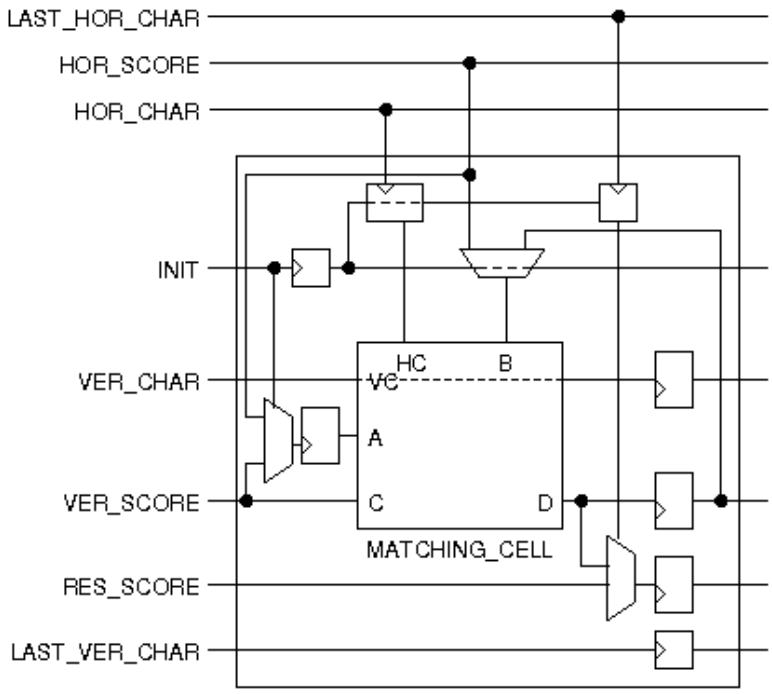


Schéma 3.3: Architektura procesního elementu.

Procesní elementy jsou v systolickém poli umístěny v řadě za sebou, v každém z nich jsou postupně počítána parciální skóre pro jeden sloupec tabulky (delší sekvence – referenční – se nachází podél její vertikální osy, testovaný řetězec pak horizontálně nad tabulkou). Navíc musí být zajištěno ukládání hodnot potřebných pro další výpočty do registrů, a to z důvodů zamezení vzniku kritických cest a možnosti kdykoli pozastavit výpočet (což se děje signálem PIPE_EN, kterým jsou povolovány všechny datové i řídící vstupy v komponentě).

Jedním z hlavních signálů zřetězeného zpracování je INIT, který postupně spouští výpočet v jednotlivých výpočetních prvcích (je registrován a předáván s každým hodinovým taktem, kdy je nastaven PIPE_EN). Podle něj jsou multiplexovány vstupy registrů uchovávajících skóre z buňky diagonálně a nad aktuální pozicí. Při inicializaci se načítá horizontální, pak již vždy vertikální skóre. Navíc tento signál povolí uložení aktuálního znaku testovaného řetězce, který je pro celý sloupec tabulky stejný, spolu s příznakem, zda se jedná o poslední prvek sekvence.

Dalšími signály zasílanými do řetězce procesních elementů jsou aktuální prvek referenční sekvence (vertikální znak prostupující všemi komponentami) a vertikální skóre (v tabulce buňka vlevo od právě vyhodnocované), které je navíc uloženo i před samotným porovnáním. Tímto se zajistí posouvání částečných skóre (hodnota, která v jednom taktu reprezentuje buňku nalevo od aktuální pozice je v následujícím kroku použita jako vstup z diagonály).

Uvedená architektura (zobrazená ve schématu 3.3) v sobě skrývá i logickou funkci výpočtu skóre, tedy inkrementaci vstupních parciálních výsledků o pokuty za mezeru (a případně za neshodu znaků) a výběr nejmenší hodnoty, která reprezentuje ohodnocení aktuální buňky v tabulce. Její určení zajišťuje podkomponenta mcell (z angl. matching cell – porovnávací buňka). Na vstup dostává trojici vstupních skóre, potřebných k výpočtu nového, a dvojici znaků, které porovná.

Postupné vyplnění celé tabulky je tedy zajištěno, zbývá určit výsledné ohodnocení podobnosti porovnávaných řetězců. Tím je hodnota v posledním sloupci posledního řádku. Je tedy potřeba tuto pozici identifikovat. Určení procesního elementu, ze kterého skóre vyjde, je zajištěno načítáním signálu LAST_HOR_CHAR při inicializaci (označíme tím komponentu, která počítá poslední sloupec tabulky). Společně s posledním znakem referenčního řetězce vstupuje do systému signál LAST_VER_CHAR, který po průchodu zřetězeným zpracováním označí platné skóre pro podobnost (identifikuje poslední řádek). Je však zapotřebí vzít v úvahu, že procesních elementů může být víc než znaků testované sekvence. V tom případě se některé z nich výpočtu vůbec neúčastní a požadovaný výsledek jimi musí pouze „projít“ (jinak by bylo nutno vést výstup každé komponenty do zbytečně velkého multiplexoru a vybírat jeho správný vstup). K tomu slouží signál RES_SCORE (pochopitelně registrovaný), do kterého označený procesní element (signálem LAST_HOR_CHAR) posílá výsledek svého výpočtu a ostatní jej pouze propagují dále.

Komponenta potřebuje pro vykonávání požadované funkce zadat, na kolika bitech jsou uloženy znaky obou sekvencí a datovou šířku skóre. Podle konstant v balíčku jsou pak uplatněny pokuty za vložení mezer a neshodu znaků.

3.3 Celkové zapojení

Akcelerační jednotka je připravena, zbývá ji zapojit do platformy NetCOPE, která zprostředkuje jak zásobování daty pro porovnání tak sběr výsledných skóre. Karta (a následně FPGA čip) komunikuje s počítačem přes systémovou sběrnici, kterou může být např. PCI či PCI Express. Data procházejí

přes komponenty zvané DMA buffery (zásobníky přímého přístupu do paměti), které přijímají tok ze systémové sběrnice (v NetCOPE je do čipu vedena jako interní sběrnice) a vysílají data ve formátu protokolu FrameLink v jednom směru a analogicky přijaté rámce umožňují zaslat zpět na sběrnici.

Jelikož je žádoucí na čip umístit větší množství akceleračních jednotek (a to i s různě nastavenými generickými parametry), musíme se architekturou DMA bufferů zabývat podrobněji. Šířka interní sběrnice je 64 bitů, stejnou šířku pak musí dohromady mít všechny FrameLinkové toky (jednotné datové šířky) na opačné straně příslušné komponenty. K jednomu páru bufferů (jeden pro vysílání a druhý pro příjem rámců) je tedy možno připojit např. 16 jednotek, které porovnávají dvě sekvence dvoubitových znaků a ohodnocují je čtyřbitovým skóre (vysílá se 2x2x16, tj. 64 a přijímá se 4x16, tedy opět 64 bitů). Aby byla možno splnit podmínku šířky rozhraní bufferů, existuje možnost některé datové toky nevyužit a ponechat je nezapojené.

Bloků s různou konfigurací datových šířek lze na čip umístit libovolné množství, limitujícím faktorem je pouze dostupnost zdrojů. Obecné schéma zapojení jednoho bloku je ve schématu 3.4. Jako TX je označována jednotka vysílající data, RX je pak přijímací komponenta, přičemž počet vysílaných datových toků je dvojnásobný vzhledem k přijímaným (vstupem jednotky jsou dva řetězce, výstupem jedno skóre).

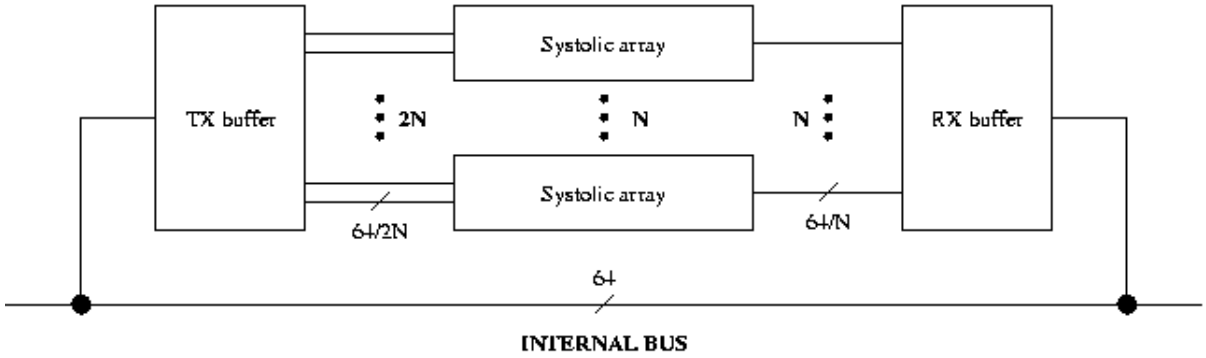


Schéma 3.4: Obecné zapojení jednotek s DMA buffery.

4 Aplikace na různé typy úloh

Vytvořená jednotka je velmi flexibilní a je možné ji použít na široké spektrum úloh. Díky nastavitelnému počtu procesních elementů a datovým šířkám vstupu a výstupu umožňuje hledat globální zarovnání prakticky libovolných řetězců (kromě biologických především textových, a to nezávisle na znakové sadě). Shrnutí celkové výkonnosti a vlastností systému není tedy možné provést bez uvedení specifikace dané úlohy. Navíc záleží i na konkrétním použitém čipu. Akcelerační jednotka byla proto vysyntetizována pro několik typických bioinformatických úloh převzatých z článku [3] a dále pro porovnání textových řetězců.

Parametrem používaným pro srovnání výkonnosti podobných systémů je BUps (z angl. billions updates per second), který udává, kolik miliard parciálních skóre vyhodnotí daná architektura za jednu vteřinu. Pro srovnání lze uvést, že optimalizovaný program dosahuje na počítači s procesorem Intel Pentium 4 (3,2 GHz, 2 MB cache) a 2 GB RAM v průměru 0,06 BUps. Hodnota 60 BUps tedy udává, že porovnávané řešení nabízí tisícinásobné zrychlení oproti softwarovému výpočtu.

Otázkou stále zůstává cílová platforma. Vzhledem k širokým možnostem využití akcelerační jednotky bylo pro porovnání parametrů u jednotlivých typů úloh zvoleno zařízení z nově nastupující rodiny FPGA Virtex5 firmy Xilinx, a to konkrétně čip XC5VLX50T (střední velikost – 7 200 polí neboli „slice“), kterým je osazena karta ML555 od stejného výrobce. Do budoucna se počítá i s využitím nové generace Combo karet vyvíjených na projektu Liberouter (tzv. Combo v2), které mohou být dle požadavků osazeny i většími čipy než má ML555 (např. XC5VLX110T či dokonce XC5VLX155T s 24 320 poli) a pojmout tak větší množství jednotek. Ve vývoji je i platforma NetCOPE, v dohledné době přibude např. možnost genericky nastavit šířku interní sběrnice, čímž se zásadně zjednoduší zapojování jednotek s různě širokými vstupy a výstupy.

šířka v bitech		procesní elementy	„slice“ pro jednotku	jednotek na čipu	maximální frekvence	BUps
znaku (vstupu)	skóre (výstupu)					
2 (2)	3 (4)	40	433	16	232 MHz	148,5
2 (2)	5 (8)	80	1 743	4	208 MHz	66,5
5 (8)	4 (4)	8	209	34	189 MHz	51,4
8 (8)	8 (8)	1 000	1 874	3	145 MHz	435
8 (8)	5 (8)	20	574	12	173 MHz	41,5
16 (16)	5 (8)	20	664	10	173 MHz	34,6

Schéma 4.1: Využití zdrojů čipu pro různé nastavení jednotky.

Dalším faktorem ovlivňujícím získané hodnoty je použitý syntetizační nástroj, kterých existuje celá řada. Zde byl použit integrovaný doplněk XST programu Xilinx ISE. Výsledky uvedené ve schématu 4.1 jsou pouze odhadem využití prostředků čipu, jejich skutečné množství by ukázalo až

vytvoření přesného designu pro FPGA. Určitá část zařízení bude navíc obsazena platformou NetCOPE (DMA buffery a řadiče, PCI-E bridge atd.), velikost tohoto prostoru bude opět záviset na řešené úloze a použitém zařízení a není ji tedy možno jednoduše promítnout do uvedených hodnot.

Z hlediska zabraných zdrojů jsou nejdůležitějšími parametry jednotky počet procesních elementů v systolickém poli (udávající maximální délku kratší z porovnávaných sekvencí) a počty bitů, na kterých jsou uloženy vstupní znaky a výstupní skóre. Datové šířky FrameLinkových toků (tedy „zaokrouhlení“ šířky znaku a skóre na nejbližší vyšší mocninu dvou) mají minimální vliv stejně jako konstanty použité pro pokuty (zde lze nanejvýš „ušetřit“ jeden generátor skóre v případě, že jsou pokuty za vložení mezer do obou sekvencí stejné – optimalizaci zajistí syntetizátor). Pro všechny uvedené příklady byla použita pokuta pro neshodu znaků 2 a pokuty za vložení mezery do obou řetězců shodně 1.

Již před rozborem algoritmů bylo řečeno, že nejčastějšími biologickými řetězci jsou posloupnosti nukleotidů v DNA a RNA (vždy 4 různé dusíkaté báze, šířka znaku i vstupu jsou tedy 2 bity) a sekvence aminokyselin tvořících proteiny (pro zakódování dvaceti základních stavebních prvků stačí 5 bitů, vstupní FrameLinkové toky budou tedy osmibitové). Šířka výstupního skóre byla určena experimentálně či výpočtem, stejně tak počet použitých procesních elementů vychází ze statistických údajů.

U textových řetězců jsou pak typickými šířkami vstupních znaků hodnoty 8 (běžná kódování jako ASCII, ISO-8859-2 apod.), případně 16 bitů (Unicode obsahující i speciální abecedy). Vzhledem k tomu, že uvažujeme porovnávání slov, můžeme počítat s nanejvýš dvaceti znaky (a tedy procesními elementy) a maximálním skóre 30 (resp. 32 – 5 bitů).

5 Závěr

Akcelerační jednotka byla navržena na základě studia bioinformatických algoritmů, zejména Needleman-Wunsch a Smith-Waterman [4]. Následně byla provedena její implementace pro čip FPGA (za použití platformy NetCOPE vyvinuté na projektu Liberouter) v jazyce VHDL. Správnost řešení byla úspěšně ověřena simulačním nástrojem ModelSim 6.3c, odhad použitých zdrojů čipu Virtex5 na kartě ML555 byl proveden syntetizátorem XST integrovaným v nástroji Xilinx ISE 9.1i.

Ze získaných údajů bylo zjištěno, že zrychlení systému oproti výpočtu na jednoprocesorovém počítači se může pohybovat až v řádu tisíců. Konkrétně pro úlohu porovnání struktury dvou proteinů dosahuje jednotka hodnoty 435 BUps, což je 7 250krát více než referenční softwarové řešení.

Celá porovnávací jednotka funguje na velmi obecné úrovni a je tedy možné použít ji na široké spektrum nejen bioinformatických úloh. Jelikož nevyžaduje pravidelné zásobování daty, je celková režie řízení (která je navíc implementována přímo v hardwaru) velmi malá.

Pro ještě větší využití by bylo vhodné doimplementovat specifika algoritmu Smith-Waterman a umožnit tak hledání lokálního zarovnání řetězců (jedná se především o úpravu výstupu, kterým již není jediná hodnota). Dále se velmi často objevuje požadavek na zpětnou rekonstrukci optimálního (případně optimálních) zarovnání porovnávaných sekvencí, nepřítomnost této funkce v prezentované jednotce je zajisté jedním z hlavních možných směrů dalšího vývoje práce.

Literatura

[1] Yu, C. W., Kwong, K. H., Lee, K. H., Leong P. H. W., A Smith-Waterman Systolic Cell. In: *Proc. 13th Field-Programmable Logic and Applications (FPL2003)*. Springer, Berlin, 2003

[2] Van Court, T., Herbordt, M. C., Families of FPGA-Based Algorithms for Approximate String Matching. In: *15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04)*. IEEE Computer Society, Washington, DC, USA, 2004

[3] Martínek, T., Lexa, M., Beck, P., Fučík, O., Automatic Generation of Circuits for Approximate String Matching. In: *Design and Diagnostics of Electronic Circuits and Systems*, 2007, s. 203

[4] Krane, D., Raymer, M., *Fundamental Concepts of Bioinformatics*. San Francisco: Benjamin Cummings, USA, 2003

Seznam příloh

Příloha 1. CD se zdrojovými texty a touto zprávou v elektronické podobě (formát .doc a .pdf).