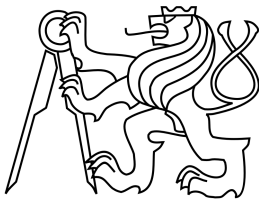
České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Začleňování analýzy měkkého rentgenového záření do systému zpětnovazebního řízení tokamaku COMPASS

VÝZKUMNÝ ÚKOL

Autor: bc. Viktor Löffelmann
Vedoucí práce: doc. RNDr. Jan Mlynář, Ph.D.
Konzultant: Ing. Ondřej Mikulín
Rok: 2014

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ
PRAHA 1 - STARÉ MĚSTO, BŘEHOVÁ 7 - PSČ 115 19

Katedra: fyziky ***Akademický rok:*** 2013/14

VÝZKUMNÝ ÚKOL

Posluchač: Bc. Viktor Löffelmann

Obor: Fyzika a technika termojaderné fúze

Vedoucí úkolu: RNDr. Jan Mlynář, Ph.D.

Konzultant: Ing. Ondřej Mikulín

Název úkolu (česky/anglicky):
Začleňování analýzy měkkého rentgenového záření do systému zpětnovazebního řízení tokamaku COMPASS
Partial implementation of the Soft X-ray analyses into the real-time control system of the COMPASS tokamak

Pokyny pro vypracování:
Provést rešerši na téma hardwarového řešení zpětnovazebního řízení tokamaku COMPASS a řešení inverze matice v obvodech FPGA. Sestavit jednoduchý testovací obvod s FPGA pro zpracování prostorově rozlišených dat měkkého rentgenového záření, ověřit jeho funkčnost na jednoduché úloze, například průběžného určování polohy plazmatu podle kanálu s maximálním signálem. Na základě rešerše rozhodnout, zda je implementace algoritmu pro tomografii v reálném čase (který byl vyvinut v rámci BP) do jednotky FPGA realistická. V pozitivním případě provést implementaci výpočetní části algoritmu, v negativním případě navrhnout náhradní řešení, obhájit ho a myšlenku na úrovni algoritmu pro zpětnovazební řízení realizovat. Ověřit funkčnost (slaboproudé testy). V rámci řešení se pokusit spolupracovat s týmem IST.
Součástí zadání výzkumného úkolu je jeho uložení na webové stránky katedry fyziky.

Literatura:
1. M. Hron et al.: Overview of the COMPASS CODAC system. Fusion Engineering and Design (submitted)
2. K. Behler et al, Real-time diagnostics at ASDEX Upgrade—Architecture and operation, Fus Eng Design 83, 2–3, (2008) 304-311
3. Carvalho P. et al.: Real-time plasma control based on the ISTTOK tomography diagnostic, Rev. Sci. Instrum. 79 10 (2008) 10F329

Datum zadání: 18.10.2013 .....

Datum odevzdání: 27.06.2014 ***Vedoucí katedry***

Poděkování

Děkuji docentu RNDr. Janu Mlynářovi, Ph.D. za to, že mi umožnil pracovat na takovém tématu.

Děkuji Ing. Ondřeji Mikulínovi za zapůjčení programovatelného hradlového pole a za trpělivé uvedení do problematiky jeho programování.

Viktor Löffelmann

Abstrakt

Byl navržen obvod pro hardwarové urychlení tomografie plazmatu. Design využívá dříve vyvinutého tomografického algoritmu, založeného na řešení regularizované soustavy lineárních rovnic. Obvod byl implementován a otestován s využitím programovatelného hradlového pole. Na dostupném hardwaru bylo dosaženo rychlosti jedné tomografické inverze o rozlišení přibližně 60 pixelů za 50 mikrosekund, což činí obvod potenciálně vhodným pro řízení polohy plazmatu na tokamaku COMPASS v reálném čase.

Obsah

1 Úvod

Plazma v tokamaku a podobných zařízeních podléhá řadě nestabilit. Nestability jsou vesměs nevítané, protože zhoršují udržení plazmatu, případně poškozují zařízení. Proti některým z nich byly vyvinuty statické metody, spočívající například ve vhodném tvarování magnetického pole. Další jsou nepředvídatelné, při každém pokusu jiné, a není známo žádné univerzální opatření, které by jim jednou provždy zabránilo. Namísto toho je třeba detekovat výskyt těchto nestabilit v průběhu experimentu a včas vyvinout reakci odpovídající situaci.

Mezi nestability vyžadující dynamické řízení patří nestabilita známá jako *vertikální*. Spočívá v pohybech celého sloupce plazmatu ve směru kolmém k rovině prstence tokamaku. Existuje i analogická *radiální* nestabilita, tedy smršťování a roztahování plazmatického prstence. Větší význam té vertikální je dán tvarováním plazmatu, používaném v moderních tokamacích. Jejich toroidální pole má (mimo jiné) kvadrupólovou složku, která zužuje průřez plazmatu v radiálním směru, zatímco na výšku jej roztahuje [1, 2, 3]. Tento vliv započtený samotný by měl za následek existenci stabilní rovnovážné polohy v tom směru, v němž je plazma stlačováno, zatímco v kolmém směru by jediná rovnováha byla nestabilní. V reálném zařízení působí na polohu plazmatu i jiné vlivy, ovšem značný nepoměr mezi vertikální a radiální nestabilitou je i tak pozorován [3].

Proti rychlým pohybům plazmatického sloupce pomáhá vyrobit komoru tokamaku vodivou. Představu o jejím vlivu lze získat z předpokladu, že plazma se pohybuje společně se svým magnetickým polem. To indukuje vířivé proudy ve stěně komory, a jejich vlastní magnetické pole pohyby plazmatu brzdí. Časová konstanta vertikální nestability u tokamaku COMPASS (vybaveného vodivou komorou) se pohybuje v řádu desetin milisekundy; Hypotetická časová konstanta s nevodivou komoru by byla o dva až tři řády kratší. Přitom platí, že větší tokamaky mají větší časovou konstantu [4].

Proti pomalejším pohybům je stabilizace vířivými proudy méně účinná, protože materiál komory mívá nenulovou rezistivitu a vířivé proudy vyvolané pomalým pohybem plazmatu se v něm stačí utlumit dříve, než by stačily ten pomalý pohyb zbrzdit [1]. Je tedy vhodné doplnit ji aktivní stabilizací využívající senzory polohy, elektroniku a cívky schopné dostatečně rychlých změn magnetického pole. Aktivní stabilizace se pro změnu hodí k ovlivňování pomalejších pohybů plazmatu. Její reakční doba je totiž dána nejen rychlostí šíření elektromagnetického pole mezi plazmatem a komorou, ale také různě složitým zpracováním signálu, setrvačností řídicích cívek a dobou pronikání pole řídicích cívek skrz vodivou stěnu (to poslední v případě, že jsou cívky umístěny mimo komoru) [4].

Je řada možností, jak získat údaje o poloze plazmatu, vhodné pro nějaký navazující regulátor. V zájmu optimalizace rychlosti se většinou omezujeme na takové metody, které nevyžadují moc dlouhé zpracování signálu. Obvykle používanou diagnostikou jsou různé druhy senzorů magnetického pole. Přepočítat výstup vhodně zvolených senzorů na polohu plazmatu může být poměrně přímočaré, často na to stačí analogové obvody [2]. Dosud neustávající vývoj elektroniky nám ovšem dovoluje i komplikovanější postupy.

Detektory měkkého rentgenového záření představují další z diagnostik použitelných k určení

polohy plazmatu. Zatímco s magnetickými senzory lze odhadovat polohu místa s nejvyšší proudovou hustotou [5], při použití detektorů měkkého rentgenového záření lze jako polohu plazmatu chápat například polohu maxima emisivity v zaznamenávaném spektru. Signál detektorů záření není výsledkem časové integrace (na rozdíl od v současnosti používaných magnetických senzorů), a odchylka od správné hodnoty se tudíž s časem nezvětšuje. Pro účely této práce podstatnou nevýhodou je fakt, že je poměrně složité získat z jejich dat nějaké praktické údaje, jako je právě poloha.

Patrně nejsložitější, ale také nejuniverzálnější cestou k určení polohy plazmatu z detektorů záření je tomografie. V minulosti byla vyvinuta řada algoritmů pro různě rychlou a různě kvalitní tomografii. Některé z nich jsou již nyní natolik časově optimalizované, že při použití přiměřeně výkonného hardwaru lze uvažovat o jejich uplatnění v řízení tokamaku v reálném čase. Mimo to se nabízí další optimalizace s využitím hardwaru specializovaného na daný úkol.

Vhodným hardwarem se zdají být programovatelná hradlová pole, jejichž technologie se v poslední době rychle rozvíjí. Tato zařízení mají některé schopnosti podobné integrovaným obvodům vyráběným na míru, ale z výroby jsou univerzální a na míru konkrétním účelům se pouze programují. Díky tomu jsou dostupná i pro malé aplikace (vyžadující použití speciálního obvodu jen na jednom nebo několika místech), mezi které patrně řízení tokamaku patří.

2 Zpětnovazební řízení tokamaku COMPASS

Tokamak COMPASS je vybaven řadou diagnostik, které mají dohromady více než 1000 výstupních kanálů, digitálních i analogových (posléze digitalizovaných). Vzorkovací frekvence těchto kanálu jsou poměrně různorodé. Nejvyšší z nich dosahují 2 GHz, což je užitečné pro měření Thomsonova rozptylu nebo pro reflektometrii [6]. Detektory měkkého rentgenového záření a bolometry pracují s frekvencí 2 MHz. Taková frekvence umožňuje studium rozličných nestabilit a turbulencí v plazmatu, ale také poskytuje značnou rezervu pro případné využití ve zpětné vazbě.

V současnosti je na COMPASSu použita zpětná vazba k řízení vertikální a horizontální polohy plazmatu, proudu plazmatem, hustoty a tvarovacího magnetického pole [6]. Některé z těchto veličin se vyvíjejí pomaleji než jiné, a jejich zpětnovazební řízení tedy nevyžaduje tak rychlé výpočty, případně tak časté opakování výpočtů. Na COMPASSu je tudíž zpětná vazba rozdělena na rychlou a pomalou, přičemž ta rychlá řídí polohu plazmatu, zatímco pomalá řídí ostatní parametry [6, 7].

Algoritmy počítající veličiny určené k regulaci (a další věci potřebné v reálném čase) jsou realizovány oddělenými bloky napsanými v C++, které jsou během výboje spouštěny ve frameworku jménem *MARTe* (*Multithreaded Application Real-Time executor*). Podle požadované rychlosti jsou tyto bloky rozděleny do dvou vláken, z nichž jedno opakuje své výpočty každých 50 mikrosekund a druhé každých 500 mikrosekund (těmto dvěma vláknům odpovídá rozdělení algoritmů na rychlou a pomalou zpětnou vazbu).

Na výstupy zmíněných algoritmů navazují regulátory a na ně aktuátory, které svou činností ovlivňují plazma. K řízení polohy (vertikální a radiální) se používají dva "rychlé" tranzistorové zesilovače, pracující s frekvencí 40 kHz [8]. Každý ze zesilovačů napájí zvolenou kombinaci cívek (*vinutí F*) vytvářejících v komoře tokamaku poloidální pole; Cívky pro řízení vertikální polohy generují pole s převažující radiální složkou, cívky řídící radiální polohu mají pole přibližně vertikální. Další sady poloidálních cívek, sloužící k řízení proudu plazmatem (*M*), tvaru plazmatu (*S*) a k vytvoření horizontální rovnovážné polohy (*E*), jsou napájeny pomalejšími tyristorovými zdroji [8]. Kromě cívek existují i další prvky považované za aktuátory, například hustota je řízena napouštěním plynu přes piezoelektrický ventil [6].

Hardware pro algoritmy zpětné vazby má podobu dvou skříní standardu ATCA. Každá skříň je vybavena deskou s procesorem, na němž běží MARTe, a několika dalšími deskami sloužícími ke sběru dat. Na deskách sbírajících data není procesor, zato je tam programovatelné hradlové pole řady Xilinx Virtex 4 a 32 osmnáctibitových AD převodníků se vzorkovací frekvencí 2 MHz. Programovatelná hradlová pole jsou využívána mimo jiné k časové filtraci výstupů AD převodníků a následnému snížení vzorkovací frekvence na 20 kHz. Data se sníženou vzorkovací frekvencí představují vstup pro MARTe [7].

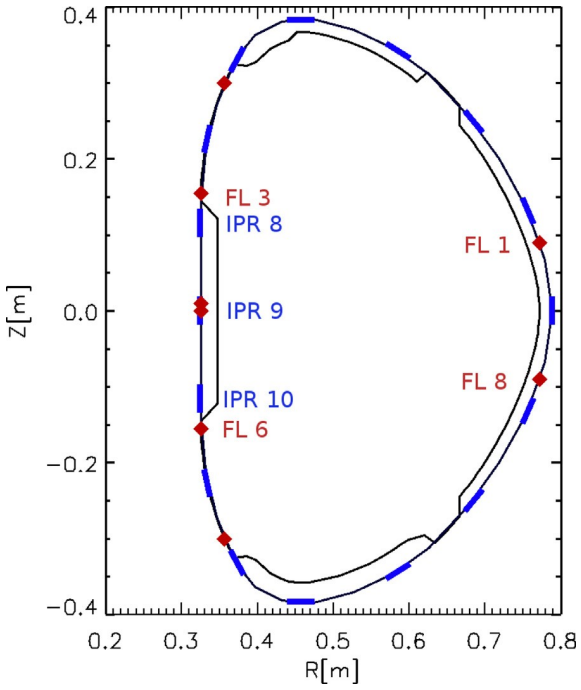
2.1 Výpočet polohy plazmatu

Z více než 400 magnetických senzorů instalovaných na tokamaku COMPASS [9] se dnes k detekci polohy plazmatu používají dva typy: *Rogowského cívky* (*IPR*, Internal Partial

Rogowski) a cívky zvané *flux loops*. Cívky IPR se nacházejí na vnitřní stěně komory a měří tu složku poloidálního pole, která je rovnoběžná se stěnou. Je jich 16, v poloidálním směru pravidelně rozmístěných kolem komory [9]. Napětí na nich měřené je úměrné časové derivaci magnetického pole, takže k určení pole samotného je potřeba jejich signál integrovat. Flux loops jsou jednoduché závity v toroidálním směru, rozmístěné na stěně komory v osmi poloidálních polohách. Měří cirkulaci elektrického pole v toroidálním směru, po časové integraci pak poloidální magnetický tok [9]. Na obrázku 1 je znázorněno umístění všech zmíněných cívek na průřezu komorou.

Vztah mezi údaji magnetických senzorů a polohou plazmatického sloupce není jednoduchý, ačkoli algoritmus používaný k výpočtu polohy je dostatečně rychlý pro použití v reálném čase. Ze signálů několika vybraných cívek se spočtou dvě lineární kombinace, o nichž je známo, že určují polohu plazmatu ve dvou směrech dostatečně jednoznačně; Dříve byly používány dvě kombinace Rogowského cívek, v současnosti jde o jednu kombinaci Rogowského cívek a jednu kombinaci flux loops [5, 7]. Získané hodnoty se na polohu přepočítají pomocí vyhledávací tabulky. Vhodné kombinace cívek byly určeny z numerického modelu, který počítá s předem definovanými profily proudové hustoty s volitelnou polohou středu. Z tohoto modelu pochází i používaná vyhledávací tabulka [5].

Výpočet, od získání dat ze senzorů po určení parametrů pro rychlé zesilovače, trvá přibližně 12 μs [7]. Další zpoždění ve smyčce zpětné vazby pochází od sériových komunikačních linek mezi počítači a tokamakem, zpracováním dat elektronikou řídící rychlé zesilovače a faktem, že zesilovače nereagují okamžitě, nýbrž jen v diskrétních okamžicích oddělených 25 μs periodou. Před nedávnou optimalizací dosahovalo celkové zpoždění (mezi naměřením dat a zásahem aktuátorů) 90 až 140 μs [7]. Časová konstanta vertikální nestability na COMPASSu činí přibližně 500 mikrosekund, přesto se však ukazuje, že v některých konfiguracích plazmatu je takové zpoždění moc velké [7].



Obr. 1: Rozmístění šestnácti IPR cívek (modře) a osmi flux loops (červeně) na řezu komorou tokamaku COMPASS; Převzato z [7]

2.2 Detekce měkkého rentgenového záření

Měkké rentgenové spektrum plazmatu v tokamaku má čárovou i spojitou složku. Čáry pocházejí především od relativně těžkých nečistot, které zůstávají i při tokamakových teplotách jen částečně ionizovány. Spojitá část je tvořena brzdným zářením, vznikajícím při srážkách elektronů s ionty. Hustota výkonu brzdného záření silně závisí na teplotě a hustotě plazmatu, což činí měkké rentgenové záření vhodným ke sledování řady nestabilit, které teplotu a hustotu ovlivňují, ale také k určování polohy nejteplejší oblasti v plazmatu [10]. Kromě toho brzdné záření závisí na obsahu těžkých nečistot; Měkká rentgenová diagnostika se používá i k jejich lokalizaci [10].

Na COMPASSu se k detekci měkkého rentgenového záření používají fotodiody řady AXUV[1]. Jde o křemíkové diody s PN přechodem, schopné měřit široké spektrum elektromagnetického záření (přibližně od jednotek elektronvoltů po jednotky kiloelektronvoltů) [11]. Vzhledem k této vlastnosti jsou tyto diody použity i jako bolometry, byť závislost jejich odezvy na energii fotonů není tak jednoduchá jako u tepelných bolometrů. Před ty detektory, které mají měřit měkké rentgenové záření, se umisťuje vhodný filtr, například beryliová fólie o tloušťce 5 μm. Takto zastíněné AXUV diody jsou citlivé na záření o energii vyšší než přibližně 500 eV [11].

Prostorové rozlišení bolometrů i měkkých rentgenových detektorů je umožněno dírkovou komorou. Detektor má podobu 20 nebo 35 AXUV diod v poloidálně orientované řadě. Mezi diodami a plazmatem je vloženo stínítko se štěrbinou propouštějící záření. Tím je dosaženo rozlišení 20 nebo 35 úhlů (měřených od štěrbiny) v poloidálním směru, rovnoběžném s páskem diod. V dalším poloidálním směru, tedy podél zorných úhlů jednotlivých diod, je signál integrován, protože plazma je pro měřené záření průhledné. Ve směru toroidálním tento druh detektoru žádné rozlišení neposkytuje, pokud není použito více detektorů v různých toroidálních úhlech (na COMPASSu není); Nezbývá tedy než předpokládat v tomto směru nějaký druh symetrie.

V současnosti jsou na COMPASSu nainstalovány tři dírkové komory pro měkké rentgenové záření, dvě z nich s 35 diodami, třetí se 20 diodami; Na obrázku 2 je schéma jejich zorných úhlů. Podobné tři dírkové komory slouží jako bolometry. Data ze všech dírkových komor jsou sbírána s frekvencí 2 MHz [6]; Za účelem omezení šumu se tato data průměrují přes delší časový interval.

2.3 Tomografie

Ze signálů měřených několika dírkovými komorami s prostorovým rozlišením v jednom směru lze (za nějakých předpokladů) rekonstruovat dvojrozměrný obraz, odpovídající rozložení emisivity na průřezu komorou. *Tomografie* je obecný název pro metody k tomu účelu používané.

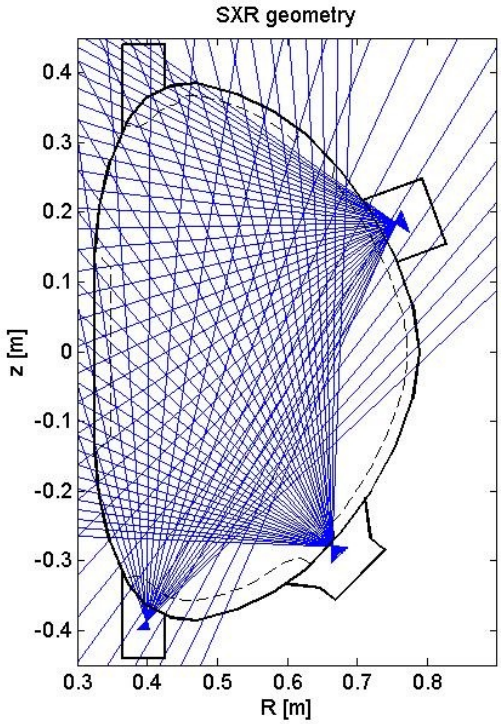
Existuje řada tomografických metod, vhodných pro různé situace. Pokud je k dispozici dostatečné množství dat, tj. detektory sledují objekt z mnoha různých směrů, dá se k rekonstrukci použít *filtrovaná zpětná projekce* nebo nebo některá z jí příbuzných metod. V zásadě jde o to, že s detektory zaznamenávajícími signál z objektu zacházíme jako se zdroji, které svůj signál do měřené oblasti promítají. Získaná rekonstrukce je přibližná (tím přesnější,

1 Vykládáno jako *Absolute X-ray UltraViolet*, *Absolute eXtreme UltraViolet*, případně *Absolute eXtended UltraViolet*

čím lépe je sledovaná oblast pokryta detektory) a vyskytují se v ní typické artefakty, které se dají částečně odstranit filtrací signálů před zpětnou projekcí. Její zásadní výhodou je nízká výpočetní náročnost, a tedy použitelnost i pro velká rozlišení výsledného obrazu [12]. Za cenu několikanásobného zvýšení výpočetní náročnosti se dají její výstupy zlepšit iterovanými opravami [12]. Popsané vlastnosti předurčují filtrovanou zpětnou projekci k použití v řadě medicínských aplikací, kde je požadováno vysoké rozlišení, a přitom je k dispozici dostatečné množství dat (například díky tomu, že se dá detektory otáčet kolem pacienta).

V tomografii plazmatu bývá dat poměrně málo (viz obrázek 2). Proto je nutné zavést určitá zjednodušení, jako jsou předpoklady o symetrii nebo hladkosti rekonstruovaného obrazu. Například pokud lze předpokládat, že profil emisivity je kruhově symetrický, lze této symetrie využít a rekonstruovat obraz z dat naměřených jedinou dírkovou komorou [10]. Matematický základ pro příslušný algoritmus představuje Abelova transformace. V řadě případů (jako je měření polohy plazmatu) ovšem takovéto přiblížení nestačí, a je třeba použít obecnější metody.

Po zavedení případných symetrií spočívá obvyklý další postup ve vyjádření všech přípustných řešení (obrazů) ve tvaru lineárních kombinací konečné množiny bázových funkcí. Bázovými funkcemi mohou být pixely, tedy funkce konstantní na zvolené oblasti, jinde nulové a navzájem se nepřekrývající, nebo nějaké hladší funkce, jako jsou Zernikovy polynomy do zvoleného řádu [10]. Dále pro každou z bázových funkcí určíme, jaký vliv má jí odpovídající profil emisivity na údaje jednotlivých detektorů. Tím získáme matici koeficientů pro přepočet obecného obrazu (z množiny přípustných) na signály detektorů. Abychom ze signálů získali obraz, je třeba vzniklou soustavu lineárních rovnic vyřešit. Kvůli tomu má tato metoda vyšší výpočetní náročnost než filtrovaná zpětná projekce, u níž se jen násobí vektor předem danou maticí[1].



Obr. 2: Směry pohledu měkkých rentgenových detektorů na tokamaku COMPASS;
Převzato z [13]

2.4 Tichonovova regularizace

Zatímco filtrovaná zpětná projekce může poskytovat nepřesná řešení, u druhé z popsaných metod to v ideálním případě nehrozí; Ideální je, když množina předpokládaných řešení zahrnuje řešení skutečné, a známe s dostatečnou přesností data i matici koeficientů. Na rozdíl od zpětné projekce může ovšem být výsledek explicitního řešení soustavy lineárních rovnic nejednoznačný. To nastává tehdy, když matice soustavy není regulární (například když je bázových funkcí více než detektorů, což je typický případ [14]).

Dalším problémem je skutečnost, že tomografická úloha bývá *špatně určená* (*ill-posed*) [10, 14]. To především znamená, že její řešení (respektive množina jejích řešení) silně závisí na datech. V případě, že je v datech přítomen šum, lze očekávat, že šum bude vidět i v nejlepším z přesných řešení, a bude zesílený. K získání použitelného výsledku i v takovémto případě se používají techniky známé jako *regularizace*. Výsledné řešení nesplňuje původní úlohu přesně, ale úroveň šumu v něm může být výrazně snížena.

Jedna z používaných regularizačních metod je založena na *rozkladu na singulární hodnoty* (*singular value decomposition*, *SVD*). Jde o vyjádření matice jako součinu tří jiných matic, z nichž dvě jsou ortogonální a jedna diagonální (čísla na její diagonále jsou ty singulární hodnoty). Lineární soustavy s ortogonální maticí se řeší snadno, proto lze SVD využít čistě jako prostředek k řešení obecných soustav. Kromě toho ale také lze po provedení rozkladu nahradit výsledné matice jinými, s vynechanými řádky a sloupci odpovídajícími malým singulárním hodnotám. Řešení takto upravené úlohy představuje přiblížení řešení přesného, ale na datech závisí stabilně. Metoda je známa jako *truncated SVD* [10].

Jednočíselné údaje, jako je poloha plazmatu (poloha těžiště profilu emisivity) nebo celková emisivita, se dají s pomocí SVD a podobných metod vypočítat samostatně, aniž bychom rekonstruovali celý obraz a tyto veličiny vypočítali z něj. Tento postup je velmi rychlý (viz např. [15]). Předpokládá se při něm ovšem, že známe matici tomografické úlohy předem a můžeme předem spočítat její SVD. Kromě toho je velmi obecný, nevyužívá žádných dodatečných znalostí, které o plazmatu můžeme mít, což je nevýhoda, pokud je máme [10].

Další, zajímavější regularizační metodou je *Tichonovova regularizace*. Oproti předchozí metodě vyžaduje mít o plazmatu i jiné informace než ty, které zaznamenávají detektory (*a-priori* informace). S těmito informacemi hledáme obraz, který s přiměřenou přesností řeší tomografickou soustavu rovnic, a přitom dostatečně dobře splňuje a-priori očekávání [10, 16]. Pokud zapíšeme rovnici neregularizované tomografické úlohy jako

$$\boldsymbol{T}\,\boldsymbol{g} \;=\; \boldsymbol{f}\,, \tag{1}$$

kde f_j je údaj j-tého detektoru a g_i koeficient i-té bázové funkce, pak úloha po Tichonovově regularizaci vypadá takto:

$$\|\boldsymbol{T}\,\boldsymbol{g} - \boldsymbol{f}\|^2 \;+\; \lambda\,\mathrm{O}(\boldsymbol{g}) \;=\; min. \tag{2}$$

Písmeno O značí *hodnotící funkcionál* (*Objective functional*) vyjadřující a-priori předpoklady, přičemž nižší hodnota O($\boldsymbol{g}$) znamená, že řešení $\boldsymbol{g}$ lépe splňuje podmínky.

Hodnotu parametru λ (kladné reálné číslo) musíme zvolit podle našich představ o

kompromisu mezi tomografickou rovnicí a hodnotícím funkcionálem. Jedním z oblíbených kritérií, jak zvolit λ, je *L-křivka*. Někdy se ukazuje, že pro hodnoty λ menší než nějaká mez je hodnota funkcionálu O ve vztahu (2) mnohem menší než hodnota prvního členu, zatímco pro λ větší než ta mez je to opačně. Vhodným kompromisem mezi řešením rovnice a splněním a-priori podmínek se pak zdá být volba λ blízkého zmíněné mezi [16]. Parametrický graf v souřadnicích odpovídajícím logaritmu členů $\| \boldsymbol{T}\,\boldsymbol{g} - \boldsymbol{f} \|^2$ a $O(\boldsymbol{g})$ parametrizovaných λ by v takovém případě měl mít tvar L. Jinou možností je zvolit λ tak, aby člen $\| \boldsymbol{T}\,\boldsymbol{g} - \boldsymbol{f} \|^2$ (kvadrát normy *rezidua*) nabýval nějaké dané hodnoty; Tato hodnota může odpovídat tomu, jakou úroveň šumu očekáváme ve vektoru $\boldsymbol{f}$ (ale ne v $\boldsymbol{g}$). V obou případech je kvůli volbě λ potřeba vyřešit úlohu (2) mnohokrát, pokaždé s jiným λ.

2.5 Minimalizace Fisherovy informace

Jednou z věcí, které o profilu emisivity plazmatu víme, je, že v místech mimo komoru tokamaku je emisivita nulová. Pokud definiční obor tomografické rekonstrukce do těch míst zasahuje, můžeme této znalosti využít jako a-priori informace. K úplné regularizaci tomografické rovnice to ovšem nestačí [10]; V pixelovém případě stále zbývá dost pixelů, k jejichž hodnotám se tato podmínka nevyjadřuje.

Dále lze předpokládat, že profil emisivity plazmatu je spojitý, ba dokonce hladký [10, 14]. Hodnotící funkcionál odpovídající této podmínce může mít podobu kvadrátu normy nějaké kombinace prostorových derivací (respektive konečných diferencí) profilu emisivity reprezentovaného vektorem $\boldsymbol{g}$. Zhruba platí, že nulté a první derivace vedou k výsledkům s podhodnocenými hodnotami emisivity; Často používanou volbou jsou druhé derivace, které podhodnocují emisivitu o něco méně. Vzhledem k tomu, že rekonstruovaný obraz má dva prostorové rozměry, lze volit derivace izotropní nebo preferovat některý směr. Pokud známe tvar magnetického pole, můžeme předpokládat, že ve směru sledujícím povrchy konstantního magnetického toku bude profil hladší než v kolmém směru [10, 14].

Práce [17] navrhuje využití ještě jiného funkcionálu, který má význam hladkosti; Ve dvojrozměrné pixelové podobě ho lze zapsat takto:

$$O(\boldsymbol{g}) = \sum_i \frac{(\partial_x g_i)^2 + (\partial_y g_i)^2}{g_i} \quad . \tag{3}$$

Tomuto funkcionálu se říká Fisherova informace, pro jeho souvislost se stejnojmennou veličinou z matematické statistiky[2]. Podobá se kvadrátu normy první derivace, ale příspěvky derivací v jednotlivých pixelech jsou děleny hodnotou daného pixelu. Motivací pro zavedení takto definovaného funkcionálu byla snaha omezit podhodnocení výšky píků v profilu emisivity, ke kterému dochází při použití obyčejné první nebo druhé derivace; Hladkost profilu je požadována přednostně v místech, kde je absolutní velikost emisivity malá.

Praxe (testování na umělých datech) ukazuje, že použití Fisherovy informace v roli hodnotícího funkcionálu skutečně zlepší přesnost rekonstruovaných výšek píků a rozlišení

2 V matematické statistice Fisherova informace vyjadřuje vztah mezi pravděpodobnostním rozdělením závislém na nějakém parametru a hodnotou toho parametru; Zde bereme za parametr i náhodný jev polohu pixelu, a hodnotu emisivity v pixelu interpretujeme jako pravděpodobnost realizace dané polohy [17]

některých detailů ve srovnání s neváženou první a druhou derivací [17]. Její nevýhodou je fakt, že umí velikosti píků kromě podhodnocování i nadhodnocovat, takže nemůže být interpretována jako jejich dolní odhad [10].

2.6 Časová optimalizace výpočtu

Výpočet vektoru konečných diferencí z vektoru $\boldsymbol{g}$ (s vhodným ošetřením podmínek na okraji rekonstruovaného profilu) lze zapsat jako součin $\boldsymbol{g}$ s vhodnou maticí. Celá levá strana úlohy (2) je tudíž kvadratickou funkcí $\boldsymbol{g}$, a úlohu lze vyřešit poměrně přímočaře. Levou stranu analyticky zderivujeme podle jednotlivých složek $\boldsymbol{g}$, čímž dostaneme soustavu lineárních rovnic pro $\boldsymbol{g}$, kterou následně vyřešíme některým z algoritmů k tomu určených. Když lze hodnotící funkcionál vyjádřit jako $\mathrm{O}(\boldsymbol{g}) = \|\boldsymbol{H}\,\boldsymbol{g}\|^2$, má úloha po derivaci tvar

$$(\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T} + \lambda \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})\boldsymbol{g} = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{f}. \tag{4}$$

Jelikož parametr λ obvykle neznáme předem, je třeba ho určit až během výpočtu, podle některého z výše popsaných kritérií. Vzhledem ke složitosti závislosti používaných kritérií na hodnotě λ k tomu úkolu přistupujeme numericky. Kritérium interpretujeme jako funkci parametru λ, jejíž hodnotu pro dané λ získáme tak, že vypočteme $\boldsymbol{g}$ z úlohy (2) s dosazeným λ, a výsledné $\boldsymbol{g}$ dosadíme do kritéria. Nyní hledáme minimum (maximum) této funkce nebo její průchod nějakou hodnotou; Na COMPASSu se používá podmínka na velikost šumu v datech, takže nejde o minimalizaci, ale o hledání kořene [18].

Výpočet funkční hodnoty popsané funkce je poměrně náročný, proto je dobré k hledání kořene rychle konvergující metodu. Nabízí se metoda sečen, která má řád konvergence přibližně 1.6, přičemž nevyžaduje znalost derivace. V některých případech se ovšem ukazuje, že z hlediska stability je lepší použít například metodu tětiv, byť formálně konverguje pomaleji [19].

Fisherova informace situaci ještě zhoršuje, protože není kvadratická, a po derivaci se z úlohy (2) stane soustava nelineárních rovnic pro $\boldsymbol{g}$, kterou je také nutné řešit iteračně. Jedním možným postupem je použít pro vážení příspěvků derivací do normy nějaké přiblížení $\boldsymbol{g}$ místo jeho skutečné hodnoty, která je v úloze neznámou. Tím se úloha vrátí do kvadratického tvaru, který vyřešíme včetně určení vhodné hodnoty λ. Vypočtené $\boldsymbol{g}$ použijeme v dalším iteračním kroku k vážení derivací. Takovýchto cyklů je třeba provést tři [20] až šest [21], podle požadované přesnosti přiblížení Fisherovy informace.

Popsaný postup počítá se dvěma vnořenými iteračními cykly. "Vnější" cyklus zpřesňuje přiblížení funkcionálu Fisherovy informace; V každém jeho kroku je třeba provést přiměřený počet kroků "vnitřního" cyklu, který hledá λ; V každém kroku vnitřního cyklu se řeší soustava lineárních rovnic. Existuje řada možností, jak tento postup urychlit.

Zmíněná soustava lineárních rovnic má jednu rovnici pro každý pixel výsledného obrázku. Nabízí se tedy zrychlit výpočet snížením počtu pixelů. Poměrně snadné je vynechat z výpočtu ty pixely, jejichž emisivita musí být nulová (mimo komoru); Je pouze třeba smysluplně vyjádřit okrajové podmínky v hodnotícím funkcionálu, když okraj kopíruje tvar stěny komory. Dále můžeme snižovat rozlišení (zvětšovat pixely), ovšem zdá se, že od jistého rozlišení níže rekonstrukce postupně ztrácí svou přesnost [21]; Přesnost je zde posuzována podle normy rozdílu vektoru pixelů použitého jako fantomový profil a vektoru rekonstruovaných pixelů, a

ne například podle odchylky určené polohy plazmatu. Další problém se snižováním počtu pixelů je, že čím menší rozlišení máme, tím menšího časového zlepšení dosáhneme jeho dalším zmenšením. Celá úloha může mít asymptotickou časovou složitost například $O(n^3)$, kde n je počet pixelů, ale při malém počtu pixelů se projeví ty části výpočtu, jejichž časová složitost je v n lineární, nebo na n nezávisí vůbec [18].

Jinou možností urychlení je použít stejné váhy derivací a stejnou hodnotu λ pro více snímků. Tyto snímky by měly být podobné, například zaznamenané během krátkého časového intervalu. Jako přiblížení vah pak lze použít převrácené hodnoty emisivity průměrované přes všechny uvažované snímky [20]. Řeší se pak jedna soustava rovnic s více pravými stranami, přičemž každá pravá strana odpovídá jednomu snímku. Výsledkem je několikanásobné (v závislosti na počtu snímků najednou) urychlení celkového výpočtu za cenu nepřesnosti použitého přiblížení Fisherovy informace. Z hlediska výpočtu v reálném čase je špatné, že tato metoda nezkrátí čas mezi vstupem dat a výstupem výsledku (ve skutečnosti ho prodlouží, zejména uvážíme-li, že je třeba počkat na data z více okamžiků).

Práce [19] podává další návrh, jak urychlit výpočet. Oba cykly se dají sjednotit tím, že místo konstantního přiblížení $\boldsymbol{g}$ pro vážení derivací použijeme v každém kroku hledání λ vektor $\boldsymbol{g}$ vypočtený v kroku předchozím. Metoda hledající λ musí být poměrně robustní, aby neztratila stopu, když se zkoumaná funkce mění; Přesto se ukazuje, že celkový počet potřebných iteračních kroků není větší než u algoritmu se dvěma cykly, byť se v něm dá využít rychleji konvergující metoda pro λ [19].

Podstatného urychlení lze následně dosáhnout při výpočtu řady snímků následujících po sobě s krátkými časovými odstupy. Jako počáteční přiblížení $\boldsymbol{g}$ pro vážení derivací je možné použít výsledek z předchozího snímku; Podobně lze použít hodnotu λ z předchozího snímku. Vzhledem k tomu, že po sobě následující snímky jsou si často velmi podobné, může se snížit počet iteračních kroků používaných k upřesnění jednotlivých snímků; Pro výpočty v reálném čase může dokonce stačit jeden iterační krok na každý snímek [19]. Výsledek pro zvolený snímek přitom nezáleží na datech z následujících časů, takže snížení počtu iteračních kroků vskutku odpovídá zkrácení času mezi vstupem dat a výsledkem. Metoda se dá zřejmě použít i v případě kvadratických hodnotících funkcionálů, kdy se z předchozích snímků použije pouze odhad λ.

V případě, že je počet iteračních kroků omezen časem místo požadované přesnosti výsledku, můžou být výstupy algoritmu nepřesné. K tomu může docházet například při rychlých změnách profilu emisivity plazmatu, tedy v situaci, kdy předchozí snímek není dobrým přiblížením dalšího snímku. Práce [19] navrhuje rozpoznávat poškozené snímky právě podle vhodné normy rozdílu po sobě jdoucích snímků, a vyloučit je z případných zpětnovazebních algoritmů. K tomu je ovšem dobré, aby těch poškozených snímků bylo relativně málo.

3 Využití programovatelných hradlových polí

Programovatelné hradlové pole je integrovaný obvod, jehož struktura může být rozsáhle konfigurována. Jeden druh součástky tak může být optimalizován pro rozličné účely. Konfigurace přitom neprobíhá ve výrobě, ale u jednotlivých uživatelů, a u moderních obvodů je mnohonásobně opakovatelná. První dvě slova anglického označení *Field-programmable Gate Array* (*FPGA*) odkazují právě na tuto vlastnost.

První poměrně malá FPGA byla používána jako programovatelné propojení mezi jinými součástkami, které dovolovalo odstraňování některých designových chyb, aniž by bylo nutné navrhovat celý obvod znovu [22]. S rostoucí složitostí našla FPGA široké uplatnění v digitálním zpracování signálů či v prototypování elektronických obvodů.

Architektura moderních FPGA se u různých výrobců liší v menších i větších detailech, základ je ovšem vždy podobný. Programovatelná struktura je tvořena mřížkou *logických bloků*[3], které se dále dělí na menší obvody zvané *slices*. Každý logický blok, respektive každý slice, může vykonávat jinou funkci, danou naprogramováním; Jejich vzájemné propojení je rovněž konfigurovatelné. Typický slice zahrnuje vyhledávací tabulku, malé paměťové elementy, případně další jednoduché prvky. Logických bloků bývá v čipu několik stovek až několik stovek tisíc[4]. Kromě nich obsahují FPGA více specializované obvody, jako jsou vstupně výstupní rozhraní, paměťové bloky, sčítačky a násobičky, procesorová jádra, někdy dokonce analogové obvody [23].

Podobně jako obvody ASIC[5] umožňují FPGA významné urychlení některých výpočtů v porovnání s počítáním na konvenčním procesoru[6] [24]. Pro řadu aplikací jsou ovšem FPGA vhodnější. Návrhový cyklus jejich firmwaru (jak se říká strojovému popisu jejich konfigurace) je rychlejší a jednodušší, hlavně proto, že nezahrnuje výrobu čipu ani omezení daná obtížnou testovatelností designu (před vyrobením čipu lze nanejvýš provádět počítačové simulace) [25]. Na druhou stranu, ASIC je typicky rychlejší, má menší spotřebu, a jeho výroba ve velkých sériích je výhodnější [26]. V některých případech je možné minimalizovat náklady na vývoj obvodu tak, že počáteční verze vyvíjíme na FPGA, a po dostatečném otestování převedeme design do podoby ASIC. U malých aplikací se ovšem obvykle nevyplatí ani takový postup, a s FPGA je tedy možné se setkat i u zařízení v běžném provozu.

3.1 Vývoj pro FPGA

K designu firmwaru pro FPGA je třeba přistupovat jinak než u procesoru. Procesor umožňuje vykonávat dlouhé série operací v jediném obvodu. V principu lze zařadit do designu FPGA víceúčelové obvody, jako jsou procesory; Problém je v tom, že takový design by nebyl výkonnější než procesor realizovaný jednoúčelovým obvodem. K optimalizaci algoritmů

3 Různí výrobci nazývají své logické bloky různě; Označení *Configurable Logic Block* používá Xilinx.

4 Xilinx XC6SLX4 (Spartan 6): 600 slices [27], Xilinx XC7V2000T (Virtex 7): 305 400 slices [23]

5 *Application-Specific Integrated Circuit*, česky obvykle *zákaznický integrovaný obvod*, je zařízení vyráběné pro jednoho zákazníka a pro jednu zamýšlenou aplikaci

6 Procesory lze v jistém smyslu také považovat za ASIC, ale jejich specifickou aplikací není konkrétní výpočetní algoritmus

s využitím FPGA lze ovšem přistupovat jinak. Velikost dnešních FPGA umožňuje vysokou úroveň paralelismu. Je tedy možné (a vhodné) pro každou část problému navrhnout speciální obvod a všechny tyto jednoduché obvody nakonec začlenit do jednoho zařízení [22]. Dílčí obvody přitom mohou být jednoúčelové, nemusí obsahovat infrastrukturu potřebnou pro programovatelnost a mohou být vysoce optimalizované. V zájmu prostorové optimalizace lze některé moduly používat ve více fázích výpočtu, v řadě případů to ovšem může mít negativní vliv na rychlost[7].

Masivně paralelní algoritmy nám dávají nové možnosti transformace mezi časovou a prostorovou složitostí. Kromě toho se u nich (a rovněž u specializované elektroniky) objevuje více veličin popisujících rychlost výpočtu, které lze optimalizovat zvlášť. Z hlediska algoritmů jsou podstatné veličiny *zpoždění* a *propustnost*[8]: Zpoždění je čas mezi vstupem dat a těmto datům odpovídajícím výsledkem, zatímco propustnost označuje množství zpracovaných dat (či počet užitečných výstupů) za jednotku času [28]. Různé aplikace kladou různé požadavky na oba druhy časové složitosti; Naproti tomu prostorová složitost je omezena velikostí použitého čipu.

Podobně jako procesory využívají FPGA hodinového signálu (byť některé relativně jednoduché, *kombinatorické*, obvody mohou fungovat i bez něj). Zpoždění a propustnost je tak možné popisovat periodami tohoto signálu namísto nanosekund. To nám umožní zavést ještě jednu nezávislou veličinu popisující časovou složitost: *timing*. Zjednodušeně řečeno jde o nejvyšší frekvenci hodinového signálu (*takt*), při níž obvod pracuje správně; Při vyšších frekvencích se výstup dílčích kombinatorických obvodů nestačí ustálit na správné hodnotě během jedné periody, a obvody na ně navazující dostanou špatný vstup.

Timing lze optimalizovat rozdělováním kombinatorických obvodů na menší. Pokud na sebe tyto menší obvody navazují, má rozdělení negativní vliv na zpoždění měřené v hodinových ticích. Propustnost (v ticích) přitom může zůstat stejná, pokud využijeme *pipeliningu*. Při něm prochází řetězem navazujících obvodů více balíků dat najednou (ideálně tolik, kolik je obvodů), každý balík v jiné fázi výpočtu. Optimalizovaný obvod má vyšší propustnost měřenou v balících dat za sekundu. Mnohem detailněji je problematika časové optimalizace popsána v [28].

U FPGA je navíc obvyklé mluvit o novém druhu prostorové složitosti: Nejde pouze o místo v paměti, nýbrž především o místo, které na čipu zabírají použité logické obvody [25]. Tato prostorová složitost je omezena velikostí použitého čipu, tedy počtem dostupných konfigurovatelných součástek. V případě, že potřeby designu překročí dostupné zdroje, je třeba buď snížit prostorovou náročnost algoritmu (zpravidla za cenu zvýšení některého druhu časové náročnosti [28]), nebo použít větší čip.

Zpracovávaná data mohou být uložena ve stejných logických blocích, které jsou používány k jejich zpracování (*distribuovaná paměť*), ale existují také specializované *blokové paměti*, tedy paměťové moduly rozmístěné mezi logickými bloky, případně lze k FPGA připojovat externí paměťové moduly (pokud příliš nezáleží na rychlosti přístupu k paměti); To činí logickou

7 Pro opakované používání dílčích obvodů jsou optimalizovány například architektury GPU; Jejich hlavní nevýhodou je pomalost přenosu dat [29].

8 Často pojmenovávané anglicky, totiž *latency* a *throughput*

složitost algoritmů na FPGA do jisté míry nezávislou na složitosti paměťové.

3.2 VHDL

VHDL je jedním ze dvou nejrozšířenějších jazyků pro popis hardware; Písmena *HDL* značí *hardware description language*. Podobně jako nakreslené schéma může kód napsaný ve VHDL reprezentovat elektrický obvod (obvykle digitální). Tuto reprezentaci lze pomocí k tomu určeného softwaru zpracovat do podoby schématu a následně vyrobit obvod ASIC. Také je možné překládat VHDL do konfiguračních souborů pro různé architektury FPGA.

Překlad VHDL pro FPGA probíhá v několika krocích. Během nich se popis obvodu mění z poměrně obecného VHDL v reprezentace čím dál tím více přizpůsobené konkrétnímu zařízení. Mezi jednotlivými fázemi se dá chování obvodu simulovat. K tomu slouží například software, který místo elektronických schémat produkuje spustitelné soubory, které pak lze na běžném počítači spouštět se zvolenými vstupy a kontrolovat jejich výstupy. Simulace v jednotlivých fázích překladu jsou různě podrobné a hodí se k rozpoznávání různých druhů chyb [29]. Také se dá s rostoucí přesností odhadnout prostorová složitost navrhovaného obvodu.

Ve VHDL lze pracovat na různých úrovních abstrakce. Je možné skládat obvody z jednotlivých hradel. Kromě toho disponuje jazyk vysokoúrovňovými datovými typy (například různými reprezentacemi celých a racionálních čísel), operátory a funkcemi (sčítání, násobení, porovnávání, bitové posuny) a dalšími konstrukcemi, jejichž detailní reprezentaci odvodí software během některé z fází překladu. Dále lze zapouzdřit menší části designu do *entit*, a tyto entity používat v dalším kódu jako součástky s definovanými vstupy a výstupy, aniž by bylo třeba popisovat jejich strukturu pokaždé znovu.

Některé konstrukce ve VHDL nemají svou obdobu v elektronických obvodech, případně v tom typu obvodů, které jsme se rozhodli používat. Říkáme, že nejsou *syntetizovatelné*, protože *syntéza* je název té fáze překladu, v níž se objeví problémy (a někdy se tak také říká všem fázím překladu dohromady). Jde například o čtení a zápis souborů, popis zpoždění mezi po sobě následujícími událostmi nebo inicializace signálů na zvolenou hodnotu při zapnutí obvodu [29]. Tyto konstrukce ovšem nacházejí využití v testování a simulacích, kde se s nimi například generují signály, které v praxi mají přicházet do obvodu zvenčí; Za účelem testování se obvykle k testovanému designu přidávají moduly, které se pak ze syntézy vynechají.

Výrobci FPGA i jiné firmy poskytují již implementované entity (nebo software generující tyto entity na míru), které provádějí některé často používané operace. Říká se jim *IP jádra*, přičemž IP znamená *Intellectual Property*. Příkladem funkcí IP jader mohou být aritmetické operace (v nějakém směru efektivnější než ty, které vykonávají standardní operátory VHDL), komunikační protokoly nebo procesorová jádra.

Vedle VHDL se v současnosti k programování FPGA používá jazyk Verilog, s odlišnou syntaxí, ale podobným principem. Existují i prostředky umožňující sofistikované kreslení schémat, jako jsou System Generator od firmy Xilinx nebo LabVIEW FPGA od National Instruments. Tyto grafické jazyky poskytují ještě vyšší úroveň abstrakce, a v některých případech mohou být přehlednější než klasické HDL. V jejich syntéze je jeden krok navíc, totiž překlad

grafického kódu do HDL [30]. Kromě grafických jazyků se objevují i pokusy o programování FPGA ve vyšších programovacích jazycích, například v různých dialektech C [30].

3.3 Řešení soustav lineárních rovnic

Soustavy lineárních rovnic jsou často řešeným problémem, a byla pro ně vyvinuta řada různých algoritmů. Některé algoritmy jsou univerzální, zatímco jiné mohou být rychlejší, ale fungují jen pro speciální soustavy. Jednotlivé algoritmy také nabízejí různé možnosti paralelizace, konkrétněji pak různé možnosti optimalizace s použitím FPGA.

Matice soustavy řešené při tomografii s Tichonovovou regularizací má tvar $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T} + \lambda \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$, kde $\boldsymbol{T}$ je matice koeficientů přepočtu mezi daty a pixely a $\boldsymbol{H}$ je například matice konečných diferencí. Parametr λ je obecně různý pro různé snímky; Při použití Fisherovy informace se mění i $\boldsymbol{H}$. Díky regularizaci je matice regulární, a tedy úloha jednoznačně řešitelná.

Z vyjádření přes $\boldsymbol{T}$ a $\boldsymbol{H}$ je zřejmé, že matice úlohy je symetrická a pozitivně semidefinitní[9]. Vzhledem k tomu, že matice $\boldsymbol{H}$ je volena tak, aby výsledná matice nebyla singulární, a parametr λ je kladný, je matice také pozitivně definitní. To umožňuje použít k výpočtu Choleského faktorizaci.

Pro dostatečně velké λ má matice převládající diagonálu (prvek na hlavní diagonále je vždy v absolutní hodnotě největším prvkem na svém řádku). To je dáno tím, že matice $\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$ má sama převládající diagonálu, pokud je odvozena z matice konečných diferencí. Tato vlastnost zaručuje konvergenci některých iteračních metod pro řešení lineárních soustav, totiž metody Jacobiho [31] nebo Gaussovy–Seidelovy [32].

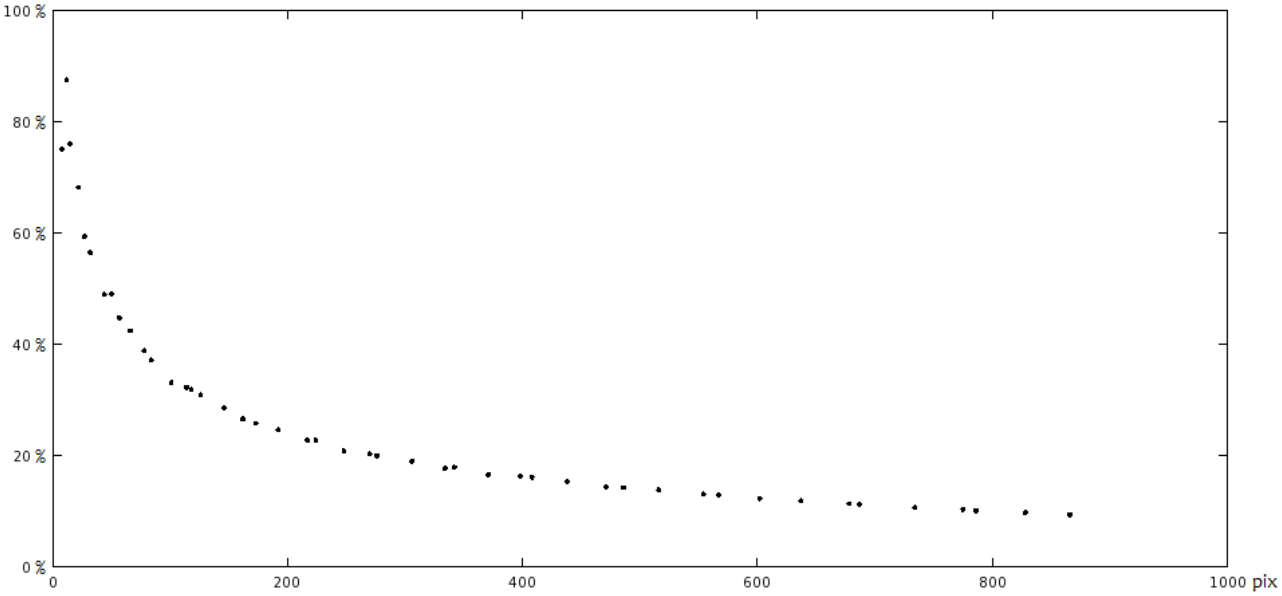
Matice $\boldsymbol{T}$ bývá řídká, protože do zorného úhlu jednotlivých detektorů zasahuje vždy jen malý počet pixelů [18]. Matice $\boldsymbol{H}$ je také řídká; Vyjadřuje-li například první konečné diference, přísluší typickému pixelu (a tudíž typickému řádku) dva nenulové prvky. Díky tomu je řídká i matice řešené úlohy. To umožňuje zavést optimalizace pro řadu výše zmíněných algoritmů [33]. Z hlediska algoritmické složitosti jde ovšem o zhoršení. Kromě toho, pro malé počty pixelů je matice $\boldsymbol{T}$ méně řídká, což vede k menšímu zlepšení při použití řídkých algoritmů; Obrázek 3 ukazuje příklad vývoje řídkosti řešené matice v závislosti na počtu pixelů.

3.4 Gaussova eliminace

Předpokládejme soustavu n rovnic pro n neznámých, tedy soustavu s maticí rozměru $n \times n$; U tomografické úlohy n odpovídá počtu pixelů. Pro zjednodušení dalších úvah předpokládejme, že matice soustavy je regulární, což matice regularizované tomografické úlohy je.

Algoritmus Gaussovy eliminace ve své nejznámější podobě počítá s obdélníkovou *rozšířenou maticí soustavy*, v níž prvních n sloupců je tvořeno sloupci původní matice soustavy, a jako poslední sloupec dosadíme vektor pravé strany. S touto rozšířenou maticí provádíme *řádkové úpravy*, čímž převedeme původní soustavu rovnic na jinou, ekvivalentní soustavu, tedy soustavu se stejnou množinou řešení. Řádkové úpravy volíme tak, aby matice nové soustavy byla trojúhelníková (tj. například všechny prvky pod hlavní diagonálou měla nulové).

9 Matice $\boldsymbol{A}$ je pozitivně semidefinitní, když $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v} \geq 0$ pro libovolný vektor $\boldsymbol{v}$; Pozitivně definitní, když $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v} > 0\ \forall \boldsymbol{v}$



Obr. 3: Možná závislost podílu nenulových prvků v matici regularizované tomografické úlohy na počtu pixelů (počítáno pro detektory měkkého rentgenového záření na COMPASSu)

Mezi řádkové úpravy patří *přičtení násobku jednoho z řádků k jinému řádku* a *záměna pořadí řádků*. Máme-li částečně upravenou rozšířenou matici, jejíchž prvních k řádků má trojúhelníkový tvar, můžeme odečíst vhodné násobky k-tého řádku od všech následujících řádků tak, aby se v k-tém sloupci každého z nich objevila nula; Tím jsme získali matici s prvními $k+1$ řádky ve správném tvaru. Pokud k-tý řádek má na k-tém místě nulu, a nelze tedy najít jeho "vhodné násobky", využijeme operaci záměny pořadí řádků.

Soustavu s trojúhelníkovou maticí umíme vyřešit pomocí *zpětného dosazování*: Vyřešíme rovnici v koutě trojúhelníka, která obsahuje jen jednu neznámou; Získanou hodnotu dosadíme do té rovnice, která má neznámé dvě, čímž získáme další rovnici s jednou neznámou; Takto postupujeme, dokud nevypočteme hodnoty všech neznámých.

Kód 1 ukazuje možnou implementaci Gaussovy eliminace v C++; Podle tří vnořených cyklů je zřejmé, že časová složitost neparalelizovaného (sekvenčního) algoritmu je $O(n^3)$. Část kódu provádějící zpětné dosazování má složitost $O(n^2)$.

Drobnou modifikací Gaussovy eliminace je metoda někdy označovaná jako Gaussova–Jordanova eliminace [34]. Při ní pomocí řádkových úprav převedeme matici soustavy do diagonálního tvaru, respektive do tvaru jednotkové matice (s využitím nové operace, *násobení řádku nenulovým číslem*). Zpětné dosazování pak není třeba, protože ve všech rovnicích se už vyskytuje jen jedna neznámá (v každé rovnici jiná). Řádkových úprav je ovšem třeba provést dvojnásobný počet, pročež je sekvenční varianta Gaussovy–Jordanovy eliminace pomalejší než první popsaný algoritmus.

Dalším algoritmem příbuzným s Gaussovou eliminací je LU faktorizace. Při ní se pracuje s nerozšířenou maticí, a je tedy nezávislá na pravé straně. Matici soustavy upravíme do trojúhelníkového tvaru, přičemž (inverzní) řádkové úpravy vyjadřujeme jako násobení nově vznikající matice jinou maticí [35]. Výsledkem je rozklad původní matice na součin dvou

```
 1 void gaussovaEliminace(double** A, double* x, int n){
 2     // A .. matice, x .. řešení, n .. počet neznámých
 3     // A[řádek][sloupec], 0 <= řádek < n, 0 <= sloupec <= n
 4
 5     // úprava do trojúhelníkového tvaru
 6     for(int k=0; k<n; k++){
 7         if(A[k][k] == 0){ /* záměna řádků */ }
 8
 9         for(int l=k+1; l<n; l++){
10             for(int m=n; m>=k; m--){
11                 A[l][m] -= A[k][m] / A[k][k] * A[l][k];
12             }
13         }
14     }
15
16     // zpětné dosazení
17     for(int k=n-1; k>=0; k--){
18         x[k] = A[k][n];
19         for(int m=n-1; m>k; m--){
20             x[k] -= A[k][m] * x[m];
21         }
22         x[k] /= A[k][k];
23     }
24 }
```

Kód 1: Algoritmus Gaussovy eliminace v C++;
Pro zjednodušení je vynechána implementace záměny pořadí řádků

trojúhelníkových matic, z nichž jedna odpovídá řádkovým úpravám; Jedna z matic má nenulové prvky na diagonále a nad ní (U), druhá na diagonále a pod ní (L). Pro získání řešení ke zvolené pravé straně provedeme dvakrát po sobě zpětné dosazování.

3.5 Paralelizace Gaussovy eliminace

Algoritmus Gaussovy eliminace poskytuje v zásadě dvě možnosti paralelizace. První možností je paralelizovat řádkovou operaci přičtení násobku jednoho řádku ke druhému, tedy provádět tuto operaci pro více prvků řádku najednou. Pokud ji dokážeme provést pro všechny prvky řádku najednou (pokud máme k dispozici dostatečný počet procesorových jader nebo odpovídajících obvodů v FPGA), má operace konstantní složitost, zatímco její sekvenční varianta má průměrnou složitost $O(n)$.

Možnou komplikací je potřeba distribuce dat mezi n paralelně pracujícími procesory; Podle kódu 1 jde o prvky `A[k][k]` a `A[l][k]` (respektive jen jejich podíl), použité ve všech krocích nejvnitřnějšího cyklu (na řádku 11). Na architekturách podporujících komunikaci pouze mezi jednotlivými uzly má tato operace (*broadcast*) logaritmickou složitost. Protože se broadcast provádí uvnitř dvou vnořených cyklů, má výsledný algoritmus časovou složitost $O(n^2 \log n)$; Prostorová složitost (např. počet procesorů) je $O(n)$.

Časovou složitost lze vylepšit s použitím pipeliningu (za cenu mírného zesložitění algoritmu, prostorová složitost zůstává stejná). Opět použijeme n procesorů, ale neprovádíme broadcast. Místo toho předáme prvky `A[k][k]` a `A[l][k]` jen prvnímu procesoru a necháme jej provést jeho část řádkové operace; Pak první procesor předá svá dvě čísla dalšímu procesoru v řadě, zatímco sám dostane nová dvě, potřebná k následující řádkové operaci. Každý procesor tak

předává data vždy jen jednomu dalšímu procesoru, a časová složitost celého algoritmu je $O(n^2)$. Podrobněji (včetně ukázek kódu) je jedna z možných podob pipeliningu popsána v [36]. Při použití FPGA lze typicky použít sběrnici, která provede broadcast v jednom kroku, takže zavádění tohoto pipeliningu není nutné (předpokládá se ovšem, že kapacita použité sběrnice roste s velikostí řešené soustavy).

Druhou možností paralelizace Gaussovy eliminace je paralelizovat oba vnitřní cykly zároveň, v kódu 1 řádky 9 až 13. Je to možné díky tomu, že výpočet pro jednotlivé prvky matice nezávisí na výsledcích ostatních výpočtů prováděných v rámci těchto dvou cyklů (závisí pouze na výsledcích z předchozích kroků vnějšího cyklu). Procesorů je v tomto případě potřeba $O(n^2)$. Při použití sběrnic nebo (dvojitého) pipeliningu je časová složitost v nejlepším případě $O(n)$; Nejhorší případ nastává, když je třeba často provádět operaci záměny řádků, a s ní spojené hledání řádku s nenulovým prvkem v aktuálním sloupci. Zpomalení v takovém případě závisí na postupu použitém pro to hledání.

Když provádíme řádkové operace algoritmem rychlejším než $O(n^2)$, je potřeba upravit i tu část algoritmu, která provádí zpětné dosazování. Je možné ji také paralelizovat, nebo použít Gaussův–Jordanův algoritmus, ve kterém tento krok není třeba. Při použití n^2 procesorů s pevnou distribucí prvků matice mezi procesory vyžaduje navíc první část Gaussova–Jordanova algoritmu stejný počet operací jako první část algoritmu Gaussova, takže paralelní Gaussův–Jordanův algoritmus je v tomto případě dokonce rychlejší (o čas potřebný ke zpětnému dosazování).

3.6 Numerická stabilita

Gaussova eliminace prováděná na počítači není zcela přesná; Přesnost vstupů, různých mezivýsledků i konečných výsledků je omezena přinejmenším použitou reprezentací čísel. Vzhledem k tomu, že dílčí výpočty navazují na výsledky předchozích výpočtů, můžou se chyby hromadit, a to do té míry, že konečný výsledek je nepoužitelný například pro tomografii. Ukazuje se, že algoritmus Gaussovy eliminace tak, jak byl výše popsán, tímto problémem trpí; Existují matice, pro které vychází řešení poněkud odlišné od řešení správného [37]. Pro větší matice je tento efekt výraznější.

Kromě zvyšování přesnosti reprezentace čísel se lze numerické nestabilitě vyhnout použitím některé z metod známých jako *pivotace*. Řádková pivotace spočívá v sofistikovaném výběru řádku, jehož násobek budeme odčítat od dalších řádků. Pokud vybereme řádek, který má na místě aktuálního sloupce nenulové, ale malé číslo, bude v kódu 1 na řádku 11 probíhat dělení tímto malým číslem; Při tom dojde ke zvětšení absolutní chyby v příslušném prvku matice (nemluvě o případném riziku přetečení reprezentace čísel). Místo řádku, který je na řadě, tedy raději vybereme ten řádek, který má na pozici aktuálního sloupce v absolutní hodnotě největší číslo (*pivot*) [34]. Tento postup pomáhá proti numerické nestabilitě Gaussova i Gaussova–Jordanova algoritmu přinejmenším u těch soustav lineárních rovnic, které nejsou příliš špatně určené [34].

Kromě řádkové pivotace existuje řada podobných metod, z nichž některé poskytují ještě vyšší stabilitu, zatímco jiné se více hodí pro paralelizaci. Při úplné pivotaci se vybírá absolutně největší prvek z celého dosud neupraveného zbytku matice, tj. nejen z aktuálního sloupce. Aby

vyšla trojúhelníková matice, je třeba sloupce (a pořadí neznámých) zaměňovat, případně lze použít takový algoritmus pro zpětné dosazování, který si poradí i s "permutovanou trojúhelníkovou" maticí. Algoritmus je ovšem stabilní i pro řadu matic, pro které řádková pivotace nestačí. Několik dalších možností pivotace je popsáno v [38].

3.7 Choleského faktorizace

Podobně jako LU faktorizace rozkládá Choleského faktorizace matici na součin dvou trojúhelníkových matic (horní a dolní), případně v jiné variantě na součin tří matic, z nichž prostřední je diagonální. Na rozdíl od dříve popisovaných metod se dá použít pouze pro matice symetrické a pozitivně definitní. Speciální vlastnosti matic umožňují, aby výstup Choleského transformace měl tvar

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{L}^{\mathrm{T}}, \tag{5}$$

kde $\boldsymbol{A}$ je původní matice soustavy a $\boldsymbol{L}$ je dolní trojúhelníková matice.

Po rozkladu na trojúhelníkové matice se postupuje zpětným dosazováním; Pro získání řešení soustavy $\boldsymbol{A}\,\boldsymbol{g} = \boldsymbol{f}$ se vyřeší nejprve soustava $\boldsymbol{L}\,\boldsymbol{h} = \boldsymbol{f}$, a následně $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{g} = \boldsymbol{h}$.

Algoritmy používané pro Choleského faktorizaci obecné (v rámci dosavadních předpokladů) matice mají kubickou asymptotickou časovou složitost, ale absolutní počet numerických operací je oproti Gaussově eliminaci přibližně poloviční [39]. Navíc jsou tyto algoritmy numericky stabilní i bez pivotace [39]. Drobnou nevýhodou může být v některých případech (jako je implementace na FPGA) složitější struktura přístupu do paměti.

4 Hardwarové testy

K vývoji a testování obvodu pro tomografii v reálném čase byl použit vývojový kit Atlys s FPGA čipem XC6SLX45 od firmy Xilinx. Dále je deska vybavena řadou periferií; Kromě několika konfigurovatelných tlačítek, přepínačů a LED diod je to sběrnice USB, vstupní i výstupní rozhraní HDMI, gigabitový Ethernet, audio konektory pro vstup i výstup a několik desítek volných pinů pro další využití [40]. Nejde tedy o zařízení určené k řízení tokamaku (to by vyžadovalo spíše samé volné piny), ale pro testování rozličných algoritmů vhodné je. Frekvence dostupného hodinového signálu je 100 MHz.

Čip na desce patří do nízkonákladové řady Spartan 6, uvedené na trh v roce 2009 (v současnosti nahrazena řadou Kintex 7). Má 6 822 obecných slices, každý slice vybavený čtyřmi vyhledávacími tabulkami (se šestibitovým vstupem) a osmi flip-flopy (paměťovými buňkami na jeden bit). Dále je k dispozici 116 paměťových bloků, každý s kapacitou 1024 osmnáctibitových slov, 58 DSP bloků (každý s osmnáctibitovou násobičkou a sčítačkou) a 358 pinů pro vstup i výstup [41].

K návrhu obvodu a k simulacím byla použita freeware varianta prostředí Xilinx ISE. Jejím hlavním omezením oproti variantě placené je fakt, že neumožňuje vývoj pro všechny čipy od Xilinxu; Čip XC6SLX45 mezi podporované patří.

Pro implementaci byla zvolena tomografická úloha s minimalizací normy prvních konečných diferencí a výpočtem regularizačního parametru z podmínky na velikost šumu. V případě potřeby lze hodnotící funkcionál upravit na minimalizaci Fisherovy informace s využitím postupu z práce [19] (za cenu mírného zvýšení prostorové složitosti, zejména počtu násobiček). Cílem bylo vypočítat jednu tomografickou rekonstrukci za nejvýše 50 mikrosekund, což je doba srovnatelná se současným zpožděním zpětné vazby na tokamaku COMPASS. Předpokládá se přitom, že pro výpočet jednoho snímku stačí jedna iterace algoritmu (především vyřešení jedné soustavy lineárních rovnic). Vzhledem k tomu, že deska ani čip nezahrnuje žádné hardwarové procesorové jádro, byly všechny části algoritmu realizovány jednoúčelovými obvody v FPGA.

Praxe potvrdila očekávání, že implementačně nejsložitější částí algoritmu bude řešení soustavy lineárních rovnic. Pro tento úkol byly pro svou přehlednost vybrány dvě varianty Gaussovy eliminace: maticově paralelizovaná bez pivotace a řádkově paralelizovaná s řádkovou pivotací.

Byla použita osmnáctibitová reprezentace čísel s pevnou desetinnou čárkou a dvojkovým doplňkem pro reprezentaci znaménka. Její hlavní výhodou je hardwarová podpora na výše zmíněném čipu (osmnáctibitové násobičky, blokové paměti). Reprezentace s plovoucí desetinnou čárkou by byla z numerického hlediska lepší (srovnatelná relativní přesnost u všech hodnot, při vhodné volbě exponentu menší riziko přetečení), ale pro sčítání a násobení takových čísel by bylo třeba implementovat tyto operace ze základních konstrukcí VHDL (prostorově neefektivní) nebo využít proprietární IP jádra.

4.1 Maticově paralelizovaná Gaussova–Jordanova eliminace

Jako první byla implementována Gaussova–Jordanova eliminace s lineární časovou složitostí a bez pivotace. Oproti popisu uvedenému v předchozí kapitole bylo provedeno několik úprav za účelem lepšího využití architektury FPGA, respektive zjednodušení struktury výpočtů obecně.

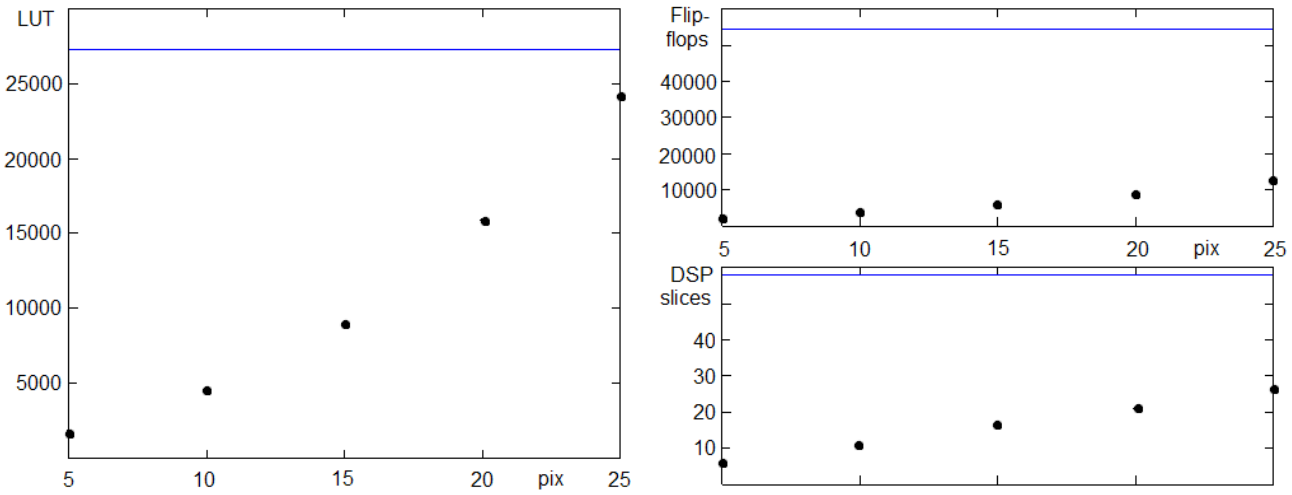
První změnou je normalizace řádku určeného k přičítání k ostatním řádkům. Jde o to, že všechny prvky na řádku se vydělí prvkem v aktuálním sloupci, takže v aktuálním sloupci je jednička. Následně se násobky tohoto řádku přičtou ke všem ostatním řádkům, přičemž díky předchozímu kroku už nyní není nutné žádné dělení. Vzhledem k tomu, že přičtení probíhá pro všechny řádky současně, a to vždy pro celý řádek zároveň, je pro tento krok třeba n^2 sčítaček a stejný počet násobiček. Děličky jsou třeba jen pro ten řádku, který se před úpravami normalizuje, což je dobré, protože jsou tvořeny IP jádry zabírajícími na čipu hodně místa (hardwarové děličky k dispozici nejsou).

Prostorová složitost se dá dále omezit tak, že místo dělení jednotlivých prvků pivotem vypočítáme převrácenou hodnotu pivotu, a touto hodnotou násobíme ostatní prvky. Aby se omezila numerická chyba, je možné použít k uchování převrácené hodnoty pivotu přesnější reprezentaci čísel než pro prvky matice samotné; Kvůli tomu nebude možné využít hardwarové násobičky (pracují jen do 18 bitů), ale i IP násobička zabírá méně místa než IP dělička.

Další změna spočívá v uložení matice do struktury umožňující operaci rotace ve vertikálním směru. Při této operaci se obsah každého řádku kromě prvního posune o pozici výš, a první řádek se vloží na místo původního posledního řádku. Struktura entity řešící soustavu se díky tomu kupodivu zjednoduší. Obvod provádějící normalizaci řádku stačí připojit jen k jednomu řádku (prvnímu), a obvody přičítající násobek řádku k těm ostatním. Podobnou strukturu, podporující navíc šikmou rotaci, navrhuje práce [42]; Tam ovšem nejde o zjednodušení normalizačních obvodů, protože se počítá nad číselným tělesem zahrnujícím jen nulu a jedničku.

Průběh algoritmu se skládá z n kroků s následující strukturou: Pomocí rotací dostaneme na první místo řádek vhodný k odčítání od ostatních (časová složitost v nejlepším případě konstantní, v nejhorším lineární v n); Tento řádek normalizujeme (konstantní složitost) a odečteme od ostatních (konstantní složitost). Pokud při rotacích nenajdeme žádný řádek s nenulovým prvkem na správném místě, je matice singulární (z hlediska používaného algoritmu). Po skončení posledního kroku je vhodné provést ještě takový počet rotací, aby se původní první řádek matice vrátil na první místo, a řešení soustavy (v posledním sloupci matice) se srovnalo do správného pořadí.

Pokud je použita dělička s latencí 4 cykly, je nejlepší časová složitost algoritmu $11n + 5$ cyklů, tedy $0.11n + 0.05$ µs. Do limitu 50 µs se hypoteticky vejde výpočet matice o velikosti 450×450 ($n = 450$). Problém je s kvadratickou prostorovou složitostí (viz obrázek 4). Největší matice, která se vejde do čipu, má rozměr přibližně 25×25 prvků a odpovídá rekonstrukci o 25 pixelech. V případě potřeby velmi rychlých rekonstrukcí s rozumnějším rozlišením lze uvažovat o použití většího čipu; Pro řízení tokamaku se ovšem zdá být praktičtější zvolit jiný algoritmus.



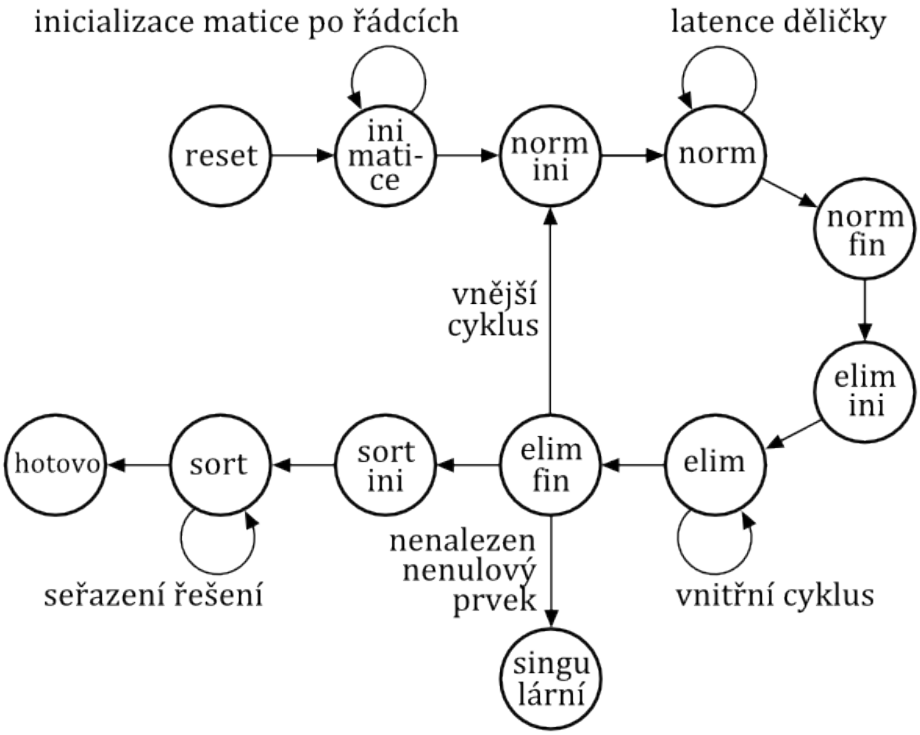
Obr. 4: Využití hardwarových zdrojů v závislosti na počtu pixelů rekonstrukce pro maticově paralelizovaný algoritmus; Modrá čára vždy značí celkový počet součástek v čipu; Je zřejmé, že první dosáhnou limitu vyhledávací tabulky (LUT).

Dalším problémem algoritmu je numerická stabilita, vzhledem k chybějící pivotaci. Pro matice velikosti 25×25 často odchylky vypočteného řešení od řešení referenčního (vypočteného v MATLABu) dosahují desítek procent, pro větší matice budou větší i relativní odchylky. Lze uvažovat o zavedení pivotace, která by pravděpodobně mírně zhoršila asymptotickou časovou složitost (najít nejlepší prvek mezi n prvky se nestihne v jednom kroku), ale i přesto by algoritmus mohl být velmi rychlý.

4.2 Řádkově paralelizovaná Gaussova–Jordanova eliminace

Druhý navržený obvod se ukazuje být algoritmicky složitější, zato však prakticky použitelnější než ten z minulého odstavce. K uložení prvků matice využívá paměťové bloky, zatímco předchozí design počítal pouze s distribuovanou pamětí. Každý z těchto bloků (v konfiguraci "simple dual-port") umožňuje v každém kroku načíst jedno číslo ze zvolené adresy, a zároveň na obecně jinou adresu jedno číslo zapsat. Bloků je k dispozici 116, každý na 1024 osmnáctibitových čísel. Design počítá s využitím jednoho paměťového bloku pro každý sloupec matice; Nelze tedy provádět řádkové operace pro mnoho řádků najednou. Přístup k blokovým pamětem je řešen přes IP jádro; Další dvě v designu použitá IP jádra slouží jako celočíselná dělička a násobička.

Algoritmus pracuje se dvěma vnořenými cykly. Vnější cyklus prochází všechny řádky v pořadí daném řádkovou pivotací. Vybraný řádek se normalizuje a uloží do registru vytvořeného z distribuované paměti. Pak následuje vnitřní cyklus, který znovu prochází všechny řádky a přičítá k nim násobky toho normalizovaného. Vnitřní cyklus je časově optimalizován pomocí pipeliningu tak, aby jedna řádková úprava včetně čtení a zápisu pamětí zabrala jeden tik (plus několik inicializačních tiků na začátku cyklu). Zároveň s cyklem řádkových úprav se provádí výběr příštího řádku pro normalizaci, takže to nemá vliv na výslednou časovou složitost. Ta je přes vší snahu kvadratická.



Obr. 5: Stavový automat řídící průběh řádkově paralelizovaného Gaussova–Jordanova algoritmu (inicializační fáze cyklů s pipeliningem zjednodušeny na jeden stav)

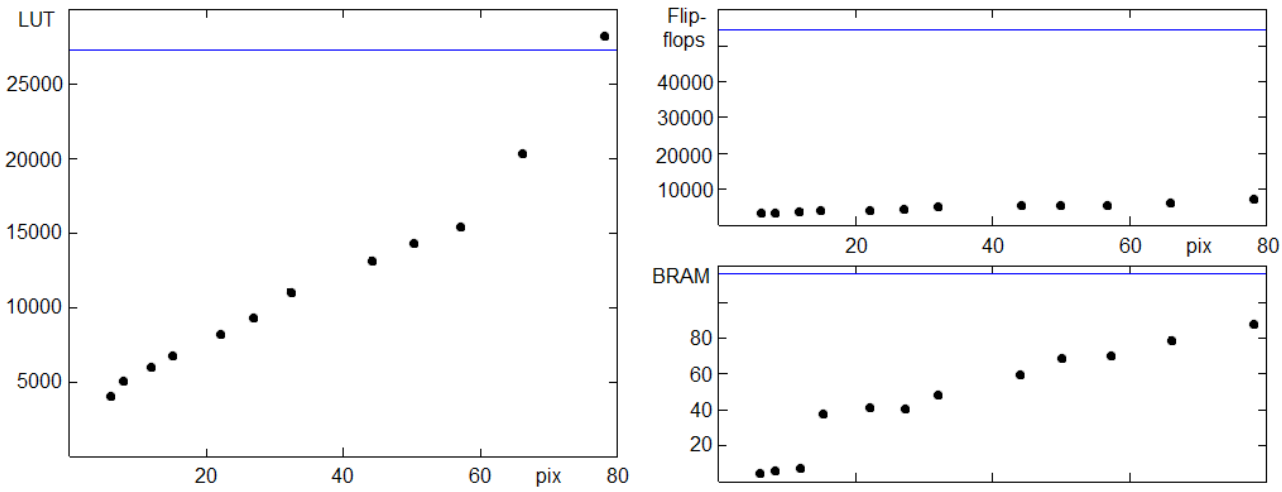
Kvůli pivotaci není upravená matice jednotková, nýbrž má permutované řádky. Proto se po skončení řádkových úprav provádí ještě jeden cyklus přes všechny řádky, v němž se složky řešení rovnají na správná místa ve výstupu (podle toho polohy jedničky na daném řádku).

Vzhledem k organizační složitosti algoritmu je posloupnost jeho kroků řízena stavovým automatem (ve vývoji pro FPGA běžná praxe); Jeho zjednodušená verze je znázorněna na obrázku 5. Průběh stavů téměř nezávisí na řešené soustavě (jen na její velikosti a regularitě); Časová složitost nemá žádný rozdíl mezi nejlepším a nejhorším případem, a pro regulární matici je vždy rovna $n^2 + 18n + 6$ cyklů. Hranici 50 µs se blíží matice o velikosti 60 × 60; Jak naznačuje statistika na obrázku 6, tato matice se také vejde do dostupného hardwaru.

4.3 Zbytek tomografického algoritmu

Kromě řešení soustavy lineárních rovnic je v každém cyklu tomografického algoritmu potřeba vykonat několik dalších maticových operací. Jde především o sestavení matice úlohy a její pravé strany před zahájením Gaussovy–Jordanovy eliminace, a po jejím vyřešení o výpočet normy rezidua, podle které se volí regularizační parametr.

K sestavení matice úlohy je třeba sečíst dvě matice, $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}$ a $\lambda\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$; Parametr λ je pokaždé jiný (v případě použití Fisherovy informace je navíc třeba násobit řádky matice $\boldsymbol{H}$ proměnnými koeficienty). Matice $\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$ vyjadřující konečné diference obsahuje pouze malá celá čísla, díky čemuž lze při výpočtu jejího λ-násobku ušetřit násobičky a využít bitové posuny (u Fisherovy informace nejde).



Obr. 6: Využití zdrojů tomografickým algoritmem s řádkově paralelizovanou Gaussovou–Jordanovou eliminací; DSP bloky tentokrát použity nebyly, místo nich je znázorněno využití blokových pamětí (BRAM); Do statistiky je započten celý obvod, ne jen část řešící soustavu lineárních rovnic; Statistika je odvozena z designu po syntéze, v dalších fázích překladu se může mírně změnit.

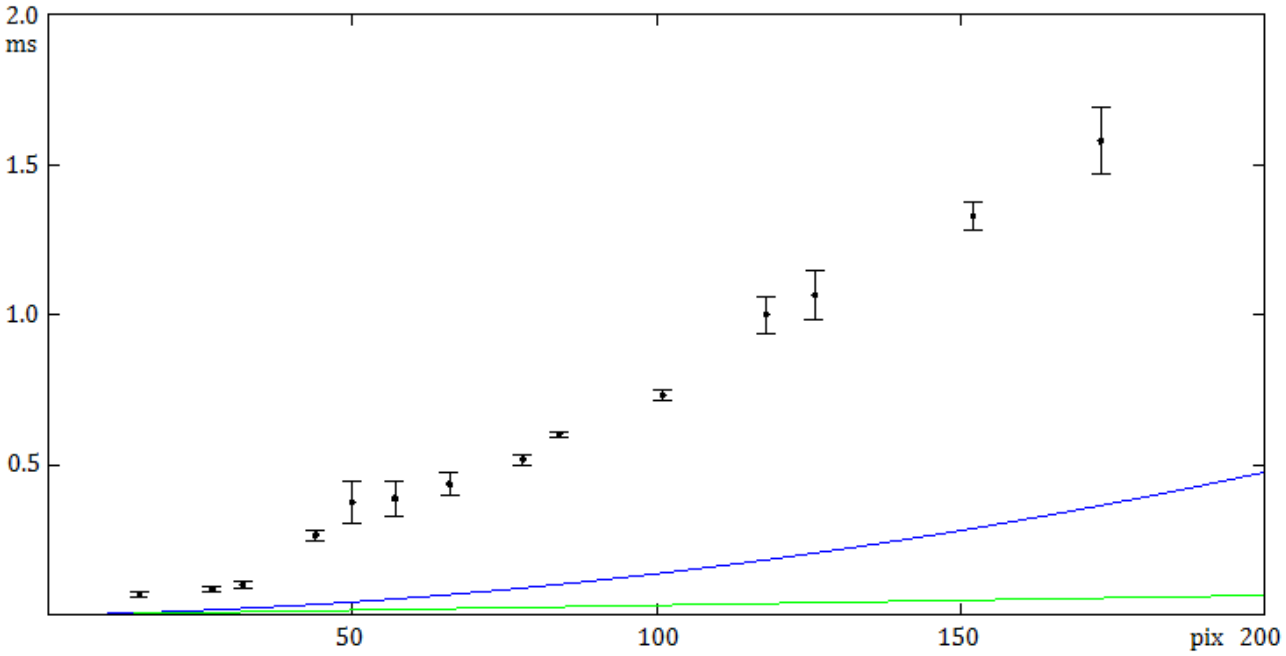
Sestavování matice úlohy probíhá po řádcích (jeden řádek za jeden tik). Tvar obou matic k sečtení je uložen v distribuované paměti[10], ve VHDL zapsané pomocí číselných konstant. Za účelem usnadnění přepisu těchto konstant z MATLABu do VHDL byl napsán skript v Pythonu, který čte textové soubory vytvořené MATLABem a generuje odpovídající VHDL. Podobně je vyřešeno násobení vektoru maticí, které je třeba k výpočtu pravé strany. Tato část designu má prostorovou (či paměťovou) složitost vyšší než lineární, protože počet nenulových prvků v matici $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}$ roste rychleji než lineárně (ačkoli ta matice je pro velká n řídká).

Možné hodnoty parametru λ jsou v současné implementaci poněkud omezené reprezentací čísel (maximální hodnota použité osmnáctibitové reprezentace činí $2^{8} - 2^{-9} \approx 256$). Aby nedocházelo k přetečení v matici $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T} + \lambda\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$, může λ nabývat hodnot přibližně mezi 0 a 64 (přitom škálování matice $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}$ bylo zvoleno tak, aby eliminace dosahovala přiměřené přesnosti). Pokud bude někdy potřeba větší rozsah pro λ, lze uvažovat o rozdělení regularizačního parametru na dva ($\mu\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T} + \lambda\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$) a místo jejich hodnot nastavovat jejich podíl. Hodnoty nového parametru lze přitom omezit na mocniny dvou, díky čemuž nebudou potřeba nové násobičky ani děličky (mocninami dvou se dá násobit a dělit pomocí bitových posunů).

4.4 Porovnání s MATLABem

S referenčním řešením stejné úlohy napsaným v MATLABu byl navržený obvod porovnáván ve dvou ohledech, rychlosti a přesnosti. Byl použit MATLAB ve verzi R2009b na procesoru Intel® Core™ 2 Duo E6550, taktovaném na 2.33 GHz. Soustava lineárních rovnic byla řešena Choleského faktorizací optimalizovanou pro řídké matice (viz např. [43]).

10 Odhady prostorové složitosti ukazují na to, že ve skutečnosti jsou data uchovávána v blokových pamětech (patrně následkem nějaké optimalizace během syntézy); To do jisté míry vysvětluje nerovnoměrnou závislost počtu blokových pamětí na počtu sloupců matice, patrnou na obrázku 6



Obr. 7: Porovnání časů výpočtu tomografické úlohy v MATLABu (černě) a FPGA (modře řádkově paralelizovaný, zeleně maticově paralelizovaný algoritmus) v závislosti na počtu pixelů

Obrázek 7 ukazuje výsledek porovnání rychlosti obou navržených obvodů s implementací v MATLABu (časové údaje pro MATLAB byly změřeny na zmíněném hardwaru, pro FPGA vypočteny ze vzorců pro počet tiků). U obou obvodů a u všech testovaných velikostí matic se rychlejším ukázalo být FPGA; Limit 50 μs nesplnil MATLAB ani u nejmenší matice, odpovídající 15 pixelům. Tento výsledek ovšem závisí na použitém hardwaru (u MATLABu i FPGA).

Přesnost FPGA obvodů (předpokládá se, že řešení MATLABu je správně) byla testována pro matice velikosti 57×57. Jako pravá strana soustavy byla použita umělá data, odpovídající gaussovskému fantomovému profilu emisivity; Data odpovídala výstupům 70 detektorů, umístěných ve dvou portech tokamaku COMPASS (na obrázku 2 jde o ty dva porty, v nichž jsou instalovány dírkové komory se 35 detektory). Testování probíhalo v simulačním programu ISim, dodávaném spolu s prostředím Xilinx ISE; Opravdový čip byl použit pouze k namátkové kontrole shodnosti výsledků.

Směrodatná odchylka relativní chyby výsledků řádkově paralelizovaného algoritmu byla změřena jako 1.7 % (pro vyzkoušená umělá data), zatímco u maticově paralelizovaného algoritmu (bez pivotace) postrádá střední kvadratická odchylka vypovídací hodnotu; Většina prvků výsledku končí na hodnotě 256 nebo -256. Je zřejmé, že bez pivotace se použitelný obvod pro tomografii neobejde.

Jelikož problém výpočtu polohy plazmatu z obrazu získaného tomografií není dosud uspokojivě vyřešen, bylo pro účely testování přesnosti počítáno s těžištěm profilu emisivity. V tomto případě jde spíše než o testování přesnosti algoritmů na FPGA o rozhodování, zda je pro určení polohy dostatečné rozlišení 57 pixelů. Těžiště vypočítané z 57pix obrazu tedy bylo porovnáváno s těžištěm z tomografie s větším rozlišením (988 pix), která byla provedena

v MATLABu.

Použitá data byla opět umělá, takže není zcela jasné, jak se bude obvod chovat v opravdovém tokamaku. Mimo jiné se předpokládalo, že vývoj profilu emisivity je natolik pomalý, že se algoritmus vždy stačí ustálit po jednom iteračním kroku; V praxi by patrně část obrazů byla rekonstruována nepřesně (a při použití vhodného kritéria většina nepřesných rekonstrukcí rozpoznána) [19]. Rovněž byl testován jen určitý rozsah pohybu plazmatu, jehož emisivita měla vždy tvar gaussovského píku. Dále lze očekávat, že pro případné použití ve zpětné vazbě bude vyvinut vhodnější algoritmus k přepočtu profilu emisivity na polohu. U reálných dat se navíc vyskytují problémy jako nepřesné určení geometrie detektorů nebo záření divertoru či limiterů, které nesplňuje a-priori předpoklady.

Výsledky tohoto omezeného testování naznačují, že rozlišení 57 pixelů je pro určení polohy podobně vhodné jako to větší rozlišení, použité jako referenční. Směrodatná odchylka rozdílu polohy těžiště rekonstrukce s větším a menším rozlišením činí 0.65 cm ve vertikálním směru a 0.53 cm ve směru radiálním. Tato hodnota je srovnatelná s nejspíš náhodnými fluktuacemi údajů z obou rozlišení, které se navíc zdají být navzájem nezávislé; Není zřejmé, které rozlišení poskytuje přesnější údaj o poloze.

4.5 Možná vylepšení

Jeden z navržených obvodů se podle dosavadních poznatků zdá být použitelný pro určování polohy v reálném čase. V případě potřeby lze ovšem vymyslet několik vylepšení rychlosti, a tím pádem rozlišení na zařízeních s dostatečnou kapacitou.

V řádkově paralelizovaném algoritmu se přičítání násobku jednoho řádku ke druhému provádí rychlostí jednoho upraveného řádku za tik. Tyto úpravy přitom probíhají v cyklu, v němž přičítaný řádek zůstává stejný. Tímto způsobem by šlo upravovat v každém kroku dva řádky najednou (nebo více), čímž by se z asymptotického hlediska dosáhlo dvojnásobného (nebo vícenásobného) urychlení výpočtu. Prostorová složitost by vzrostla, ale zůstala by lineární. Pro stejně velkou matici by bylo potřeba víc paměťových bloků, protože v rámci jednoho bloku nelze přistupovat k mnoha adresám (prvkům odpovídajícího sloupce matice) zároveň. Tato úprava nebyla implementována, protože s použitým čipem by vedla ke zbytečnému omezení velikosti matice.

K řešení soustavy lineárních rovnic se dají použít jiné algoritmy, které by i v paralelním hardwaru mohly být rychlejší než Gaussova eliminace. Kromě Choleského faktorizace jde o iterační metody, jejichž použití v tomografii ovšem dosud nebylo prozkoumáno ani na konvenčním procesoru.

Drobných zlepšení by šlo dosáhnout důslednějším využíváním pipeliningu, například použitím děličky s větším zpožděním, ale i větší propustností (lze dosáhnout propustnosti až jedno dělení za tik [44]). Bylo by přitom třeba vyřešit problém pivotace tak, aby vstup děličky nebyl (příliš) závislý na předchozím výstupu, protože v době, kdy se vstup zadává, ještě není předchozí výsledek znám. Navržená úprava by pomohla zejména u maticově paralelizovaného algoritmu, protože kritická část řádkově paralelizovaného algoritmu už úplného pipeliningu využívá.

5 Shrnutí

Byly navrženy a implementovány dva obvody pro FPGA, provádějící tomografickou rekonstrukci profilu emisivity plazmatu v "reálném čase" (na tokamaku COMPASS přibližně do 50 mikrosekund). Zamýšleným využitím těchto obvodů je rychlá zpětná vazba, tedy řízení vertikální a radiální polohy plazmatu na základě měření vyzařování plazmatu v měkkém rentgenovém oboru.

Oba obvody jsou založeny na dříve vyvinutém tomografickém algoritmu, který používá Tichonovou regularizaci s požadavkem na hladkost rekonstruovaného profilu, a iteraci na proměnných datech (výsledek rekonstrukce z minulého okamžiku se používá jako počáteční přiblížení pro další rekonstrukci). Tento algoritmus byl v minulosti časově optimalizován, ale stále nebyl vhodný pro použití v reálném čase. S využitím FPGA byl nyní algoritmus optimalizován hardwarově. S touto optimalizací se zdá, že rychlost výpočtu může být dostatečná pro zpětnou vazbu i na tak malém tokamaku, jakým je COMPASS (přinejmenším v radiálním směru, v některých případech zřejmě i ve vertikálním); Smysluplných výsledků přitom bylo dosaženo s FPGA z řady Spartan 6 (Xilinx), optimalizované na jiné parametry, než je výpočetní výkon.

Kritickým místem ve zmíněném tomografickém algoritmu je řešení soustavy lineárních rovnic. Byly implementovány dva obvody řešící tento problém. První z nich je vysoce paralelní, jeho časová složitost roste lineárně s řádem soustavy. Pro tomografii nakonec použit nebyl kvůli své vysoké prostorové složitosti (množství potřebných logických prvků). Navíc jeho současná verze není numericky stabilní, takže pro případné využití by potřebovala další optimalizace (zavedení nějaké formy pivotace).

Druhý z implementovaných algoritmů má kvadratickou časovou složitost, ale pro dostatečně malé rozlišení tomografické rekonstrukce (na použitém hardwaru kolem 60 pixelů) je dostatečně rychlý pro reálný čas. Tento algoritmus je numericky stabilní a jeho prostorová složitost je dostatečně nízká, aby se jeho verze pro 60 pixelů vešla do výše zmíněného FPGA (což u prvního designu neplatilo).

Tomografický obvod využívající druhý z navržených řešičů lineárních soustav byl testován jak prostředky numerických simulací, tak implementací v reálném FPGA; V simulacích byla určována numerická stabilita a přesnost, zatímco testy v hardwaru sloužily především pro ověření splnění podmínek na timing v kritických částech obvodu (a tedy realističnosti simulací). Všechny testy využívaly umělá data. Jejich výsledky naznačují, že výstupy obvodu jsou dostatečně přesné pro určování polohy plazmatu.

K dosud implementovanému obvodu bylo navrženo několik vylepšení, která by mohla zvýšit spolehlivost obvodu, jeho rychlost pro danou velikost matice, respektive velikost matice pro daný časový limit. Jejich případná implementace závisí na konkrétních aplikacích.

Je zřejmé, že před případným praktickým využitím bude nutné provést důkladnější odladění obvodu, s méně zjednodušujícími předpoklady (nejlépe s pomocí dat naměřených skutečnými detektory měkkého rentgenového záření). Pro použití ve zpětné vazbě bude navíc třeba navrhnout spolehlivý přepočet tomografického obrazu na polohu, respektive vyřešit problém s nepřesně rekonstruovanými snímky, které algoritmus občas produkuje.

Reference

[1] O. Kudláček. *Řízení polohy plazmatu na tokamaku COMPASS*. ČVUT, FJFI, 2010, výzkumný úkol. fttf.fjfi.cvut.cz/StPrace/VyzkUk/2010/KudlacekOndrej.pdf

[2] R. Beňo. *Řízení tokamaku COMPASS. ČVUT, FEL, 2011, diplomová práce*. support.dce.felk.cvut.cz/mediawiki/images/4/49/Dp_2011_beno_radek.pdf [cit. 7. 11. 2011]

[3] J. Wesson. *Tokamaks*. Clarendon press - Oxford, 2004. ISBN 0 19 8509227

[4] R. Beňo, J. John. *Modelování zpětnovazebního řízení polohy plazmatu v tokamaku COMPASS*. Čs. čas. fyz. 59 (2009) 242 – 245

[5] F. Janky et al. *Determination of the plasma position for its real-time control in the COMPASS tokamak*. Fusion Engineering and Design, 86 (2011) 1120 – 1124

[6] M. Hron et al. *Overview of the COMPASS CODAC system*. Fusion Engineering and Design 89 (2014) 177 – 185

[7] F. Janky et al. *Upgrade of the COMPASS tokamak real-time control system*. Fusion Engineering and Design 89 (2014) 186 – 194

[8] M. Hron et al. *Power supplies for plasma column control in the COMPASS tokamak*. Fusion Engineering and Design 88 (2013) 1640 – 1645

[9] V. Weinzettl et al. *Overview of the COMPASS diagnostics*. Fusion Engineering and Design 86 (2011) 1227 – 1231

[10] L. C. Ingesson et al. *Tomography Diagnostics: Bolometry and Soft X-Ray*. Fusion Science and Technology 53 (2008) 528 – 576

[11] M. Vácha. *Detection systems for measurements of high-temperature plasma radiation on the COMPASS tokamak by fast bolometers and soft X-ray detectors*. Univerzita Karlova, Matematicko-fyzikální fakulta, 2009, diplomová práce.

[12] J. Sunnegårdh. *Iterative Filtered Backprojection Methods for Helical Cone-Beam CT*. Linköping University, Institute of Technology, 2009, disertační práce

[13] *Soft X-ray diagnostics*. http://www.ipp.cas.cz/Tokamak/euratom/index.php/en/compass-diagnostics/spectroscopic/soft-x-rays

[14] J. Mlynář et al. *Inversion Techniques in the Soft X-Ray Tomography of Fusion Plasmas: Toward Real-time applications*. Fusion Science and Technology 58 (2010) 733 – 741

[15] M. Imríšek et al. *Studies of simplified inversion methods for real time calculation of center of mass of soft x-ray radiation in tokamaks*. 8th Workshop on Fusion Data Processing, Validation and Analysis, 2013

[16] P. C. Hansen. *The L-curve and its use in numerical treatment of inverse problems*. www.sintef.no/project/eVITAmeeting/2005/Lcurve.pdf [cit. 23. 8. 2014]

[17] M. Anton et al. *X-Ray Tomography at TCV*. Plasma Physics and Controlled Fusion 38 (1996) 1849

[18] M. Odstrčil. *Tomografie plazmatu na tokamaku COMPASS*. ČVUT, FJFI, 2010, bakalářská práce

[19] V. Löffelmann. *Tomografie měkkého rentgenového záření pro řízení tokamaku v reálném čase*. ČVUT, FJFI, 2012, bakalářská práce.

[20] J. Mlynář et al. *Investigation of the consistency of magnetic and soft X-ray plasma position measurements on TCV by means of a rapid tomographic inversion algorithm*. Plasma Physics and Controlled Fusion 45 (2003) 169

[21] D. Mazon et al. *Soft x-ray tomography for real-time applications: present status at Tore Supra and possible future developments*. Review of Scientific Instruments 83 (2012)

[22] R. Woods. *FPGA-Based Implementation of Signal Processing Systems*. Wiley 2008. ISBN 978-0-470-03009-7

[23] Xilinx. *7 Series FPGAs Overview*. www.xilinx.com [cit. 13. 8. 2014]

[24] C. Cullinan et al. *Computing Performance Benchmarks among CPU, GPU and FPGA*. www.wpi.edu/Pubs/E-project/Available/E-project-030212-123508/unrestricted/Benchmarking_Final.pdf [cit. 18. 4. 2014]

[25] Xilinx. *Xilinx Design Reuse Methodology for ASIC and FPGA Designers*. www.xilinx.com [cit. 18. 4. 2014]

[26] I. Kuon. *Measuring the Gap Between FPGAs and ASICs*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 2 (2007), 203 – 215

[27] Xilinx. *Spartan 6 Family Overview*. www.xilinx.com [cit. 11. 3. 2014]

[28] S. Kilts. *Advanced FPGA design: Architecture, Implementation, and Optimization*. Wiley 2007. ISBN: 978-0-470-05437-6

[29] P. P. Chu. *FPGA Prototyping by VHDL Examples*. Wiley 2008. ISBN: 978-0-470-18531-5

[30] Wikipedia. *Field Programmable Gate Array*. de.wikipedia.org/wiki/Field_Programmable_Gate_Array [cit. 14. 8. 2014]

[31] Wikipedia. *Jacobi method*. en.wikipedia.org/wiki/Jacobi_method [cit. 25. 8. 2014]

[32] Wikipedia. *Gauss–Seidel method*. en.wikipedia.org/wiki/Gauss–Seidel_method [cit. 25. 8. 2014]

[33] M. T. Heath et al. *Parallel Algorithms for Sparse Linear Systems*. SIAM Review 33 (1991) 420 – 460

[34] G. Peters, J. H. Wilkinson. *On the Stability of Gauss–Jordan Elimination with Pivoting*. Communications of the ACM 18 (1975) 20 – 24

[35] Wikipedia. *Gaußsches Eliminationsverfahren*. http://de.wikipedia.org/wiki/Gaußsches_Eliminationsverfahren [cit. 25. 8. 2014]

[36] J. Goodman. *Introduction of Pipelinig Optimisations into Gaussian Elimination*. Draft Proceedings of the 3rd Scottish Functional Programming Workshop, 2001

[37] N. J. Higham. *How Accurate is Gaussian Elimination?* Cornel University, 1989, Technical Report

[38] S. Donfack et al. *On Algorithmic Variants of Parallel Gaussian Elimination: Comparison of Implementations in Terms of Performance and Numerical Properties*. University of Tennessee, EECS Department, 2013, Technical Report

[39] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2002, ISBN 0-521-43108-5

[40] Digilent Inc. *Atlys Reference Manual*. www.digilentinc.com [cit. 27. 8. 2014]

[41] Xilinx. *Spartan-6 Family Overview*. www.xilinx.com [cit. 11. 3. 2014]

[42] A. Bogdanov et al. *SMITH - A Parallel Hardware Architecture for fast Gaussian Elimination over GF(2)*. Workshop on Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS 2006), Conference Records, 2006

[43] J. R. Gilbert et al. *Sparse Matrices in MATLAB: Design and Implementation*. SIAM Journal of Matrix Analysis and Applications 13 (1992), 333 – 356

[44] Xilinx. *LogiCORE IP Divider Generator v5.1; Product Guide for Vivado Design Suite*. www.xilinx.com [cit. 20. 3. 2014]