Technická zpráva

Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Akcelerátor výpočtu věrohodnostní funkce pro systémy pasivní radiolokace

Funkční vzorek

Dr. Michal Kvasnička *,
Ing. Antonín Heřmánek, Ph.D. **,
Ing. Michal Kuneš ***

** ERA a.s., Poděbradská 186/52, 12000 Praha 2, Česká Republika, Tel.: +420 266107 441, FAX: +420 281868025, email: m.kvasnicka@era.cz , WWW: www.era.cz .*
*** ÚTIA AV ČR, Pod vodárenskou věží 4, 18208 Praha 8, Česká Republika, Tel.: +420 266052470, FAX: +420 266052511, email: hermanek@utia.cas.cz , WWW: www.utia.cas.cz/ZS .*
**** ÚTIA AV ČR, Pod vodárenskou věží 4, 18208 Praha 8, Česká Republika, Tel.: +420 266052470, FAX: +420 266052511, email: hermanek@utia.cas.cz , WWW: www.utia.cas.cz/ZS .*

Obsah

Revize

Revize	Datum	Autor	Popis změn v dokumentu
0	16.2.2007	M.K., A.H.	Vytvoření dokumentu
1			
2			

1. Úvod

Jedním z klíčových problémů v pasivní koherentní lokaci (PCL) je efektivní a numericky přesný výpočet věrohodnostní funkce (CAF). Tato funkce souvisí s přímím signálem a signálem odraženým od hledaných objektů. PCL systémy využívají komerční vysílače (TV, FM atd.) s vysokým výkonem a relativně nízkých frekvencích. Další výhodou je, že vysílač nemusí spolupracovat s přijímačem. CAF representuje výkonové spektrální rozložení vzájemné korelace mezi přímím a odraženým signálem. Je závislá na vzájemném časovém zpoždění a frekvenčním posunu vstupních signálu a je považována za primární informaci pro detekci, lokalizaci a identifikaci sledovaných objektů. Z těchto důvodů vyplívá důležitost efektivní implementace CAF. V této zprávě presentujeme první výsledky implementace CAF funkce na platformě FPGA. Těchto výsledů bylo dosaženo na základě úzké spolupráce mezi ERA, a.s. a UTIA AV ČR, v.v.i..

One of key problem in passive coherent location (PCL) is effective and accurate computation of the cross ambiguity function (CAF). This function is related to the direct signal and signals reflected from localized targets. PCL systems exploit high-power commercial transmitters of opportunity (FM, TV, etc.) to take advantage of lower frequencies, multistatic geometries and covert deployment. The transmitter does not have to cooperate with the receiver. The CAF represent power spectral density distribution of the cross-correlation between direct and reflected signals. It depends on mutual time delay and frequency shift of the input signals and is considerate as primary information for detection, localization and identification of the tracked targets. Regarding above mentioned reasons has to be important develop optimal (numerically effective and sufficiently accurate) implementation of the HW architecture based on FPGA for CAF computation, which will be suitable for future real-time PCL systems. As a first result which originates on the ongoing mutual cooperation between ERA a.s. and UTIA is design of the PC accelerator card for CAF computation based on Xilinx FPGA processor. The presented contribution gives overall information about used algorithms, FPGA accelerator card design and achieved performance.

2. Funkce CAF pro PCL systémy

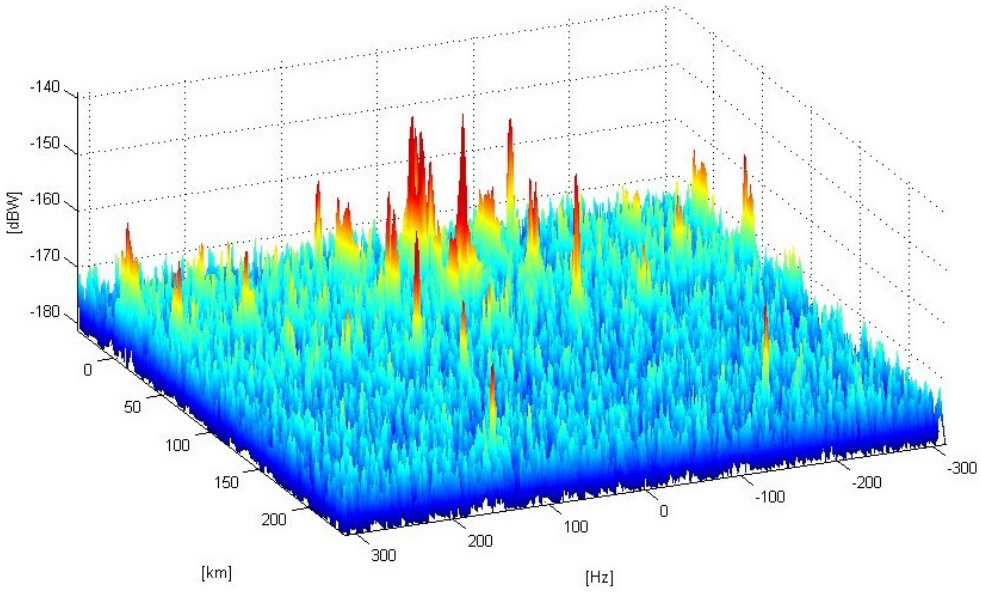
Výpočet vzájemné funkce nejednoznačnosti (dále CAF – Cross Ambiguity Function) v systému pasivní koherentní radiolokace (dále PCL – Passive Coherent Location), je jednou z nejdůležitějších a současně výpočetně nejnáročnějších operací používaných v procesu digitální detekce a vyhodnocení signálů (viz. Obr.1.1).
Funkce CAF je definována následujícím způsobem:

$$CAF(\tau, f) = \int_0^T s_1(t)\, s_2^*(t+\tau)\, e^{-j2\pi ft}\, dt\,, \qquad (1.1)$$

kde s_1 a s_2 jsou časově spojité signály v analytickém tvaru, T je integrační perioda, τ a f je jejich vzájemné časové zpoždění (TDOA) resp. frekvenční (dopplerovský) posun (FDOA). Přejdeme-li do diskrétní reprezentace v časové oblasti, pak můžeme psát, že $t = nT_S$ a $f = \frac{kf_S}{N}$, kde T_S je vzorkovací perioda, $f_S = 1/T_s$ je vzorkovací frekvence, n reprezentuje pořadové číslo vzorku a N je celkový počet vzorků. Obecnou CAF, viz. rovnice (1.1), transformujeme do diskrétního tvaru:

$$CAF(\tau, k) = \sum_{n=0}^{N-1} s_1(n)\, s_2^*(n+\tau)\, e^{-j2\pi \frac{kn}{N}}\,, \qquad (1.2)$$

kde $\frac{k}{N}$ má význam diskrétního frekvenčního kroku resp. zlomku vzorkovací frekvence. Hodnota $CAF(\tau,k)$ nebo $|CAF(\tau,k)|$ dosahuje maxima pro hodnoty τ a $\frac{k}{N}$, které odpovídají TDOA a FDOA analyzovaných signálů s_1 a s_2 (viz. Obr. 1). Poznamenejme, že samotná přítomnost výrazných maxim ve funkci CAF umožňuje v principu velice robustní detekci signálů jako takových a to i v případě extrémně nízkých SNR.



Obrázek 1: Vzájemná funkce nejednoznačnosti CAF pro přímý a odražený FM signál (časové zpoždění τ je zde zobrazeno jako tzv. eliptická vzdálenost)

Numerická efektivnost výpočtu CAF je klíčovým faktorem pro aplikace v pasivních radiolokačních systémech s korelační detekcí signálů, protože je nutno prohledávat relativně velké intervaly TDOA resp. FDOA. V definici (2.2) pro diskrétní CAF jsou uvažovány TDOA v rozsahu $-N \le \tau \le N$ a FDOA v rozsahu $-\frac{N}{2}+1 \le k \le \frac{N}{2}$. Na kompletní prohledání všech možných časových zpoždění a frekvenčních posunů je tedy potřeba cca $2N^2$ vyčíslení funkce CAF, což představu pro dlouhé integrační periody T resp. velké celkové počty vzorků N extrémní nároky na výpočetní výkon.
Optimálním algoritmem pro efektivní výpočet CAF je metoda rychlé diskrétní Fourierovy transformace (FFT) aplikovaná na tzv. signálový součin, tedy

$$CAF(\tau,k) = FFT\left(s_1(n)\, s_2^*(n+\tau)\right), \quad (1.3)$$

kde jedna aplikace FFT je realizována pro všechny požadované hodnoty k současně a jednu hodnotu τ. FFT je tedy nutno aplikovat pro každou hodnotu τ separátně.
Jednou z velice perspektivních cest pro efektivní výpočet CAF je hardwarová implementace vhodného výpočetního algoritmu (viz. rovnice (1.3)) pomocí tzv. programovatelných hradlových polí (FPGA), která pro vhodně navržený algoritmus poskytují vysoký výpočetní výkon.
V současné době proto probíhá výzkum (spolupráce mezi ERA, a.s. a ÚTIA AV ČR, v.v.i.) efektivní hardwarové implementace akcelerátoru pro výpočet CAF s vysokou přesností na omezeném intervalu frekvenčních posunů $\langle -f_{\max}, +f_{\max} \rangle$ resp. časových zpoždění $\langle 0, \tau_{\max} \rangle$.

Nároky na vysokou přesnost výpočtu jsou kladeny jak ze strany numerické přesnosti výpočtů samotných, tak ve smyslu požadovaného frekvenčního rozlišení $\Delta f = \frac{f_S}{N}$.

Požadované parametry akcelerátoru pro výpočet CAF jsou následující:

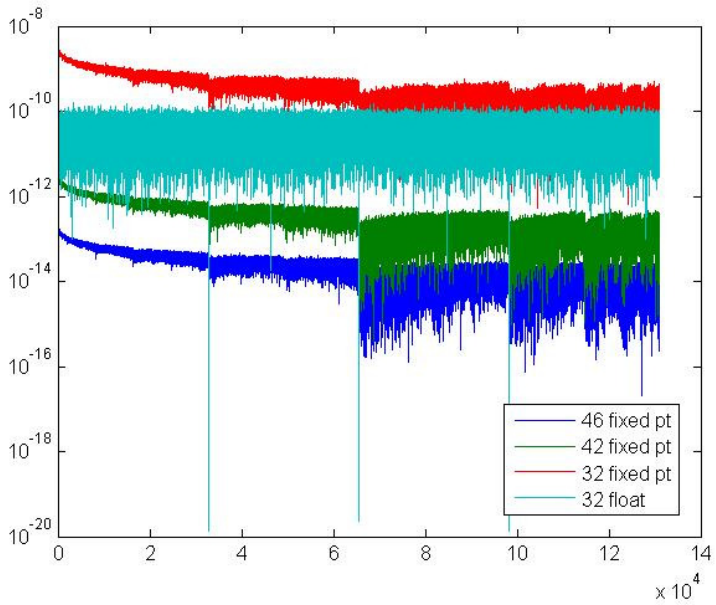
- vzorkovací kmitočet: 100-200 kHz
- bitová hloubka vstupních dat: 18 – 24 bitů
- celkový počet vzorků resp. délka integrační periody: 217 = 131 072 vzorků, cca 1sec.
- frekvenční rozlišení: cca 1Hz
- přesnost výpočtu FFT: chyba $10^{-9} \div 10^{-12}$ vůči IEEE 64-bitovému floating pointu
- maximální počet hodnot časových zpoždění resp. separátních výpočtů FFT: až 1024
- maximální frekvenční interval: $\langle -300, +300 \rangle$ Hz, tj. 600 spektrálních čar
- celková doba výpočtu: cca 1sec

Takto definovaný problém je velmi náročný ze dvou důvodů. Prvním je velmi velký objem dat, druhým je extrémně vysoká přesnost výpočtu CAF. V současné době je na trhu mnoho výkonných řešení pro výpočet FFT ve formě optimalizovaných maker (tzv. IP cores). Žádný z dodavatelů těchto maker, ale nepodporuje implementaci FFT vyhovující výše uvedeným požadavkům. A to jak z hlediska požadavků na přesnost výpočtu, tak požadavků na délku bloku pro výpočet FFT. Přední výrobce v oblasti FPGA, firma Xilinx (USA), nabízí v rámci svého produktu "Core Generator" (knihovna optimalizovaných funkčních maker) blok FFT o maximální délce 65536 vzorků a maximální datové přesnosti 24-bit.

Doba výpočtu CAF (viz. výše uvedené parametry akcelerátoru) je na současných PC v rozmezí 20 až 30 sekund. Cílem prezentovaného výzkumu je návrh akcelerátoru pro PC, založeném na technologii FPGA, který by výpočet CAF provedl v řádu cca 1sec. Takové řešení by bylo možno použít v reálném systému PCL, který by byl provozován v režimu reálného času.

Byla provedena analýza numerických vlastností FFT pro aritmetiky pracující s pevnou řádovou čárkou (FP) a s plovoucí řádovou čárkou (FLP). Výsledky této analýzy byly ještě ověřeny řadou experimentálních simulací. Z výsledků analýzy a experimentů vyplývá:

1. 32 bitová aritmetika s plovoucí řádovou čárkou (IEEE single precision FLP) vykazuje chybu v vůči referenční hodnotě řádu 10^{-10}, což je z hlediska požadované přesnosti nedostatečné. Navíc, vzhledem k charakteru zpracovávaných dat, není bitový rozsah optimálně využit.
2. aritmetika s pevnou řádovou čárkou se změnou měřítka (scaling) výstupů motýlku faktorem $1/2$, vykazuje menší aritmetickou chybu, než v případě bez změny měřítka. Zmiňovaná chyba je v případě 32 bitové aritmetiky řádu 10^{-9}, což také nesplňuje požadavky na přesnost výpočtu. Proto je nutné použít 42 nebo 46 bitovou aritmetiku, kde se chyba pohybuje v řádu 10^{-12} až 10^{-13}. Dále uveďme, že u FP aritmetiky lze dělení dvěmi jednoduše implementovat jako bitový posuv napravo.



Obrázek 2: Porovnání chyby výpočtu FFT délky $N = 131\,072$ pro různé velikosti fixed point aritmetiky a pro 32. bitovou aritmetiku s plovoucí čárkou (32 bit FPL)

3. Návrh architektury pro výpočet PCL na FPGA

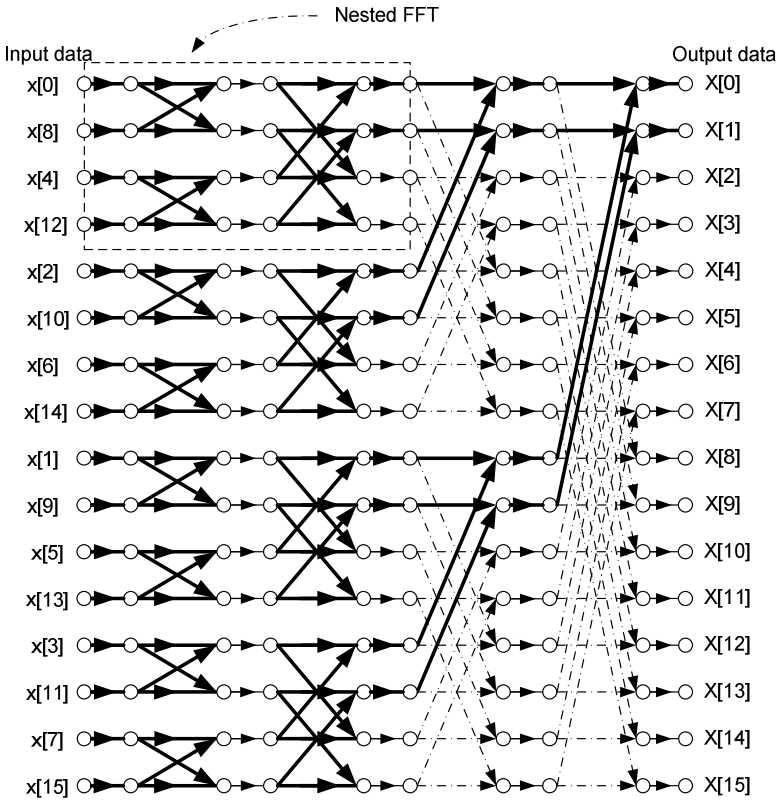
V případě, že není potřeba vypočítat celé spektrum signálu, ale jen jeho malou část (v našem případě se jedná o cca 1%), lze počet operací výrazně redukovat. Existuje řada metod pro výpočet části spektra (Getzel algoritmus, FFT s kmitočtovou lupou atd.) avšak tyto metody zanášejí do výpočtu systematickou chybu a nelze je v PCL systémech prakticky použít.

Proto jsme navrhli metodu výpočtu frekvenčního intervalu s redukcí počtu operací přímo z definice FFT tak, že jsou počítány jen ty motýlky/operace, které jsou potřeba pro výpočet dané spektrální čáry, respektive frekvenčního intervalu. Tento případ je zobrazen na obrázku 3.
Na tomto obrázku je nakreslen graf datových toků, kde jednotlivé uzly grafu representují výpočet jednoho motýlku FFT (radix-2). Zvýrazněné uzly pak representují ty uzly, které je třeba počítat pro zadaný interval frekvenčních čar. Jak je z obrázku patrné, až do určitého kroku je třeba počítat všechny motýlky ve „sloupci“, tj. počítá se sada vnořených FFT délky N_f. Jakmile je velikost vnořeného FFT větší než počet požadovaných spektrálních čar, počet aktivních motýlků se výrazně redukuje. Počet operací u tohoto algoritmu lze vyjádřit pro $\Delta < N_f$ takto:

$$N \log_2(N_f) + (2^\nu - 1)\Delta ,$$

kde N je počet bodů “původní“ FFT, N_f je počet bodů vnořeného FFT, Δ je počet spektrálních čar a $\nu = \log_2(N / N_f)$. V zadaném případě je pak redukce výpočtu dána poměrem

$$\frac{N \log_2 N_f + (2^\nu - 1)\Delta}{N \log_2 N} = \frac{1386920}{2228224} = 62\% .$$



Obrázek 3: Redukce výpočtů v rámci FFT

3.1 Architektura návrhu

Vzhledem k velkému objemu vstupních dat, předpokládáme jejich uložení do rychlých externích pamětí v akcelerátoru. V rámci optimalizace výpočetní náročnosti, bude přenos dat mezi hostitelským PC a externími paměťmi akcelerátoru řešen pomocí DMA přenosu po 64-bitové PCI sběrnici. Vstupní vektory pro signálový součin budou uloženy do čtyřech samostatných externích pamětí, čímž se umožní současné načítání reálné a imaginární složky obou datových vektorů.
Vlastní jádro algoritmu se sestává ze tří základních částí:

- výpočet signálového součinu
- výpočet vnořených FFT délky N_f
- výpočet zbylých motýlků FFT délky N

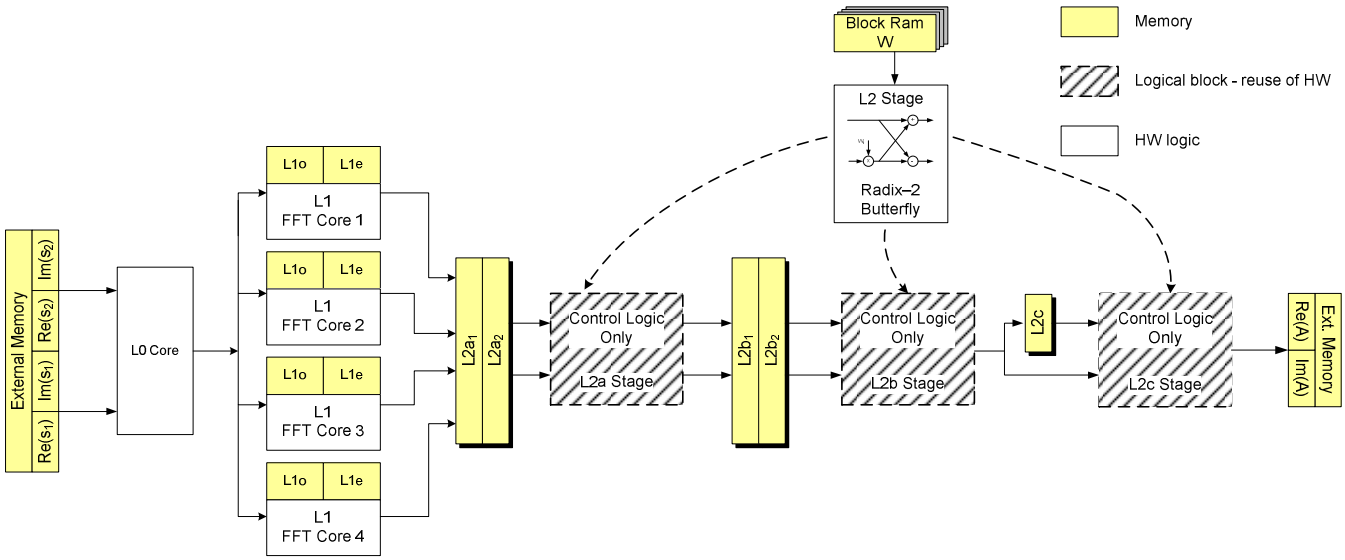
Blok výpočtu signálového součinu načítá hodnoty prvků vstupních vektorů z externích pamětí s určitým časovým/adresovým posunutím a provádí jejich skalární součin. Tento blok je tedy representován komplexní násobičkou a příslušným stavovým automatem obsluhující adresaci pamětí. Pro efektivní implementaci bloku FFT je do úvodního bloku signálového součinu zahrnut i první krok výpočtu FFT, ve kterém je hodnota koeficientu motýlku rovna hodnotě 1. Tím se operace motýlku redukuje na operaci součtu a rozdílu vstupních hodnot. Výstupní data jsou paralelně ukládána každý druhý hodinový cyklus do vnitřních blokových pamětí FPGA dedikovaných pro výpočet vnořeného FFT. Toto předzpracování dat (výpočet prvního motýlku) způsobí prodloužení dopravního zpoždění mezi načítáním vstupních dat a ukládáním dat pro zpracování FFT o jeden takt, ale výsledný datový tok zůstane stejný a navíc způsobí zkrácení výpočtu jednoho krátkého FFT o $N_f / 2$ cyklů.

Blok výpočtu jednoho vnořeného FFT se sestává ze dvou sad vnitřních pamětí pro uložení sudých a lichých vzorků vektorů. Každá sada pamětí obsahuje paměti pro uložení reálné a imaginární složky. Tímto způsobem lze zajistit plynulý tok dat do bloku pro výpočet motýlku a zároveň plynulé ukládání jeho výsledů. Každý blok motýlku je realizován příslušnými aritmetickými operacemi, bloky zpoždění pro synchronizaci zápisu výsledků do pamětí a tabulkou koeficientů W_N^{nk}. Vzhledem k symetričnosti tabulky koeficientů, byl použit efektivní způsob výpočtu koeficientu, který redukuje velikost tabulky koeficientů na $N_f/4$.Výpočet jednoho vnořeného FFT trvá přibližně $N_f(\log_2(N_f)-1)$ cyklů. Pro výpočet daného intervalu frekvenčních čar je potřeba vypočítat N/N_f těchto vnořených FFT.

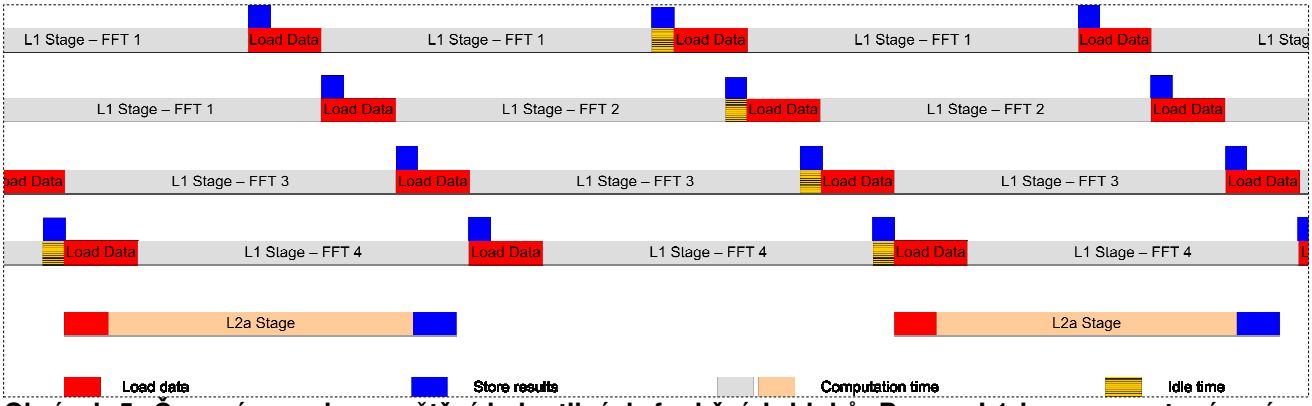
Poslední blok jádra algoritmu – výpočet zbylých motýlků – je realizován obdobným způsobem. Využívá pouze jeden blok (motýlek), který tentokrát obsahuje několik sad tabulek koeficientů W_N^{nk}. Finální výpočet je prováděn v několika sekvenčních blocích a jeho konkrétní realizace je závislá na konkrétních hodnotách N, N_f a Δ. V následujícím odstavci proto probereme konkrétní případ pro $N=2^{17}$ a frekvenční rozsah $\langle -300,+300\rangle$ Hz (tj. $\Delta=601$ spektrálních čar).

Pro výpočet 601 spektrálních čar je požadovaná nejmenší velikost vnořených FFT $N_f=1024$. Výpočet jednoho vnořeného FFT tedy zapere přibližně $N_f(\log_2(N_f)-1)=4608$ cyklů. Těchto krátkých FFT je třeba vypočítat celkem $N/N_f=128$ což representuje celkovou dobu výpočtu 589824 cyklů. Výpočet zbývajících motýlků pak obsahuje $(2^v-1)\Delta=76928$ cyklů. (Pozor! Uvedené údaje jsou odhadované a neobsahují časové intervaly nutné k načtení a uložení datových vektorů). Z uvedeného vyplývá, že nejnáročnější částí je výpočet krátkých FFT. Pro zvýšení efektivity je možné počítat několik těchto FFT paralelně. V našem případě jsme zvolily výpočet 4 paralelních bloků FFT, čímž se celková doba výpočtu první části zkrátí na 147456 cyklů.Výsledný návrh architektury je zobrazen na Obr. 4.

Čtyři bloky vnořených FFT se spouštějí se zpožděním/rozestupy 1024 cyklů, což je doba potřebná k naplnění příslušných pamětí. Po dokončení výpočtu vnořeného FFT je 601 koeficientů uloženo do vyrovnávací paměti. Doba výpočtu 4 po sobě spouštěných FFT trvá přibližně 10 000 taktů. Vždy po výpočtu 8 bloků FFT délky $N_f=1024$ je spuštěn výpočet tří stupňů zbývajících motýlků (viz. Obr. 4). Tím se nám redukuje velikost paměti pro ukládání mezivýsledků velkého FFT na $8\Delta=4808$ hodnot. Doba výpočtu tohoto prvního troj-stupně je přibližně $\Delta(2^3-1)=6615$ cyklů (což je doba přibližně stejná, jako doba potřebná pro výpočet osmi bloků vnořených FFT). Výsledky výpočtů jsou ukládány do druhé mezipaměti o stejné délce. Tento výpočet je prováděn paralelně s výpočtem vnořených FFT. Po dokončení výpočtů pro vektor délky N jsou provedeny zbývající výpočty sestávající ze čtyř stupňů motýlků, které trvají přibližně $\Delta(2^4-1)=9015$ taktů. Celkové funkční schéma je zobrazeno na Obr. **4** a časový rozvrh spouštění jednotlivých funkčních bloků je na Obr. 5.



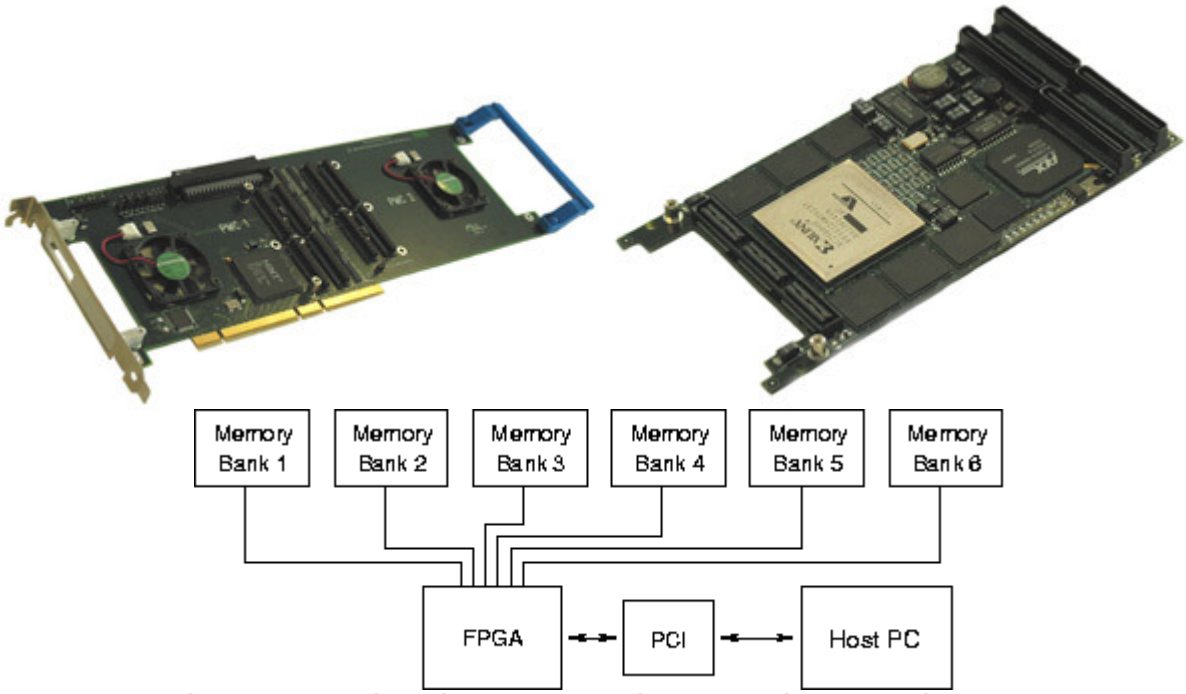
Obrázek 4: Blokové funkční schéma výpočtu frekvenčního intervalu pomocí FFT s redukcí počtu operací.



Obrázek 5: Časový rozvrh spouštění jednotlivých funkčních bloků. Proces L1 je representován výpočty FFT 1 až FFT 4. Bloky L2b a L2c se spouští po dokončení výpočtu všech bloků L1 a L2a, nutných pro výpočet jednoho FFT délky N a proto nejsou v rozvrhu zobrazeny.

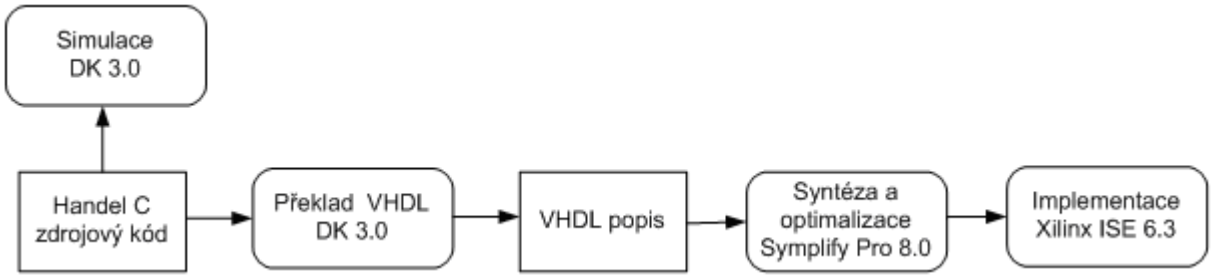
4. Použitá platforma a design flow

Jádro akcelerátoru pro výpočet CAF tvoří modulární vývojová deska RC2000 firmy Alpha Data. Deska je s hostitelským PC propojena pomocí 64-bitové PCI sběrnice a může obsahovat až dva vývojové moduly. Jeden modul obsahuje FPGA obvod firmy Xilinx XC2V6000 a 6 paměťových banků o velikosti 4Mbyte (ZBT SRAM), které jsou paralelně připojeny k FPGA obvodu. Dále je k dispozici jeden 128Mb DDR SDRAM modul a dva přídavné 6Mb paměťové moduly (ZBT RAM).



Obrázek 6: Blokové schéma hardwarového akcelerátoru pro výpočet CAF

Pro implementaci návrhu akcelerátoru byl použit programovací jazyk Handel-C. Návrh byl překládán do jazyka VHDL překladačem obsaženým v návrhovém prostředí DK 3.1 (oboje fy Celoxica). Následně byl použit nástroj pro syntézu a optimalizaci Symplify Pro 8.0 (fy Synplicity) a pro vlastní implementaci byl použit Xilinx ISE 6.3.



Obrázek 7: Schéma vývojového cyklu

V současné době je výpočet CAF implementován v 36-bitové aritmetice s pevnou řádovou čárkou s použitím obvodu XC2V6000 a to především z důvodů kratší doby syntézy a implementace (syntéza a implementace pro 42 bitů trvá přibližně o 1 hodinu déle a proto není vhodná pro ladící a experimentální účely). Návrh zabírá přibližně 61% logiky čipu, 66% hardwarových násobiček a 88% blokových SRAM pamětí. Implementace je provozována na 33MHz a to z důvodu stability DMA přenosu, který v současné době na vyšších frekvencích nefunguje spolehlivě. Výpočet jednoho FFT trvá 6,5ms, výpočet celého CAF pole trvá 4s. Celková doba výpočtu včetně přenosu dat a inicializace systému činí cca 7s.

V současné době je implementace akcelerátoru ve fázi ladění a testování. Reálné hodnoty doby trvání jednotlivých částí jsou shrnuty v Tabulce 1.

Proces	Počet cyklů	Doba výpočtu [33MHz]	Doba výpočtu [66MHz]
Příprava dat pro L1	1033	31,3 us	15,6 us
Výpočet L1	4825	146,2 us	73,1 us
Uložení výsledků L1	304	9,2 us	4,6 us
Příprava dat pro L2a / L2b	624	18,9 us	9,45 us
Výpočet L2a / L2b	4263	129,2 us	64,6 us
Uložení výsledků L2a / L2b	606	18,3 us	9,2 us
Příprava dat pro L2c	609	18,4 us	9,2 us
Celkem 1 x FFT o velikosti Nf	219084	6,6 ms	3,3 ms
Celkem 600x FFT o velikosti Nf	131450402	3,98 s	1,99 s

Tabulka 1: Doba trvání jednotlivých procesů výpočtu PCL. Pro vyjádření jednotlivých procesů jsou v tabulce použity následující zkratky: L1 – paralelní výpočet 4 vnořených FFT, L2a výpočet prvního troj-stupně, L2b – výpočet druhého troj-stupně, L2c – výpočet posledního stupně FFT.

5. Interface akcelerátoru CAF pro Matlab

Softwarová část projektu je navržena pro platformu PC a je určena pro testování projektu. Projekt byl vytvořen pod Microsoft VC++ .NET a je určen pro poslání dat do ADMXRCII karty a následné vyčtení výsledků. Výsledná dynamická knihovna obsahující funkce pro komunikaci s akcelerátorem je použita v komunikačních funkcích pro Matlab (.mex soubory). Tato makra umožňují celý návrh velice efektivně testovat přímo z prostředí Matlabu a tím efektivně využít všech jeho možností. Používání v **prostředí Matlab vyžaduje instalaci** podpory pro Matlab od firmy **Alpha Data** pro kartu ADM-XRC II (viz. Přiložené CD).

Upozornění: *Akcelerátor provádí po každém výpočtu jednoho motýlku FFT dělení dvěmi. Z tohoto důvodu je jeho výstup definován vztahem:*

$$FFT(\mathbf{x}) = \frac{1}{N}\sum x_n e^{j2\pi kn/N}$$

Matlab používá definici

$$FFT(\mathbf{x}) = \sum x_n e^{j2\pi kn/N}$$

Dělení je použito pro snížení chyby vlivem konečné délky slova a k celkovému snížení HW nároků.

Významе jednotlivých funkcí je následující:

d2f.m

Pomocné makro pro převod čísel double ⇒ float.

```
function out=d2f(x,bits)
out=floor(x*2^(bits-1));
```

f2d.m

Pomocné makro pro převod čísel float ⇒ double

```
function out=f2d(x,bits)
out=x/2^(bits-1);
```

clear_card.m

Makro nahraje prázdný design do FPGA. Používejte v případě problémů s přehříváním obvodu.

test.m

- Příklad jakým správným způsobem používat CAF akcelerátor v Matlabu.
- Makro vytvoří testovací vstupní data, nakonfiguruje ADMXRCII kartu, spustí výpočet a výsledky zobrazí do grafu.

```
%% Přednastavení proměnných
N=2^17;
Nlog=ceil(log2(N));
m=600;
   .
%% Příprava vstupních dat
t=1:N;
x=sin(2*pi*t*.001);
x1=sin(2*pi*t*.0005);
x1=[ x1 zeros(1,m) ];
   .
%% Inicializace karty ADM XRC II
h=admxrc_init;                          %Inicializace HW karty (ADMXRCII)

admxrc_config(h,'fft-36b.bit');     %Konfigurace HW karty předvytvořeným bitstreamem. !!
%admxrc_config(h,'fft-32b.bit');

admxrc_setclockrate(h,0,33);        % Přednastavení hodinové frekvence karty. !!

admxrc_close(h);            % Ukončení práce s HW kartou. (Aby s ním mohla začít pracovat funkce – caf() )

%% Vlastní výpočty CAF

tic;
%c=afft_send_data(d2f(x,28),d2f(x1,28));
c=caf(d2f(x,28),d2f(x1,28),0);   % Volání funkce caf() – výpočet caf (cross ambiguity function)
toc

%% Vykreslení výsledků
oo=f2d(c,32);
figure(1),plot(abs(oo(1:601)))
%figure,plot(abs(real(oo(1:601))))
%figure,plot(abs(imag(oo(1:601))))
ooo=reshape(oo,601,600);
```

```
figure(2),plot(abs(ooo))
```

6. Obsah a popis přiloženého balíku

Pokud je přiložen nějaký software, tak by zde měly být popsány všechny jeho součásti – adresářový strom, popis parametrů jednotlivých programů, atd.

```
cdrom - CAF    - obsahuje soubory s bitstreamy akcelerátoru a podpůrné
                 programy pro Matlab
      - doc    - obsahuje dokumentaci a přiložené články týkající se
                 akcelerátoru
      - adm_xrc_2  - Obsahuje podporu karty ADM XRC 2 od firmy Alpha Data.
                 Aktuální verzi lze stáhnout z ftp.alpha-data.com
```

7. Závěr

V současné době je výzkum ve stavu dokončení implementace navrženého řešení s redukcí počtu operací. Byla provedena numerická analýza požadované přesnosti aritmetických operací, navržen způsob redukce operací a byl proveden a analyzován návrh architektury akcelerátoru. Taktéž byla provedena implementace a celého výpočtu na hardwaru.

Celková doba výpočtu pole CAF je přibližně 4s. Je však nutné připomenout, že zatím nebylo využito plného paralelismu algoritmu - bloky L2b a L2c v současné době nepracují paralelně s výpočtem bloku L1 a fáze přípravy dat pro bloky L2b lze zcela zredukovat. V současném stavu se podařilo docílit zrychlení vůči PC implementaci (Matlab) o 80 až 90%.

Akcelerátor CAF je v současné době provozován s hodinovým kmitočtem 33MHz. Přechodu na 66MHz (maximální kmitočet PCI) zatím brání problémy s výkonnostními ztrátami v použitém typu FPGA obvodu, které způsobují přehřívání obvodu a poté jeho nekorektntí chování. Tyto jevy byli odstraněny v následné verzi pro obvody řady Virtex IV a Stratus II.

V neposlední řadě je součástí prezentovaného výzkumu návrh a realizace uživatelského prostředí pro práci s akcelerátorem. Vzhledem k tomu, že akcelerátor je určen k vývoji nových systémů pasivní radiolokace, předpokládáme jeho integraci s vývojovým systémem Matlab/Simulink. V končené fázi návrhu prvního řešení bylo otestováno i jednoduché rozhraní umožňují integraci akcelerátoru se systémem Matlab.

LITERATURA

[1] Jacobsen, E., Lyons, R.: *The Sliding DFT, Signal Processing Magazine, IEEE, vol. 20, No. 2, March 2003*

[2] Oppenheim, A. V., Einstein, C. J*: Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform, Proceedings of the IEEE, vol. 60, no. 8, August 1972*

[3] Knight, W. R., Kaiser R.: *A Simple Fixed-Point Error Bound for the Fast Fourier Transform, Acoustics, IEEE Trans. , vol. ASSP-27, no. 6, December 1979*

[4] Sorensen, H. V., Heideman, M. T., Burnus, C. S.: *On Computing the Split-Radix FFT, IEEE Trans., vol ASSP-34, no. 1, Feburuary 1986*

[5] Takahashi D.: *An Extended Split-Radix FFT Algorithm, Signal processing letters, IEEE, vol. 8, no. 5, May 2001*

[6] www.alpha-data.com

[7] www.celoxica.com