Západočeská univerzita v Plzni

Fakulta elektrotechnická

Katedra aplikované elektroniky a telekomunikací

DISERTAČNÍ PRÁCE

k získání akademického titulu doktor v oboru
Elektronika

Ing. Pavel Fiala

Efektivní řešení synchronizace v rámci softwarově definovaného rádia na FPGA

Školitel: Doc. Ing. Jiří Masopust, CSc.
Datum státní doktorské zkoušky:
Datum odevzdání práce:

ZČU, Plzeň 2016

Anotace

Cílem této doktorské práce je navrhnout vysoce efektivní synchronizační struktury, které lze využít v rámci softwarově definovaného rádia částečně implementovaného na hradlovém poli FPGA. Zaměřuji se na algoritmy pro symbolovou synchronizaci a synchronizaci fáze nosné vlny. V současnosti většina číslicového zpracování signálu v softwarově definovaném přijímači probíhá až na cílovém počítači a hradlové pole bývá využito pro signálové zpracování pouze částečně. Synchronizační algoritmy potřebují pro svoji správnou činnost digitální filtry a z tohoto důvodu je výhodné využít paralelního zpracování na FPGA. Hlavním limitujícím faktorem implementace rozsáhlých digitálních rádiových systému na osobním počítači je právě nedostatečný výkon související s omezenou možností paralelního zpracování dat. Hlavní náplní práce je rozbor vhodných synchronizačních algoritmů, návrh efektivní synchronizačních modelů a jejich následné ověření.

Klíčová slova

Číslicové zpracování signálu, hradlové pole, FPGA, komunikační systém, softwarově definované rádio

Abstract

This doctoral thesis is devoted to the proposal of highly efficient synchronization structures, which can be used within the concept of software defined radio partially implemented on an FPGA. I focus on algorithms for symbol synchronization and carrier phase synchronization. At present, the majority of digital signal processing in software-defined receiver runs on a personal computer and FPGA section is used for signal processing only partially. Synchronization algorithms need for their proper operation digital filters and FPGAs are suitable for parallel data processing. The main limiting factor in the implementation of extensive digital radio systems on a personal computer is just poor performance in parallel data processing. The object of interest will be analysis of appropriate synchronization methods, design an efficient synchronization models and their verification.

Key words

Communication system, Digital signal processing, FPGA, gate array, software defined radio

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této práce.

V Plzni dne …… ……

Ing. Pavel Fiala

Poděkování

Chtěl bych poděkovat vedoucímu práce doc. Ing. Jiří Masopustovi, CSc. a dalším kolegům z oddělení telekomunikační a multimediální techniky za konzultace během doktorského studia. Zvláštní poděkování patří zvláště Ing. Richardu Linhartovi, PhD.

Obsah práce

1 Úvod

Motivací pro vznik konceptu softwarově definovaného rádia (dále SDR) kolem roku 1991 byla snaha o realizaci víceúčelových komunikačních zařízení, která budou podporovat řadu přenosových standardů. Hlavním podnětem bylo nahradit pevné obvodové řešení rekonfigurovatelnou částí, která bude využívat číslicového zpracování signálu. Nástup digitálních a mobilních technologií v této době urychlil dále rozvoj tohoto nového oboru radiotechniky. Využití číslicového zpracování bylo zprvopočátku omezené a bylo dáno vlastnostmi digitalizačních obvodů, případně nízkým výkonem tehdejších signálových procesorů. Docházelo k postupnému zdokonalování bloků číslicového zpracování signálu a to především v základním pásmu (demodulace, synchronizace, zdrojové kódování, kanálové kódování atd.). Především s rozvojem hradlových polí FPGA a se zdokonalením ADC resp. DAC převodníků bylo možné posunout digitální zpracování směrem k anténnímu vstupu a to především do mezifrekvenční oblasti. Čistě digitální řešení, označované jako ideální softwarové rádio bez kmitočtové konverze v analogové části přijímače, je ve většině případů i v současnosti stále velmi obtížně realizovatelné.

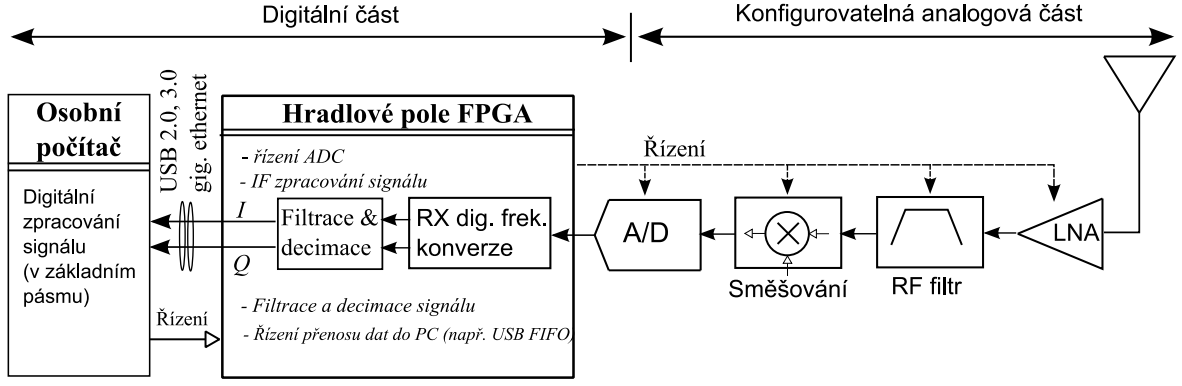
1.1 Zpracování číslicového signálu v dostupných přijímačích SDR

V současné době je podstatná část číslicového zpracování signálu v komerčně dostupných přijímačů SDR prováděna až na koncovém počítači. Počítačem nemusí být nutně míněn klasický osobní počítač, ale může se jednat například o specializovaný vestavěný počítač s jádrem ARM. Pouze malá část tohoto řetězce zpracování je ponechána na vlastním rádiu, které k tomuto účelu využívá hradlové pole, případně signálový procesor dle potřebného výkonu. Obvykle jsou zde pouze implementovány bloky řízení analogově digitálních převodníků, digitálního směšování signálu, decimace signálu a blok řízení přenosu signálu směrem do koncového počítače. Do počítače jsou potom přenášeny vlastní I/Q složky vzorkovaného signálu. To obvykle znamená, že potřebujeme přenášet velké objemy dat. V případě, že budeme uvažovat ADC s rozlišením 14-*bitů* (obvykle přenášeno do počítače jako 16-*bitů*) bude propustnost zhruba 10 MSPS v případě sběrnice USB 2.0 a do 25 MSPS v případě gigabitové ethernetu. Tento limit je sice možné v současné době překonat s využitím novějších sběrnic (např. USB 3.0, PCI-E nebo 10 gigabitové ethernetu), ale hlavní problém stále spočívá v nedostatečném výkonu x86 resp. x86-64 procesorů využívaných

v osobních počítačích pro zpracování signálu v reálném čase. To platí také ještě s větším důrazem pro vestavěné platformy, dnes ve značné míře postavené na architektuře ARM procesorů. Tento problém se plně projeví v případě komplexních přenosových systémů, které vyžadují vysokou paralelizaci výpočtů. Z tohoto důvodu je výhodné část signálového zpracování přesunout na hradlové pole FPGA. Příkladem komplexního přenosového systému může být přijímač digitální televize DVB-T nebo DVB-T2.

1.2 Cíle práce

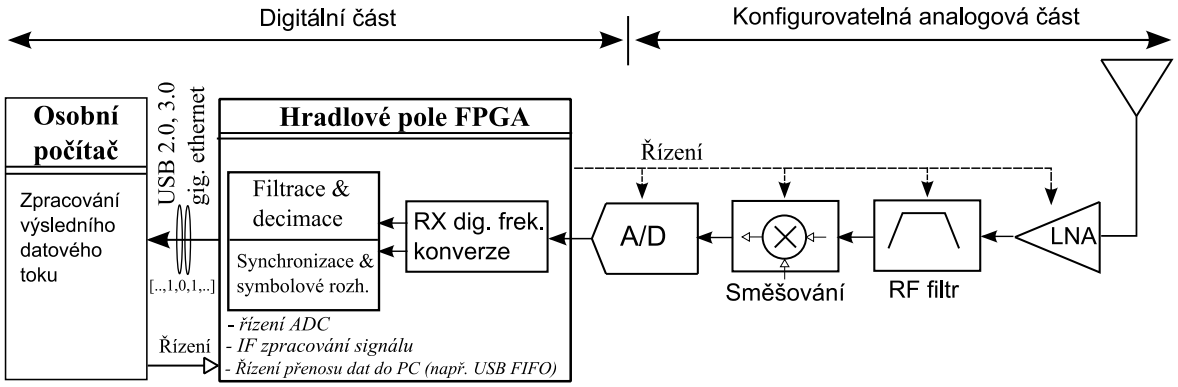
Cílem této disertační práce je návrh a analýza bloků číslicového zpracování signálu, které je možné přesunout na hradlové pole FPGA v rámci SDR přijímače. Tento přesun umožní uvolnění systémových zdrojů na PC a rozšíří například možnosti ukládání dat na pevný disk nebo lepší vizualizaci zpracovávaných dat pro uživatele. Obecně se reálné SDR rádio skládá ze dvou částí. První část lze označit jako rekonfigurovatelný digitální subsystém a druhou část lze nazvat jako softwarově konfigurovatelný analogový subsystém. Analogový subsystém obsahuje především bloky nízkošumových zesilovačů (LNA - low noise amplifier), selektivních filtrů a směšovačů. Blokový koncept je zobrazen na obrázku č. 1.1.



Obr. 1.1 Funkční schéma reálného SDR

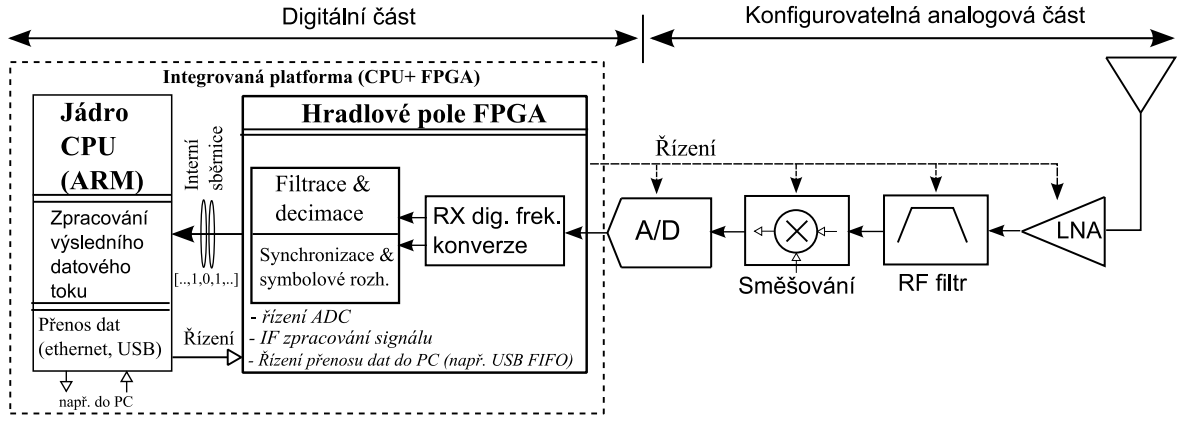
V práci se chci zaměřit na digitální část SDR přijímače a především na možnosti zpracování číslicového signálu na vlastním hradlovém poli FPGA přijímače. V kapitole 1.1 již byly popsány běžné implementované bloky na FPGA SDR. Každý digitální přijímač musí pro úspěšnou demodulaci signálu obsahovat část synchronizace. Konkrétně se jedná o bloky symbolové synchronizace a synchronizace fáze nosné vlny, které následují ihned po provedení základního zpracovanému signálu jako např. decimace signálu nebo digitální kmitočtová konverze. V případě, že provedeme kompletní demodulaci signálu (tj.

synchronizace + symbolové rozhodování) do počítače již přenášíme pouze výsledný datový tok. Na počítači potom můžeme implementovat bloky související s vlastním zpracováním datového signálu. Příkladem může být rámcová synchronizace nebo interní dekódování. Funkční schéma takového SDR je uvedeno na obrázku č. 1.2. Výhodou bude značné snížení požadavků na propustnost datového kanálu mezi FPGA a PC. V případě číslicového zpracování na FPGA implementujeme algoritmy často z důvodu dosažení maximálního výkonu v pevné řádové čárce, ale na moderním počítači pracujeme ve většině případů s plovoucí řádovou čárkou. Implementace některých algoritmů v pevné řádové čárce může být velice složitá a zdlouhavá. Z tohoto důvodu mohou být implementovány pouze na PC. Záměrem práce je právě navrhnout takové synchronizační algoritmy, které bude možné implementovat v pevné řádové čárce v určitém HDL jazyce a provést syntézu na FPGA.



Obr. 1.2 Modifikované schéma reálného SDR; do PC přenášíme výsledný datový tok

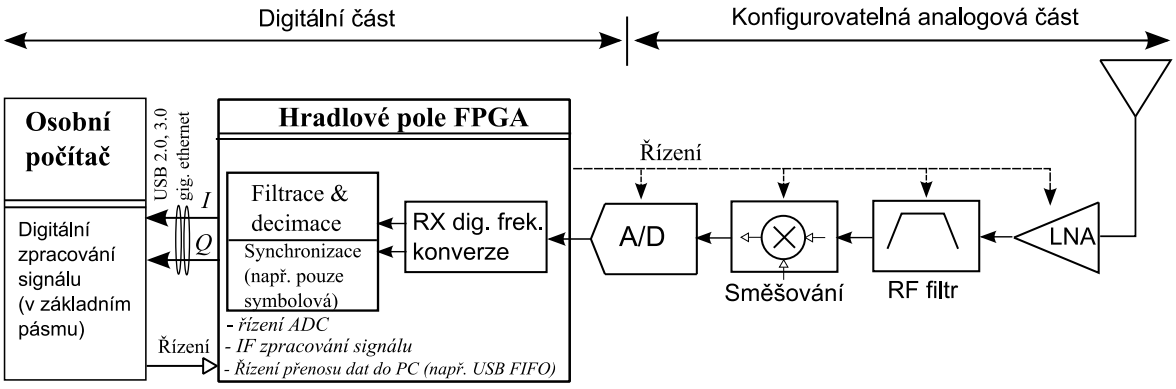
V průběhu psaní práce se na trhu objevily integrované platformy, které na jednom čipu obsahují integrovaný vestavěný procesor (jádro ARM – např. platforma *SocKit* od společnosti Altera) a vlastní hradlové pole FPGA. Blokové schéma je uvedeno na obrázku č. 1.3.



Obr. 1.3 Modifikované schéma reálného SDR s integrovanou platformou (CPU+FPGA)

Integrovaný procesor má přístup k paměťovému rozsahu, kde se nachází FPGA. Lze tedy interně předávat a řídit bloky implementované na FPGA. Výkon vestavěného procesu nemusí být dostatečný a přesunutí signálového zpracování na FPGA může být velice přínosné. Tato platforma umožní efektivní rozdělení zpracování signálu v pevné (FPGA) a pohyblivé řádové čárce (CPU ARM).

Další možností modifikace SDR je provedení synchronizace bez symbolového rozhodování a dále jsou přenášeny přímo I/Q vzorky do počítače (viz. obrázek 1.1). Tato vlastnost může být zajímavá při vývoji, neboť koncept SDR slouží často jako testovací platforma před návrhem zákaznického řešení na čipu (ASIC). Pro úplnost je tato modifikace ukázána na obrázku č. 1.4. Příkladem může být testování bloku symbolové synchronizace, kde potřebujeme rychle zobrazit vzorky signálu v I/Q diagramu.



Obr. 1.4 Modifikované schéma reálného SDR, synchronizace je provedena bez symbolového rozhodování, výhodné jako vývojová a testovací platforma

V teoretické i praktické části práce se budu zabývat rozborem a řešením následujících témat:

a) Návrh synchronizační modelů - na základě podrobného teoretického rozboru chybových detektorů pro symbolovou synchronizaci a synchronizaci fáze nosné vlny, budou nejprve vytvořeny jednotlivé synchronizační modely. Při odvození vztahů pro chybové detektory využiji metod maximální věrohodnosti. V průběhu návrhu se přednostně zaměřuji na jednotlivé aspekty implementace v pevné řádové čárce pro následnou efektivní syntézu na FPGA. Důležitým aspektem je možnost rychlé úpravy navrženého synchronizačního modelu pro konkrétní přijímač. Tento požadavek je velice důležitý, neboť implementace číslicové

zpracování signálu na FPGA může být velice zdlouhavý proces. Proces zahrnuje simulace v pevné i pohyblivé řádové čárce, zapsání algoritmu v HDL jazyku, návrh testovací procedury (testbench) a následnou syntézu. V práci se tedy zaměřuji i na automatizaci uvedeného procesu. Navržená struktura musí být rychle modifikovatelná pomocí generovaných parametrů. Navržené modely musí být pro svoji správnou funkci doplněny o podpůrné bloky. Jedná se především o bloky digitálních filtrů. Z tohoto důvodu se zaměřím na struktury FIR filtrů, které budou z důvodu paralelizace vhodné pro optimální syntézu na FPGA. Tyto filtry naleznou uplatnění jako přizpůsobené (polyfázové) filtry a interpolační filtry. Dalším důležitým podpůrným subsystémem je blok automatického řízení zisku.

a) Porovnání a optimalizace synchronizačních modelů – cílem je porovnat navržené modely z hlediska možnosti zařazení v demodulačním řetězci, chybovosti symbolů při využití daného chybového detektoru, složitosti implementace a výsledné syntézy na hradlovém poli FPGA. Na základě simulací ověřím, zdali je vhodné zařadit blok symbolové synchronizace před synchronizaci fáze nosné vlny nebo naopak. Dále bude vyhodnocována chybovost symbolů při využití určitého detektoru. Následně provedená syntéza pro určitý model, který bude na základě simulací popsán v HDL jazyce, ukáže využití systémových zdrojů FPGA (tj. počet obsazených logický prvků, vestavěných násobiček a paměťových bloků). Nakonec ověřím navržené synchronizační algoritmy v reálném přenosovém řetězci. Tento řetězec bude postaven na konceptu SDR z popsaných blokových schémat z obrázků č. 1.2 resp. 1.4. Dále bude doplněn o modulátor přenášeného signálu a simulátor kanálu s bílým šumem.

2 Digitální zpracování signálu pro softwarově definovaný přijímač

V následující kapitole budou popsány významné bloky číslicového zpracování signálu, které je nutné řešit v rámci digitální části softwarově definovaného přijímače. Zaměřuji se na aplikace fázového závěsu pro synchronizaci fáze nosné vlny a také na symbolovou synchronizaci. Hilbertova transformace je zmíněna z důvodu využití v rámci bloku synchronizace nosné vlny (Costasova smyčka).

2.1 Aplikace fázového závěsu

Fázový závěs (Phase-locked loop – dále *PLL*) lze v digitální komunikaci použít pro synchronizaci fáze nosné vlny nebo pro synchronizaci hodin přijímače se zdrojem ve vysílači (symbolová synchronizace) [1]. Základní stavební prvky *PLL* tvoří fázový detektor, zpětnovazební filtr (většinou) typu dolní propust (tzv. PLL loop filter) a napětím řízený oscilátor (VCO) a jeho základní struktura je zobrazena na obr. 1.1a.

Předpokládejme, že máme k dispozici vstupní signál $x(t)$ ve tvaru

$$x(t) = Asin\big(\omega_0 t + \theta(t)\big) \tag{2.1}$$

a výstup z VCO

$$y(t) = sin\left(\omega_0 t + \bar{\theta}(t)\right) \tag{2.2}$$

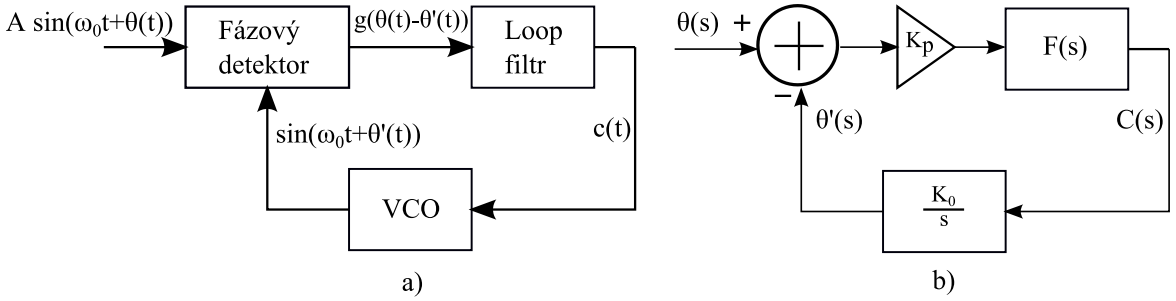
Fázový detektor je blok, který na svém výstupu vytváří funkci $g()$ tvořenou fázovým rozdílem obou vstupů. Výstupem je funkce $g(\theta(t) - \bar{\theta}(t))$ a rozdíl $\theta(t) - \bar{\theta}(t))$ se nazývá fázová chyba a lze jí označit jako θ_e. Výstup z fázového detektoru prochází filtrem, který produkuje řídící napětí pro VCO c(t). Výstup z VCO $y(t)$ v souvislosti se vstupem $c(t)$ lze vyjádřit za pomoci fázového odhadu $\bar{\theta}(t)$,

$$\bar{\theta}(t) = k_0 \int_{-\infty}^{t} c(x)dx \tag{2.3}$$

kde k_0 je konstanta úměrnosti nazývající se zisk VCO.

Pokud bude PLL pracovat správně, filtr bude na svém výstupu poskytovat napětí $c(t)$, které bude řídit VCO a produkovat odhad $\bar{\theta}(t)$ a tak snižovat fázovou chybu k nule. Fázový závěs je nelineární zpětnovazební systém, protože parametry fázového detektoru jsou

ovlivňovány nelineárními funkcemi jeho vstupů [1]. Z nelineárního systému lze provést transformaci na systém lineární tak, že systém linearizujeme k určitému pracovnímu bodu (zde θ_e=0). Ve většině případů je $g(\theta_e) \approx k_p\theta_e$ pro malé θ_e PLL lineární zpětnovazební systém a lze použít Laplaceovu transformaci (respektive Z transformaci pro diskrétní systémy) pro získání přenosové funkce.



Obr. 2.1 a) Základní struktura PLL b) Odvození přenosové funkce PLL

Základní PLL je možné charakterizovat pomocí fázové chyby $\theta_e(t)$ a VCO výstupu $\bar{\theta}(t)$. Přenosovou funkci pro obr. 1b lze odvodit následujícím způsobem.

$$\theta_e(p) = \theta(p) - \bar{\theta}(p), \text{přičemž } \bar{\theta}(p) = k_p F(p)\frac{k_0}{p}\theta_e(p) \tag{2.4}$$

$$\theta_e(p) = \theta(p) - k_p F(p)\frac{k_0}{p}\theta_e(p) \tag{2.5}$$

$$\theta_e(p) + k_p F(p)\frac{k_0}{p}\theta_e(p) = \theta(p) \tag{2.6}$$

$$\theta_e(p)(1 + k_p F(p)\frac{k_0}{p}) = \theta(p) \tag{2.7}$$

$$\frac{\theta_e(p)}{\theta(p)} = \frac{p}{p + k_0 k_p F(p)} \tag{2.8}$$

2.2 Aplikace Hilbertovy transformace

Hilbertova transformace je široce používána pro analýzu a zpracování pásmových signálu. Předpokládejme, že $x(t)$ je vstupní signál a $\bar{x}$(t) je jeho Hilbertova transformace. U Hilbertovy transformace je jak vzor, tak jeho obraz funkcí času. V časové oblasti je Hilbertova transformace definována integrálním vztahem,

$$\bar{x}(t) = x(t) * \frac{1}{\pi t} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau \tag{2.9}$$

kde * představuje konvoluci.

Frekvenční charakteristiku lze vyjádřit následujícím způsobem

$$H_h(\omega) = -j\,sign(\omega) = \begin{cases} -j\ pro\ \ \omega > 0 \\ 0\ pro\ \ \omega = 0 \\ j\ \ pro\ \omega < 0 \end{cases} \tag{2.10}$$

Hilbertovu transformaci signálu $x(t)$ lze realizovat tzv. Hilbertovým transformátorem (filtrem), který přenáší v celém frekvenčním pásmu $-\infty < \omega < \infty$ složky signálu $x(t)$ s nezměněnou amplitudou. Fázi složek signálu s kladnými frekvencemi ($\omega>0$) obrací o -90°.

Analytický signál (pre-envelope) lze definovat jako

$$x_+(t) = x(t) + j\bar{x}(t) \tag{2.11}$$

Ve frekvenční oblasti znamená konstrukce analytického signálu eliminaci záporných složek spektra a zdvojnásobení kladných složek spektra signálu. Komplexní obálka signálu (complex envelope) $x(t)$ vůči frekvenci ω_c je možné definovat tímto způsobem.

$$\tilde{x}(t) = x_+(t)e^{-j\omega_c t} \tag{2.12}$$

Pokud bude ω_c zvolena jako nosná frekvence, potom signál $\tilde{x}(t)$ bude přeložen do základního pásma. Vytvoření komplexní obálky má za následek demodulaci přenášeného signálu na nosné frekvenci ω_c. Impulzní odezva Hilbertovy transformace je nekonečná a proto nelze Hilbertovu transformaci přímo implementovat. Pro implementaci prostřednictvím FIR filtru lze například použít Remezova výměnného algoritmu [2].

2.3 Costasova smyčka jako aplikace fázového závěsu

Pro koherentní demodulaci je nutné znát velmi přesně frekvenci a fázi nosné vlny. Tyto parametry lze získat prostřednictvím *Costasovy smyčky*, která představuje modifikaci fázového závěsu [3]. Prostřednictvím Costasovy smyčky lze také sledovat dynamické změny frekvence vstupního signálu. Toto je důležité v satelitní komunikaci pro kompenzaci Dopplerova jevu. Dále bude představen koncept Costasovy smyčky, který je vhodný pro implementaci na FPGA, případně na signálovém procesoru.

Předpokládejme, že přijatý signál bude popsán následujícím výrazem,

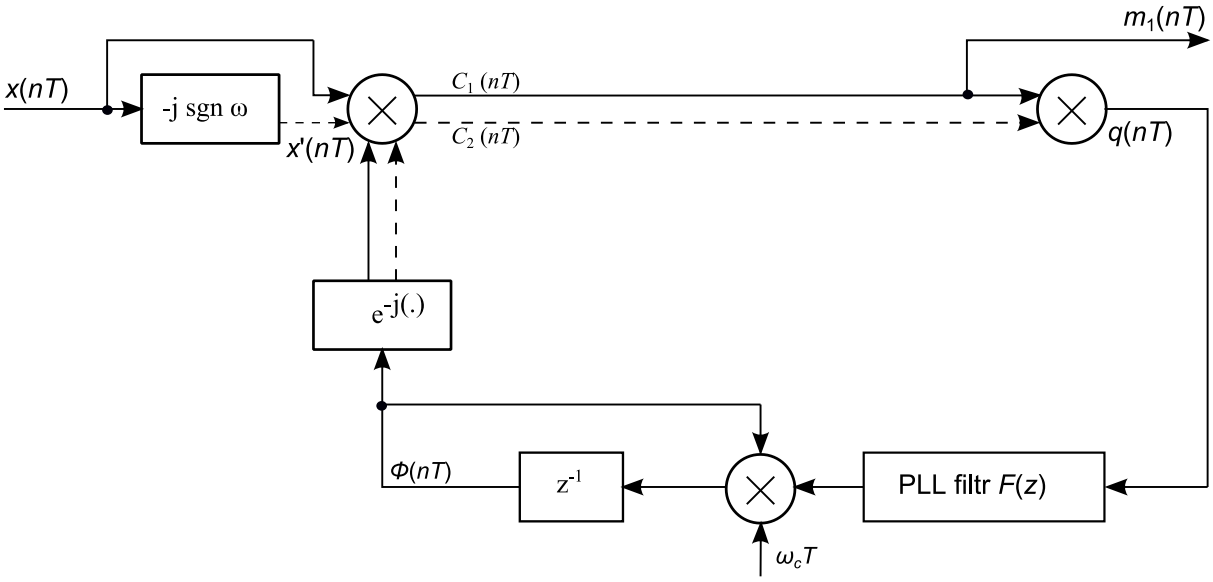
$$x(nT) = A_c m(nT)\cos(\omega_c nT + \theta_1) \tag{2.13}$$

kde ω_c je nosná frekvence a θ_1 je konstantní nebo pomalu se měnící fáze signálu.

Nyní lze vytvořit analytický signál (pre-envelope) prostřednictvím Hilbertovy transformace.

$$x_+(nT) = x(nT) + j\bar{x}(nT) = A_c m(nT) e^{j(\omega_c nT + \theta_1)} \quad (2.14)$$

Na obr. 2.2 je naznačen koncept diskrétní podoby Costasovy smyčky. V této podobě je ji možné přímo použít pro demodulaci BPSK. Plnou čarou je zobrazena reálná a čárkovanou čarou imaginární část komplexního signálu.



Obr. 2.2 Diskrétní podoba implementace Costasovy smyčky

Systém generuje odhad Φ(nT) pro určitý fázový posuv přijatého signálu,

$$\Phi(nT) = \omega_c nT + \theta_2(nT) \quad (2.15)$$

který je posléze násoben komplexním výrazem $e^{-j\Phi(nT)}$.

Signál lokálního oscilátoru je násoben vytvořeným analytickým signálem. Komplexní násobení lze zapsat takto:

$$c(nT) = A_c m(nT) e^{j(\omega_c nT + \theta_1)} e^{-j(\omega_c nT + \theta_2 nT)} = A_c m(nT) e^{j(\theta_1 - \theta_2 nT)} \quad (2.16)$$

Za použití výrazu $e^{j\Phi} = \cos(\Phi) + j\,sin(\Phi)$ lze zapsat reálnou a imaginární část výrazu $c(nT)$.

$$c_{1(real)} = A_c m(nT) \cos(\theta_1 - \theta_2(nT)) \quad (2.17)$$

$$c_{2(imag)} = A_c m(nT) \sin(\theta_1 - \theta_2(nT)) \quad (2.18)$$

Pokud bude fázová chyba θ_1-θ_2 malá, bude Costasova smyčka zavěšena. Pokud chyba bude rovna nule, demodulovaný signál bude roven výrazu c_{1real} a výraz c_{2imag} bude nulový. Signály c_{1real} a výraz c_{2imag} jsou nyní násobeny a poskytují signálovou složku q_n.

$$q(nT) = c_{1real} . c_{2imag} = A_c^2 m^2(nT) \sin\{2[\theta_1 - \theta_2(nT)]\} \quad (2.19)$$

Spodní část Costasovy smyčky představuje zpětnovazební (loop) filtr. Tento filtr musí mít nenulový stejnosměrný zisk, aby v ustáleném stavu byla za přítomnosti fázového ofsetu chyba fáze nulová. Dále musí mít nekonečný stejnosměrný zisk, aby v ustáleném stavu byla za přítomnosti frekvenčního ofsetu chyba fáze nulová.

Nejprve předpokládejme, že vstup do PLL se liší od výstupu VCO o fázový rozdíl $\Delta\theta$. To lze modelovat pomocí výrazu,

$$\theta(t) = \Delta u(t) \quad (2.20)$$

kde $u(t)$ je jednotkový skok. Tento případ je znázorněn na obr 2.3a. Strmost vstupního signálu (předpokládejme sinusovku) je úměrná frekvenci, která se v čase nemění. Laplaceova transformace pro jednotkový skok vypadá takto

$$\Xi(s) = \frac{\Delta\theta}{p} \quad (2.21)$$

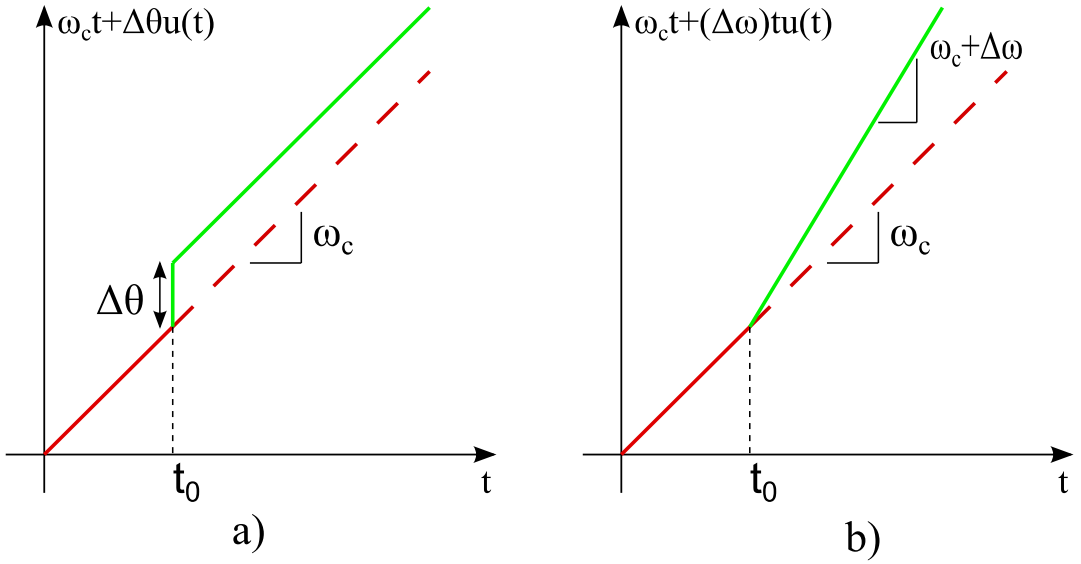
Další důležitý případ nastane, když se bude vstupní sinusový signál lišit proti výstupu z VCO o frekvenční ofset Δω rad/s. Tento případ je znázorněn na obr. 2.3b.

$$\sin\big((\omega_c + \Delta\omega)t\big) = \sin(\omega_c t + \Delta\omega t) \quad (2.22)$$

Laplaceova transformace je pro lineárně rostoucí funkci vyjádřena:

$$\Xi(s) = \frac{\Delta\omega}{p^2} \quad (2.23)$$

Dosazením (2.8) do (2.20) a řešením pro Ξ_e Laplaceova transformace pro jednotkový skok určité fázové chyby lze vyjádřit následujícím způsobem (pro ustálený stav $t \rightarrow \infty$).



Obr. 2.3 a) fázový rozdíl Δθ b) frekvenční ofset Δω

$$\Xi_{e_{step}}(\infty) = \lim_{s\to 0} p\Xi_e(p) = \lim_{s\to 0} \frac{p\,\Delta\theta}{p + k_0 k_p F(p)} = 0\ pro\ F(0) \neq 0 \tag{2.24}$$

Pokud bude mít filtr nenulový stejnosměrný zisk, bude v ustáleném stavu pro jednotkový skok chyba fáze nulová. Obdobně lze postupovat v případě frekvenčního ofsetu (místo jednotkového skoku použijeme rampu).

$$\Xi_{e_{ramp}}(\infty) = \lim_{s\to 0} p\Xi_e(p) = \lim_{s\to 0} \frac{p\,\Delta\omega}{p + k_0 k_p F(p)} = 0\ pro\ F(0) = \infty \tag{2.25}$$

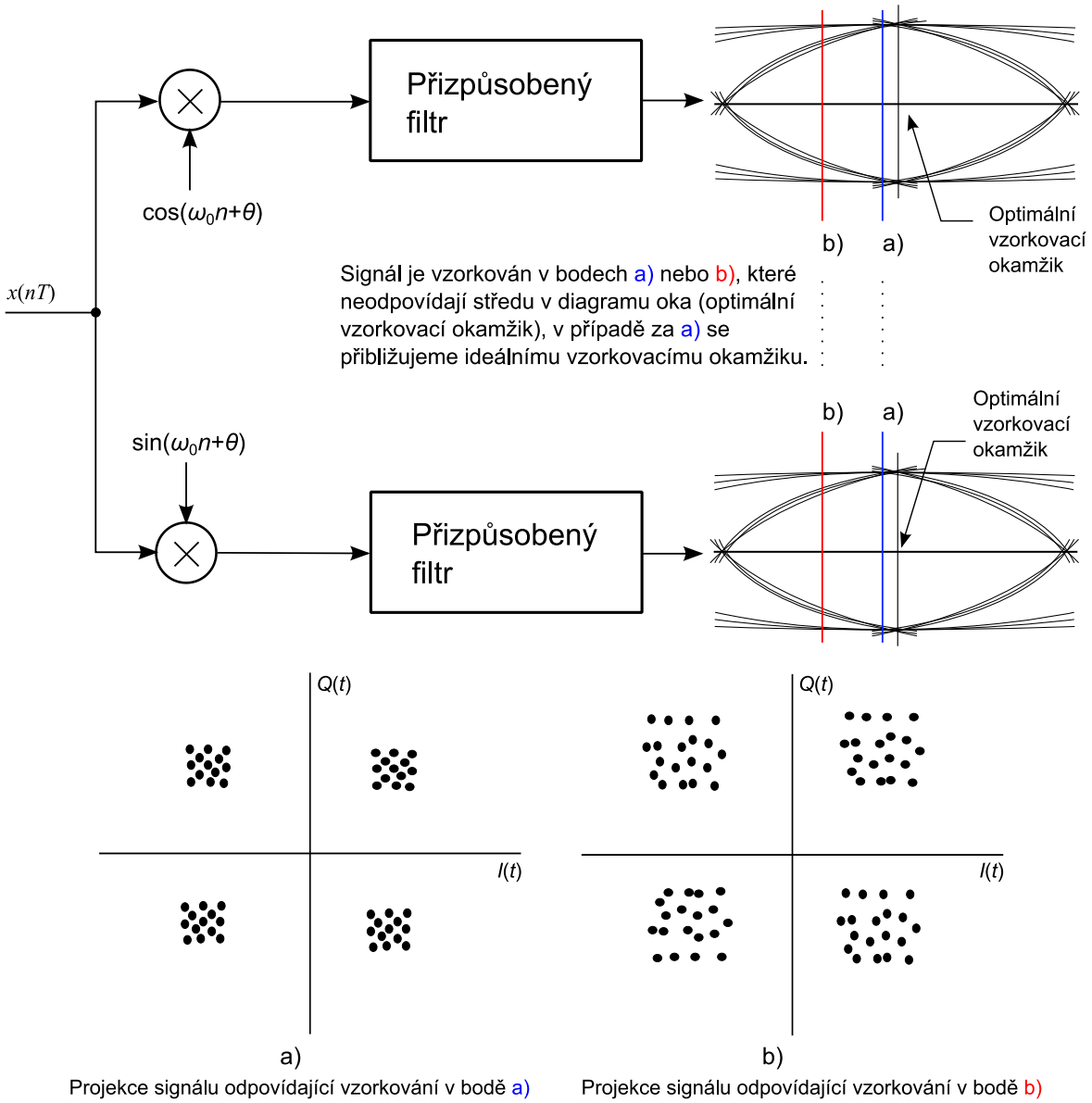
Pokud bude mít filtr nekonečný stejnosměrný zisk, bude v ustáleném stavu pro lineární funkci chyba fáze nulová. Tomuto vyhovuje např. proporcionálně integrační člen s přenosovou funkcí:

$$F(p) = k_1 + \frac{k_2}{p} \tag{2.26}$$

Napětím řízený oscilátor je charakterizován jedním pólem. Použitím filtru s p póly se vytvoří PLL jehož lineárně fázově ekvivalentní systém je řádu p+1. Běžně používaný systém druhého řádu dokáže sledovat odlišnosti fáze a frekvence. Je nutné poznamenat, že se musí jednat o konstantní frekvenční ofset. Pro kompenzaci dynamického frekvenčního ofsetu je nutné implementovat systém třetího řádu.

2.4 Symbolová synchronizace využívající fázového závěsu a interpolace

Proces obnovení časování symbolů představuje určení (odhad) hodinového signálu, který je shodný (ve fázi i frekvenci) s hodinovým signálem na straně vysílače [4]. Není efektivní přenášet samostatný hodinový signál v dalším rádiovém kanálu, proto musí být synchronizační informace získána z přenášeného datového signálu. Optimální vzorkovací okamžik pro soufázovou I a kvadraturní složku Q signálu z výstupu přizpůsobeného filtru odpovídá středu v digramu oka. Důvod využití bloku symbolové synchronizace je uveden na obrázku č. 2.4.



Obr. 2.4 Efekt chyby časování pro obecný signál *m-QAM* a projekce v *I*/*Q* diagramu

Chyba časování symbolů může být v diagramu oka chápána jako vzorkovací okamžik, kdy nedojde k maximálnímu otevření oka v centrální části digramu [5]. Tento efekt je možné blíže pozorovat po provedení projekce signálu v *I/Q* rovině. Ačkoliv je předpokládána dokonalá synchronizace fáze nosné vlny a k signálu v přenosovém kanálu nebyl přidán žádný šum, body v *I/Q* digramu budou „rozptýlené“. S rostoucí chybou časování symbolů bude tento efekt stále více patrný, jak je znázorněno v obrázku č. 2.4 – případ b).

Pokud budu uvažovat přijatý *PAM* signál popsaný následující rovnicí, kde

$$r(t) = \sum_k a(k)p(t - kT_s - \tau) + w(t), \tag{2.27}$$

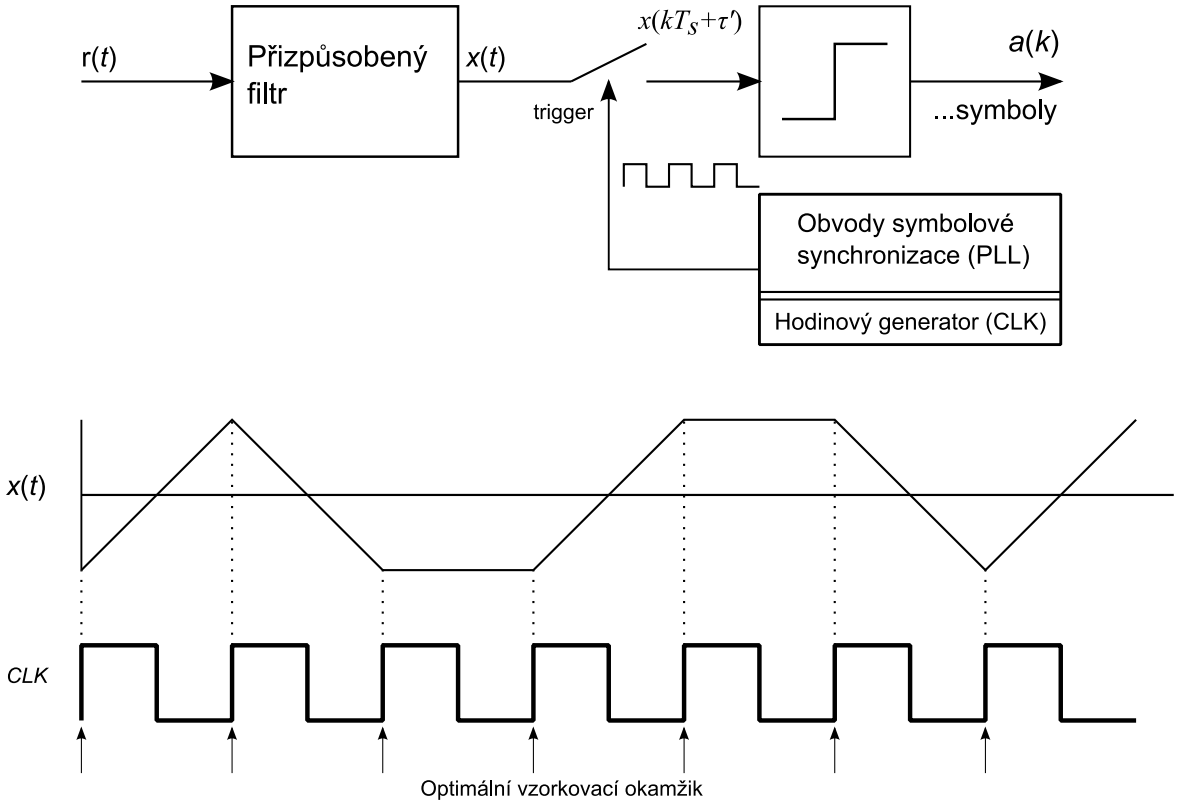
kde $a(k)$ je *k-tý* PAM symbol, T_s je symbolová perioda, τ je neznámé časové zpoždění, $p(t)$ je jednotková energie pulsu tvarovacího signálu na intervalu $-LT_S \leq T \leq LT_S$ a je $w(t)$ Gausovo bílý šum. Přijatý signál prochází dále přizpůsobeným filtrem, jehož impulsní odezva je $p(-t)$. Výstup filtru $x(t)$ může být vyjádřen rovnicí, kde

$$x(t) = \sum_k a(k)r_p(t - kT_s - \tau) + w(t) \tag{2.28}$$

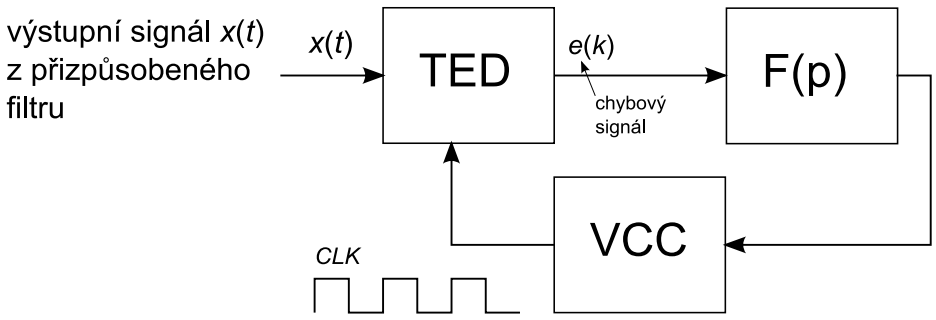
r_p je autokorelační funkce tvaru pulsu a $v(t) = w(t) * p(-t)$ označuje šum na výstupu přizpůsobeného filtru. V ideálním případě by měl být výstupní signál filtru vzorkován v čase $t=kT_s+\tau$ pro správnou detekci, jestliže je τ známé. V případě, že τ neznámé, je nutné použít předpokládanou hodnotu (odhad) $\bar{\tau}$ z výstupu synchronizátoru. Výstup filtru může být popsán v okamžiku $t = kT_s + \bar{\tau}$

$$x(kT_s + \bar{\tau}) = a(k)r_p(-\tau_e) + \sum_{m \neq k} a(m)r_p\big((k-m)T_s - \tau_e\big), \tag{2.29}$$

kde chyba určení časové polohy symbolu je $\tau_e=\tau-\bar{\tau}$. Smyslem symbolové synchronizace je vytvoření hodinové signálu, který je synchronizovaný (zarovnaný) se změnami datového signálu $x(t)$, jak je znázorněno na obrázku č. 2.5 (zde na náběžnou hranu hodinového signálu). Je možné použít fázový závěs pro získání hodinového signálu pro symbolovou synchronizaci. *PLL* pro obnovení symbolové synchronizace se obecně skládá z detektoru synchronizační chyby (*TED*), zpětnovazebního filtru s přenosovou funkcí $F(p)$ a z napětí řízeného hodinového generátoru (*VCC*). TED zjišťuje fázovou chybu mezi výstupem VCC a hodinovým signálem, který je zakódovaný v přijatém signálu. Tato chyba symbolového časování je po průchodu filtrem použita pro přizpůsobení VCC. Obecné schéma zpětnovazební synchronizace je uvedeno na obrázku č. 2.5.



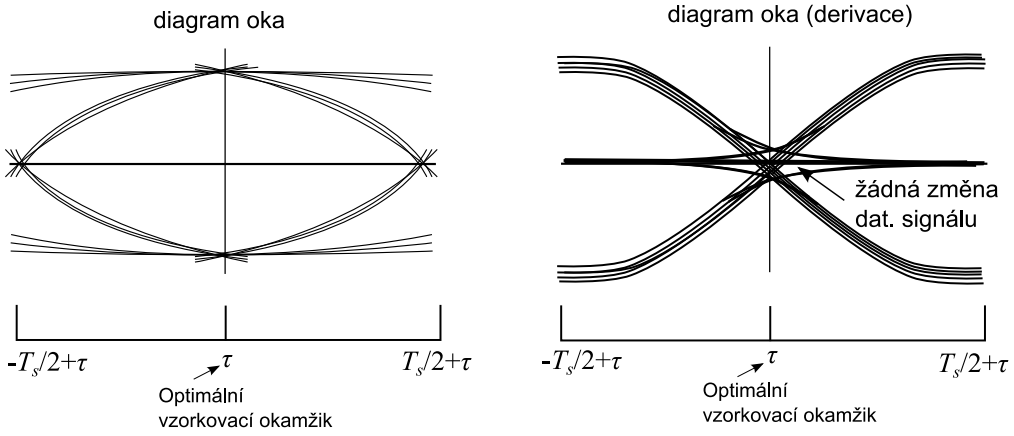
Obr. 2.5 Vytvoření hodinového signálu a jeho vztah k datovému signálu



Obr. 2.5 Obecné schéma zpětnovazební synchronizace

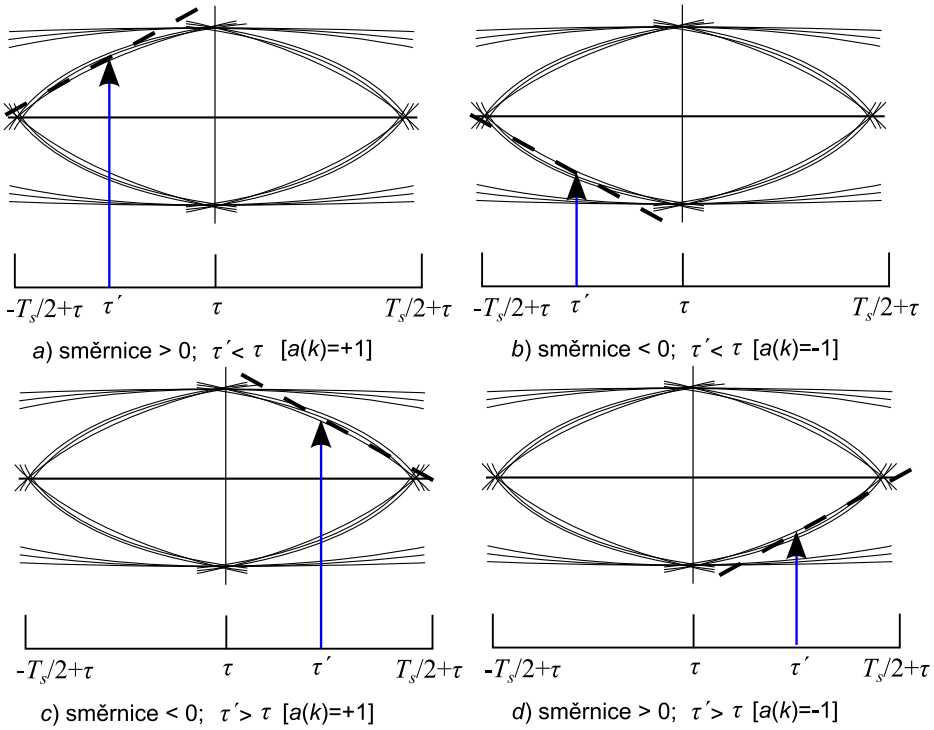
Optimální vzorkovací okamžik odpovídá v digramu oka jeho průměrnému maximálnímu otevření. Maximálního otevření oka nastává v časovém okamžiku, kdy (průměrná) směrnice tečny v diagramu oka je rovna nulové hodnotě. Nenulová směrnice v $t = \tau$ nastane v bodech odpovídající změně datového signálu (tj. změna znaménka po které nenásleduje změna datového signálu a naopak). Tento princip je naznačen také na obrázku č. 2.6. Pomocí směrnice tečny v digramu oka lze vyjádřit chybu časování symbolů, jak bude dále ukázáno. Blok symbolové synchronizace bude pracovat správně pouze tehdy, jestliže bude zajištěna

v pravidelných změna znaménka datového signálu. Signál, ve kterém se nacházejí velké shluky ‘1’ a ‘0’, je nutné transformovat pomocí skrambleru.



Obr. 2.6 Význam digramu oka pro odvození chybového signálu

Odvození základního vztahu pro chybový signál $e(k)$ je vysvětleno prostřednictvím obrázku č. 2.7 na základě grafických průběhů v diagramu oka.



Obr. 2.7 Využití směrnice tečny pro odvození chybového signálu $e(k)$. V případě za *a*) a *b*) nastane vzorkovací okamžik příliš brzo a symbol má hodnotu a(k)=+1 resp. a(k)=-1. V případě za c) a d) vzorkovací okamžik nastává příliš pozdě a symbol nabývá hodnot a(k)=+1 resp. a(k)=-1. Znaménko symbolu hraje důležitou roli v případech za *b*) a *d*).

V případě za *a*) nastane vzorkovací okamžik příliš brzy ($\hat{\tau}(k) < \tau\ a\ \tau_e(k) > 0$) a symbol nabývá hodnot *a*(*k*)=+1. Směrnice tečny v bodě $\bar{\tau}(k)$ bude mít kladnou hodnotu a může být využita přímo k výpočtu chybového signálu *e*(*k*). Protože vzorkovací okamžik v tomto případě nastal příliš brzy, další vzorkovací okamžik $\bar{\tau}(k+1)$ musí být zvětšen oproti $\bar{\tau}(k)$. To lze zajistit zvýšením periody výstupu *VCC*. V dalším případě za *c*) nastane vzorkovací okamžik příliš pozdě ($\hat{\tau}(k) > \tau\ a\ \tau_e(k) < 0$) a symbol nabývá hodnot *a*(*k*)=+1. Směrnice tečny v bodě $\bar{\tau}(k)$ bude mít zápornou hodnotu a může být využita přímo k výpočtu chybového signálu *e*(*k*). Další vzorkovací okamžik $\bar{\tau}(k+1)$ musí nastat dříve než $\bar{\tau}(k)$. Perioda *VCC* musí být snížena.

Nyní se zaměřím na případy za *b*) a *d*), kde bude *a*(*k*)=-1. V případě možnosti za *b*) nastává vzorkovací okamžik opět příliš brzy ($\hat{\tau}(k) < \tau\ a\ \tau_e(k) > 0$) a perioda *VCC* musí být zvýšena pro splnění podmínky $\bar{\tau}(k+1) > \bar{\tau}(k)$. Směrnice tečny v bodě $\bar{\tau}(k)$ bude mít zápornou hodnotu a nemůže být tedy přímo využita pro výpočet chybového signálu *e*(*k*). Správnou hodnotu *e*(*k*) lze ovšem získat změnou znaménka směrnice tečny v bodě $\bar{\tau}(k)$ s pomocí hodnoty symbolu *a*(*k*)=-1 (pouhým vynásobením). Podobně lze postupovat v případě za *d*), kdy vzorkovací okamžik nastává příliš pozdě ($\hat{\tau}(k) > \tau\ a\ \tau_e(k) < 0$). Směrnice tečny v bodě $\bar{\tau}(k)$ bude mít kladnou hodnotu a musí být opět modifikována za pomoci hodnoty symbolu *a*(*k*)=-1. Z předchozích komentářů k obrázku č. 12 lze odvodit vztah pro chybový signál (PAM)

$$e(k) = a(k)\dot{x}(kT_s + \bar{\tau}) \tag{4.30}$$

pro případ známé datové posloupnosti (Data Aided Detector – DAD). Pro případ neznámé datové posloupnosti (Non Data Aided - NDA) lze využít přístup s využitím rozhodovací úrovně (Decision Directed – DD) kde chyba *e*(*k*) je vyjádřena jako

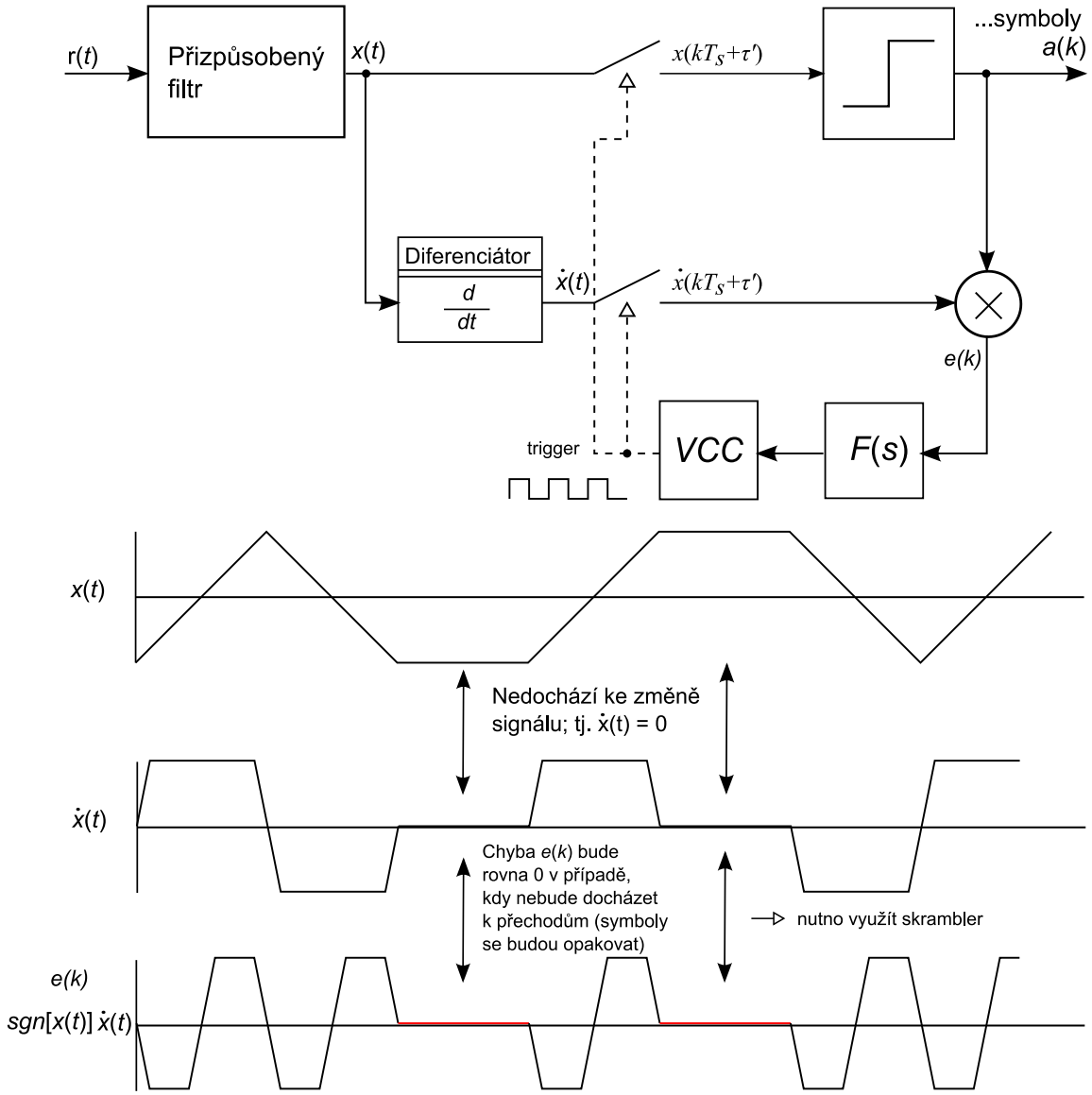
$$e(k) = \hat{a}(k)\dot{x}(kT_s + \bar{\tau}) \tag{2.31}$$

a symbol vyjádřit pomocí znaménkové funkce

$$\hat{a}(k) = sgn[\dot{x}(kT_s + \bar{\tau})]. \tag{2.32}$$

Blokové schéma synchronizačního systému s PLL postavené na výrazu z rovnice (2.32) je uvedeno na obrázku č. 2.8. Synchronizační schéma vyžaduje blok diferenciátoru připojený na výstup přizpůsobeného filtru. Za diferenciátorem je nutné signál vzorkovat v čase $t = kT_s + \bar{\tau}$ pro získání směrnice tečny v bodě $\hat{\tau}(k)$. Posledním krokem je v případě využití *DD* principu vynásobení signálu *e*(*k*) znaménkovou funkcí vzorkované signálu z přizpůsobeného filtru.

Dále bude ukázáno, že tento typ detektoru využívá metod maximální věrohodnosti (Maximum Likelihood).



Obr. 2.8 Význam digramu oka pro odvození chybového signálu

Uvedený typ detektoru je založen na výpočtu derivace signálu z výstupu přizpůsobeného filtru. Je zřejmé, že detektor tohoto typu bude pro svoji správnou funkci vyžadovat relativně vysoký počet vzorků na periodu. Komplexnost detektoru lze snížit nahrazením derivace diferencí, jak je naznačeno na obrázku č. 2.9 a 2.10. Protože platí

$$\dot{x}(t_0) = \lim_{n \to 0} \frac{x(t_0 + \Delta) - x(t_0 - \Delta)}{2\Delta}, \tag{2.33}$$

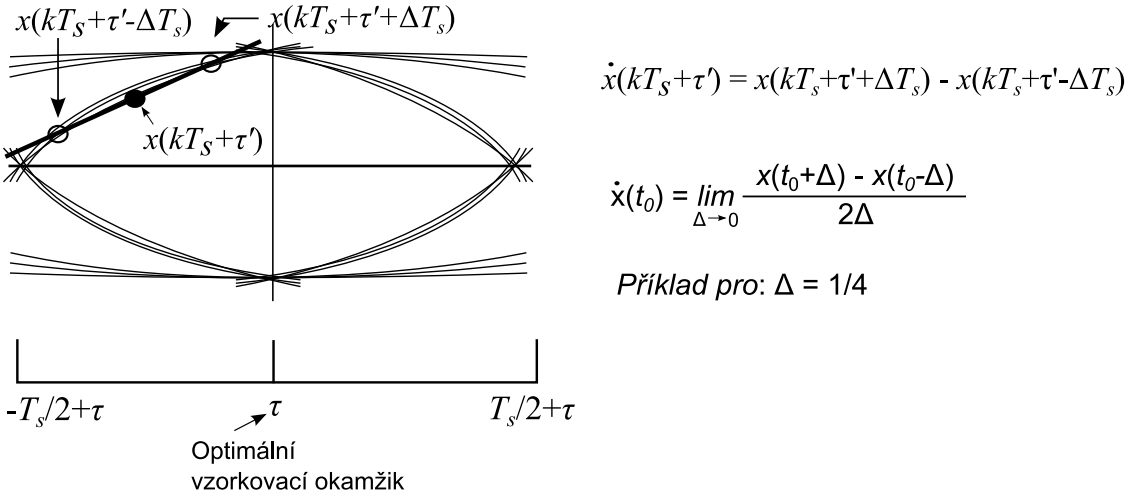
derivace signálu může být aproximována diferencí pro snížení komplexnosti detektoru. Na tomto principu je založen detektor s původním anglickým názvem „*early-late gate detector*“ [6]. V případě známé datové posloupnosti lze chybový signál vyjádřit jako

$$e(k) = a(k)[x(kT_s + \hat{\tau} + \Delta T_s) - x(kT_s + \hat{\tau} - \Delta T_s)] \tag{2.34}$$

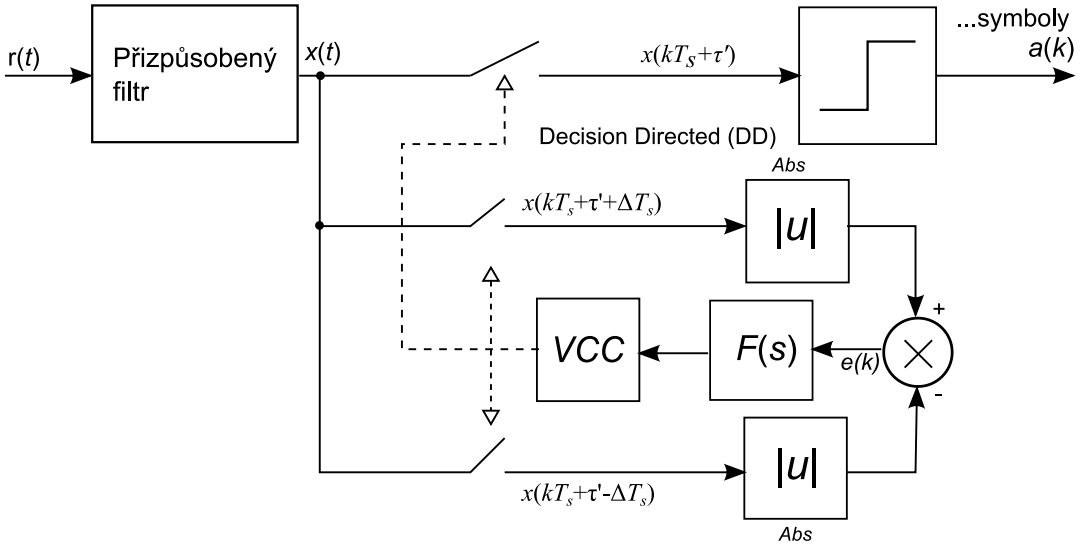
V případě DD lze chybový signál přepsat do podoby

$$e(k) = a(k)|x(kT_s + \hat{\tau} + \Delta T_s)| - |x(kT_s + \hat{\tau} - \Delta T_s)| \tag{2.35}$$

Tento typ detektoru je základem pro další detektory (např. detektor průchodu nulou). Může pracovat s malým počtem vzorků na symbolovou periodu, z principu funkce to jsou minimálně 2 vzorky / symbol (Δ = ½). Nevýhodou tohoto detektoru je jeho vlastní šum, který může mít vliv na smyčku fázového závěsu v případě nedostačených změn datového signálu.



Obr. 2.9 Využití diference pro aproximaci derivace pro „*early-late*“ detektor

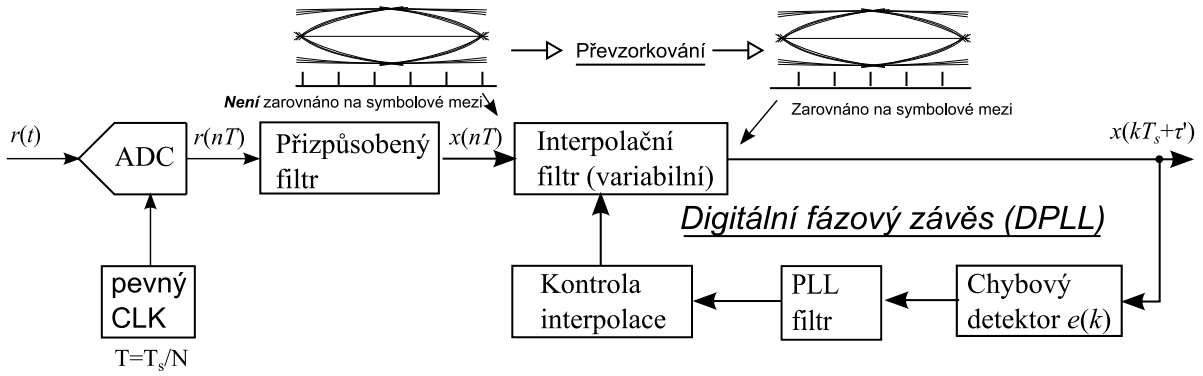


Obr. 2.10 Smyčka fázového závěsu s „*early-late*“ (DD) detektorem

Předchozí případy synchronizačních systémů předpokládali, že přizpůsobený filtr je implementován v analogové podobě a je modifikována vzorkovací perioda *ADC* na základě výstupu *VCC* fázového závěsu. Tento přístup je ovšem v reálných podmínkách velice obtížně realizovatelný. Vzorkovací kmitočet ADC je většinou odvozen od krystalového oscilátoru, který má daný mikroprocesor nebo obvod FPGA k dispozici. Je možné sice modifikovat tento kmitočet (i ve zlomkovém poměru) prostřednictvím interních PLL daného obvodu, ale tyto integrované PLL bývá nutné inicializovat před vlastním během programu resp. logického obvodu. Rekonfigurovatelné integrované PLL jsou v současné době dostupné u některých obvodů FPGA [7], stále je nutné počítat s časovým intervalem zavěšení PLL. Z toho vyplývá skutečnost, že vzorkovací perioda T musí být pevná. Přizpůsobený filtr bude implementován číslicově a signál PAM z výstupu přizpůsobeného filtru je možné následně popsat vztahem

$$x(nT) = \sum_{k} a(k)p(nT - kT_s - \tau) + w(nT) \tag{4.36}$$

kde $a(k)$ je *k-tý* PAM symbol; T je vzorkovací perioda; T_s je symbolová perioda; τ je neznámé časové zpoždění; $p(nT)$ je jednotková energie pulsu tvarovacího signálu na intervalu $-LT_S \leq T \leq LT_S$ a $w(nT)$ je Gausovo bílý šum s výkonovou spektrální hustotou $\mathrm{N}_0/2\ \mathrm{W/Hz}$. Vzorkovací rychlost $1/T$ je asynchronní vůči symbolové rychlosti $1/T_s$. Neznámé časové zpoždění τ je nutné určit z dostupných vzorků $x(nT)$. Tyto vzorky nejsou zarovnány vůči symbolové mezi. Signál je tedy nutno převzorkovat (ve zlomkovém poměru) prostřednictvím adaptibilního interpolátoru resp. interpolačního filtru. Vzorky z výstupu interpolačního filtru již budou v ideálním případě zarovnány na symbolové mezi. Principiální blokové schéma tohoto interpolačního řetězce je uvedeno na obrázku č. 2.11. Nevýhodou tohoto přístupu je interpolační jitter, který nastane v případě $T_i \neq NT$ [8].



Obr. 2.11 Zpětnovazební synchronizace s interpolačním filtrem

3 Návrh struktur FIR filtrů vhodných pro implementaci na hradlovém poli FPGA

Digitální filtry představují jeden ze základních stavebních bloků digitálního přijímače i vysílače. Filtry s konečnou impulsní odezvou (FIR) jsou velice často používány pro modifikování parametrů signálu v oblasti SDR. Jedna z hlavních výhod FIR filtrů je, že lineární fázové charakteristiky filtru je možné dosáhnout, pokud koeficienty filtru splňují jednu z následujících symetrií (kauzální formulace) [9]

$$h(M-n) = h(n) \quad pro \ n = 1,2,\dots,M \tag{3.1}$$

nebo

$$h(M-n) = -h(n) \quad pro \ n = 1,2,\dots,M. \tag{3.2}$$

FIR filtry jsou v digitální komunikaci často používány jako tvarovací filtry (pulse shaping filters) resp. jako přizpůsobené filtry [10] (matched filters) nebo pro změnu vzorkovací frekvence (decimační a interpolační filtry). Dále jsou uvedeny tři důležité aplikace FIR filtrů pro oblast SDR, které budou později diskutovány.

- *Square-Root Raised Cosine Filter* (SRRC) kosinové filtry typu „*roll-off*“ rozdělují filtraci mezi vysílací a přijímací stranu s typickým koeficientem $\alpha = 0.5$ nebo $\alpha = 0.35$. S využitím stejného filtru na obou stranách komunikačního řetězce lze dosáhnou minimálních mezisymbolových interferencí (ISI).
- Digitální *Hilbertovy transformátory* tvoří speciální třídu digitálních filtrů, které provádí fázový posuv vstupního signálu o $\pi/2$. Primárně jsou tyto filtry využity pro vytvoření imaginární části komplexního signálu na základě jeho reálné části. Digitální implementace *Costasovy smyčky* na bázi *PLL* využívá právě tento typ filtru.
- Subsystém symbolové synchronizace také využívá digitální filtry za účelem interpolačního procesu. Zde aplikovaná *Farrow struktura* [6] je velice efektivní pro implementaci prostřednictvím hardwaru (například hradlového pole FPGA nebo ASIC). Je zde možné efektivně využívat aplikaci parabolického nebo kubického interpolátoru implementovaného jako FIR filtr.

Všechny uvedené aplikace lze implementovat právě jako filtry s konečnou impulsní odezvou. Z tohoto důvodu se zaměřuji na návrh struktur FIR filtrů, které lze efektivním

způsobem implementovat v HDL jazyce a následně provést RTL syntézu na FPGA. Základním stavebním prvkem navržených struktur jsou matematické metody distribuované aritmetiky, které jsou popsány v následující kapitole.

3.1 Aplikace distribuované aritmetiky

Distribuovaná aritmetika (DA) [11] představuje obecně skupinu algoritmů, které pro výpočet výsledku násobení a následného přičtení do akumulátoru (Multiply and accumulate – MAC) používají vyhledávací paměťové tabulky (Lookup table – LUT). Lineární konvoluci lze formulovat jako součet částečných produktů násobení

$$y = \sum_{n=0}^{N-1} c[n] \times x[n] = c[0]x[0] + c[1]x[1] + .. + c[N-1]x[N-1]. \tag{3.3}$$

Ve velkém počtu DSP aplikací není třeba používat obecné násobičky a filtrace signálu není výjimkou. Pokud koeficienty filtru $c[n]$ budou konstantní, potom částečný MAC produkt obecného násobení $c[n]\times x[n]$ přejde na prosté násobení s konstantou. To je základní předpoklad pro aplikaci metod distribuované aritmetiky. Proměnou $x[n]$ můžeme v systému DA bez znaménka vyjádřit jako

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \times 2^b \; kde \; x_b[n] \in \{0,1\} \tag{3.4}$$

kde x_b je b bit $x[n]$. Konečný produkt konvoluce y lze vyjádřit jako

$$y = \sum_{n=0}^{N-1} c[n] \times \sum_{b=0}^{B-1} x_b[n] \times 2^b. \tag{3.5}$$

Redistribucí pořadí sčítaní ve výrazu (6.5) se dostaneme ke konečnému výsledku

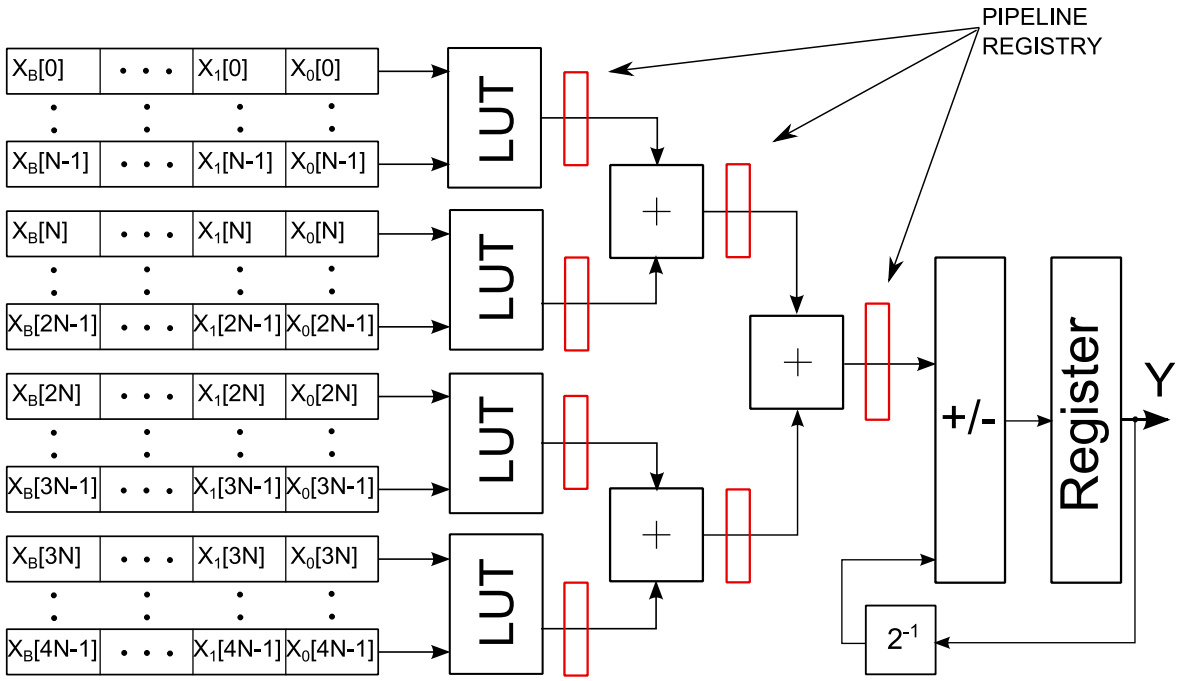
$$y = \sum_{n=0}^{N-1} c[n] \times \sum_{b=0}^{B-1} x_b[n] \times 2^b \; = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n] \times x_b[n]) \, . \tag{3.6}$$

Implementace funkce nebo přesněji mapování $f(c[n]\times x[n])$ v rovnici (3.6) vyžaduje zvláštní pozornost. Na FPGA lze provést toto mapování pomocí LUT tabulky. Další možností je využití integrovaných bloků paměti RAM, kde příkladem mohou být bloky M4K, M9K nebo M512K [12], které se nacházející na hradlových polích od společnosti Altera. Problémem ovšem v tomto případě může být jejich omezené množství. To platí především pro základní a levnější řady FPGA [13]. Proto implementace mapování pomocí LUT tabulky může být preferovaným řešením a toto řešení je dále popsáno. Tabulka 2^N LUT je naprogramována tak, aby na vstupu akceptovala vektor x_b=(x_b[0],x_b[1],

$x_b[2],\ldots x_b[N\text{-}1]$) a na výstupu poskytovala požadované mapování $f(c[n]\times x[n])$. Dále jsou tato mapování násobena odpovídající mocninou o základu dva a akumulována. Výsledná struktura nevyužívá násobiček. Je využito pouze bitového posuvu. Tyto násobičky mohou být s výhodou využity ve specializovaných DSP blocích SDR, kde jsou nezbytně potřeba. Na první pohled je sice možné konstatovat, že řešení je vhodné pouze pro velmi krátké filtry z důvodu omezeného adresovatelného paměťového prostoru jedné LUT tabulky. Protože FIR filtry patří do skupiny lineárních filtrů, kombinací N filtrů nižšího řádu lze vytvořit filtr vyššího řádu [14]. Rychlost výpočtu není závislá na počtu koeficientů filtru v případě důsledného využití zřetězeného zpracování. Výpočetní systém DA lze rozšířit pro práci s čísly se znaménkem (dvojkovém doplněk), neboť MSB bit čísla ve dvojkovém doplňku rozlišuje mezi zápornými a kladnými čísli následujícím způsobem

$$y = -2^B \times f(c[n] \times x_b[n]) + \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n] \times x_b[n]). \tag{3.7}$$

Obecná struktura takového filtru s DA (se znaménkem) je uvedena na obrázku č. 3.1. Využití zřetězeného zpracování (pipeline registry) umožňuje dramaticky zvýšit maximální frekvenci zpracování dat.

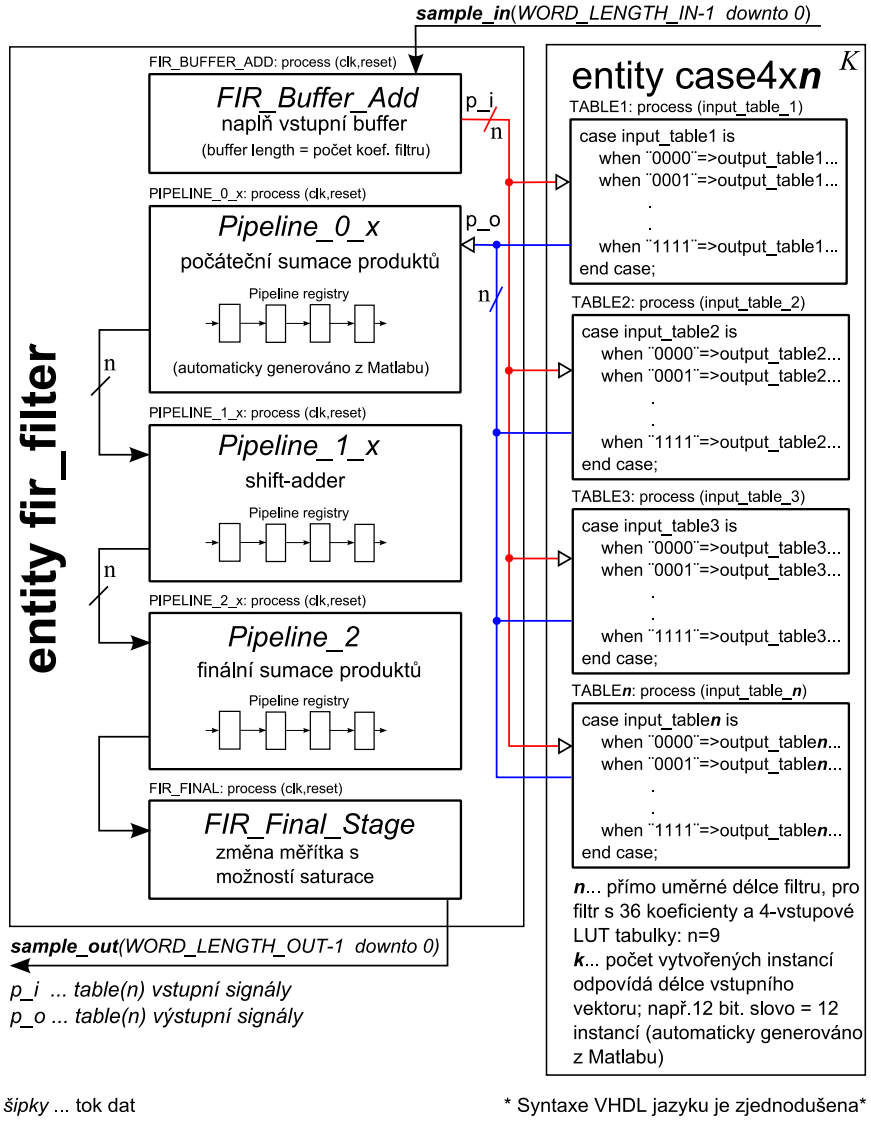


Obr. 3.1 Obecná struktura DA FIR filtru se zřetězeným zpracováním dat

V dalších kapitolách budou popsány navržené struktury FIR filtrů založené na distribuované aritmetice, které lze aplikovat do oblasti softwarově definovaného rádia. Je popsána navržená plně paralelní a sériová struktura. Rozšířením plně paralelní struktury vznikly polyfázové struktury pro decimační a interpolační filtry. Jednotlivé modely filtrů byly navrženy v jazyce VHDL.

3.2 Plně paralelní model DA FIR filtru

VHDL struktura plně paralelního filtru se zřetězeným zpracováním je uvedena na obrázku č. 3.2. Tento typ filtru provádí výpočet během jednoho hodinového cyklu a skládá se ze dvou hlavních VHDL entit: *fir_filter* a *case4x**n***. Entita *fir_filter* reprezentuje hlavní část tohoto modelu a tvoří šablonu, která se nepatrně modifikuje na základě koeficientů určitého filtru.



Obr. 3.2 Plně paralelní model DA FIR filtru

Vstupní vektor signálu (*std_logic_vector*) je uložen do vstupní vyrovnávací paměti, jejíž velikost odpovídá počtu koeficientů daného filtru. Tento vektor je dále konvertován na znaménkový typ (VHDL *signed* typ). V každém hodinovém cyklu je načteno nové slovo a předchozí vektory jsou posunuty. Tento blok v podstatě implementuje posuvný registr. Potom jsou individuální mapování $f(c[n]\times x[n])$ provedena pomocí instancí *case4x**n*** VHDL entity, kde *n* je přímo úměrné délce FIR filtru. Fragment VHDL kódu ilustruje vytvoření *case4x**6*** instance vzhledem ke vstupnímu bufferu (*fir_buffer*).

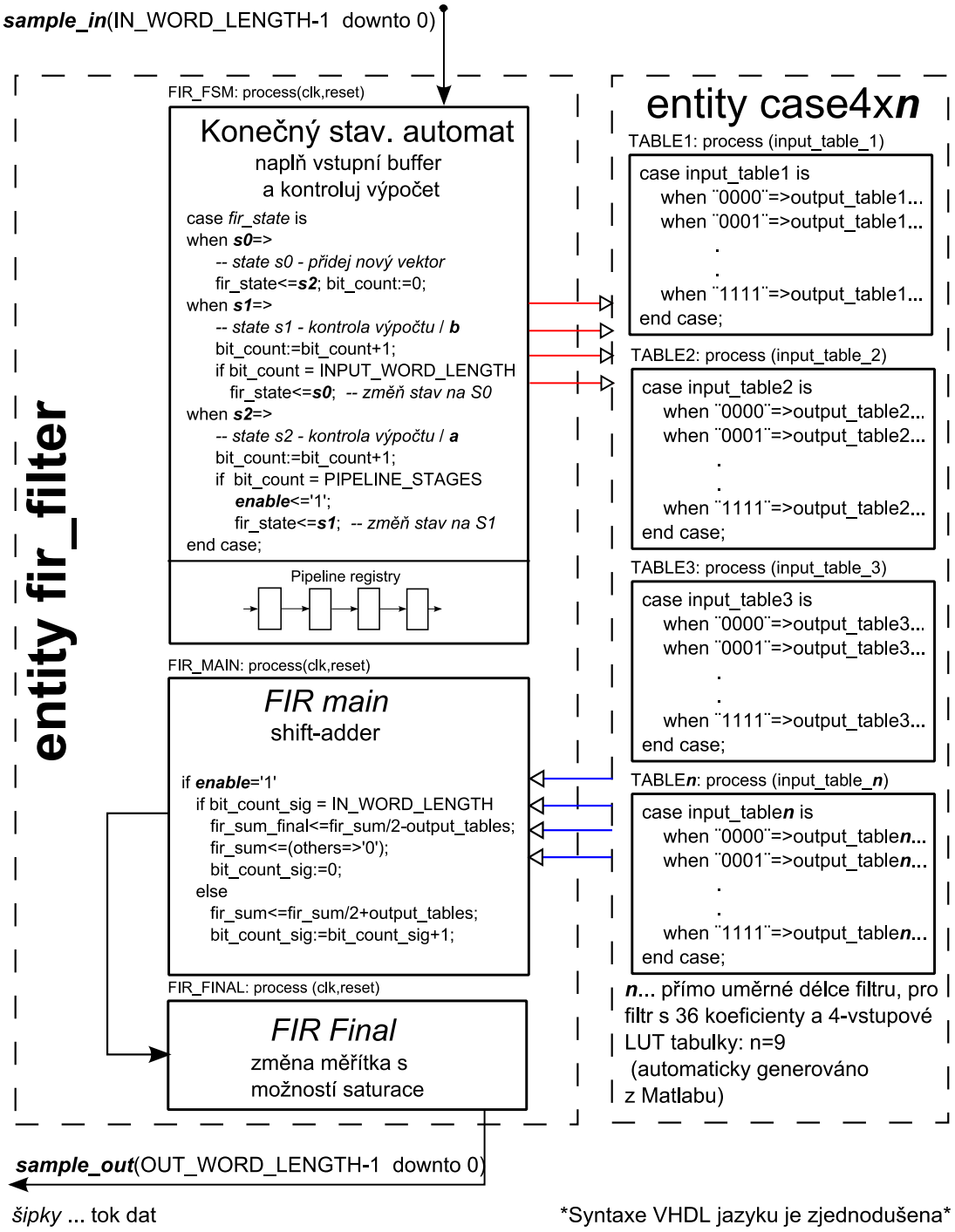
```
182   TABLE_INST0: entity work.case4x6(filter_coeff)
183   port map(input_table1=>fir_buffer(0)(3 downto 0),
184       input_table2=>fir_buffer(0)(7 downto 4),
185       input_table3=>fir_buffer(0)(11 downto 8),
186       input_table4=>fir_buffer(0)(15 downto 12),
187       input_table5=>fir_buffer(0)(19 downto 16),
188       input_table6=>fir_buffer(0)(23 downto 20),
189       output_table1=>output_table0(0),
190       output_table2=>output_table0(1),
191       output_table3=>output_table0(2),
192       output_table4=>output_table0(3),
193       output_table5=>output_table0(4),
194       output_table6=>output_table0(5));
```

Obr. 3.3 Fragment VHDL kódu pro vytvoření *case4x**n*** instance

Pro plně paralelní implementaci potřebujeme vytvořit *k* instancí. To znamená, že pro každý bit vstupního slova je potřeba vytvořit unikátní *case4x**n*** instanci. Tento princip je blíže vysvětlen v pravém dolním rohu obrázku č. 3.2. Entita *case4x**n*** je automaticky generována z programu Matlab. Sekce *Pipeline_0* provádí výpočet prvotního součtu mapování. Tato sekce může být velice rozsáhlá a z tohoto důvodu je také automaticky vygenerována. Příslušný VHDL kód je poté vložen do entity *fir_filter*. V sekci *Pipeline_1* je implementována operace bitového posuvu a přičtení do akumulátoru (shift-adder), integrované násobičky nejsou využity. *Pipeline_2* s *FIR_Final* sekcí tvoří poslední část plně paralelního modelu FIR filtru se zřetězeným zpracováním. Přizpůsobené filtry mohou s výhodou využít této struktury, protože vzorkovaný signál na vstupu přijímače z ADC je obvykle vysoce přezvorkován a rychlosti zpracování signálu přesahující 100MSPS nebo i více jsou nyní běžné.

3.3 Sériový model DA FIR filtru

Navržená struktura sériového DA FIR filtru je zobrazena na obrázku č. 3.4. Sériová struktura poskytuje výslednou hodnotu na svém výstupu po l+1 hodinových cyklech, kde l značí délku vstupního slova.



Obr. 3.4 Sériový model DA FIR filtru

Proces výpočtu je řízen konečným stavovým automatem (FSM) se třemi hlavními stavy označenými jako *s0*, *s1* a *s2*. Stav *s0* je zodpovědný za načtení nového vstupního slova a uložení do posuvného registru. Systém nyní přejde do stavu *s2*. Výpočet je řízen pomocí signálu *enable*, který je nastaven (*enable* = 1) po průchodu vlastní sekcí zřetězeného zpracování. Tento signál je nezbytný pro správnou funkci posuvného registru. Vlastní operace bitového posuvu a přičtení do akumulátoru je implementována pomocí procesu *FIR_MAIN*. Tato sériová struktura FIR filtru stále počítá částečný výstup (pro jeden bit vstupního slova) přes všechny koeficienty najednou, ale je nezbytné spustit výpočet v l+1 hodinových cyklech pro celkový výstup filtru. Tuto strukturu je možné použít tam, kde nám stačí celková rychlost zpracování maximálně 10MSPS - 30MSPS v závislosti na typu FPGA. Výhodou může být, že ušetříme velké množství logických bloků a registrů při srovnání s plně paralelní verzí DA filtru.

3.4 Polyfázová dekompozice

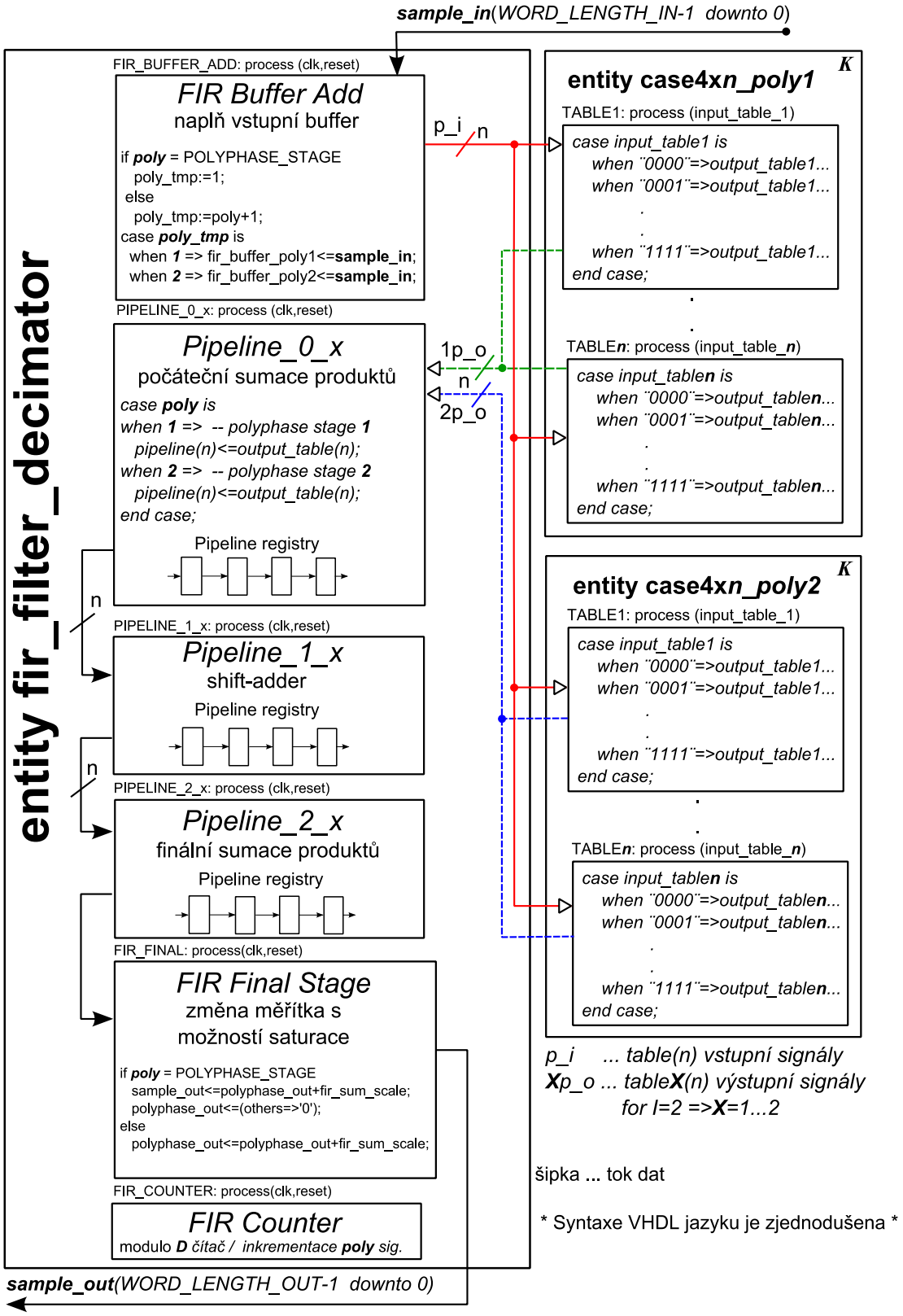
Polyfázová dekompozice je velice užitečná, pokud implementujeme decimační nebo interpolační struktury pomocí FIR filtrů. Pokud budeme chtít například aplikovat decimační faktor R prostřednictvím FIR filtru, zjistíme, že je nutno provést výpočet $y[n]$ pouze v těchto časových okamžicích

$$y[0], y[R], y[2R], \dots . \tag{3.8}$$

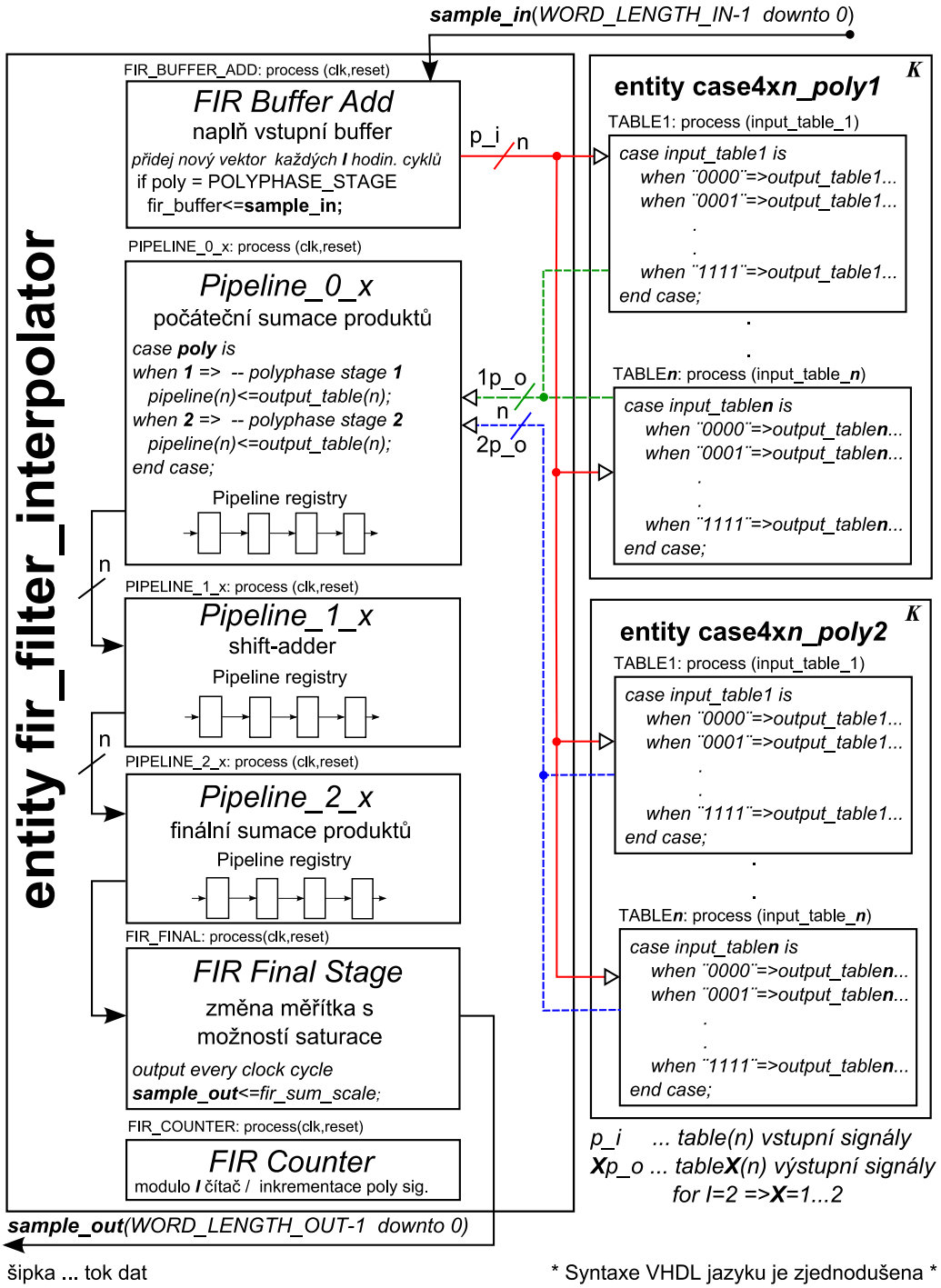
Je proto vhodné rozdělit vstupní signál do R samostatných sekvencí a také rozdělit filtr $f[n]$ (koeficienty filtru) do R sekvencí resp. sub-filtrů nazvaných $f_r[n]$. Tyto filtry mají potom stejnou amplitudou frekvenční charakteristiku, ale jsou od sebe odděleny zpožděním o určitý vzorek signálu, který zavádí žádaný fázový posuv [15].

Polyfázová decimační struktura DA FIR filtru je zobrazena na obrázku č. 3.5. Pro přehlednost je aplikován decimační faktor D = 2. Tato struktura je založena na již popsaném plně paralelním modelu DA FIR filtru. Pro každý sub-filtr $f_r[n]$ musíme vytvořit nezávislou *case4xn_poly_**D*** entitu, kde D je decimační faktor. Ačkoliv celková struktura představuje systém, který provádí změnu vzorkovacího kmitočtu, je v samotném modulu využit pouze jeden hodinový signál. Z tohoto důvodu je přidán čítač, který pomocí výrazu *case-when* volí příspěvky od jednotlivých sub-filtrů. Tento čítač je také zodpovědný za přidání nového vektoru a řízení výstupu filtru, kde je nový vzorek signálu dostupný každých D hodinových cyklů. Polyfázový interpolační filtr je navržen na podobeném

principu, ale celková struktura tohoto filtru je jednodušší. Struktura interpolačního DA filtru je zobrazena na obrázku č. 3.6. Interpolační faktor I=2 je zde opět zvolen pro názornost.



Obr. 3.5 Polyfázový model DA FIR filtru – decimátor



Obr. 3.6 Polyfázový model DA FIR filtru – interpolátor

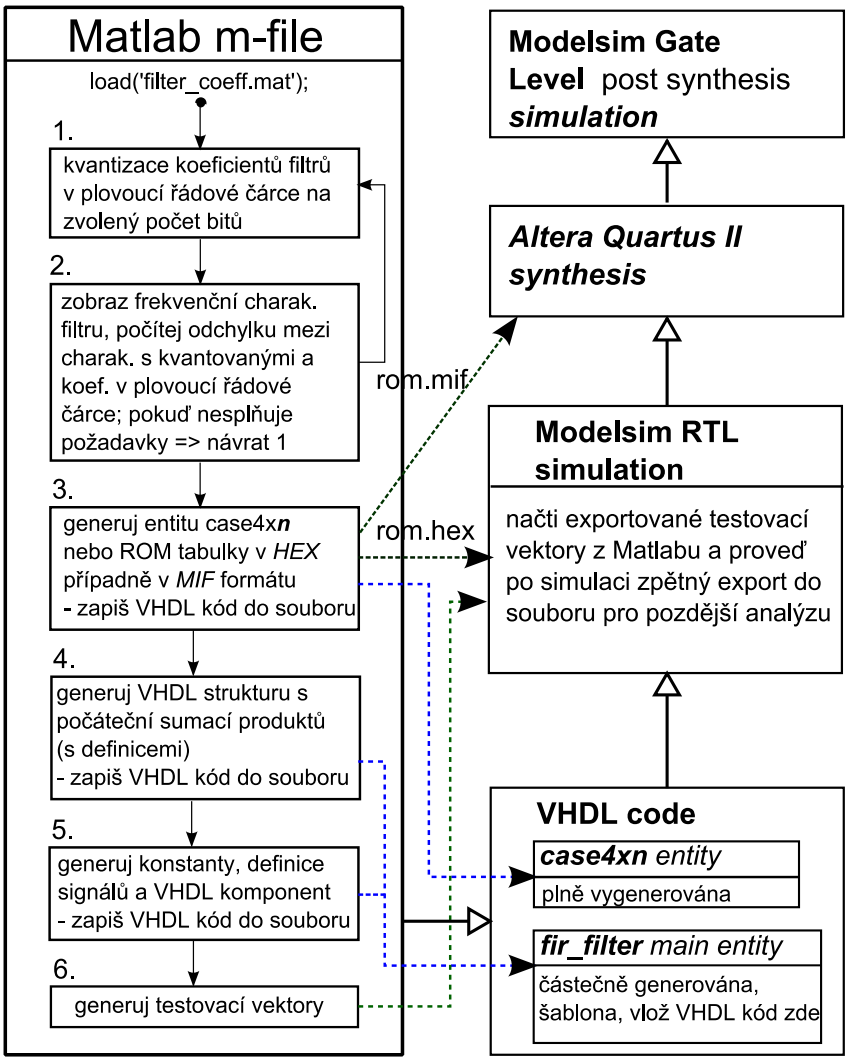
3.5 Modifikace předchozích struktur

Navržené předchozí struktury mohou využívat místo LUT tabulek integrovaných RAM bloků FPGA konfigurovaných jako „single-port“ ROM paměť (např. Altera

Altsyncram Megafunction). Sériová struktura byla také modifikována a umožňuje zpracovávat více bitů (např. 2) najednou paralelně.

3.6 Vývojový cyklus a simulace

Vývojový cyklus začíná již navrženým filtrem s koeficienty v plovoucí řádové čárce v programu Matlab. Navržený skript (*m-file*) provádí funkce, které jsou popsány prostřednictvím vývojového diagramu na obrázku č. 3.7. Simulace byla provedena prostřednictvím programu Matlab s testovacím programem (testbench), který načítá vstupní testovací vektory z textového souboru. Tyto testovací data jsou generována také z programu Matlab. Testbench umožňuje také zpětný export výstupních vektorů do textového souboru, které lze potom analyzovat v prostředí programu Matlab.



Obr. 3.7 VHDL vývojový cyklus DA FIR filtru

3.7 Syntéza FIR filtrů na hradlovém poli FPGA

Ukázková syntéza byla provedena na vývojovém kitu Terasic DE0 FPGA s Altera Cyclone IV (s Cyclone IV EP4CE22F17C6). Výsledky v tabulkách jsou uvedeny pro plně paralelní, sériovou a polyfázové struktury (decimační i interpolační filtr).

Tabulka I. Výsledky syntézy pro plně paralelní strukturu DA FIR filtru

Výsledky syntézy: vstupní data – **12 bitů** koeficienty – **17 bitů**	**Cyclone IV EP4CE22F17C6 (22.320 LEs)** *Vývojové prostředí Altera Quartus 12.0sp2*			
Hilbertův FIR filtr - 36 koeficientů	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
LUT verze f_{MAX}=196.5 MHz	3232 (14%) / 0	2715 (12%)	3183 (14%)	0/132 (0%)
ROM M9K verze f_{MAX}=210.8 MHz	1686 (8%) / 222.264 (37%)	1239 (6%)	1661 (7%)	0/132 (0%)

Tabulka II. Výsledky syntézy pro sériovou strukturu DA FIR filtru

Výsledky syntézy: vstupní data – **12 bitů** koeficienty – **17 bitů**	**Cyclone IV EP4CE22F17C6 (22.320 LEs)** *Vývojové prostředí Altera Quartus 12.0sp2*			
Hilbertův FIR filtr - 36 koeficientů	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
FIR LUT verze f_{MAX}=181.6 MHz	826 (4%) / 0	543 (3%)	658 (3%)	0/132 (0%)
FIR ROM M9K verze f_{MAX}=174.7 MHz	720 (3%) / 22.572 (4%)	484 (2%)	608 (3%)	0/132 (0%)
FIR LUT verze, více bitů najednou paralelně (2 bity) f_{MAX}=135.9 MHz	1405 (6%) /0	1231 (6%)	830 (4%)	0/132 (0%)

Není možné přímo srovnávat výsledky syntézy navržených modelů FIR filtrů s komerčními IP (Intellectual property) kompilátory. Přesná struktura nám není známá a konfigurace může být rozdílná. Proto zde uvádím pouze orientační srovnání s Altera IP FIR (I a II) kompilátory.

Tabulka III. Výsledky syntézy pro polyfázové struktury FIR filtrů

Výsledky syntézy: vstupní data – **12 bitů** koeficienty – **17 bitů**	**Cyclone IV EP4CE22F17C6 (22.320 LEs)** *Vývojové prostředí Altera Quartus 12.0sp2*			
Nyquistův filtr typu dolní propust - 96 koeficientů / 4 sub-filtry /24 koeficientů	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
LUT FIR interpolátor I = 4 f_{MAX}=213.9 MHz	5541 (25%) / 0	5141 (23%)	2800 (13%)	0/132 (0%)
LUT FIR decimátor D = 4 f_{MAX}=152.4 MHz	5861 (26%) / 0	5116 (23%)	3401 (15%)	0/132(0%)

Tabulka IV. Výsledky syntézy – IP Altera FIR I a II kompilátor (srovnání)

Výsledky syntézy: vstupní data – **12 bitů** koeficienty – **17 bitů**	**Cyclone IV EP4CE22F17C6 (22.320 LEs)** *Vývojové prostředí Altera Quartus 12.0sp2*			
Hilbertův FIR filtr - 36 koeficientů	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
LUT plně paralelní filtr * [2] f_{MAX}=250 MHz	3663(16%) / 0	2238 (10%)	3645 (16%)	0/132 (0%)
*LUT sériová filtr** [1] f_{MAX}=99.6MHz	1385 (6%) /0	1027 (5%)	1212 (5%)	0/132 (0%)
LUT FIR interpolátor, 96 koeficientů *I = 4* [2] f_{MAX}=250 MHz	6581 (29%) /0	5040 (23%)	6519 (29%)	0/132 (0%)
LUT FIR decimátor, 96 koeficientů *D = 4* [2] f_{MAX}=182 MHz	7566 (34%) /0	6171 (28%)	5463 (24%)	0/132 (0%)

[1]… *Altera IP FIR kompilátor I* (2015/2016 – zastaralý, ale podporuje sériovou strukturu)
[2]… *Altera IP FIR kompilátor II* (nepodporuje sériovou strukturu, nejsou použity integrované násobičky – nastavení LEs/DSP Block Multiplier Threshold je rovno 600)

Altera Quartus II nastavení syntézy: Fitter level – *standard*; Optimization technique – *balanced*

4 Návrh struktur symbolové synchronizace vhodných pro implementaci na FPGA

V této části práce se zaměřuji na návrh bloků symbolové synchronizace pro softwarově definovaný přijímač implementovaný na hradlovém poli FPGA. V první části kapitoly provádím rozbor detektorů pro určení chyby časování v oblasti symbolové synchronizace a následná část kapitoly se již plně zaměřuje na navržené synchronizační struktury.

4.1 Maximum Likelihood detektor pro určení chyby časování v oblasti symbolové synchronizace

Metoda Maximum Likelihood (ML) [16] neboli metoda maximální věrohodnosti označuje jednu z centrálních metod matematické statistiky. Dál budu uvažovat náhodný výběr $t_1, t_2, \ldots t_n$ z rozdělení s hustotou $f(t, \phi)$, kde ϕ je neznámý parametr. Předpokladem je, že známe tvar rozložení, které závisí na parametru (resp. parametrech) ϕ. Jedná se podmíněné rozložení $f(t, \phi)$, za podmínky, že známe právě parametr ϕ . Pro nezávislá pozorování je společné rozložení násobkem jednotlivých hustot pravděpodobnostních funkcí. Úkolem je nalézt funkci, kterou lze nazvat funkcí věrohodnosti danou

$$L(t_1, t_2, \ldots t_n; \phi) = f(t_1; \phi).f(t_2; \phi) \ldots f(t_n; \phi). \tag{4.1}$$

a z této funkce posléze získat $\hat{\phi}$ tak, aby $\hat{\phi} = (t_1, t_2, \ldots t_n)$ bylo co nejlepším odhadem pro ϕ. Pravá strana rovnice je sdružená hustota pravděpodobnosti *n*-nezávislých proměnných se stejným rozdělením [16]. Když mluvíme o rozložení, považujeme parametr za fixní a pozorování se mění. Jestliže mluvíme o věrohodnosti, pak jsou pozorování fixní a parametr se může měnit. Princip maximální věrohodnosti říká, že máme zvolit jako odhad parametr, který maximalizuje věrohodnost toho, že napozorujeme danou konfiguraci pozorování [16]

$$L(t_1, t_2, \ldots t_n; \hat{\phi}) \geq L(t_1, t_2, \ldots t_n; \phi). \tag{4.2}$$

Protože L je jednoduše funkcí neznámého parametru ϕ, který je odhadován, metoda maximální věrohodnosti je založena na získání takové hodnoty ϕ, která maximalizuje L. Při praktických výpočtech je výhodnější maximalizovat spíše funkci $ln\ L$

namísto *L*, jelikož obě tyto operace jsou ekvivalentní a dávají stejné výsledky (logaritmus je rostoucí funkce). Podmínkou optimality je rovnice

$$\frac{\partial lnL(t_1, t_{2,} \dots\ t_n; \phi)}{\partial \phi} = 0 \tag{4.3}$$

a hodnota z této podmínky získaná je funkcí náhodného výběru $\hat{\phi} = \hat{\phi}(t_1, t_{2,} \dots\ t_n)$.

Normální rozdělení představuje základ pro další odvození. Pro hustotu pravděpodobnosti *f*(*x*) tohoto rozdělení platí

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \tag{4.4}$$

Střední hodnotu *E*(*x*) lze definovat jako

$$E(x) = \int_{-\infty}^{\infty} xf(x)dx = \mu \tag{4.5}$$

a rozptyl σ

$$E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx = \sigma^2. \tag{4.5}$$

Výše popsaná metoda maximální věrohodnosti může být aplikována i do oblasti symbolové synchronizace, kde je možné za neznámý parametr považovat chybu v určení optimální polohy vzorkovacího okamžiku (zkráceně tedy chybu časování symbolů). Tento parametr budu označovat jako *τ*. Následující odvození je ukázáno na přijatém signálu s modulací *QPSK* v základním pásmu. Dále popsané principy je možné aplikovat i pro přenosové systémy s *m-QAM* modulačními schématy. Pokud budu uvažovat přijatý pásmově omezený signál *QPSK* vzorkovaný každých *T* sekund

$$r(nT) = s(nT) + w(nT); n = 0,1, \dots NL_0 - 1 \tag{4.6}$$

kde

$$\begin{aligned} s(nT) = \sum_{k=0}^{L_0} & a_0(k)p(nT - kT_s - \tau)\sqrt{2}\cos(\Omega_0 n) \\ & - a_1(k)p(nT - kT_s - \tau)\sqrt{2}\sin(\Omega_0 n) \end{aligned} \tag{4.7}$$

kde $a_0(k)$ a $a_1(k)$ jsou soufázová *I* a kvadraturní *Q* složky *k*-tého symbolu; $p(nT)$ je jednotková energie pulsu tvarovacího signálu na intervalu $-LT_S \leq T \leq LT_S$; T_s je symbolová perioda; τ je již zmíněný časový ofset, který má být určen a $w(nT)$ je Gausovo bílý šum s výkonovou spektrální hustotou $N_0/2$ W/Hz . Tento vektor $\boldsymbol{w}$ je možné

definovat jako posloupnost nezávisle a identicky distribuované Gaussovo proměnné s nulovou střední hodnotou a směrodatnou odchylkou

$$\sigma^2 = \frac{N_0}{2T}. \tag{4.8}$$

Pro přehlednost lze definovat následující vektory

$$\begin{aligned} \boldsymbol{r} &= [r(0), r(T), \dots, r(NL_0 - 1)T] \\ \boldsymbol{s} &= [s(0), s(T), \dots, s(NL_0 - 1)T] \\ \boldsymbol{w} &= [w(0), w(T), \dots, w(NL_0 - 1)T]. \end{aligned} \tag{4.9}$$

Na základě předchozích definicí lze zapsat funkci hustoty pravděpodobnosti $\boldsymbol{w}$ takto

$$p(\boldsymbol{w}) = \frac{1}{(2\pi\sigma^2)^{L_0 N/2}} exp\left\{ -\frac{1}{2\sigma^2} \sum_{n=0}^{NL_0-1} w^2(nT) \right\}. \tag{4.10}$$

Odhad parametru τ lze provést ze vzorků $r(nT) = s(nT; \boldsymbol{a}, \boldsymbol{\tau}) + w(nT)$, kde výraz *s*(.) je funkcí parametrů **a** a $\boldsymbol{\tau}$. Odhad na základě metody maximální věrohodnosti maximalizuje logaritmus podmíněné pravděpodobnosti $p(\boldsymbol{r}|\boldsymbol{a}, \boldsymbol{\tau})$. S využitím funkce hustoty pravděpodobnosti $\boldsymbol{w}$, lze vyjádřit podmíněnou pravděpodobnost $p(\boldsymbol{r}|\boldsymbol{a}, \boldsymbol{\tau})$ následujícím způsobem

$$p(\boldsymbol{r}|\boldsymbol{a}, \boldsymbol{\tau}) = \frac{1}{(2\pi\sigma^2)^{L_0 N/2}} exp\left\{ -\frac{1}{2\sigma^2} \sum_{n=0}^{NL_0-1} [r(nT) - s(nT; a, \tau)]^2 \right\}. \tag{4.11}$$

Přirozený logaritmus (4.11) je možné označit jako *log-likelihood* funkci $L(\mathbf{a}, \boldsymbol{\tau})$

$$L(\boldsymbol{a}, \boldsymbol{\tau}) = -\frac{L_0 N}{2} ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=0}^{NL_0-1} [r(nT) - s(nT; a, \tau)]^2 . \tag{4.12}$$

$$\{ aplikace \ln(e^x) = x;\ \log_a x^r = r \log_a x \ na\ 5.11 \}$$

Z předchozích vztahů je zřejmé, že je zde podmíněná závislost nejen na parametru $\boldsymbol{\tau}$ a také na přijatých datových symbolech $\boldsymbol{a}$. Pokud jsou symboly známé, odhad parametru $\boldsymbol{\tau}$ je funkcí datových symbolů. Příkladem mohou být paketové rámcové komunikační systémy, kde je známa hlavička paketu [4]. Druhou možností je, že datové symboly nejsou známé. V tomto případě je možné symbolovou datovou sekvenci považovat za náhodnou a funkční závislost na $\boldsymbol{a}$ odstranit. S využitím věty o úplné pravděpodobnosti lze vyjádřit funkci hustoty pravděpodobnosti $p(\boldsymbol{r}|\boldsymbol{a}, \boldsymbol{\tau})$ takto

$$p(r|\tau) = \int p(r|a,\tau)p(a)da, \tag{4.13}$$

kde $p(\boldsymbol{a})$ je funkce hustoty pravděpodobnosti datových symbolů **a.** Předpokládám, že symboly jsou nezávislé a všechny stavy mají stejnou pravděpodobnost. Pro *QPSK* lze předchozí vztah přepsat do následující podoby

$$\begin{aligned} p(r|\tau) = \int p(r|a,\tau)p(a)da &= \prod_{k=0}^{L_0-1} \int p(r|a(k),\tau)p\big(a(k)\big)da(k) \\ &= \prod_{k=0}^{L_0-1} \int \Big\{\frac{1}{4}p(r|a(k)=[1,1],\tau) + \frac{1}{4}p(r|a(k)=[1,-1],\tau) \\ &+ \frac{1}{4}p(r|a(k)=[-1,1],\tau) + \frac{1}{4}p(r|a(k)=[-1,-1],\tau)\Big\}. \end{aligned} \tag{4.14}$$

$\{\ aplikace\ \ P(A \cap B) = P(A).P(B) -$ pro nezávislé jevy A, B $\}$

Přepsáním (4.11) pro jednotlivé symboly z (4.14) platí

$$p(\boldsymbol{r}|\boldsymbol{a},\boldsymbol{\tau}) = \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r(nT) - s(nT;a,\tau)]^2\right\}. \tag{4.15}$$

S využitím substituce z (4.7) lze následně pro jednotlivé symbolové stavy *QPSK* napsat

$$\begin{aligned} p(\boldsymbol{r}|\boldsymbol{a}(\boldsymbol{k}),\boldsymbol{\tau}) = \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\Bigg\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\Big[r(nT) \\ - \big(a_0(k)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n) \\ - a_1(k)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\big)^2\Big]\Bigg\}, \end{aligned} \tag{4.16}$$

$$p(\boldsymbol{r}|\{\mathbf{1},\mathbf{1}\},\boldsymbol{\tau}) = \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) + p^2(nT-kT_s-\tau)]\right\} \times exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\} \times exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\}, \quad (4.17)$$

$\{s\ využitím\ vztahu\ (\cos x)^2 + (\sin x)^2 = 1\}$

$$p(\boldsymbol{r}|\{\mathbf{1},-\mathbf{1}\},\boldsymbol{\tau}) = \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) + p^2(nT-kT_s-\tau)]\right\} \times exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\} \times exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\}, \quad (4.18)$$

$\{s\ využitím\ vztahu\ (\cos x)^2 + (\sin x)^2 = 1\}$

$$
\begin{aligned}
&p(\boldsymbol{r}|\{-\mathbf{1},\mathbf{1}\},\boldsymbol{\tau}) \\
&\quad= \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) \right. \\
&\quad\left. + p^2(nT-kT_s-\tau)]\right\} \\
&\quad\times exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\} \\
&\quad\times exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\}, \\
&\quad\{s\ využitím\ vztahu\ (\cos x)^2 + (\sin x)^2 = 1\}
\end{aligned}
\tag{4.19}
$$

$$
\begin{aligned}
&p(\boldsymbol{r}|\{-\mathbf{1},-\mathbf{1}\},\boldsymbol{\tau}) \\
&\quad= \prod_{k=0}^{L_0-1} \frac{1}{(2\pi\sigma^2)^{L_0N/2}} exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) \right. \\
&\quad\left. + p^2(nT-kT_s-\tau)]\right\} \\
&\quad\times exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\} \\
&\quad\times exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\}, \\
&\quad\{s\ využitím\ vztahu\ (\cos x)^2 + (\sin x)^2 = 1\}
\end{aligned}
\tag{4.20}
$$

Nyní již lze dosadit odvozené výrazy (4.17-4.20) do (4.14)

$$p(\boldsymbol{r}|\boldsymbol{\tau}) = \frac{1}{4}\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{L_0N/2}}exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) + p^2(nT-kT_s-\tau)]\right\} \times\left(exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\} + exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right\}\right) \times\left(exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\} + exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right\}\right). \quad (4.21)$$

$$\{sloučení - xy + xv + yz{+}yv = (x{+}y)(z{+}v)\}$$

a dále je možné vztah upravit do podoby

$$p(\boldsymbol{r}|\boldsymbol{\tau}) = \prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{L_0N/2}}exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}[r^2(nT) + p^2(nT-kT_s-\tau)]\right\} \times\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right]\right) \times\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\right). \quad (4.22)$$

$$\left\{aplikace\ vztahu\ \cosh x = \frac{e^x + e^{-x}}{2}\right\}$$

Pro náhodné symboly lze výslednou *log-likelihood* funkci $L(\boldsymbol{\tau})$ vyjádřit takto

$$L(\boldsymbol{\tau}) = -\frac{L_0 N}{2} ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=0}^{L_0-1} \sum_{n=N(k-L_p)}^{N(k+L_p)} [r^2(nT) + p^2(nT - kT_s - \tau)]$$
$$\times \ln\cosh\left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\cos(\Omega_0 n) \right] \right)$$
$$\times \ln\cosh\left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\sin(\Omega_0 n) \right] \right). \quad (4.23)$$

$$\{ aplikace \ln(e^x) = x;\ \log_a x^r = r \log_a x \ na\ 5.22 \}$$

Dále je nutné rozlišit, zdali synchronizační systém pracuje se známou posloupností přijatých symbolů nebo naopak.

4.1.1 ML - neznámá symbolová posloupnost

Pokud bude přijatá symbolová posloupnost neznámá (tj. nezávislost na datových symbolech), odhad na základě metod maximální věrohodnosti je taková hodnota τ, která maximalizuje *log-likelihood* funkci $L(\boldsymbol{\tau})$ danou vztahem (4.23). Parciální derivace (složené) funkce (4.23)

$$\frac{\partial}{\partial\tau} L(\boldsymbol{\tau}) = \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\cos(\Omega_0 n) \right] \right)$$
$$\times \frac{\partial}{\partial\tau} \left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\cos(\Omega_0 n) \right] \right)$$
$$+ \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\sin(\Omega_0 n) \right] \right)$$
$$\times \frac{\partial}{\partial\tau} \left(\frac{1}{\sigma^2} \sum_{n=N(k-L_p)}^{N(k+L_p)} \left[r(nT) p(nT - kT_s - \tau)\sqrt{2}\sin(\Omega_0 n) \right] \right). \quad (4.24)$$

Poznámka 1. *h(x)=h(f(x)g(x)) ... využití pravidla pro derivaci složené funkce*

$$h'(x) = \frac{\partial h}{\partial x} = \frac{\partial h}{\partial f}\frac{\partial f}{\partial x} + \frac{\partial h}{\partial g}\frac{\partial g}{\partial x}$$

Poznámka 2. *Derivace ln(cosh(x)) ... využití řetízkového pravidla*

$$\frac{d}{dx} ln\big(cosh(x)\big)dx = \frac{d}{du} ln(u)\frac{du}{dx} = \frac{1}{u}\sinh(x) = \frac{\sinh(x)}{\cosh(x)} = \tanh(x)$$

$$u = \cosh(x);\ \frac{du}{dx} = \cosh(x)$$

4.1.2 ML - známá symbolová posloupnost

Pokud bude přijatá symbolová posloupnost známa (předpokládá se závislost na datových symbolech; je k dispozici například opakující se záhlaví paketu), odhad na základě metod maximální věrohodnosti je taková hodnota τ, která maximalizuje *log-likelihood* funkce $L(\mathbf{a},\boldsymbol{\tau})$ danou vztahem (4.12). Parciální derivaci funkce (4.12) lze vyjádřit následujícím způsobem

$$\begin{aligned}\frac{\partial}{\partial\tau}L(\boldsymbol{a},\boldsymbol{\tau}) &= -\frac{1}{2\sigma^2}\frac{\partial}{\partial\tau}\sum_{n=0}^{NL_0-1}[r(nT)-s(nT;a,\tau)]^2 \\ &= -\frac{1}{2\sigma^2}\frac{\partial}{\partial\tau}\sum_{n=0}^{NL_0-1}[r^2(nT)-2r(nT)s(nT;a,\tau)+s^2(nT;a,\tau)]\end{aligned} \tag{4.25}$$

Parciální derivace prvního výrazu v hranaté závorce je nulová, protože energie přijatého signálu nezávisí na ofsetu τ. Dále parciální derivace třetího výrazu bude přibližně nulová, protože je zde velice malá závislost na τ. Dále tedy budu pracovat pouze s druhým členem v hranaté závorce. Po substituci (4.7) do (4.25) lze parciální derivaci $L(\mathbf{a}, \boldsymbol{\tau})$ vyjádřit jako

$$\begin{aligned}\frac{\partial}{\partial\tau}L(\boldsymbol{a},\boldsymbol{\tau}) = \frac{1}{\sigma^2}\frac{\partial}{\partial\tau}\sum_{n=0}^{L_0-1}a_0(k)\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right] \\ -\frac{1}{\sigma^2}\frac{\partial}{\partial\tau}\sum_{n=0}^{L_0-1}a_1(k)\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s\right. \\ \left.-\tau)\sqrt{2}\sin(\Omega_0 n)\right]\end{aligned} \tag{4.26}$$

4.2 Analýza detektorů a problematika jejich realizace

V této části se nejdříve zaměřuji na možnou realizaci detektorů s využitím metod maximální věrohodnosti (ML). Na získané poznatky navazuji při analýze dalších odvozených detektorů (např. detektor průchodu nulou – *Zero crossing detector* [6]).

4.2.1 Analýza realizace ML - neznámá symbolová posloupnost

Výstupní signál QPSK z přizpůsobeného filtru lze pro soufázovou a kvadraturní složku vyjádřit následujícím způsobem

$$x(kT_s+\tau)=\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\cos(\Omega_0 n)\right], \tag{4.27}$$

$$y(kT_s+\tau)=\sum_{n=N(k-L_p)}^{N(k+L_p)}\left[r(nT)p(nT-kT_s-\tau)\sqrt{2}\sin(\Omega_0 n)\right]. \tag{4.28}$$

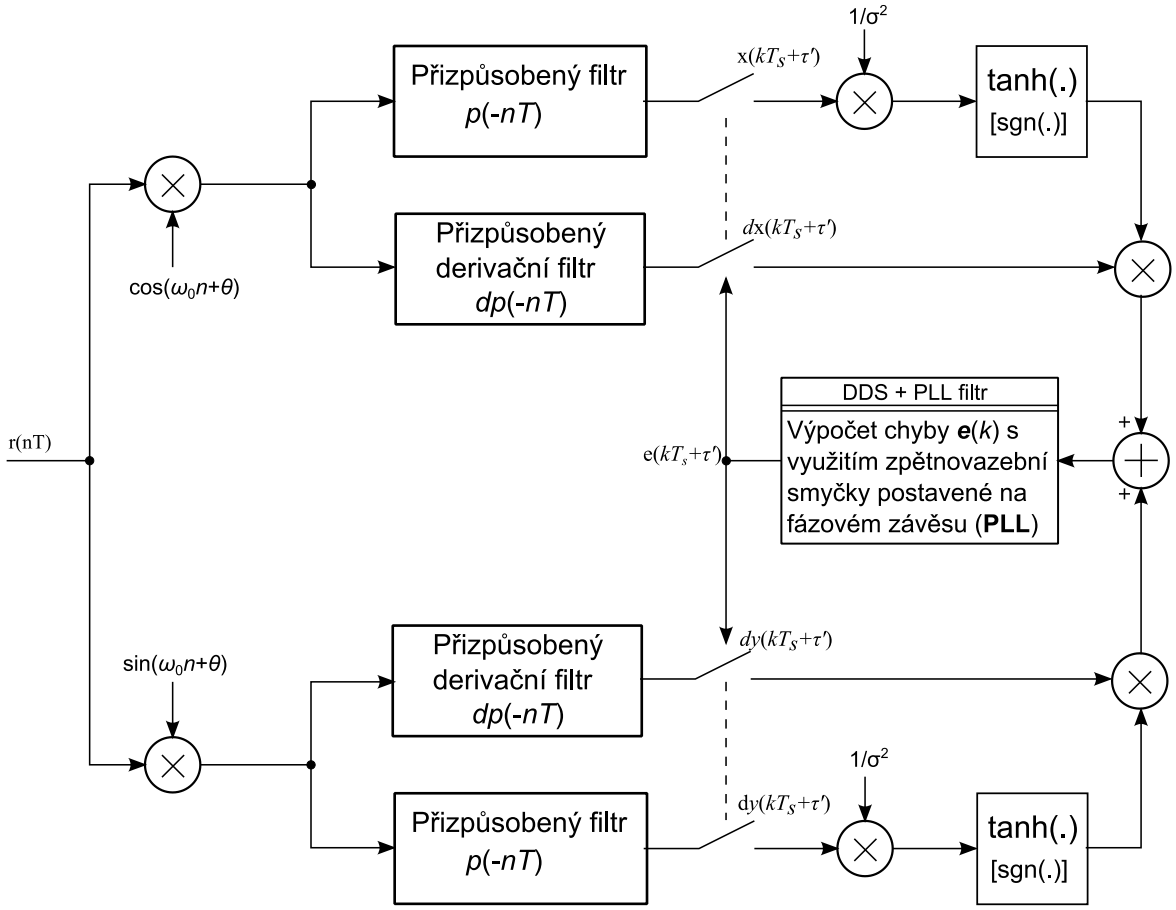
Výraz (4.24) lze za pomoci (4.27 a 4.28) přepsat do zjednodušeného tvaru

$$\begin{aligned}\frac{\partial}{\partial\tau}L(\boldsymbol{\tau})=&\sum_{k=0}^{L_0-1}\tanh\left(\frac{1}{\sigma^2}x(kT_s+\tau)\right)\times\frac{\partial}{\partial\tau}\left(\frac{1}{\sigma^2}x(kT_s+\tau)\right)\\&+\sum_{k=0}^{L_0-1}\tanh\left(\frac{1}{\sigma^2}y(kT_s+\tau)\right)\times\frac{\partial}{\partial\tau}\left(\frac{1}{\sigma^2}y(kT_s+\tau)\right).\end{aligned} \tag{4.29}$$

Jestliže nahradíme parciální derivace časovými derivacemi, musí odhad $\hat{\tau}$ na základě metod maximální věrohodnosti splňovat vztah (tj. derivace musí být rovna nule)

$$\begin{aligned}0=&\sum_{k=0}^{L_0-1}\tanh\left(\frac{1}{\sigma^2}x(kT_s+\hat{\tau})\right)\times\left(\frac{1}{\sigma^2}\dot{x}(kT_s+\hat{\tau})\right)\\&+\sum_{k=0}^{L_0-1}\tanh\left(\frac{1}{\sigma^2}y(kT_s+\hat{\tau})\right)\times\left(\frac{1}{\sigma^2}\dot{y}(kT_s+\hat{\tau})\right).\end{aligned} \tag{4.30}$$

Odhad parametru $\hat{\tau}$ je nutno provést iterativně, neexistuje přímé řešení. Lze využít zpětnovazební smyčky na principu fázového závěsu, jak je schematický uvedeno na obrázku č. 4.1. Získaný vztah (4.29) lze již relativně jednoduše implementovat prostřednictvím běžného procesoru pracujícího s plovoucí řádovou čárkou, ale přímá implementace v hardwaru (resp. na hradlovém poli FPGA) může být velice neefektivní. Jedná se především o funkci *tanh*(x), kde pro výpočet určitého argumentu této funkce je nutno využít například algoritmu CORDIC (Coordinate Rotation Digital Computer) [17].



Obr. 4.1 Blokové schéma synchronizačního systému s ML – neznámá symbolová posloupnost

Využití zřetězeného zpracování nemusí sice ovlivnit maximální dosaženou pracovní frekvenci systému, ale dojde ke značnému nárůstu spotřeby logických prvků. Z tohoto důvodu je vhodné tuto funkci aproximovat. Z průběhu funkce *tanh*(x) je zřejmé, že pro velké hodnoty argumentu (tj. pro větší odstup SNR) lze tuto funkci aproximovat pomocí znaménkové funkce *sgn*(x). Tuto funkci lze jednoduše implementovat například za pomoci digitálního komparátoru. Lze tedy přepsat vztah (4.30) s využitím funkce *sgn*(x) pro signál typu *QPSK*

$$e(k) \approx sgn[x(kT_s + \tau)]\dot{x}(kT_s + \tau) + sgn[y(kT_s + \tau)]\dot{y}(kT_s + \tau). \tag{4.31}$$

V případě prostého signálu s PAM (resp. BPSK) se výraz zjednoduší na (využije se pouze první část výrazu)

$$e(k) \approx sgn[x(kT_s + \tau)]\dot{x}(kT_s + \tau). \tag{4.32}$$

4.2.2 Analýza realizace ML - známá symbolová posloupnost

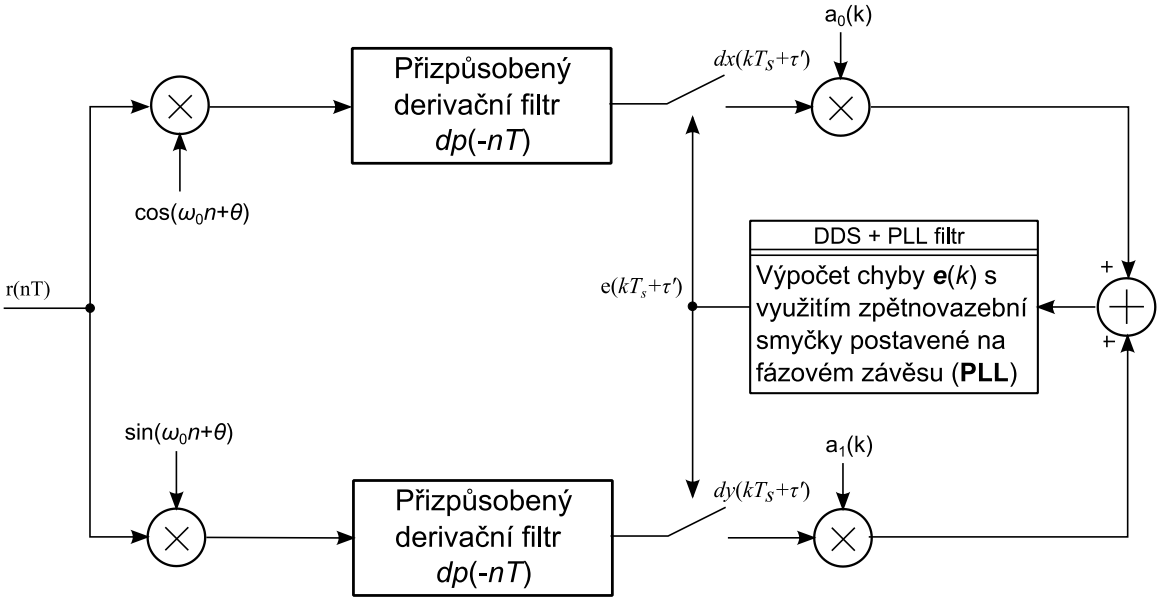
Výstupní signál z přizpůsobeného filtru lze opět popsat za pomoci vztahů (4.27) a (4.28). Parciální derivaci *log-likelihood* funkce $L(\mathbf{a},\boldsymbol{\tau})$ z výrazu (4.12) lze poté přepsat do zjednodušeného tvaru

$$\begin{aligned}\frac{\partial}{\partial\tau}L(\boldsymbol{a},\boldsymbol{\tau}) &= \frac{1}{\sigma^2}\frac{\partial}{\partial\tau}\sum_{n=0}^{L_0-1} a_0(k)\,x(kT_s+\tau)+a_1(k)y(kT_s+\tau)\\ &= \frac{1}{\sigma^2}\sum_{n=0}^{L_0-1} a_1(k)\,\dot{x}(kT_s+\tau)+a_1(k)\dot{y}(kT_s+\tau),\end{aligned} \tag{4.33}$$

kde $\dot{x}(kT_s+\tau)$ a $\dot{y}(kT_s+\tau)$ jsou vzorky časové derivace z výstupu přizpůsobeného filtru (soufázová a kvadraturní složka). Je tedy nutno za přizpůsobený filtr zařadit derivační filtr (platí samozřejmě i pro neznámou posloupnost symbolů). V aplikacích citlivých na časové zpoždění, je nutné, aby tyto dva filtry pracovali paralelně. Odhad $\hat{\tau}$ na základě metod maximální věrohodnosti musí opět splňovat vztah (tj. derivace musí být rovna nule)

$$0 = \sum_{n=0}^{L_0-1} a_1(k)\,\dot{x}(kT_s+\tau)+a_1(k)\dot{y}(kT_s+\tau), \tag{4.34}$$

Odhad parametru $\hat{\tau}$ je nutno provést opět iterativně, obrázek 4.2 ukazuje možnou implementaci synchronizačního systému na bázi PLL.



Obr. 4.2 Blokové schéma synchronizačního systému s ML – známá symbolová posloupnost

4.2.3 Detektor průchodu nulou a Gardnerův detektor

Detektor průchodu nulou dále označený jako ZCTED (*Zero-crossing Timing Error Detector*) je založený na detekci průchodu signálu nulou v diagramu oka. Pracuje se dvěma vzorky na symbol a poskytuje nulovou chybovou hodnotu v případě, že každý druhý vzorek signálu je časově zarovnán vůči nulové hodnotě na výstupu přizpůsobeného filtru. Dále budu předpokládat, že přizpůsobený filtr poskytuje na svém výstupu 2 vzorky/symbol a posloupnost těchto vzorků je číslována pomocí indexu *k*.

$$\begin{aligned} \dots, x\big((k-1)T_s-\tau\big), x\big((k-1/2)T_s-\tau\big), x(kT_s-\tau), \\ x\big((k+1/2)T_s-\tau\big), x\big((k+1)T_s\big), \dots \end{aligned} \tag{4.35}$$

Chybu časování symbolů *e*(*k*) pro ZCTED lze potom vyjádřit jako

$$e_{ZCTED}(k) = x\big((k-1/2)T_s+\hat{\tau}\big)[a(k-1)-a(k)] \tag{4.36}$$

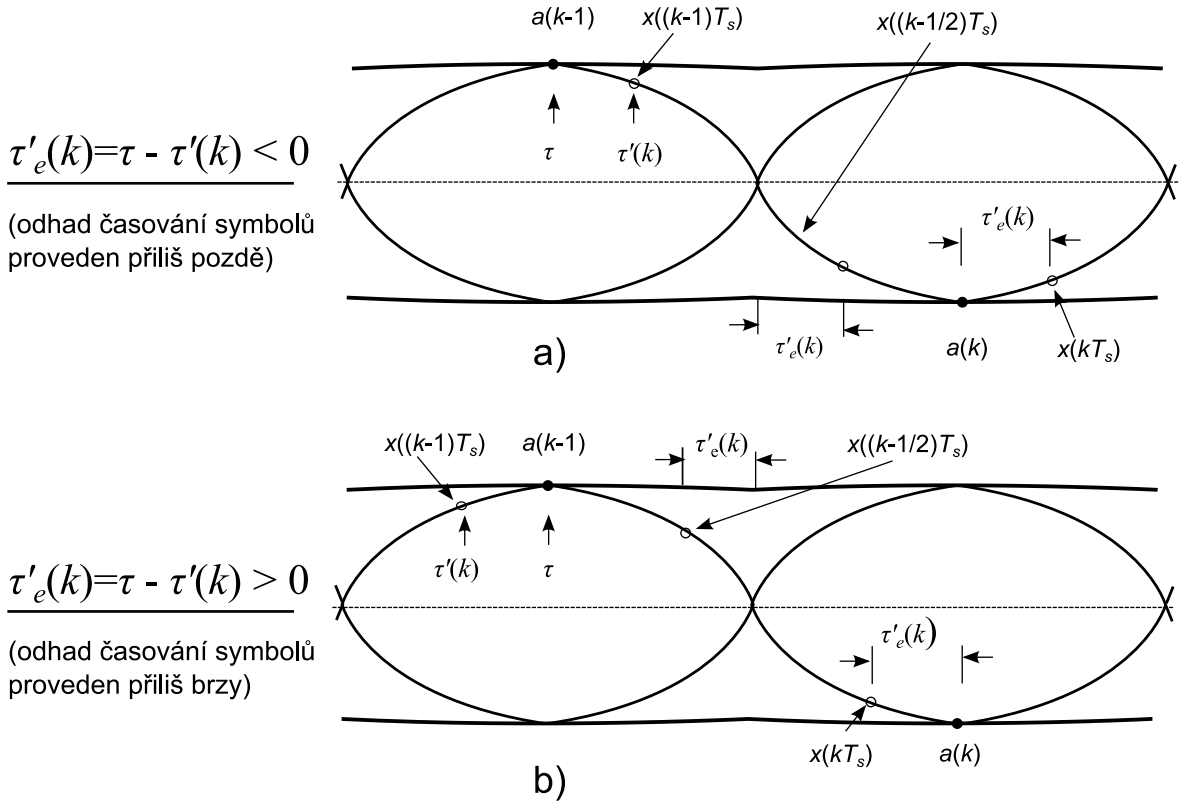
kde T_s je symbolová perioda; τ je neznámý časový ofset a $\hat{\tau}$ je odhad časového zpoždění pro ZCTED detektor. V případě DD detektoru platí pro jednotlivé symboly *a*

$$\begin{aligned} a(k-1) &= sgn\big[x\big((k-1)T_s+\hat{\tau}\big)\big] \\ a(k) &= sgn[x(kT_s+\hat{\tau})]. \end{aligned} \tag{4.37}$$

Speciální modifikací ZCTED detektoru je detektor nazvaný GTED (*Gardner Timing Error Detector*) [18]. Tento výhradně *NDA* detektor je nezávislý na jakékoliv rotaci fáze nosné vlny a je možné jej zařadit před blok synchronizace fáze nosné vlny. Detektor GTED pracuje také se dvěma vzorky na symbol a chyba časování *e*(*k*) je

$$e_{GTED}(k) = x\big((k-1/2)T_s+\hat{\tau}\big)\big[x\big((k-1)T_s+\hat{\tau}\big)-x(kT_s+\hat{\tau})\big]. \tag{4.38}$$

Princip tohoto detektoru je ukázán na obrázku č. 4.3 pro dva případy. V případě *a)* byl odhad časování symbolů proveden příliš pozdě a v případě *b)* naopak příliš brzy.



Obr. 4.3 Princip funkce detektoru průchodu nulou (ZCTED)

4.2.4 S-křivka a filtr fázového závěsu

Parametry chybového signálu *e(k)* jsou dále modifikovány pomocí filtru fázového závěsu (*PLL loop filter*). Tento filtr je z důvodu minimalizace časového zpoždění nutno implementovat jako filtr s nekonečnou impulsní odezvou (IIR). S výhodou lze tedy využít proporcionálně integračního filtru *IIR*. Proporcionální konstanta K_p (zisk detektoru) musí být vypočtena pro zajištění správné funkce použitého detektoru. Tato konstanta může být určena prostřednictvím tzv. *S-křivky* (v angličtiny obecně označována jako *S-curve*) [6]. Tu lze pro určitý detektor definovat jako hodnotu *e(k)*, která je podmíněna hodnotou chyby časování symbolu. Nejdříve se zaměřím na detektor *ZCTED* (*GTED*) a poté na detektor *MLTED*.

S-křivku pro detektor *ZCTED* lze definovat jako odhad strmosti $r_a(-\tau_e)$ s využitím hodnot $r_a(t)$ jednu poloviny symbolové periody před a za $-\tau_e$, kde r_a je autokorelační funkce tvarovacího pulsu a τ_e je chyba vyjádřená jako

$$\tau_e = \tau - \hat{\tau}. \tag{4.39}$$

Přizpůsobený filtr typu zvednutý kosinus odmocněný druhou mocninou (*SRRC*-square-root raised cosine) s 50% rozšířenou šířkou pásma je použit v následných

simulacích. Filtrace signálu za pomoci „zvednutého“ kosinového filtru je rozdělena mezi vysílač a přijímač. Je využito stejného *SRRC* filtru na obou stranách přenosového řetězce, což zajišťuje minimalizaci mezisymbolových interferencí [19]. Experimentálně jsem data pro grafické znázornění *S-křivky* získal otevřením rekurzivní smyčky (fázového závěsu) a měřil jsem průměrnou hodnotu chybového signálu *e(k)*. *S-křivku*, označenou jako $S(\tau_e)$, lze pro detektor *ZCTED* formulovat tímto způsobem

$$S(\tau_e) = KE_{avg}\left[r_a\left(\frac{T_s}{2} - \tau_e\right) - r_a\left(-\frac{T_s}{2} - \tau_e\right)\right], \tag{4.40}$$

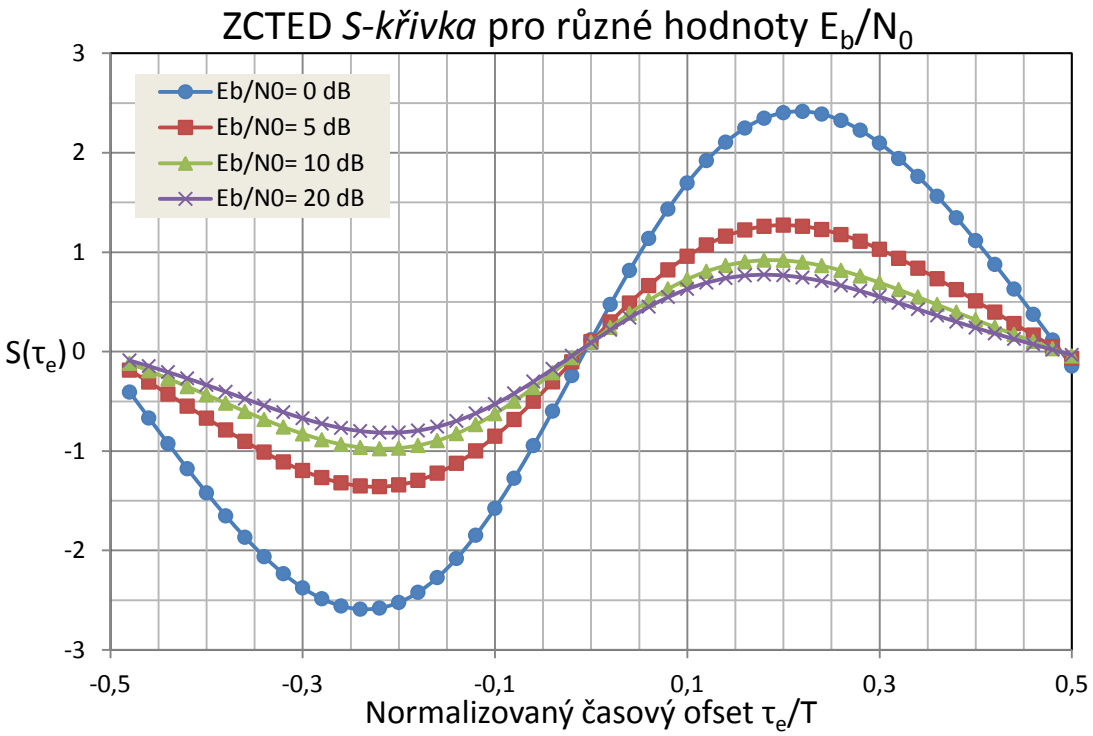
kde K je amplituda signálu a E_{avg} je průměrná energie symbolu. Protože se jedná o autokorelační funkci, $r_a(t)$ bude symetrická okolo t=0. Z toho plyne, že pro τ_e=0 bude výraz

$$\left[r_a\left(\frac{T_s}{2} - \tau_e\right) - r_a\left(-\frac{T_s}{2} - \tau_e\right)\right] = 0 \tag{4.41}$$

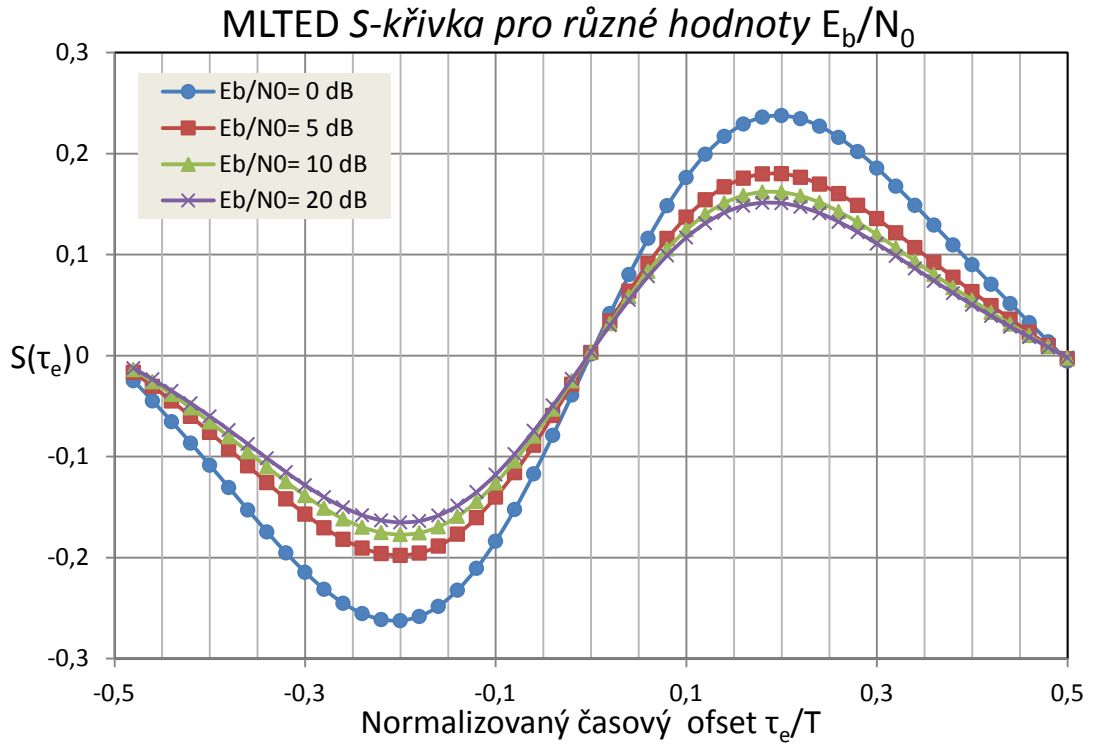
a následně i *S-křivka* v bodě τ_e=0 rovna nule. Parametr K_p (záměrně zatím neuvádím, že se jedná o konstantu) je potom směrnice tečny $S(\tau_e)$ v bodě τ_e=0. Pro detektor *MLTED* bude *S-křivka* formulována následovně

$$S(\tau_e) = E\left[a(k)\frac{d}{dt}x(kT_s + \tau)\right] = KE_{avg}\dot{r}_a(-\tau_e), \tag{4.42}$$

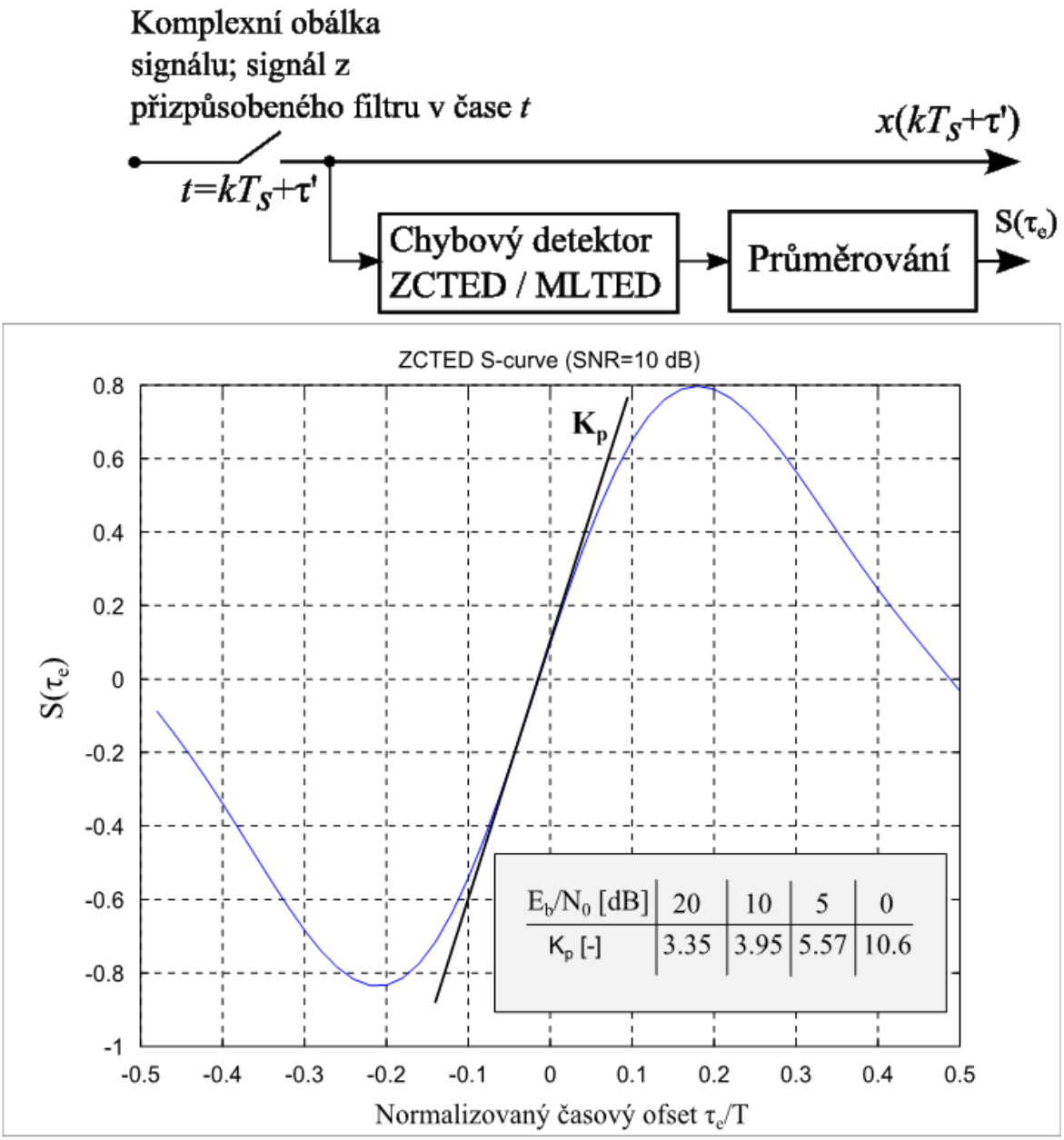
kde $\dot{r}_a(-\tau_e)$ je časová derivace autokorelační funkce tvarovacího pulsu vyhodnocená pro -τ_e. Výpočtovou simula*ci S-křivky* pro různé poměry signál-šum a signál s modulací *QPSK* jsem provedl v prostředí programu Matlab. *S-křivka* pro detektor ZCTED je zobrazena na obrázku č. 4.4 a pro MLTED na obrázku č. 4.5. Obrázek č. 4.6 popisuje experimentální měření S-křivky a uvádí v tabulce příklady získaných hodnot parametrů K_p.



Obr. 4.4 Simulace *S-křivky* pro detektor ZCTED, použit *SRRC* filtr s koeficientem 0.5 a normovanou jednotkovou energií



Obr. 4.5 Simulace *S-křivky* pro detektor MLTED, použit *SRRC* filtr s koeficientem 0.5 a normovanou jednotkovou energií



Obr. 4.6 Experimentální měření S-křivky, princip platný pro ZCTED i MLTED, údaje v tabulce v dolní části grafu jsou platné pro detektor ZCTED

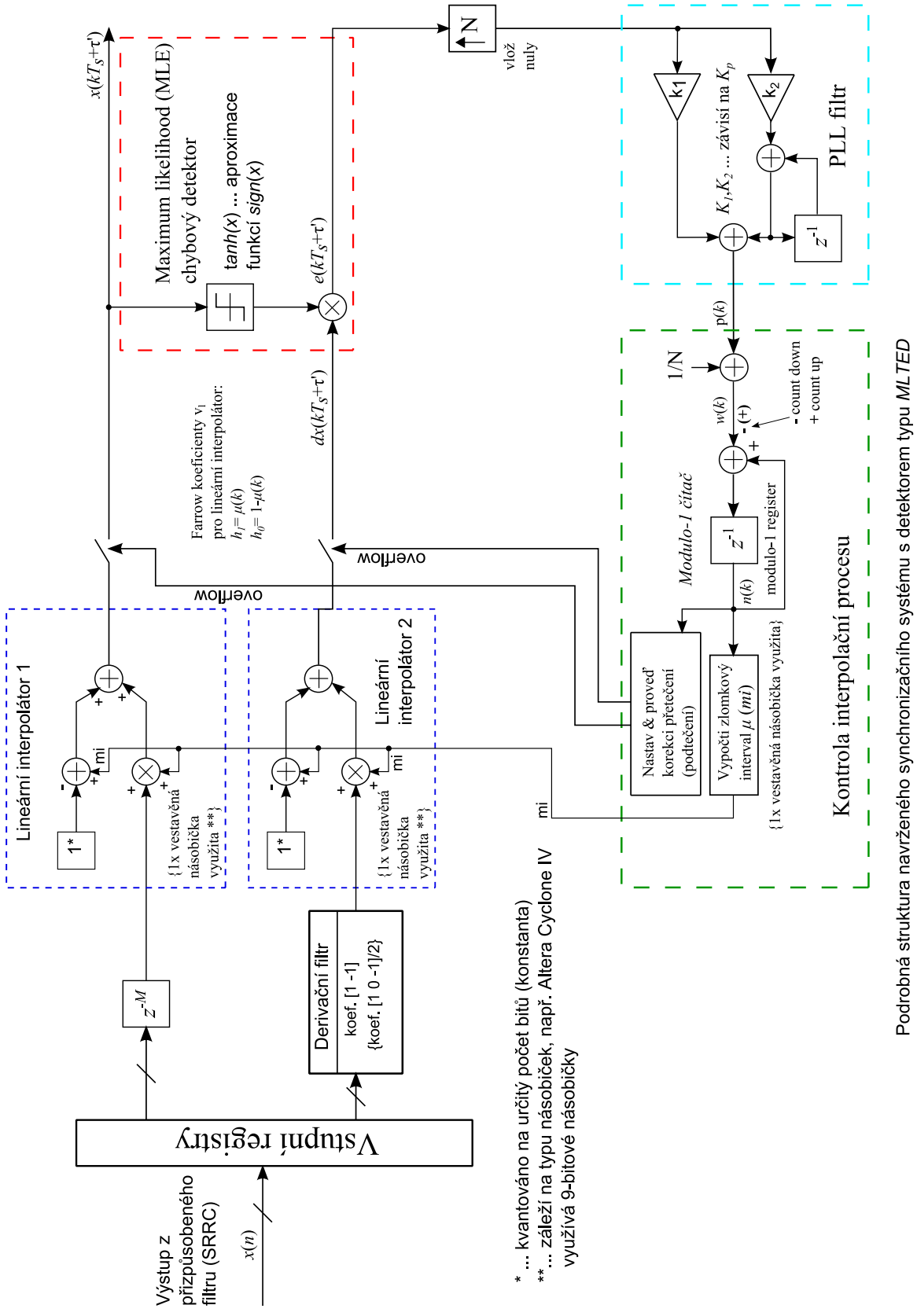
Z důvodu, že parametr K_p závisí na úrovni přijatého signálu, je nutno tento parametr v reálném dynamickém prostředí vždy přepočítat. Tento princip je možné použít při simulacích synchronizačního systému, ale pro efektivní implementaci je nutno s K_p pracovat jako s konstantou. Blok symbolové synchronizaci bude potom potřebovat pro svoji správnou funkci systém automatického řízení zisku (*AGC* – Automatic Gain Control) [4]. Tento blok lze implementovat v analogové i digitální podobě. Blok *AGC* musí být vždy zařazen před symbolovou synchronizací.

4.3 Modely symbolové synchronizace pro implementaci na hradlovém poli FPGA

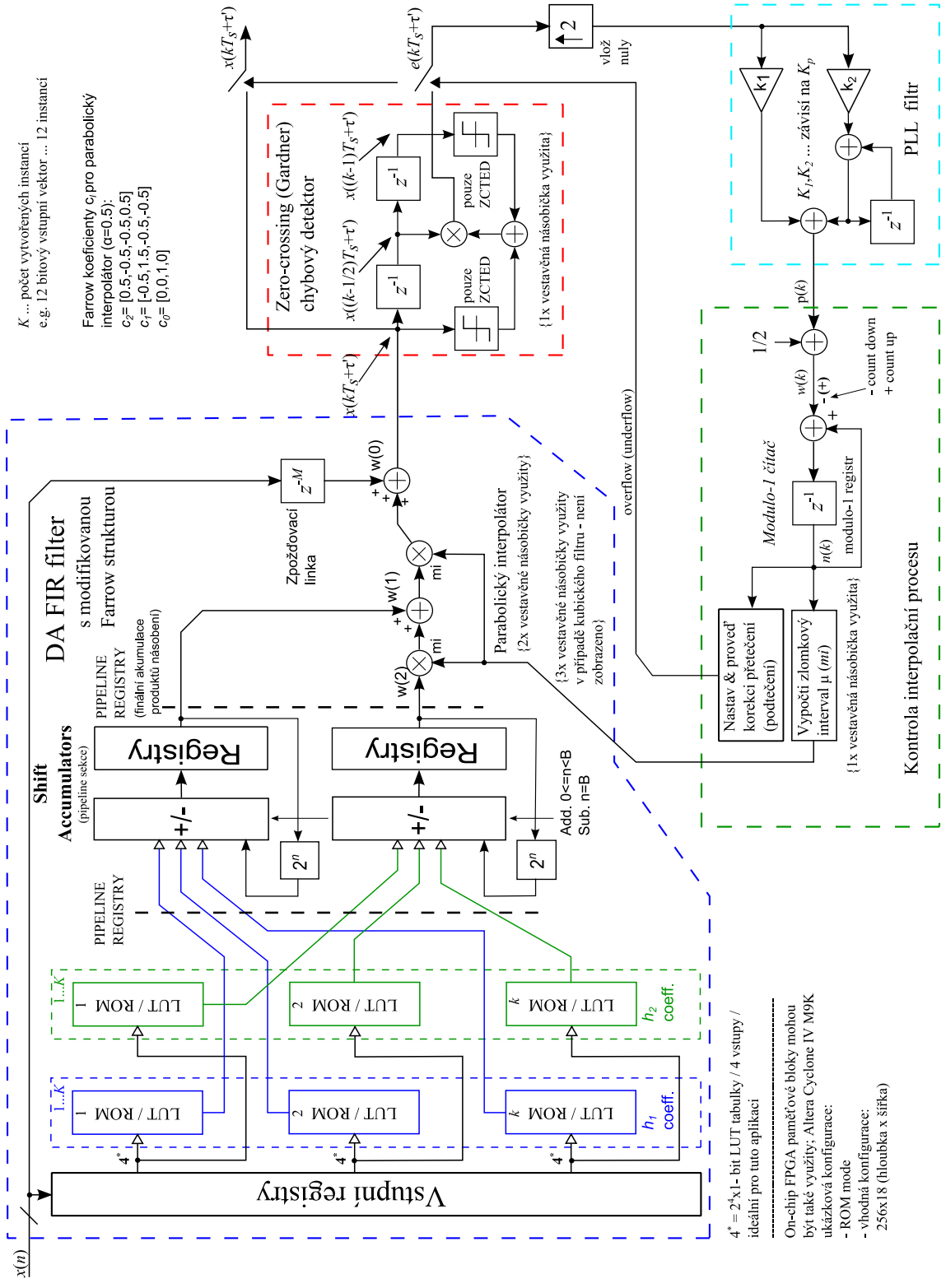
V předchozích kapitolách jsem provedl rozbor chybových detektorů určení časové polohy symbolů a nyní je možné na základě uvedených předpokladů vytvořit modely symbolové synchronizace pro efektivní implementaci na hradlovém poli FPGA případně ASIC. Modely jsou postaveny na detektorech MLTED a ZCTED (GTED) a využívají zpětnovazební smyčky postavení na fázovém závěsu. V případě detektoru MLTED je z důvodu větší obecnosti vytvořen model pro případ neznámé symbolové posloupnosti. Model s detektorem průchodu nulou ZCTED je možné modifikovat na čistě NDA synchronizační systém s detektorem GTED. Předpokladem pro navržené modely je, že přijatý signál x(t) je vzorkován s fixní rychlostí $1/T$, která je asynchronní vůči symbolové rychlosti $1/T_s$. Časové zpoždění τ musí být určeno ze vzorků $x(nT)$. Jedná se o asynchronní vzorky z výstupu přizpůsobeného filtru. Jak již bylo uvedeno v kapitole 2.4, vzorky nebudou zarovnány na symbolové mezi a musí být „přemístěny“ na správnou časovou polohu. Signál je nutné převzorkovat ve zlomkovém poměru a to je úkol právě pro interpolační filtr, který musí být zařazen do synchronizačního systému. Modely se skládají z následujících částí:

1. *Interpolační filtr* – provádí změnu vzorkovací kmitočtu ve zlomkovém poměru.
2. *Chybový detektor určení časové polohy symbolů* – MLTED, ZCTED resp. GTED.
3. *PLL filtr* (*PLL loop filter*) – proporcionálně integrační filtr typu *IIR*.
4. *Blok kontroly interpolačního procesu*

Navržené modely jsou zobrazeny na obrázku č. 4.7 pro detektor MLTED a pro ZCTED (GTED) na obrázku č. 4.8. Schémata modelů jsou uvedeny z důvodu přehlednosti pouze pro jednu větev I/Q signálu. V uvedené podobě je možné modely například použít pro signály typu *PAM* nebo *BPSK*. V případě vícestavových modulací typu *QPSK* resp. *m-QAM* je možné při vlastní realizaci na FPGA vytvořit dvě instance některých částí modelu, které potom budou společně provádět výpočet chyby $e(k)$. Uvedené rozšíření je popsáno v kapitole č. 7. V aktuální kapitole se zaměřuji na efektivní návrh interpolačního filtru a celkové řízení interpolačního procesu. Realizace chybových detektorů je založena na již představených principech (viz. kapitola č. 4.2) .



Obr. 4.7 Podrobná struktura navrženého synchronizačního systému s detektorem MLTED



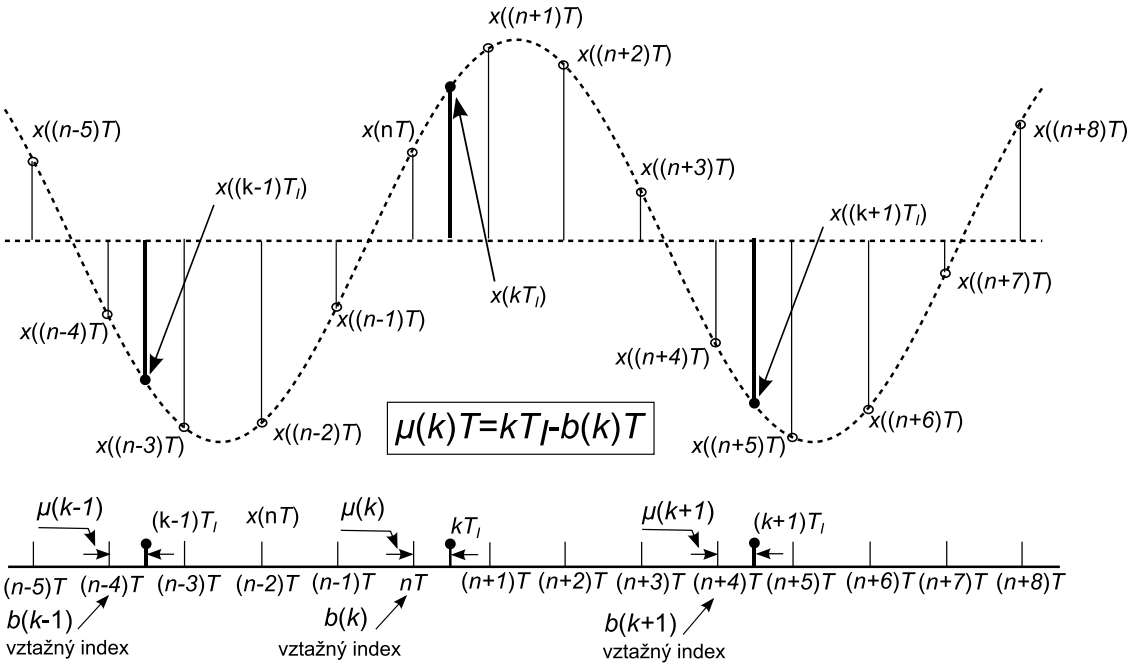
Obr. 4.8 Podrobná struktura navrženého synchronizačního systému s detektorem ZCTED/GTED

4.3.1 Interpolační filtr

Interpolační proces je možné znázornit pomocí obrázku č. 4.9 a opětovně předpokládám pásmově omezený signál vzorkovaný s periodou T

$$\dots, x\big((n-2)T\big), x\big((n-1)T\big), x(nT), x\big((n+1)T\big), x\big((n+2)T\big), \dots \tag{4.43}$$

Požadovaný vzorek v čase kT_I bude dále označován jako k-tý interpolační vzorek. Když se bude k-tý interpolační vzorek bude nacházet mezi vzorky $x(nT)$ a $x(n+1)T$, vzorek s indexem n lze nazvat jako k-tý vztažný referenční bod a v obrázku je č. 4.9 je označen jako $b(k)$. Zlomkový k-tý interval $\mu(k)$ je možné potom vztáhnout k referenčnímu bodu $b(k)$ a tento interval představuje určitou část vzorkovací periody T_s.



Obr. 4.9 Znázornění interpolační procesu a interval $\mu(k)$

Z grafického znázornění je možné odvodit vztah pro zlomkový interval $\mu(k)$

$$\mu(k)T = kT_I - b(k)T \rightarrow \mu(k) = k\frac{T_I}{T} - b(k), \tag{4.44}$$

který splňuje podmínku $0 \leq \mu(k) < 1$. Základní vztah pro interpolační proces je vhodné vyjádřit jako interpolační filtr [20]. Fiktivní systém zobrazený na obrázku č. 4.10 nejprve převede vzorky $x(nT)$ pomocí D/A konverze na váženou řadu impulsů

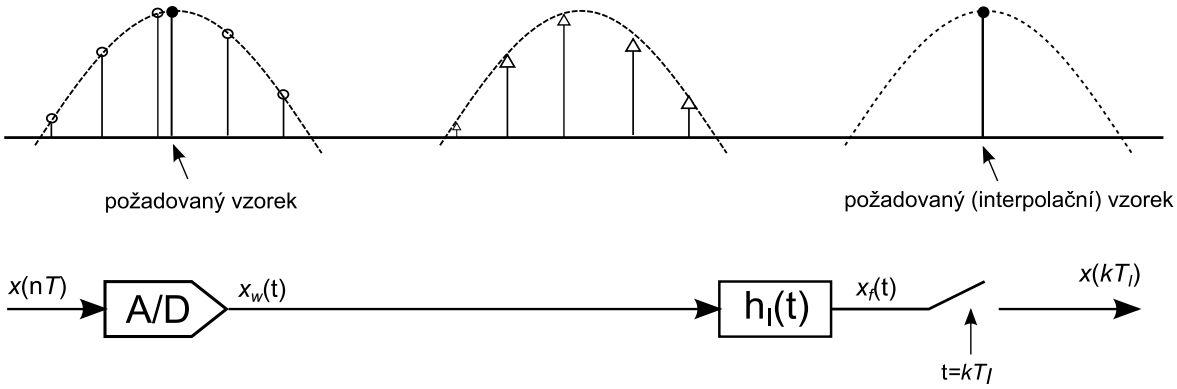
$$x_w(t) = \sum_n x(nT)\delta(t - nT) \tag{4.45}$$

a následně interpolační filtr s impulsní odezvou h_I vytvoří v čase spojitý výstup $x_f(t)$.

$$x_f(t) = \sum_n x(nT) h_I(t - nT) \tag{4.46}$$

Pro získání požadovaného interpolačního vzorku, signál $x_f(t)$ musí být vzorkován (resp. převzorkován) na intervalu kT_I. Výstupem je interpolační vzorek v čase $t=kT_I$ vyjádřený jako

$$x_f(kT_I) = \sum_n x(nT) h_I(kT_I - nT) \tag{4.47}$$



Obr. 4.10 Odvození interpolačního procesu

S využitím indexu filtru I, který je možné také označit jako interpolační index, lze rovnici (4.47) přepsat do tvaru

$$x_f(kT_I) = \sum_I x\big((b(k) - I)T\big) h_I\Big(\big(I + \mu(k)\big)T\Big), \tag{4.48}$$

kde $I = b(k) - n$ a $kT_I = \big(b(k) + \mu(k)\big)T$. Metoda interpolace (Lagrangeova) polynomiálními funkcemi [20] je dále využita pro odvození struktur interpolačního filtru, který bude možné efektivně implementovat. Libovolnou spojitou křivku lze v čase t aproximovat polynomem

$$x(t) \approx c_p t^p - c_{p-1} t^{p-1} + \cdots + c_1 t + c_0. \tag{4.49}$$

Koeficienty tohoto polynomu je možné získat z p+1 hodnot vzorků, které obklopují vztažný index $b(k)$. Po získání koeficientů c_p lze interpolační vzorky v čase $t = kT_I = \big(b(k) + \mu(k)\big)T$ získat s využitím

$$x(kT_I) \approx c_p (kT_I)^p - c_{p-1} (kT_I)^{p-1} + \cdots + c_1 (kT_I) + c_0. \tag{4.50}$$

V případě *p*=1 se jedná o první stupeň polynomu a tedy o lineární interpolátor

$$x(t) \approx c_1 t + c_0. \tag{4.51}$$

a požadované interpolační vzorky lze následně vypočíst z

$$\begin{gathered} x(kT_I) = c_1(kT_I) + c_0\,, resp. \\ x\Big(\big(b(k) + \mu(k)\big)T\Big) = c_1\Big(\big(b(k) + \mu(k)\big)T\Big) + c_0. \end{gathered} \tag{4.52}$$

Koeficienty c_1 a c_0 mohou být získány z dostupných vzorků a pro případ lineární interpolace uvažuji vzorky v bodech [0,1]. Rovnice v maticové podobě x = ***V*** c má rozměr *N*=2

$$\begin{gathered} \begin{bmatrix} x(b(k)T) \\ x(b(k+1)T) \end{bmatrix} = \begin{bmatrix} t_0 & 1 \\ t_1 & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} resp. \\ \begin{bmatrix} x(b(k)T) \\ x(b(k+1)T) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} \end{gathered} \tag{4.53}$$

a kde $t_k = k$. V každém řádku matice se vyskytují po sobě jdoucí členy geometrické posloupnosti počínaje jedničkou. Jedná se o Vandermonde matici [21], jak bude lépe patrno pro vyšší stupeň polynomu. Pro výpočet koeficientů je nutno vypočíst inverzní matici

$$\begin{gathered} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} x(b(k)T) \\ x(b(k+1)T) \end{bmatrix} = \\ \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} x(b(k)T) \\ x(b(k+1)T) \end{bmatrix} \end{gathered} \tag{4.54}$$

Po dosazení do (4.52) získáváme výsledný vztah pro lineární interpolátor

$$\begin{gathered} x\Big(\big(b(k) + \mu(k)\big)T\Big) \approx [\mu + 0]x(b(k+1)T) + [-\mu + 1]x(b(k)T) \\ = \mu(k)x(b(k+1)T) + [1 - \mu(k)]x(b(k)T) \end{gathered} \tag{4.55}$$

Interpolační filtr v případě lineárního interpolátoru má tvar

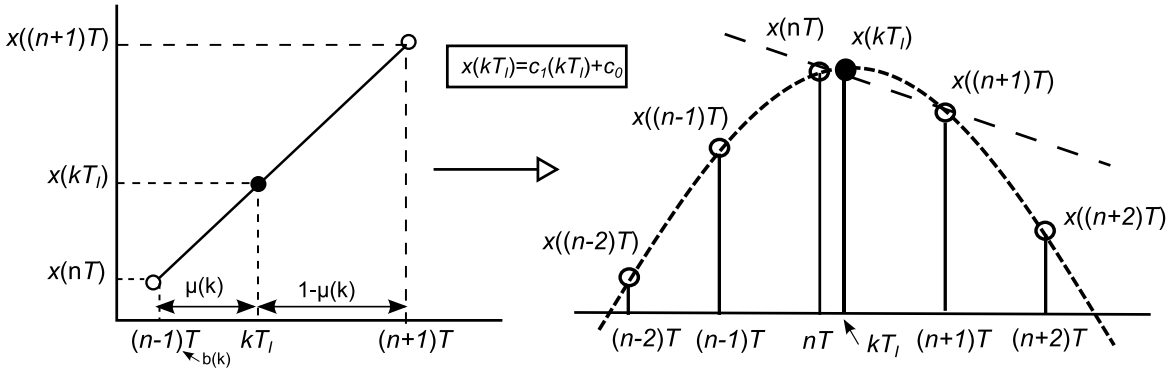
$$x_f(kT_I) = \sum_{I=-1}^{0} h_{LIN}(I)\, x\big((b(k) - I)T\big), \tag{4.56}$$

kde

$$\begin{gathered} h_{LIN}(-1) = 1 - \mu(k) \\ h_{LIN}(0) = \mu(k). \end{gathered} \tag{4.57}$$

Z odvozené struktury (4.56) je zřejmé, že se jedná o FIR filtr s lineární fázovou charakteristikou. To lze zdůvodnit sudým počtem vzorků pro výpočet interpolantů a s tím

související symetrií filtru kolem $\mu(k)$=1/2. Z toho vyplývá požadavek na lichý stupeň interpolačního polynomu. Dále je zřejmé, že koeficienty tohoto filtru nejsou funkcí vztažného bodu $b(k)$, ale pouze zlomkového intervalu $\mu(k)$. To jsou velice výhodné vlastnosti pro konkrétní implementaci; s výhodou lze například využít struktury filtru, který je postavena na metodách distribuované aritmetiky s využitím zřetězeného zpracování. Filtr má také jednotkový zisk a nemění amplitudu přenášeného signálu. Pro názornost ještě uvádím na obrázku č. 4.11 grafickou interpretaci odvození koeficientů pro lineární interpolátor. Lineární interpolátor v oblasti symbolové synchronizace lze použít tehdy, když je k dispozici dostatečný počet vzorků na symbol. Simulace v následujících kapitolách prokáže, že minimálním počet vzorků pro lineární interpolátor je 8 a optimální 16 nebo 32.



Obr. 4.11 Grafická interpretace odvození koeficientů pro lineární interpolátor

Dalším lichým stupněm polynomu je kubický polynom (p+3). Požadované interpolační vzorky lze vypočítat z

$$x(kT_I) \approx c_3(kT_I)^3 + c_2(kT_I)^2 + c_1(kT_I) + c_0. \tag{4.58}$$

Matice pro výpočet koeficientů c_3,c_2,c_1,c_0 v bodech [$-N$/2-1,…,N/2 ={ -1,-0,1,2}] bude mít následující tvar

$$\begin{bmatrix} x(b(k-1)T) \\ x(b(k)T) \\ x(b(k+1)T) \\ x(b(k+2)T) \end{bmatrix} = \begin{bmatrix} t_{-1}^3 & t_{-1}^2 & t_{-1} & 1 \\ t_0^3 & t_0^2 & t_0 & 1 \\ t_1^3 & t_1^2 & t_1 & 1 \\ t_2^3 & t_2^2 & t_2 & 1 \end{bmatrix} \times \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} resp.$$

$$\begin{bmatrix} x(b(k-1)T) \\ x(b(k)T) \\ x(b(k+1)T) \\ x(b(k+2)T) \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}. \tag{4.59}$$

Po výpočtu inverzní matice získáme koeficienty kubického polynomu

$$\begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} x(b(k-1)T) \\ x(b(k)T) \\ x(b(k+1)T) \\ x(b(k+2)T) \end{bmatrix}$$
$$= \begin{bmatrix} -1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/3 & -1/2 & 1 & -1/6 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x(b(k-1)T) \\ x(b(k)T) \\ x(b(k+1)T) \\ x(b(k+2)T) \end{bmatrix} \quad (4.60)$$

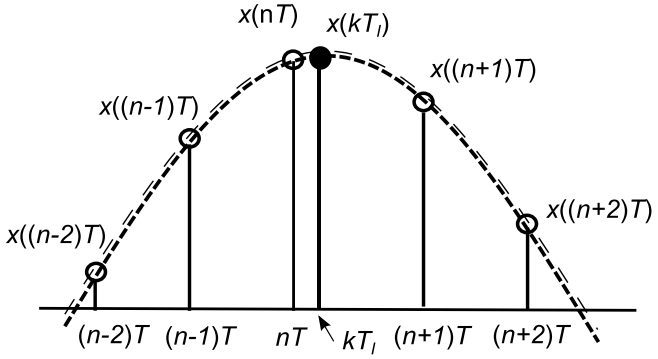
a proces interpolace kubickým polynomem je znázorněn na obrázku č. 4.12 včetně přehledné tabulky s koeficienty. Interpolační filtr v případě kubického interpolátoru má tvar

$$x_f(kT_I) = \sum_{I=-2}^{1} h_{CUB}(I)\, x\big((b(k)-I)T\big), \quad (4.61)$$

kde

$$\begin{aligned} h_{CUB}(-2) &= \frac{\mu(k)^3}{6} - \frac{\mu(k)}{6} \\ h_{CUB}(-1) &= -\frac{\mu(k)^3}{2} + \frac{\mu(k)^2}{2} + \mu(k) \\ h_{CUB}(0) &= \frac{\mu(k)^3}{2} - \mu(k)^2 - \frac{\mu(k)}{2} + 1 \\ h_{CUB}(1) &= -\frac{\mu(k)^3}{6} + \frac{\mu(k)^2}{2} - \frac{\mu(k)}{3} \end{aligned} \quad (4.62)$$

Interpolace pomocí kubického polynomu je vhodná v případě, že je k dispozici malý počet vzorků na symbol. To je případ detektoru ZCTED (GTED), který pracuje vždy se dvěma vzorky na symbol.



$x(kT_I)=c_3(kT_I)^3+c_2(kT_I)^2+c_1(kT_I)+c_0$

I	$c_0(I)$	$c_1(I)$	$c_2(I)$	$c_3(I)$
-2	0	-1/6	0	1/6
-1	0	1	1/2	-1/2
0	1	-1/2	-1	1/2
1	0	-1/3	1/2	-1/6

Obr. 4.12 Kubický interpolátor s tabulkou koeficientů

Při odvození lineárního a kubického interpolačního filtru byl úmyslně vynechán kvadratický polynom (p+2)

$$x(kT_I) \approx c_2(kT_I)^2 + c_1(kT_I) + c_0. \quad (4.63)$$

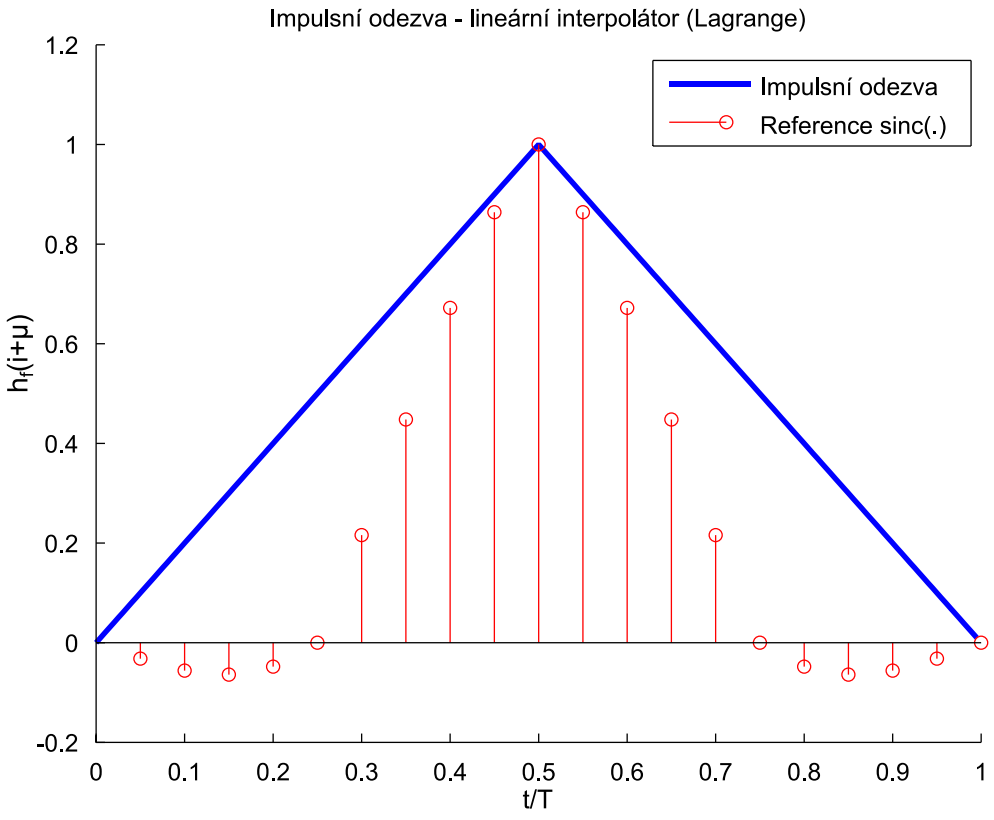
Pro výpočet interpolačních vzorků je potřeba využít tři vstupní vzorky. Počet vzorků je lichý a z tohoto důvodu není splněna podmínka symetrie filtru vůči $\mu(k)$=1/2. Autoři z [20] ukázali, že je možné využít čtyři vstupní vzorky a strukturu parabolického interpolačního filtru vyjádřit stejným způsobem jako u kubické struktury

$$x_f(kT_I) = \sum_{I=-2}^{1} h_{KVAD}(I)\, x\big((b(k) - I)T\big), \tag{4.64}$$

kde

$$\begin{aligned} h_{KVAD}(-2) &= \beta\mu(k)^2 - \beta\mu(k), \\ h_{KVAD}(-1) &= -\beta\mu(k) + (1+\beta)\mu(k), \\ h_{KVAD}(0) &= -\beta\mu(k) - (1-\beta)\mu(k), \\ h_{KVAD}(1) &= \beta\mu(k)^2 - \beta\mu(k). \end{aligned} \tag{4.65}$$

a β je parametr, který je potřeba vzít v úvahu z důvodu využití čtyř vstupních vzorků. Simulace ukázala, že hodnota parametru β=0.5 je vhodná i z důvodu implementace v hardwaru. Impulsní odezvy pro jednotlivé interpolační filtry s referenční impulsní odezvou *sinc*(.) jsou zobrazený na následujících grafech. Na obrázku č. 4.13 je zobrazena simulovaná impulsní odezva pro lineární interpolační filtr

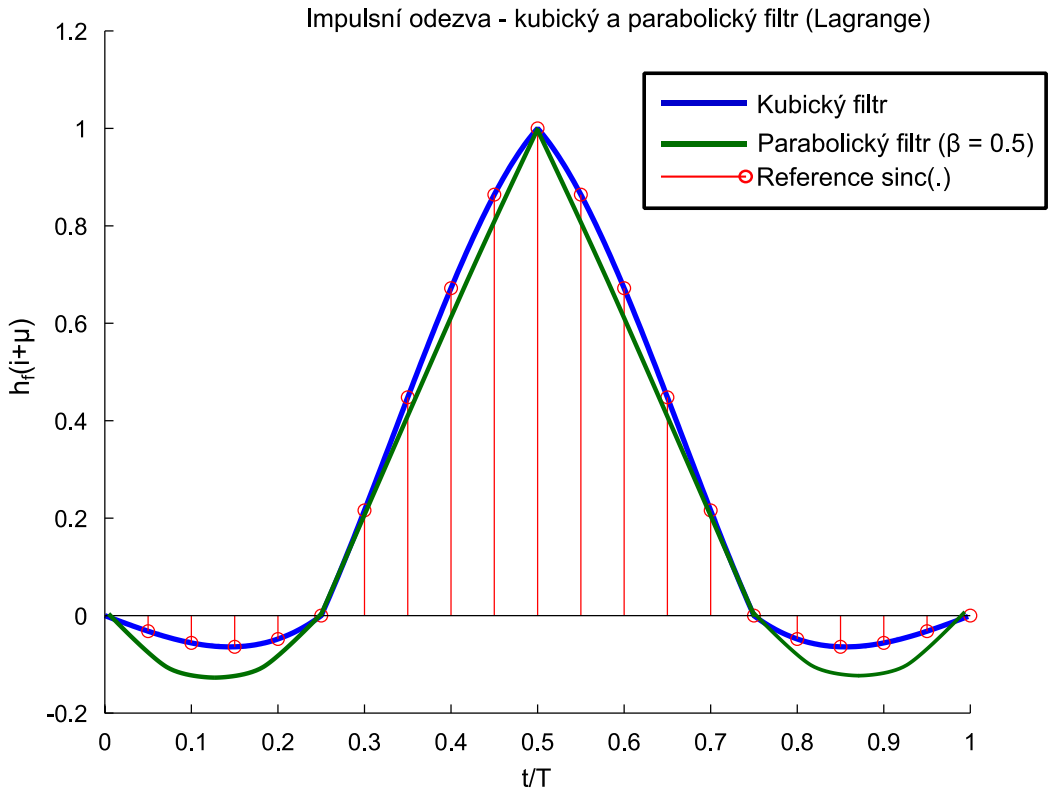


Obr. 4.13 Impulsní odezva pro lineární interpolační filtr

Impulsní odezvy pro kubický a parabolický (β=0.5) interpolační filtr jsou pro porovnání uvedeny na obrázku č. 4.14. Impulsní odezva kubického filtru je téměř shodná s impulsní odezvou filtru *sinc*(.). Tento filtr byl zvolen jako referenční, neboť je se jedná o ideální interpolační filtr typu dolní propust s impulsní charakteristikou

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T}. \tag{4.66}$$

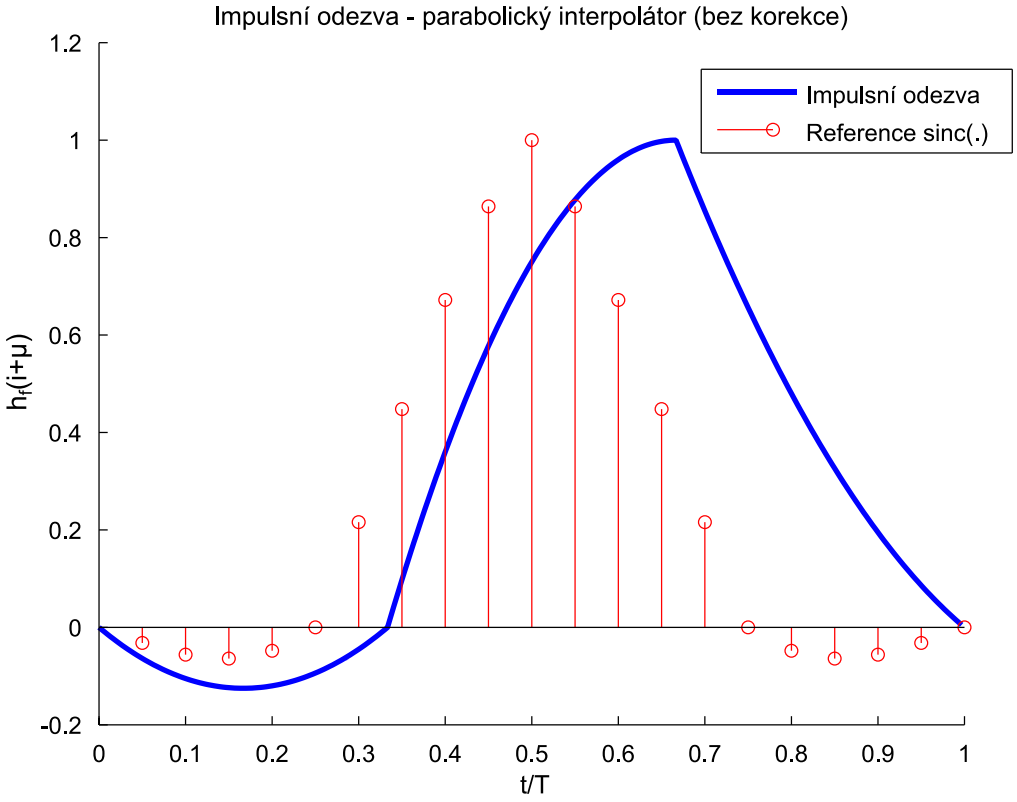
Pro úplnost uvádím na obrázku č. 4.15 impulsní odezvu pro parabolický filtr bez provedených úprav podle (5.66) a (5.67), která je asymetrická a není tedy možné bez provedených úprav tento filtr přímo použít.



Obr. 4.14 Impulsní odezva pro kubický a parabolický interpolační filtr (β=0.5)

Přímá implementace v hardwaru vyžaduje minimálně dvě obecné násobičky se dvěma obecnými vstupy. Z rovnice (4.64) lze vyjádřit h_I jako

$$h_I\left(\left(I + \mu(k)\right)T\right) = \sum_{l=0}^{p} c_l(I)\mu(k). \tag{4.67}$$



Obr. 4.15 Impulsní odezva pro parabolický interpolační filtr (bez korekce)

Po zpětném dosazení (4.67) do (4.64) a po redistribuci členů (lineární operace) lze výraz pro interpolační filtr přepsat do podoby

$$x_f(kT_I) = \sum_{l=0}^{p} \mu(k) \underbrace{\sum_{I=I_1}^{I_2} c_I(I)x\big((b(k)-I)T\big)}_{w(l)}. \tag{4.68}$$

Nyní již je nutno použít minimálně jednu násobičku s obecnými vstupy a druhá násobička reprezentující výraz $w(l)$ z (4.68) bude mít jeden vstup pevný. Potom tuto násobičku lze nahradit bitovými posuvy (násobení mocninou o základu dva) a aplikovat metodu distribuované aritmetiky. Výsledný vztah pro kubický interpolační filtr bude mít podobu

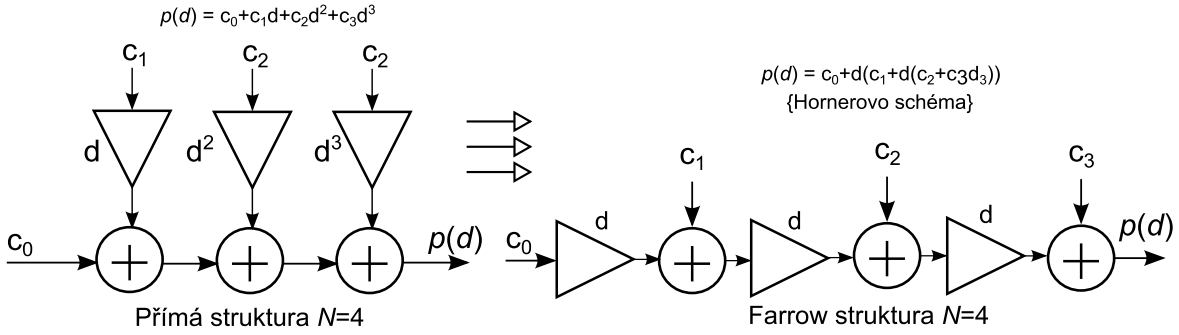
$$x(kT_I) = w(0) + \mu(k)\big(w(1) + \mu(k)\big(w(2) + w(3)\mu(k)\big)\big) \tag{4.69}$$

a pro parabolický

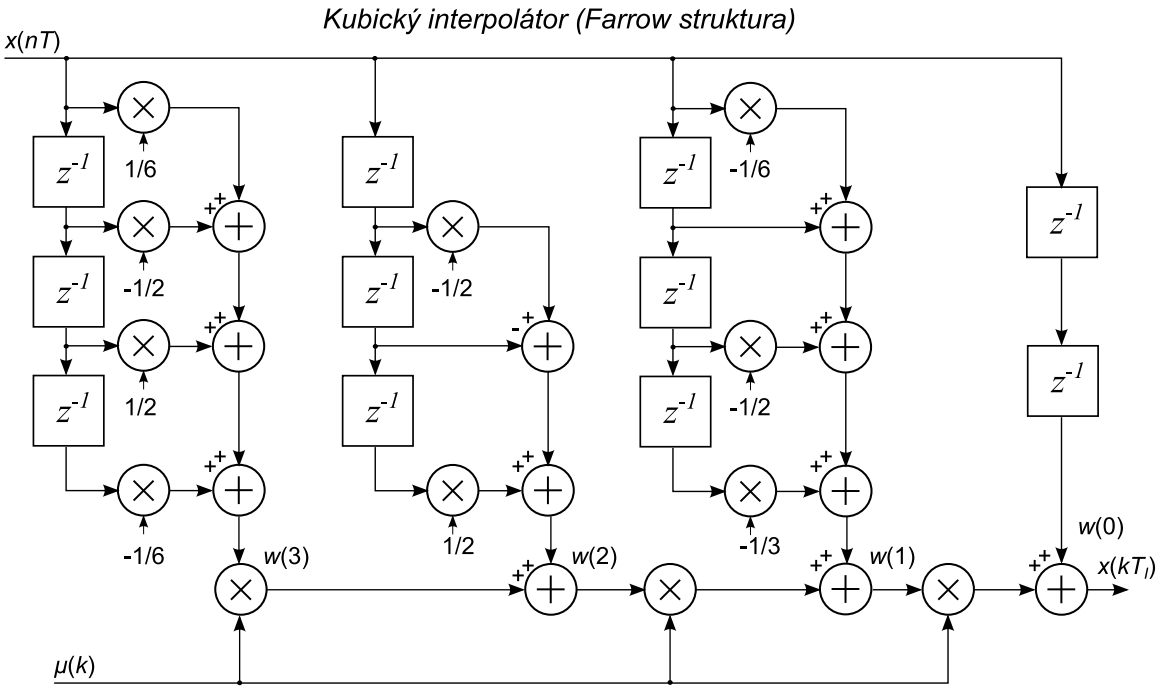
$$x(kT_I) = w(0) + \mu(k)(w(1) + \mu(k)w(2)\,). \tag{4.70}$$

Z výrazů (4.69) a (4.70) je zřejmé, že se jedná o aplikaci *Hornerova* schéma z teorie lineární algebry. V oblasti signálového zpracování se tato struktura označuje jako *Farrow*

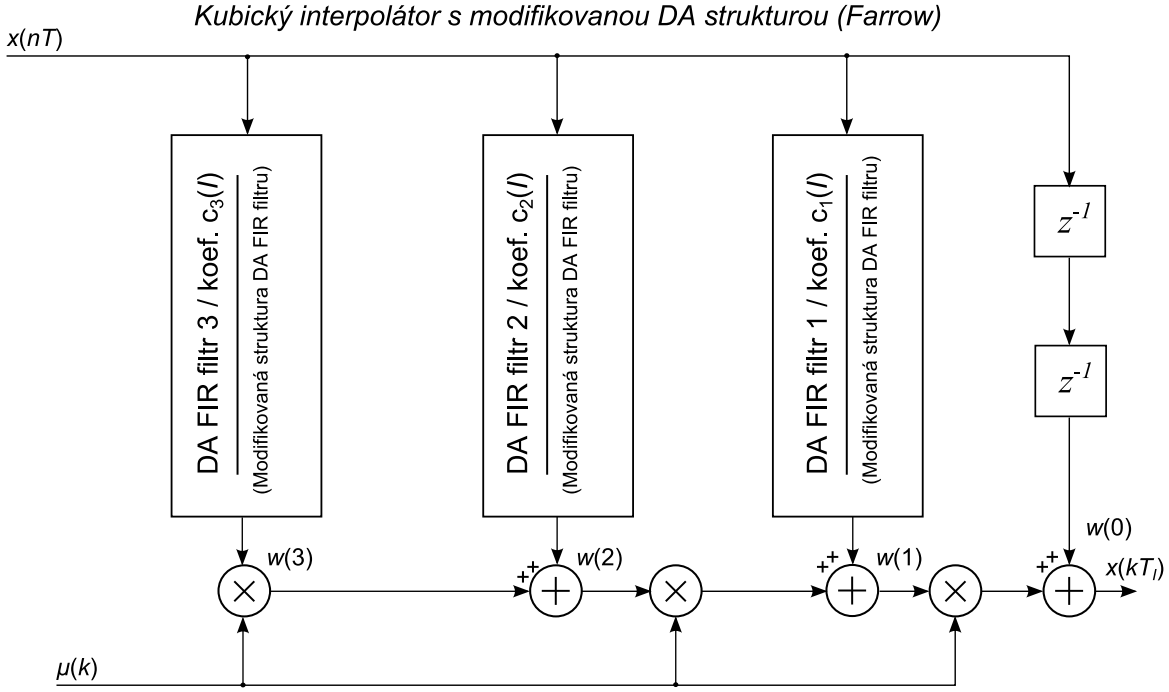
struktura a je optimální pro přímou implementaci na *FPGA* či *ASIC*. Na obrázku č. 4.16 je naznačen převod přímé struktury s využitím Hornerova schéma na Farrow strukturu. Obrázek č. 4.17 dále představuje přímo implementovatelnou Farrow strukturu pro kubický interpolátor a na obrázku č. 4.18 dále ukazuji využití modifikované struktury DA FIR filtru ve struktuře interpolátoru. Je aplikována plně paralelní DA FIR struktura a modifikace spočívá ve vynechání části *FIR_FINAL* (tj. změna měřítka + saturace) z původního modelu.



Obr. 4.16 Polynomiální interpolace $N = 4$ - přímá implementace a převod na Farrow strukturu



Obr. 4.17 Farrow struktura pro implementaci kubického interpolačního filtru



Obr. 4.18 Farrow struktura pro implementaci kubického interpolačního filtru (DA FIR)

4.3.2 Řízení interpolačního procesu

Řízení interpolačního procesoru je řešeno pomocí *modulo-1* čítače. V principu lze tento čítač implementovat jako inkrementující nebo dekrementující. Inkrementující čítač je inkrementován v průměru vždy o *1/N* (při práci s pevnou řádovou čárkou kvantováno na určitý počet bitů) a naopak. Protože například ZCTED (GTED) pracuje se 2 vzorky/symbol, *N* bude rovno 2. Pro MLTED, kde je využito lineárního interpolátoru může *N* nabývat hodnot 8, 16 nebo 32. Jedná se o testované hodnoty, které jsem použil při simulacích. Čítač přeteče každých *N* vzorků a výstup PLL filtru nastaví hodnotu, o kterou tento čítač přetekl. Zlomkový interval μ je počítán při každé periodě přetečení a zmíněný princip je pro inkrementující čítač ilustrován na obrázku č. 4.19.

Velikost tohoto intervalu lze odvodit ze struktury bloku kontroly interpolačního procesu a obrázku č. 4.19. Po přetečení inkrementujícího modulo-1 čítače platí

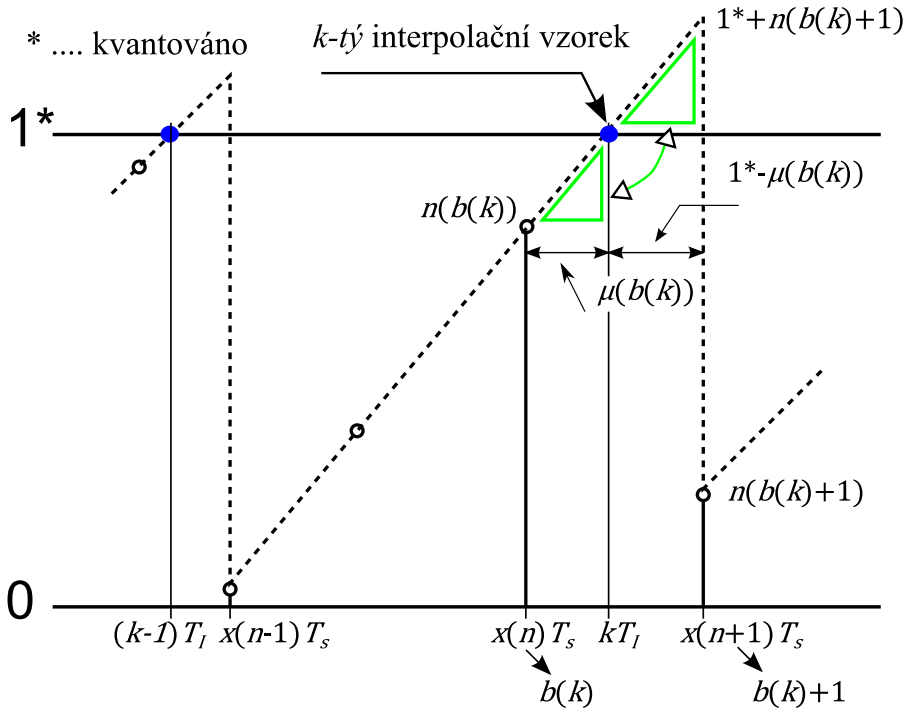
$$n(b(k)+1) = n\big(b(k)\big) + w\big(b(k)\big) - 1. \tag{4.71}$$

Jak je naznačeno na obrázku č. 4.19, hodnoty v bodech $n(b(k)+1)$ a $n(b(k))$ tvoří podobné trojúhelníky. Tuto identitu je možné přepsat pomocí vztahu

$$\frac{\mu(b(k))}{1-n(b(k))} = \frac{1-\mu(b(k))}{n(b(k)+1)}. \tag{4.72}$$

Po roznásobení lze $\mu(b(k))$ vyjádřit jako

$$\mu(b(k)) = \frac{1-n(b(k))}{n(b(k)+1)-n(b(k))+1} = \frac{1-n(b(k))}{w(b(k))}. \tag{4.73}$$



Obr. 4.19 Výpočet zlomkového intervalu μ s využitím principu inkrement. modulo-1 čítače

Při konkrétní implementaci může být výhodnější využití principu dekrementujícího modulo-1 čítače a proto pro úplnost uvádím jeho odvození. Princip je uveden na obrázku č. 4.20. Po podtečení dekrementujícího modulo-1 čítače platí

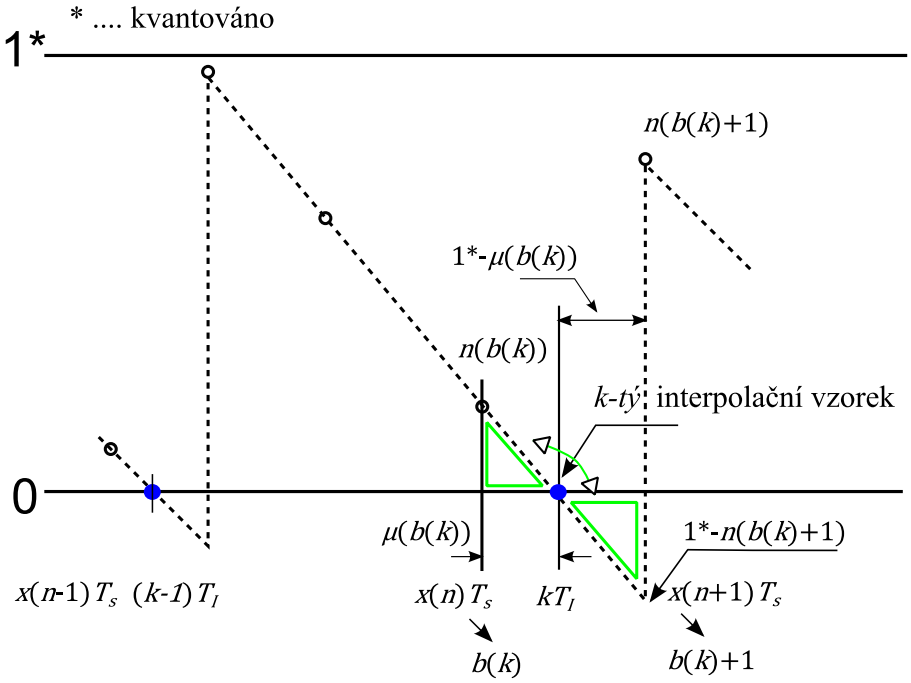
$$n(b(k)+1) = n(b(k)) - w(b(k)) + 1. \tag{4.74}$$

Hodnoty v bodech $n(b(k)+1)$ a $1-n(b(k)+1)$ tvoří podobné trojúhelníky. Tuto identitu je možné opět přepsat pomocí vztahu

$$\frac{\mu(b(k))}{n(b(k))} = \frac{1-\mu(b(k))}{1-n(b(k)+1)}. \tag{4.75}$$

Po roznásobení lze $\mu(b(k))$ vyjádřit pro dekrementující čítač napsat

$$\mu(b(k)) = \frac{n(b(k))}{n(b(k))-n(b(k)+1)+1} = \frac{n(b(k))}{w(b(k))}. \tag{4.76}$$



Obr. 4.20 Výpočet zlomkového intervalu μ s využitím principu dekrement. modulo-1 čítače

Výrazy (4.73) a (4.76) obsahují obecné dělení. Implementace obecného dělení v pevné i pohyblivé řádové čárce je v HDL jazycích podporována, ale přímá syntéza na hradlovém poli FPGA nikoli. Operaci obecného dělení je tedy možné přímo pouze simulovat. Na tomto místě je vhodné poznamenat, že přímá syntéza dělení je možná pouze pro dělitele s mocninou o základu dva. Dělení poté přechází na bitový posuv. Nástroje pro syntézu často nabízejí IP blok pro implementaci obecného dělení, případně je možné tento blok implementovat prostřednictvím sériové děličky nebo za pomoci modifikovaného algoritmu CORDIC. Dělení je samo o sobě pomalou operací, sekvenčním výpočtem na několik period hodin, a proto je vhodné tuto operaci ve vztazích (4.73) a (4.76) aproximovat pro rychlou implementaci v pevné řádové čárce a optimální syntézu na FPGA.

Z principu výpočtu zlomkového intervalu lze vyvodit, že $1/N \approx T_l/T_s$. Ačkoliv je T_l/T_s neznámý a iracionální, jmenovitá hodnota ξ_0 s konečnou přesností může být velmi dobrou aproximací pro skutečnou hodnotu. Zlomkový interval μ může být v případě detektoru ZCTED aproximován takto

$$\mu\big(b(k)\big) \approx \xi_0[1 - \eta(b(k)] \; where \; \xi_0 = N = 2. \tag{4.77}$$

V případě velké odchylky ξ_0, je možné aplikovat korekci prvního řádu, standardní odchylka pro μ je dále redukována na $\Delta\xi^2/\xi_0^2\sqrt{12}$ [22]. Opět není nutné provádět obecné dělení. Korekce intervalu přetečení resp. podtečení je nutné provést, protože skutečný A/D převodník nebude nikdy na svém výstupu produkovat celočíselný počet vzorků na periodu (např. 2 vzorky na periodu v případě detektoru ZCTED). Skutečná vzorkovací perioda T_r je vždy o určitý zlomkový interval menší nebo větší než požadovaná (ideální) vzorkovací perioda T_s. Korekce je nutná především při málem počtu vzorků na symbol, v případě 2 vzorků na symbol je nutno korekci provést vždy. Princip této korekce pro detektor pracujícím se 2 vzorky/symbol a s inkrementujícím modulo-1 čítačem je shrnut v následujících dvou odstavcích.

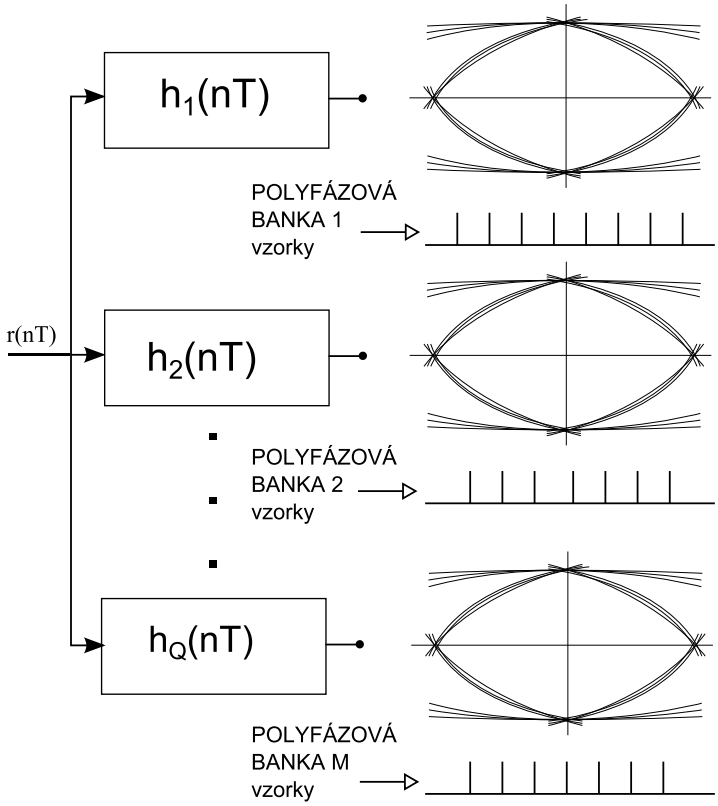
- Jestliže bude vzorkovací rychlost T_r větší, tj. $T_r > T_s/2$, potom je vložen vzorek s nulovou hodnotou z důvodu zabránění vzniku dvou po sobě jdoucích přetečení.
- Jestliže bude vzorkovací rychlost T_r menší, tj. $T_r < T_s/2$, potom je určitý vzorek vynechán z důvodu zabránění vzniku přetečení po dvou po sobě jdoucích vzorcích místo jednoho vzorku.

4.4 Symbolová synchronizace využívající bank polyfázových filtrů

Polyfázová dekompozice pomocí bank filtrů je užitečná při implementaci decimačních nebo interpolačních filtrů. Tento princip je ale možné aplikovat i do oblasti symbolové synchronizace, kde pro zvýšení vzorkovacího kmitočtu využijeme bank filtrů v konfiguraci, kterou lze označit jako polyfázový interpolátor. Diskrétní impulsní odezva m-tého filtru v dané bance filtrů lze vyjádřit jako [23]

$$h_m(mT_s) = h\left(nT_s + \frac{m}{M}T\right). \tag{4.78}$$

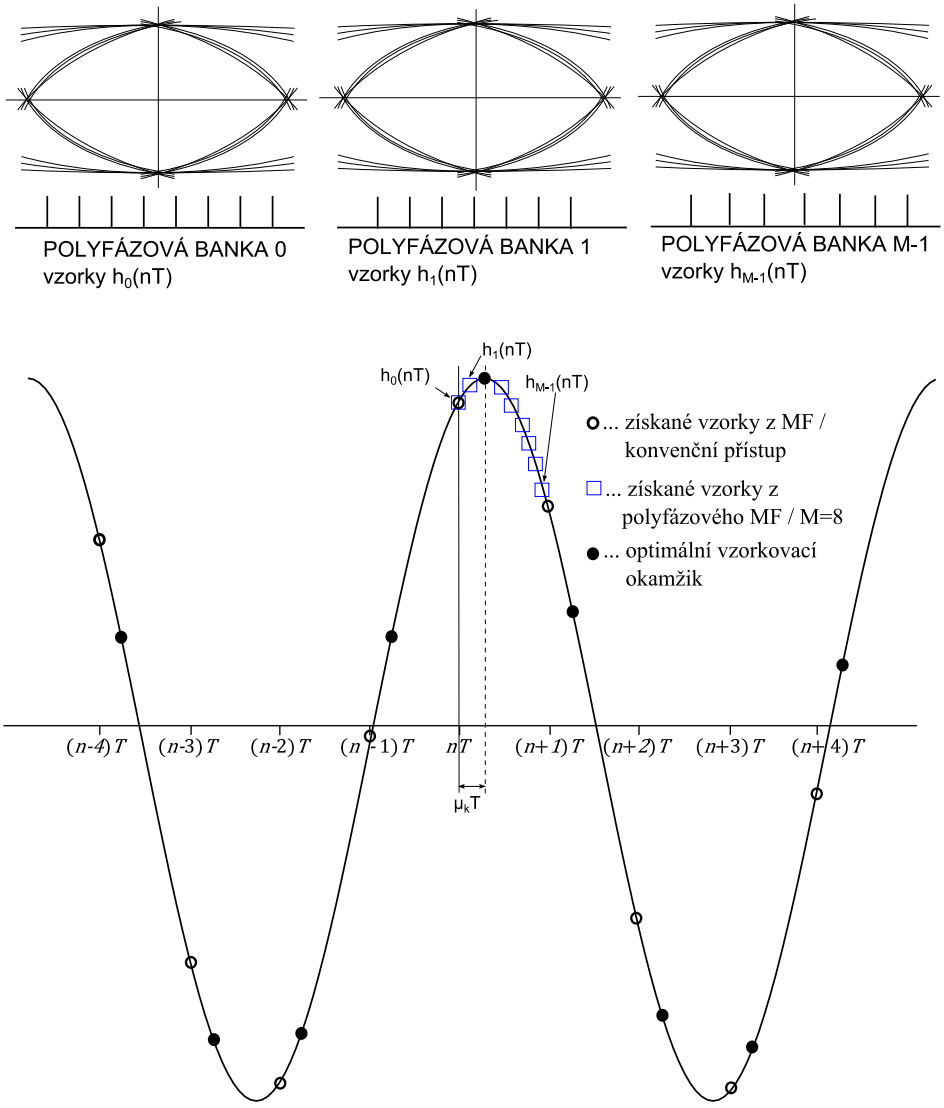
Všechny tyto filtru (sub-filtry) mají stejnou amplitudovou frekvenční charakteristiku, ale jsou vůči sobě zpožděny o určitý počet vzorků, který zavádí fázový posun [24]. Synchronizační systém musí vybrat určitý sub-filtr s indexem m tak, aby zlomková část m/M vzorkovací periody T byla v blízkosti požadovaného časového ofsetu. Tento princip je ilustrován na obrázku č. 4.21 a č. 4.22.



Obr. 4.21 Polyfázový interpolátor pro symbolovou synchronizaci – základní princip

V praxi lze implementovat pouze konečný počet sub-filtrů a z tohoto důvodu máme k dispozici pouze konečný počet interpolačních vzorků. Optimálního vzorkovacího

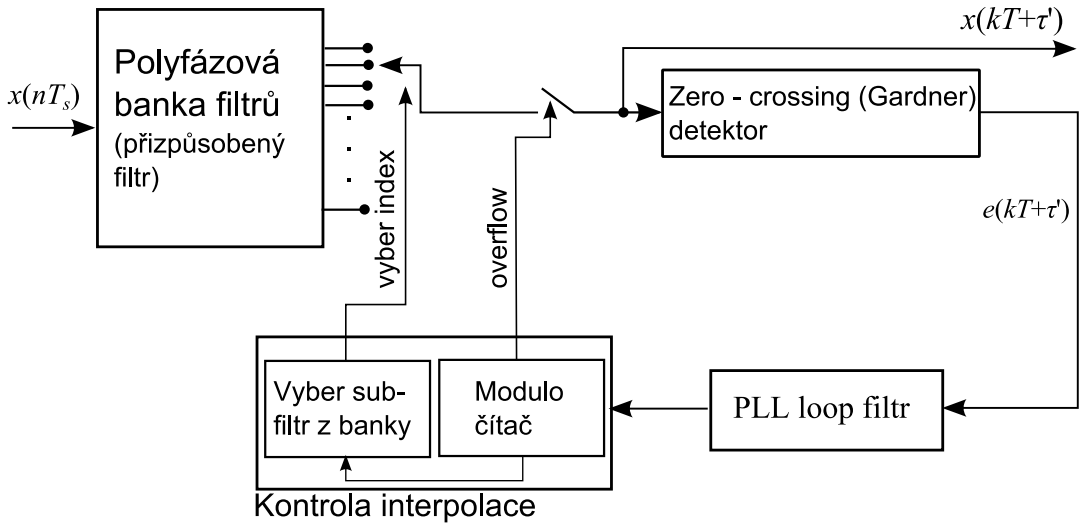
okamžiku nedosáhneme, ale můžeme se k němu velmi přiblížit. To je podobné jako v případě konvečního přístupu s využitím lineárního, parabolického nebo kubického interpolačního filtru. V případě konvenčního přístupu impulsní odezva interpolačního filtru s rostoucím stupněm polynomu h_I aproximuje se stále větší a větší přesností (4.66), než v limitě $p \to \infty$ přechází právě na funkci *sinc*(.). Při implementaci si ale často zcela vystačíme s kubickým nebo kvadratickým interpolátorem.



Obr. 4.22 Určení optimálního vzorkovacího okamžiku – srovnání s konvenčním přístupem

Na základě výše uvedeného principu jsem vytvořil systémový model s využitím polyfázového interpolačního filtru. Model se skládá ze zmíněné polyfázové banky FIR filtrů, detektoru ZCTED nebo Gardnerova (GTED) detektoru, PLL filtru a bloku kontroly interpolačního procesu. Chybový detektor a PLL filtr jsou plně převzaty z konvenčního

modelu, blok kontroly interpolačního procesu je doplněn o digitální komparátor, jehož funkce bude popsána později. Polyfázový filtr plní zároveň funkci přizpůsobeného filtru. Banka filtrů využívá metod distribuované aritmetiky a interpolační filtr z kapitoly 3.4 je upraven speciálním způsobem pro potřeby symbolové synchronizace. Blokové schéma je zobrazeno na obrázku č. 4.23.



Obr. 4.23 Navržené blokové schéma symbolové synchronizace s polyfázovým filtrem

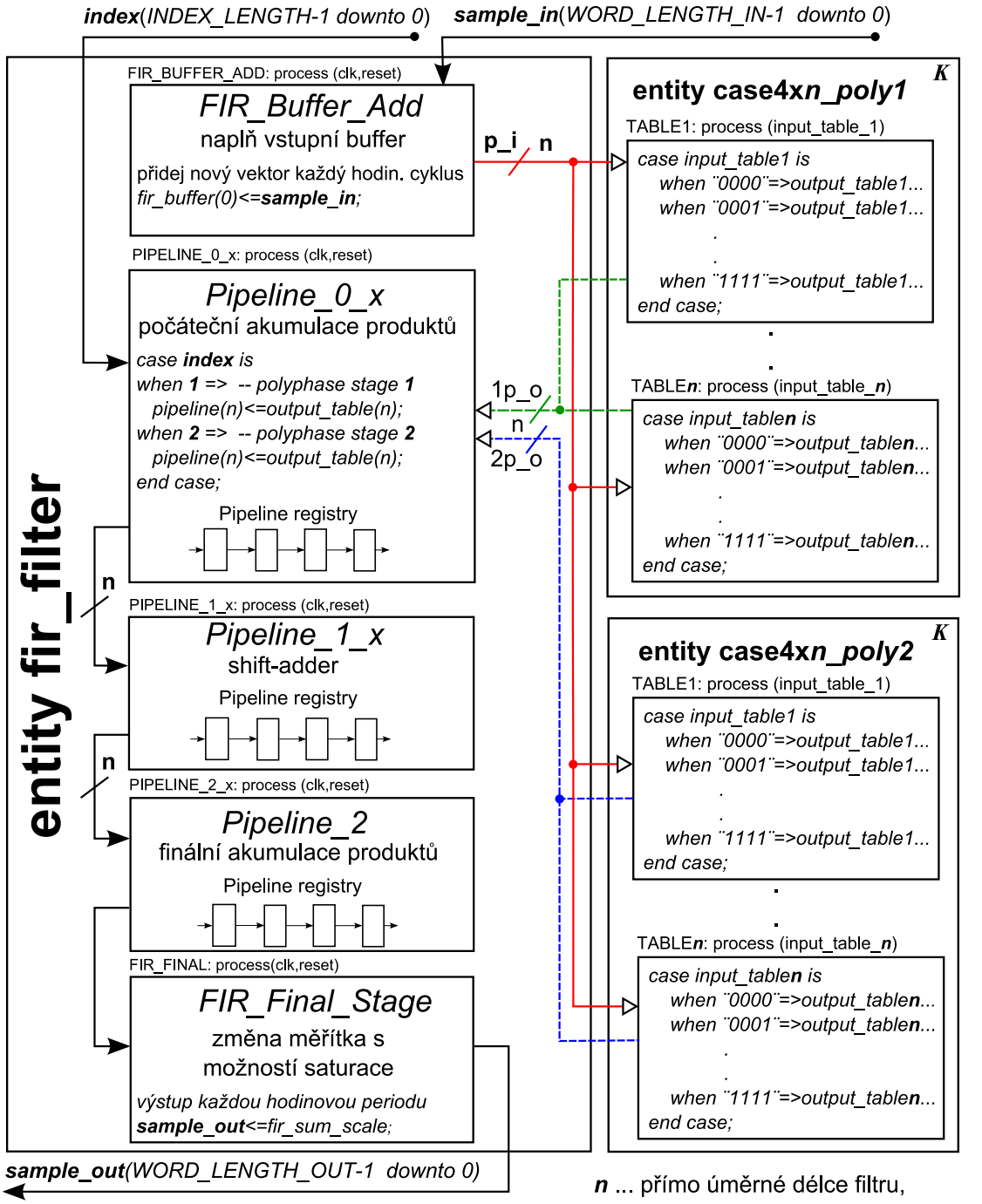
Struktura implementace zmíněného plně paralelního filtru se znaménkem (dvojkový doplněk) v jazyce *VHDL* je představena na obrázku č. 4.24. Filtr provádí výpočet a poskytuje výstup v jednom hodinovém cyklu. Dále budou popsány změny vůči modelu interpolačního filtru z kapitoly 3.4. Obecný interpolační filtr načítá nový vstupní vzorek každých *I* hodinových cyklů. Potom v hlavní entitě filtru se nachází pouze jeden hodinový signál. Tento hodinový signál je *I*-krát rychlejší, než hodinový signál, kterým je řízen originální vzorkovaný signál. Nezbytné modifikace pro symbolovou synchronizaci jsou shrnuty v následujících dvou odstavcích.

- Aktuální polyfázový sub-filtr je vybrán prostřednictvím signálu označeného jako *index*, který je generován blokem kontroly interpolačního procesu. Tento signál specifikuje adresu pro multiplexor ve struktuře filtru.
- Interní modulo *I* čítač není potřeba, protože každý výstupní vzorek je čten každou hodinovou periodu. Vždy jeden sub-filtr s požadovaným fázovým ofsetem je potřeba a poskytuje v čase svůj výstup. Tento sub-filtr je právě vybrán signálem s názvem *index*.

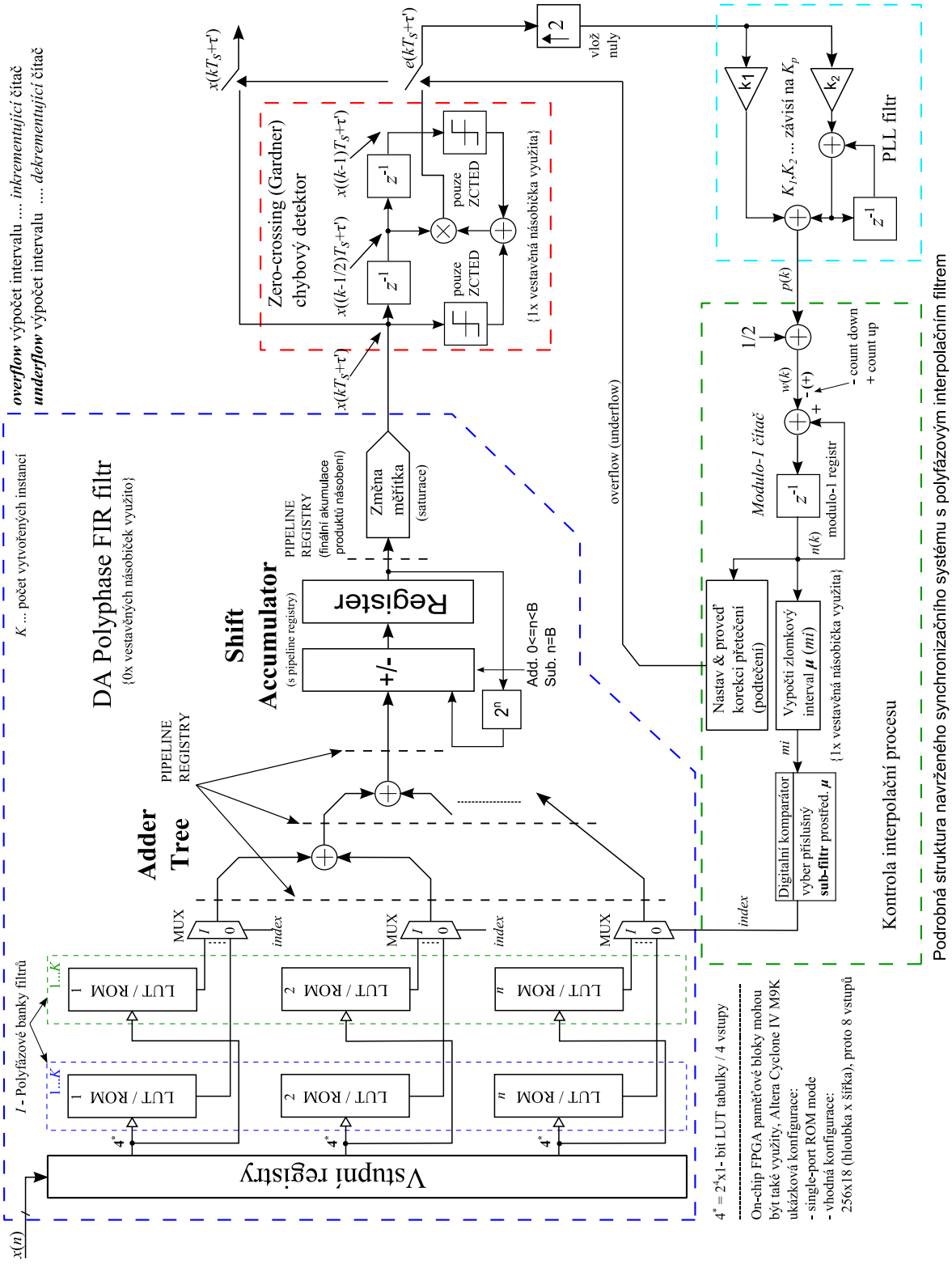
Hlavní VHDL entita označená jako *fir_filter* představuje hlavní část modelu filtru a představuje šablonu, která je modifikována pro rozdílné sady koeficientů. Signálový vektor *sample_in* je uložen do vstupního zásobníku s hloubkou, která odpovídá počtu koeficientů sub-filtru v polyfázové bance. Každou hodinovou periodu je načten nový vektor a starší vektory jsou posunuty. Proces *FIR_Buffer_Add* má v principu funkci posuvného registru. Mapování $f(c[n]\times x[n])$ je dále konkurenčně provedeno za pomoci instance entity *case4x**n**_poly_**I*** , kde n je přímo úměrné počtu koeficientů sub-filtru a I je interpolační faktor. Předpokládám, že jsou použity 2^4 x 1 - bitové *LUT* tabulky. Z tohoto důvodu jsem entity označil jako case***4***. Výběr typu *LUT* tabulky je konfigurovatelný parametr. Pro každý sub-filtr v polyfázové bance je vytvořena jedna nezávislá entita *case4x**n**_poly_**I***. To například znamená, že při interpolačním faktoru $I =$ 8 je potřeba 8 sub-filtrů a také 8 entit. Pro plně paralelní zpracování dat je potřeba vytvořit k těchto instancí, neboť pro každý bit vstupního slova je potřeba jedna instance.

Proces *PIPELINE_0_x* využívá multiplexor s adresným vstupem, který je řízen signálem *index* generovaným blokem řízení interpolačního procesu z důvodu výběru správného sub-filtru. Dále tento proces provádí prvotní úkony pro následnou akumulaci částečných produktů násobení. Malé písmeno *x* značí identifikační číslo *VHDL* procesu ve stejné sekci zřetězeného zpracování. Sekce procesů *PIPELINE_1_x* provádí násobení (pomocí bitových posuvů) a akumulaci těchto produktů. Vestavěné násobičky nejsou potřeba a je možné je vyčlenit pro specializované DSP bloky. Proces *PIPELINE_2_x* dokončuje výpočet akumulace částečných produktů násobení a proces *FIR_Final_Stage* implementuje změnu měřítka s možností saturace. Entity *case4x**n**_poly_**I*** jsou automaticky generované z prostředí programu Matlab. Sekce zřetězeného zpracování *PIPELINE_1_x* může být velice rozsáhlá a proto je také generována automaticky. Vytvořený VHDL kód s definicí signálů je poté vložen do hlavní entity s názvem *fir_filter*.

Navržená struktura může místo LUT tabulek pro uložení koeficientů využít integrovaných RAM bloků konfigurovaných jako paměť ROM (single-port). Toto rozšíření bylo testováno s *Altera* M9K integrovanými RAM bloky s využitím IP bloku *Altsyncram Megafunction* [12]. Nevýhoda tohoto rozšíření je, že musíme počítat s jedním hodinovým cyklem navíc při čtení z paměti. Výhodou může být úspora cenných logických bloků FPGA. Struktura synchronizační systému s polyfázovým filtrem je uvedena na obr. č. 4.25.



Obr. 4.24 VHDL struktura plně paralelního DA FIR polyfázového interpolátoru pro symbolovou synchronizaci (interpolační faktor I=2 je pro ilustraci)



Obr. 4.25 Podrobná struktura navrženého synchronizačního systému s detektorem ZCTED/GTED využívající bank polyfázového filtr

5 Návrh struktur synchronizace fáze nosné vlny vhodných pro implementaci na FPGA

Synchronizace nosné vlny je proces, který zabezpečuje, že lokální oscilátor na straně přijímače je synchronizovaný ve fázi i frekvenci s oscilátorem nosné vlny na straně vysílače [4]. Fázová chyba nosné vlny způsobuje rotaci signálu v I/Q diagramu. V případě, že tato rotace bude příliš velká, jednotlivé symboly se budou nacházet na nesprávných rozhodovacích oblastech I/Q diagramu. Problém synchronizace fáze nosné vlny nastává vždy i v případě ideálního komunikačního kanálu bez přidaného šumu, jestliže místní oscilátory na straně vysílače a přijímače nebudou synchronizovány [6]. To v reálných podmínkách téměř vždy z logických důvodů nelze splnit, a proto blok symbolové synchronizace musí být společně se symbolovou synchronizací obsažen v každém přijímači. Budou představeny dva přístupy. První přístup představuje v podstatě modifikaci *Costasovy* smyčky pro *m-QAM* signály. Ve druhém případě je synchronizace fáze nosné vlny provedena až po symbolové synchronizaci. Opět budou vytvořeny modely, u kterých je možné provést efektivní syntézu na hradlovém poli FPGA.

Předpokládám dále, že přijatý pásmově omezený *m-QAM* signál lze vyjádřit jako

$$r(t) = \sum_{k}^{N} a_0(k)p(t - kT_s)\sqrt{2}\cos(\omega_0 t + \theta) - a_1(k)p(t - kT_s)\sqrt{2}\sin(\omega_0 t + \theta) + w(t) \tag{5.1}$$

kde $a_0(k)$ a $a_1(k)$ jsou soufázová I a kvadraturní Q složky k-tého symbolu; $p(t)$ je jednotková energie pulsu tvarovacího signálu na intervalu $-LT_S \leq T \leq LT_S$; T_s je symbolová perioda; θ je neznámý fázový ofset nosné vlny, který má být určen a $w(t)$ je Gausovo bílý šum. Navzorkovaný signál bude mít poté podobu

$$r(nT) = \sum_{k}^{N} a_0(k)p(nT - kT_s)\sqrt{2}\cos(\Omega n + \theta) - a_1(k)p(nT - kT_s)\sqrt{2}\sin(\Omega n + \theta) + w(nT), \tag{5.2}$$

kde $\Omega = \omega_0 T$ radiánů / vzorek.

5.1 Metody synchronizace fáze nosné vlny

První uvedená metoda využívá principu modifikované Costasovy smyčky pro signály typu *m-QAM*. Přijatý pásmově omezený navzorkovaný signál označený jako *r*(*nT*) (předpokládám mezifrekvenci) je nejprve přeložen do základního pásma pomocí kvadraturních signálových složek $\sqrt{2}\cos(\Omega n + \hat{\theta} n)$ a $-\sqrt{2}\sin(\Omega n + \hat{\theta} n)$ generovaných blokem přímé digitální syntézy (*DDS - direct digital synthesis*). Signál přeložený do základního pásma bude mít podobu (uvedeno již bez složky 2**f*$_c$)

$$\begin{aligned} I(nT) &= \sum_{k}^{N} a_0(k)p(nT - kT_s)\cos(\theta - \hat{\theta} n) - a_1(k)p(nT - kT_s)\sin(\theta - \hat{\theta} n) \\ &\quad + w_I(nT)\ \{I\ složka\}\ a \\ Q(nT) &= \sum_{k}^{N} a_0(k)p(nT - kT_s)\sin(\theta - \hat{\theta} n) + a_1(k)p(nT - kT_s)\cos(\theta - \hat{\theta} n) \\ &\quad + w_Q(nT)\ \{Q\ složka\}. \end{aligned} \tag{5.3}$$

Signály dále procházejí blokem přizpůsobených filtrů s impulsní odezvou *p*(-*nT*) a produkují na svém výstupu signály pro soufázovou $x_i(nT)$ a kvadraturní složku $x_q(nT)$

$$\begin{aligned} x_i(nT) &= \sum_{k}^{N}\left[a_0(k)\cos(\theta - \hat{\theta} n) - a_1(k)\sin(\theta - \hat{\theta} n)\right] r_a(nT - kT_s) \\ &\quad + v_I(nT)\ \{I\ složka\}. \\ x_q(nT) &= \sum_{k}^{N}\left[a_0(k)\sin(\theta - \hat{\theta} n) + a_1(k)\cos(\theta - \hat{\theta} n)\right] r_a(nT - kT_s) \\ &\quad + v_Q(nT)\ \{Q\ složka\}. \end{aligned} \tag{5.4}$$

kde $r_a(m)$ je autokorelační funkce tvarovacího pulsu

$$r_a(m) = \int_{-LT_s}^{LT_s} p(t)p(t - m)dt\ \ a \tag{5.5}$$

$$v_i(nT) = p(-nT) * w_I(nT)\ a\ v_q(nT) = p(-nT) * w_q(nT).$$

Při tomto odvození se předpokládá dokonalá realizace symbolové synchronizace. Z tohoto důvodu je zde zařazen blok symbolové synchronizace a signál je převzorkován v poměru $n = kT_s/T = kN$ (*N* je celé číslo). Při skutečném řešení je blok symbolové synchronizace zařazen až za blokem synchronizace nosné vlny, která neobsahuje část s detekcí datových

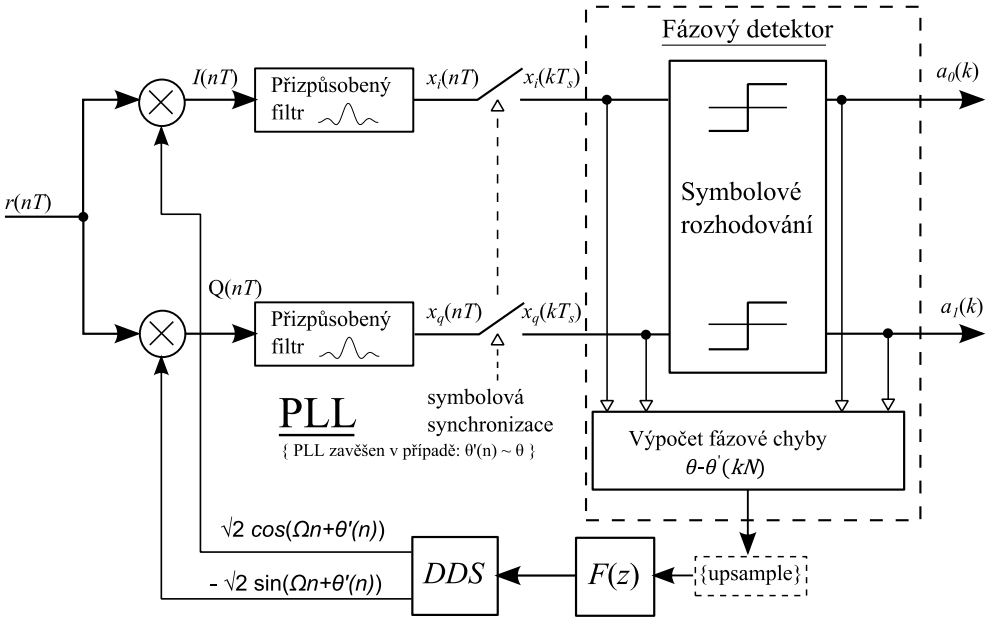
symbolů {1,0} a výstupem je pouze fázově synchronizovaný signál s určitou úrovní. Po provedené symbolové synchronizaci lze výraz z (8.4) přepsat na

$$\begin{aligned} x_i(kT_s) &= \left[a_0(k)\cos\left(\theta-\hat{\theta}(kN)\right) - a_1(k)\sin\left(\theta-\hat{\theta}(kN)\right)\right] \\ &\quad + v_I(kT_s)\ \{\,I\ složka\}, \\ x_q(kT_s) &= \left[a_0(k)\sin\left(\theta-\hat{\theta}(kN)\right) + a_1(k)\cos\left(\theta-\hat{\theta}(kN)\right)\right] \\ &\quad + v_Q(kT_s)\ \{\,Q\ složka\}. \end{aligned} \tag{5.6}$$

Po přepsání do maticového tvaru je evidentní rotační matice

$$\begin{bmatrix} x_i(kT_s) \\ x_q(kT_s) \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\left(\theta-\hat{\theta}(kN)\right) & -\sin\left(\theta-\hat{\theta}(kN)\right) \\ \sin\left(\theta-\hat{\theta}(kN)\right) & \cos\left(\theta-\hat{\theta}(kN)\right) \end{bmatrix}}_{Rotační\ matice\ pro\ komp.\ \theta-\hat{\theta}(kN)} \begin{bmatrix} a_0(k) \\ a_1(k) \end{bmatrix} + \begin{bmatrix} v_I(kT_s) \\ v_Q(kT_s) \end{bmatrix}, \tag{5.7}$$

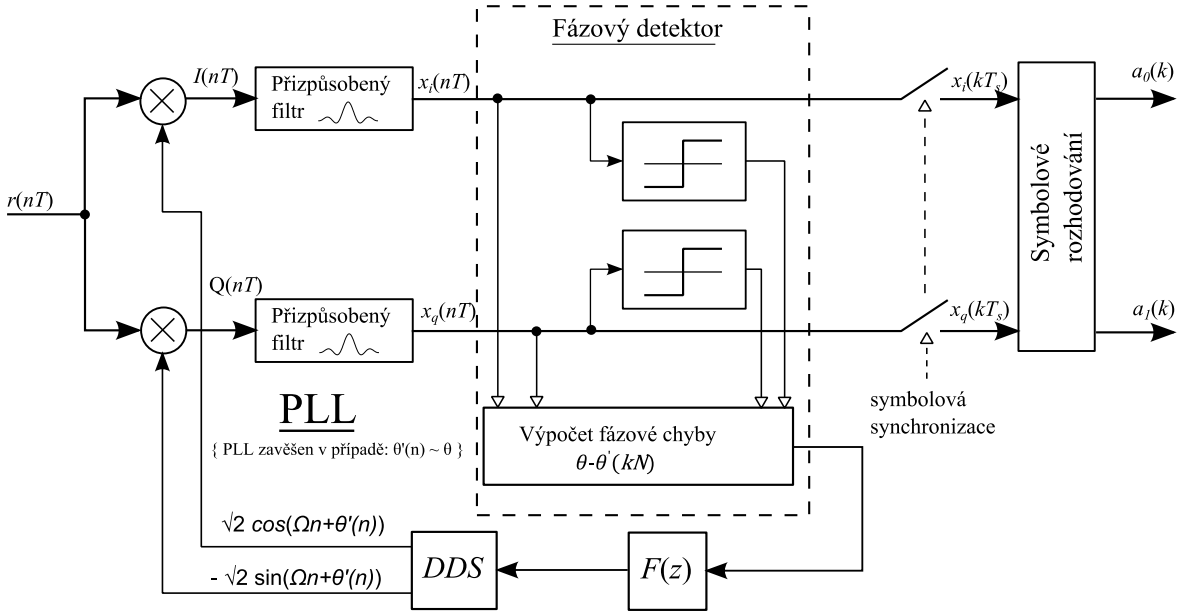
která slouží pro kompenzaci fázového ofsetu $\theta-\hat{\theta}(kN)$. Vektor $[x_I(kT_s),\ x_q(kT_s)]$ je otočená verze vektoru $[a_0(k),\ a_1(k)]$. Rotace o určitý úhel proti směru hodinových ručiček je zatím nekompenzovaná fázová chyba $\theta-\hat{\theta}(kN)$. Podobně jako v případě symbolové synchronizace je možné využít fázového závěsu pro výpočet chybového signálu. Fázový závěs bude zavěšen v případě, že chybový signál bude roven nule. Smyčka fázového závěsu bude obecně tvořena vlastním fázovým detektorem, filtrem fázového závěsu a blokem přímé digitální syntézy DDS. Obecné schéma je na obrázku č. 5.1.



Obr. 5.1 Ideální blokové schéma smyčky PLL pro synchronizaci fáze nosné vlny

Problémem naznačeného přístupu je, že se jedná o synchronizační systém s více hodinovými signály. Fázový detektor pracuje s 1 vzorkem / symbol, ale DDS pracuje s *N*

vzorky / symbol. Parametr N závisí na typu použitého detektoru pro symbolovou synchronizaci. V případě ZCTED (GTED) bude N rovno dvěma. Z tohoto důvodu je třeba chybový signál z fázového detektoru převzorkovat. Jak již bylo uvedeno, je vhodné blok symbolové synchronizace zařadit až za blok synchronizace fáze nosné vlny. Možné řešení je uvedeno na obrázku č. 5.2.



Obr. 5.2 Blokové schéma synchronizačního systému nosné vlny; symbolová synchronizace následuje až synchronizací nosné vlny

Další možností pro obnovu nosné vlny předpokládá využití volně běžícího oscilátoru (tzn. s pevně nastavenou frekvencí a fází) do základního pásma z mezifrekvenčního kmitočtu. Kompenzace fáze je provedena až za přizpůsobeným filtrem a symbolovou synchronizací v bloku nazvaném „Rotace fáze nosné vlny“. Přijatý pásmově omezený navzorkovaný signál $r(nT)$ je přeložen do základního pásma pomocí kvadraturních signálových složek $\sqrt{2}\cos(\Omega n)$ a $-\sqrt{2}\sin(\Omega n)$ generovaných pomocí volně běžícího oscilátoru, který je možné realizovat také jako DDS. Blokové schéma tohoto synchronizačního systému je zobrazeno na obrázku č. 5.3. Po provedení symbolové synchronizace budou mít signálové složky $[x_i(kT_s); x_q(kT_s)]$ podobu (zapsáno v maticové podobě)

$$\begin{bmatrix} x_i(kT_s) \\ x_q(kT_s) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a_0(k) \\ a_1(k) \end{bmatrix} + \begin{bmatrix} v_I(kT_s) \\ v_Q(kT_s) \end{bmatrix}. \tag{5.8}$$

Při odvození vztahu (5.6) vycházím z rovnic (5.3) až (5.7). Signálové složky $[x_i(kT_s); x_q(kT_s)]$ jsou dále natočeny podle odhadu fázového ofsetu $\hat{\theta}(k)$, který poskytuje blok *Rotace fáze nosné vlny*. Blok digitální syntézy DDS poskytuje odhad fázového ofsetu $\hat{\theta}(k)$

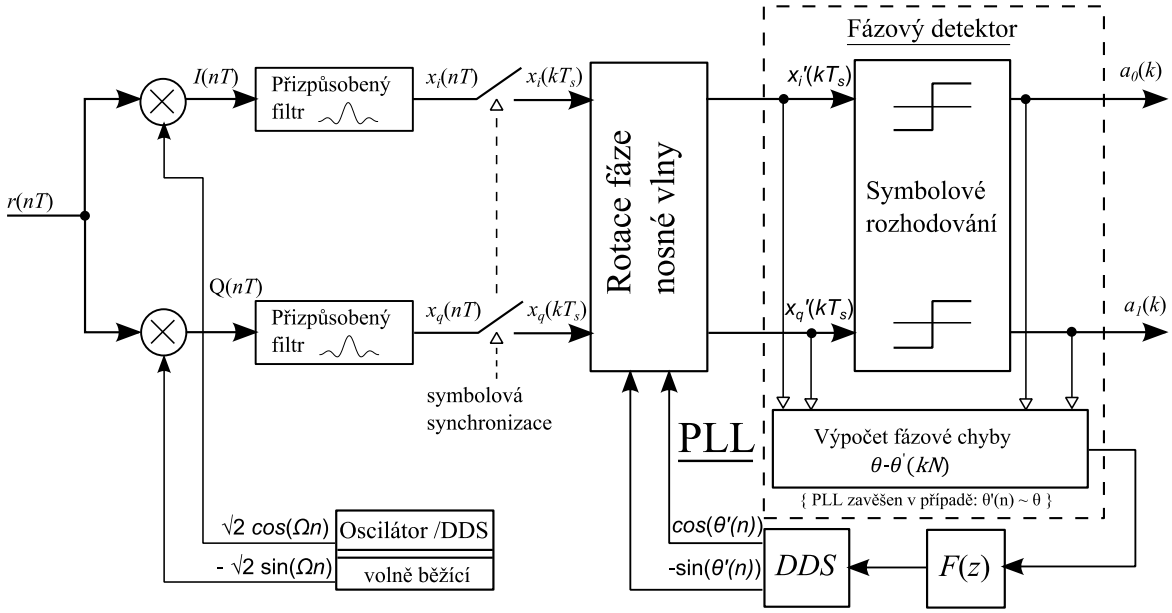
v podobě signálů $\cos(\hat{\theta}(k))$ a $-\sin(\hat{\theta}(k))$. Vektor $[x_i(kT_s); x_q(kT_s)]$ je natočen o $-\hat{\theta}(k)$ pro vytvoření rotovaného signálu $[x_i'(kT_s); x_q'(kT_s)]$

$$\begin{bmatrix} x_i'(kT_s) \\ x_q'(kT_s) \end{bmatrix} = \begin{bmatrix} \cos(\hat{\theta}(k)) & -\sin(\hat{\theta}(k)) \\ \sin(\hat{\theta}(k)) & \cos(\hat{\theta}(k)) \end{bmatrix} \begin{bmatrix} x_i(kT_s) \\ x_q(kT_s) \end{bmatrix} + \begin{bmatrix} v_I(kT_s) \\ v_Q(kT_s) \end{bmatrix}. \tag{5.9}$$

Po přepsání do tvaru s datovými symboly $[a_0(k), a_1(k)]$ získáváme vztah

$$\begin{aligned} \begin{bmatrix} x_i'(kT_s) \\ x_q'(kT_s) \end{bmatrix} &= \begin{bmatrix} \cos(\hat{\theta}(k)) & -\sin(\hat{\theta}(k)) \\ \sin(\hat{\theta}(k)) & \cos(\hat{\theta}(k)) \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a_0(k) \\ a_1(k) \end{bmatrix} \\ &+ \begin{bmatrix} \cos(\hat{\theta}(k)) & -\sin(\hat{\theta}(k)) \\ \sin(\hat{\theta}(k)) & \cos(\hat{\theta}(k)) \end{bmatrix} \begin{bmatrix} v_I(kT_s) \\ v_Q(kT_s) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta - \hat{\theta}(k)) & -\sin(\theta - \hat{\theta}(k)) \\ \sin(\theta - \hat{\theta}(k)) & \cos(\theta - \hat{\theta}(k)) \end{bmatrix} \begin{bmatrix} a_0(k) \\ a_1(k) \end{bmatrix} + \begin{bmatrix} v_I'(kT_s) \\ v_Q'(kT_s) \end{bmatrix}. \end{aligned} \tag{5.10}$$

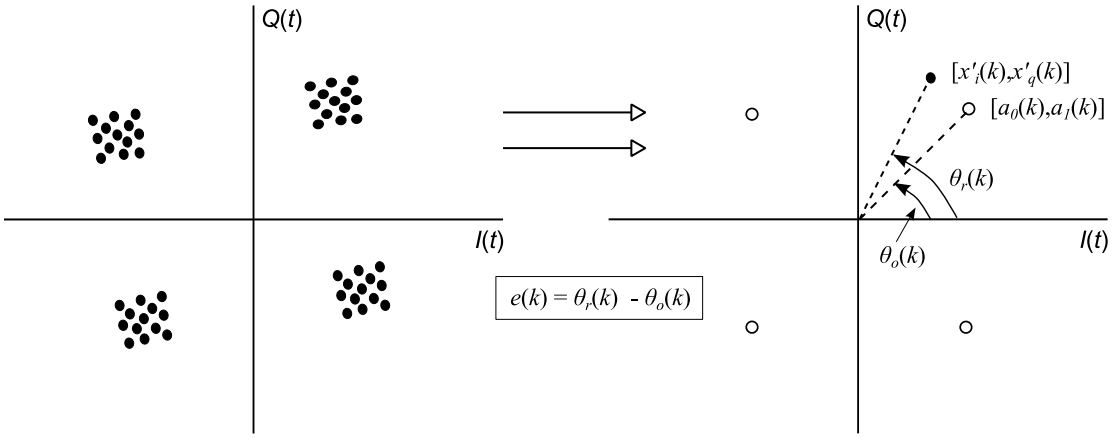
Odvozený vztah z (5.10) je ekvivalentní vůči vztahu z (5.7) s rozdílem, že odhad fáze $\hat{\theta}$ v (5.7) je vzorkován s frekvencí $1/T$ vzorků / symbol, ale v (5.10) s frekvencí $1/T_s$ vzorků / symbol. Z tohoto důvodu je synchronizační systém z obrázku č. 5.3 velice vhodný pro vlastní implementaci na FPGA nebo ASIC, neboť se jedná o čistě diskrétní přístup.



Obr. 5.3 Blokové schéma synchronizačního s blokem *Rotace fáze nosné vlny*; vlastní PLL synchronizace fáze nosné vlny je zařazen až za blokem symbolové synchronizace

5.2 Chybový detektor pro synchronizace fáze nosné vlny

Odvození chybového fázového detektoru bude opět provedeno pro přenosový systém s modulací QPSK, neboť je možné jednoduché rozšíření pro modulace typu *m*-QAM. Soufázová a kvadraturní složka signálu z výstupu přizpůsobeného filtru resp. po provedení symbolové synchronizace [$x_i(kT_s)$; $x_q(kT_s)$] jsou natočeny o $-\hat{\theta}(k)$ pro zarovnání signálové projekce [$x_i'(kT_s)$; $x_q'(kT_s)$] s konstelačními body v I/Q diagramu (4 body ±A, ±A v případě QPSK). To je přehledně ukázáno na obrázku č. 5.4.



Obr. 5.4 Geometrická interpretace fázové chyby pro QPSK

Z geometrické interpretace plyne, že fázovou chybu lze určit jako diferenci mezi body [$x_i'(kT_s)$; $x_q'(kT_s)$] a konstelačními body [$a_0(k)$; $a_1(k)$]. Předpokládám, že chybový detektor zná přenášené datové symboly. Fázový úhel $\theta_r(k)$ pro signálové složky [$x_i'(kT_s)$; $x_q'(kT_s)$] po provedené rotaci lze vyjádřit jako

$$\theta_r(k) = \tan^{-1}\left\{\frac{x'_q(kT_s)}{x'_i(kT_s)}\right\}, \tag{5.11}$$

resp. pro konstelační body [$a_0(k)$; $a_1(k)$]

$$\theta_o(k) = \tan^{-1}\left\{\frac{a_1(k)}{a_0(k)}\right\}. \tag{5.12}$$

Chybový signál z výstupu detektoru bude mít tvar

$$e(k) = \theta_r(k) - \theta_o(k) = \tan^{-1}\left\{\frac{x'_q(kT_s)}{x'_i(kT_s)}\right\} - \tan^{-1}\left\{\frac{a_1(k)}{a_0(k)}\right\}. \tag{5.13}$$

Chybový signál z (8.13) předpokládá znalost přenášených datových symbolů (trénovací sekvence). V případě, že trénovací sekvence není k dispozici, je možné použít detektor řízený rozhodovací úrovní. Detektor řízený úrovní lze vyjádřit jako

$$e(k) = \theta_r(k) - \theta_o(k) = \tan^{-1}\left\{\frac{x'_q(kT_s)}{x'_i(kT_s)}\right\} - \tan^{-1}\left\{\frac{\hat{a}_1(k)}{\hat{a}_0(k)}\right\}. \tag{5.14}$$

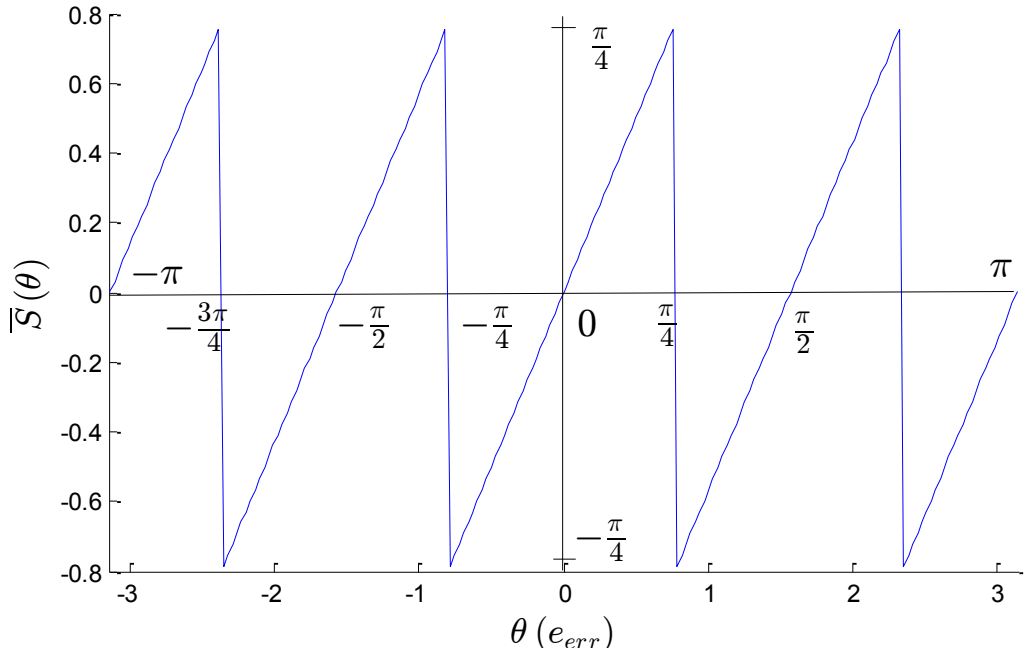
kde

$$\hat{a}_0(k) = A \times sgn\{x'_i(kT_s)\}\ a\ \hat{a}_1(k) = A \times sgn\{x'_q(kT_s)\}. \tag{5.15}$$

S-křivku pro uvedený detektor, lze vyjádřit podobným způsobem jako u detektorů pro symbolovou synchronizaci. Výstup detektoru závisí na fázové chybě, kterou dále budu označovat jako θ_{err} a datových symbolech. Závislost $e(k)$ na θ_{err} lze označit jako S-křivku pro fázový detektor a dle konvencí práce ji nazývám $S(\theta_{\text{err}})$

$$\begin{aligned} S(\theta_{err}, [a_0(k), a_1(k)]) &= \tan^{-1}\left\{\frac{x'_q(kT_s)}{x'_i(kT_s)}\right\} - \tan^{-1}\left\{\frac{a_1(k)}{a_0(k)}\right\} \\ &= \tan^{-1}\left\{\frac{a_0(k)\sin(\theta_{err}) + a_1(k)\cos(\theta_{err})}{a_0(k)\cos(\theta_{err}) - a_1(k)\sin(\theta_{err})}\right\} - \tan^{-1}\left\{\frac{a_1(k)}{a_0(k)}\right\}. \end{aligned} \tag{5.16}$$

kde za $[x_i'(kT_s); x_q'(kT_s)]$ bylo provedena substituce z (8.9). Vhodnější je pro konkrétní případ vyjádřit průměrovanou *S*-křivku označenou jako $\bar{S}(\theta_{err})$, kterou lze pro QPSK vyjádřit průměrováním přes čtyři uvažované symboly $[\{a_0(k); a_1(k)\} = \{\pm A, \pm A\}$ resp. $\{\pm 1, \pm 1\}]$. V případě úrovní řízeného detektoru je S-křivka zobrazena na obrázku č. 5.5.



Obr. 5.5 Simulace průměrované S-křivky pro lineární fázový chybový detektor řízený úrovní; modulace *QPSK*

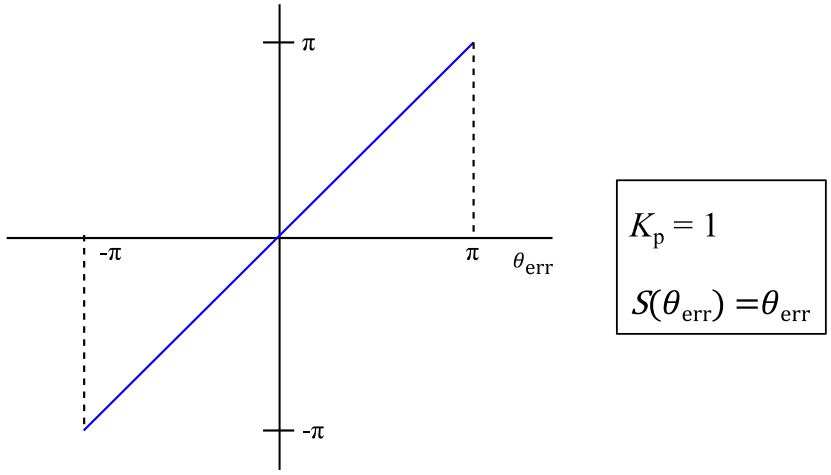
Směrnice uvedená *S*-křivky v bodě $\theta_{\text{err}} = 0$ je rovna jedné a z toho plyne, že proporcionální konstanta $K_p = 1$. *S*-křivka pro detektor řízený úrovní prochází nulovou hodnotou v bodech

$$\theta_{err} = \frac{-3\pi}{4}, \frac{-\pi}{2}, \frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi. \tag{5.17}$$

Fázový závěs bude zavěšen v případě $\theta_{err} = 0$. Z toho vyplývá, že zavěšení PLL nastane pro takové hodnoty θ_{err}, kde $S(\theta_{err})$ prochází nulou s kladnou směrnicí. Stabilní body pro zavěšení PLL tedy jsou

$$\theta_{err} = \frac{-\pi}{2}, 0, \frac{\pi}{2}, \pi. \tag{5.18}$$

Tato vlastnost způsobuje, že PLL pro synchronizaci fáze nosné vlny s modulací *QPSK* se může zavěsit se správnou fází, ±90° fázově posunut nebo 180° fázově posunut opět se správnou fází. Tento detektor má π/2 fázovou dvojznačnost a je nutné s touto vlastností při návrhu samotného synchronizačního systému počítat. Řešením může být například využití diferenciálního kódování. Pro úplnost ještě uvádím *S*-křivku pro případ známé datové posloupnosti na obrázku č. 5.6.



Obr. 5.6 Simulace průměrované S-křivky pro lineární fázový chybový detektor v případě známých datových symbolů; modulace *QPSK*

Problémem uvedeného detektoru je, že nutné počítat se dvěma výpočty funkce *arkus tangens* (ve 4 kvadrantech). Výpočet této funkce lze provést i v pevné řádové čárce například prostřednictvím algoritmu *CORDIC*, ale implementace na FPGA nebo ASIC bude velice neefektivní. Z tohoto důvodu je nutné provést modifikaci tohoto detektoru a využít metod maximální věrohodnosti. Detektor založený na metodě maximální věrohodnosti [25, 26] provádí fázový odhad $\hat{\theta}$, který minimalizuje energii fázové chyby

$\theta_{err} = \theta - \hat{\theta}$. Detektor popsaný pomocí rovnic (5.13) a (5.14) lze převést na detektor typu *ML* aplikací funkce *sinus* na aktuální fázovou chybu *e*(*k*).

$$\begin{aligned} sin\big(e(k)\big) &= sin\big(\theta_r(k) - \theta_o(k)\big) \\ &= \sin\theta_r(k)\cos\theta_o(k) \\ &\quad - \cos\theta_r(k)\sin\theta_o(k) \\ &= \frac{y'(kT_s)a_0(k) - x'(kT_s)a_1(k)}{\sqrt{x'^2(kT_s) + y'^2(kT_s)}\sqrt{a_0^2(kT_s) + a_1^2(kT_s)}} \end{aligned} \tag{5.19}$$

$$aplikace: sin(\tan^{-1}(x)) = \frac{x}{\sqrt{1+x^2}}\ a\ sin(X-Y) = sinXcosY - cosXsinY$$

Výraz v rovnici (5.19) obsahuje dělení, které představuje pomalou iterativní operaci. Dělenec v (5.19) je možné využít jako vlastní fázovou chybu a dělitel může být pohlcen v rámci proporcionální konstanty K_p. Chybu *e*(*k*) lze tedy vyjádřit jako

$$e(k) = y'(kT_s)a_0(k) - x'(kT_s)a_1(k). \tag{5.20}$$

Pro detektor řízený úrovní

$$e(k) = y'(kT_s)\hat{a}_0(k) - x'(kT_s)\hat{a}_1(k), \tag{5.21}$$

kde

$$\hat{a}_0(k) = A \times sgn\{x'_i(kT_s)\}\ a\ \hat{a}_1(k) = A \times sgn\big\{x'_q(kT_s)\big\}. \tag{5.22}$$

S-křivku pro ML detektor bude odvozena opět s využitím rovnice (5.9). Po dosazení do (5.20) získáváme

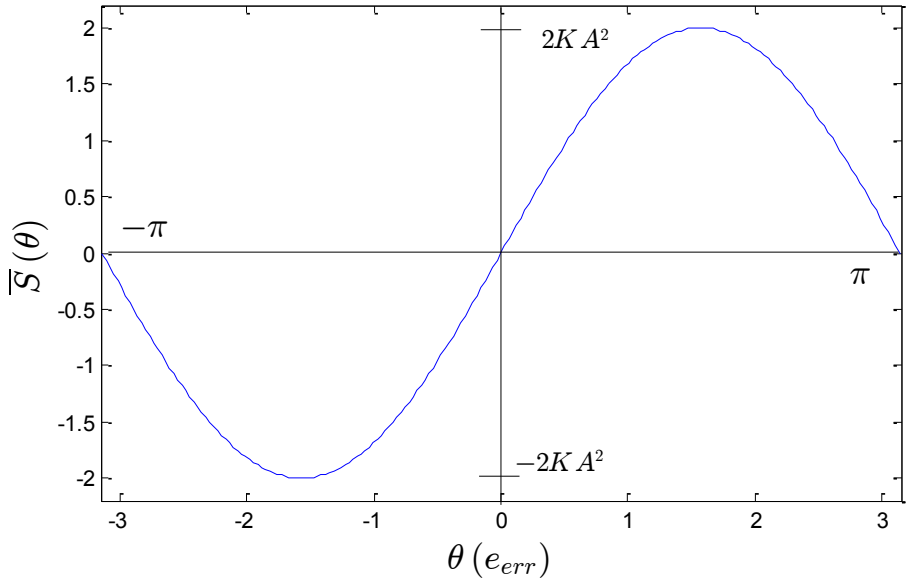
$$\begin{aligned} &S(\theta_{err}, [a_0(k), a_1(k)]) \\ &\quad = K_c[(a_0(k)\sin(\theta_{err}) + a_1(k)\cos(\theta_{err}))a_0(k) \\ &\quad - a_0(k)\cos(\theta_{err}) - a_1(k)\sin(\theta_{err})\ a_1(k)] \\ &\quad = K_c(a_0^2 + a_1^2)sin(\theta_{err}) \end{aligned} \tag{5.23}$$

Průměrováním přes čtyři uvažované symboly [$\{a_0(k); a_1(k)\} = \{\pm A, \pm A\}$] lze v případě známé datové posloupnosti S-křivku $\bar{S}(\theta_{err})$ vyjádřit jako

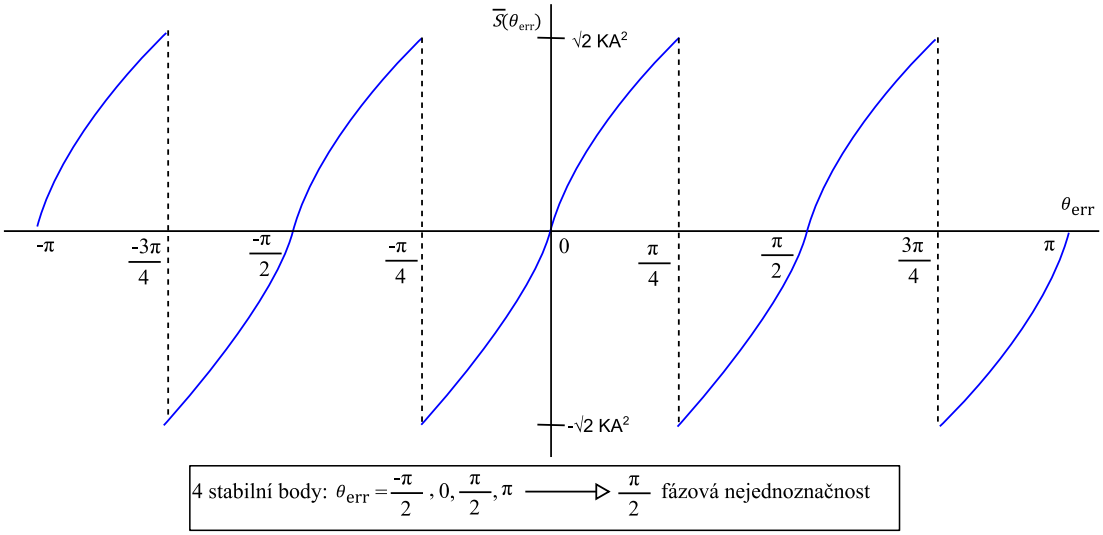
$$\bar{S}(\theta_{err}) = 2KA^2\ sin(\theta_{err}). \tag{5.24}$$

Ze vztahu (5.21) vyplývá, že proporcionální konstanta K_p bude mít tvar $K_p = 2KA^2$ ($\theta_{err} \rightarrow 0$; $\bar{S}(\theta_{err}) = 2KA^2\theta_{err}$). Konstanta K_p závisí na amplitudě přijatého signálu a je tedy jako v případě symbolové synchronizace pro správnou funkci zařadit blok automatického řízení

zisku AGC (K a A budou potom fixní). Tato vlastnost je především důležitá při vlastní implementaci na FPGA nebo ASIC. S-křivka v případě známých datových symbolů je uvedena na obrázku č. 5.7 a pro případ úrovní řízeného detektoru je S-křivka zobrazena na obrázku č. 5.8.

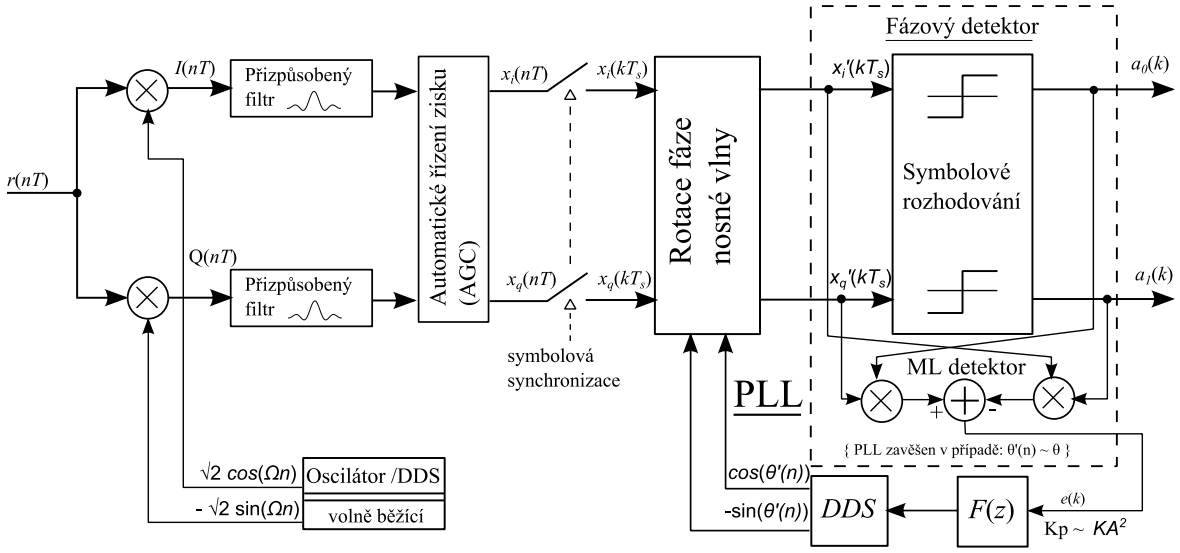


Obr. 5.7 Simulace průměrované S-křivky pro ML fázový chybový detektor v případě známých datových symbolů; modulace *QPSK*



Obr. 5.8 Simulace průměrované S-křivky pro ML fázový chybový detektor řízený úrovní; *QPSK*

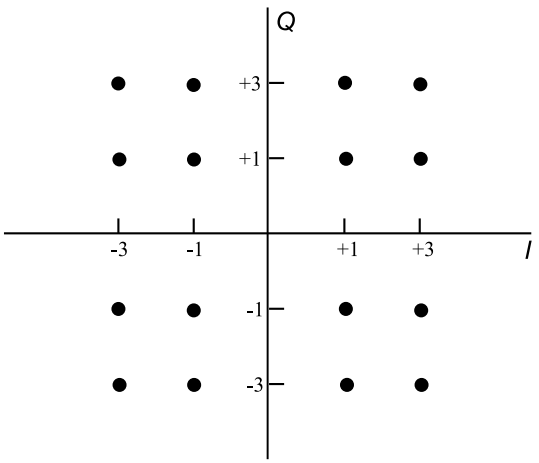
Celkové blokové schéma obsahující blok automatického řízení zisku AGC, symbolovou synchronizaci a blok synchronizace fáze nosné vlny s ML detektorem je uvedeno na obrázku č. 5.9.



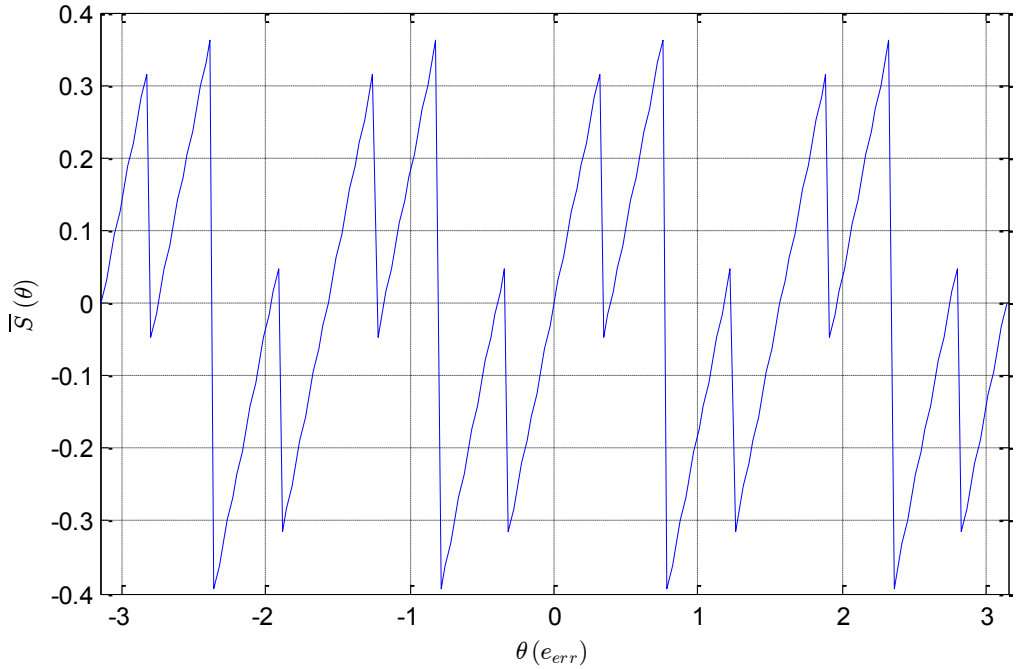
Obr. 5.9 Celkové blokové schéma synchronizačního systému s blokem *Rotace fáze nosné vlny*; je využito ML detektoru.

Protože modulace m-*QAM* představují zobecnění QPSK, je možné navržené PLL pro synchronizaci fáze nosné vlny využít i pro modulace typu m-*QAM*. Lze tedy přímo využít blokové schéma z obrázku č. 5.2 nebo 5.3. Vlastnosti *S*-křivky ovšem závisí na daném typu konstelačního diagramu. Simulace průměrované *S*-křivky s příslušným konstelačním diagramem je uvedena na obrázku č. 5.10 a 5.11. Z uvedené simulace vyplývá, že různé S-křivky budou mít tyto společné vlastnosti:

- *S*-křivka prochází nulou v bodě $\theta_{err} = 0$ vždy s kladnou směrnicí; $\theta_{err} = 0$ je vždy stabilní bod pro zavěšení PLL.
- Každá *S*-křivka bude mít více stabilních bodů pro zavěšení PLL dle typu konstelačního diagramu.



Obr. 5.10 Konstelační diagram pro dále uvedenou *S*-křivku s 16-QAM



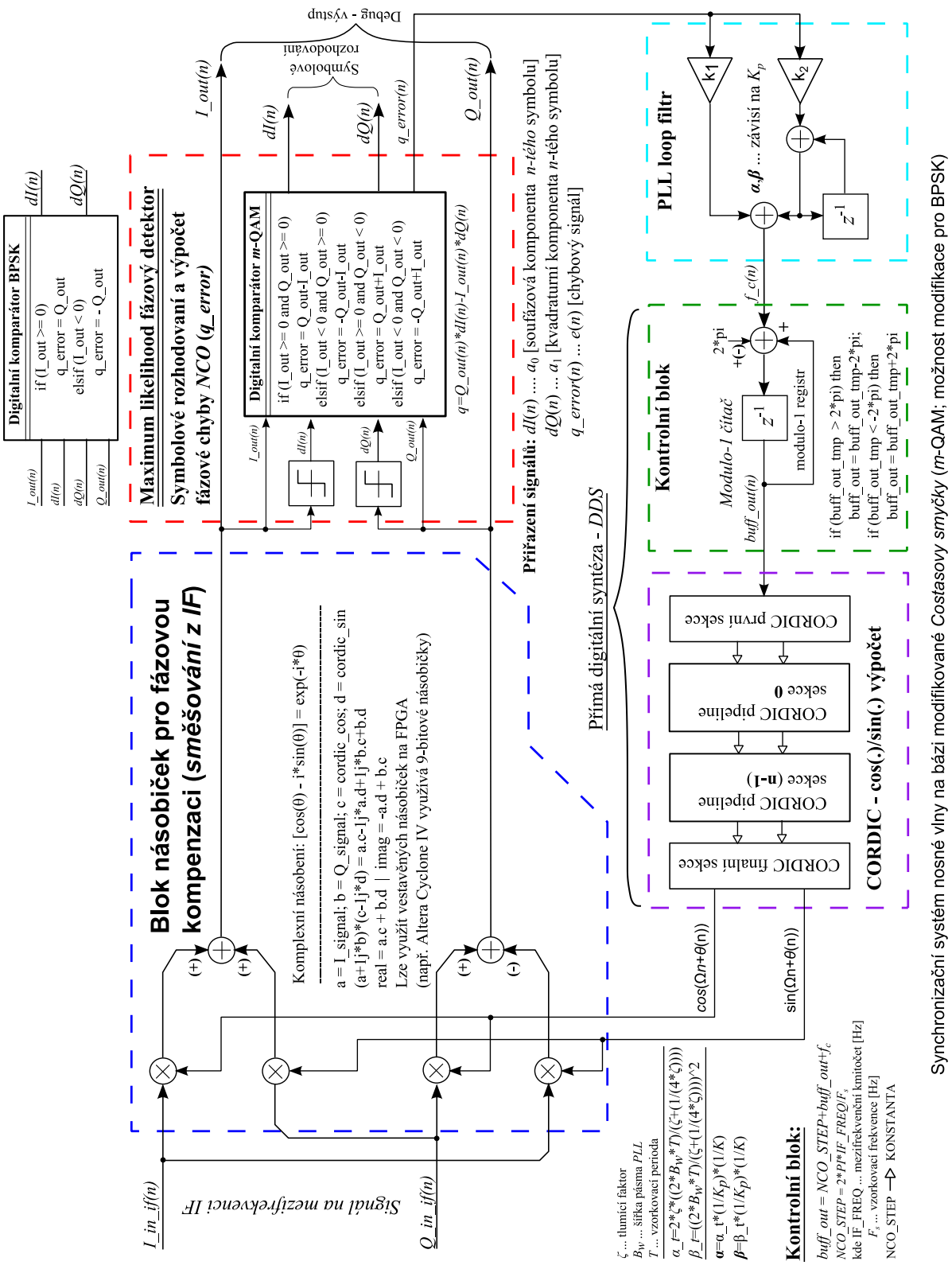
Obr. 5.11 Simulace průměrované S-křivky pro *ML* fázový chybový detektor řízený úrovní; modulace 16-*QAM*

5.3 Modely synchronizace fáze nosné vlny vhodné pro implementaci na hradlovém poli FPGA

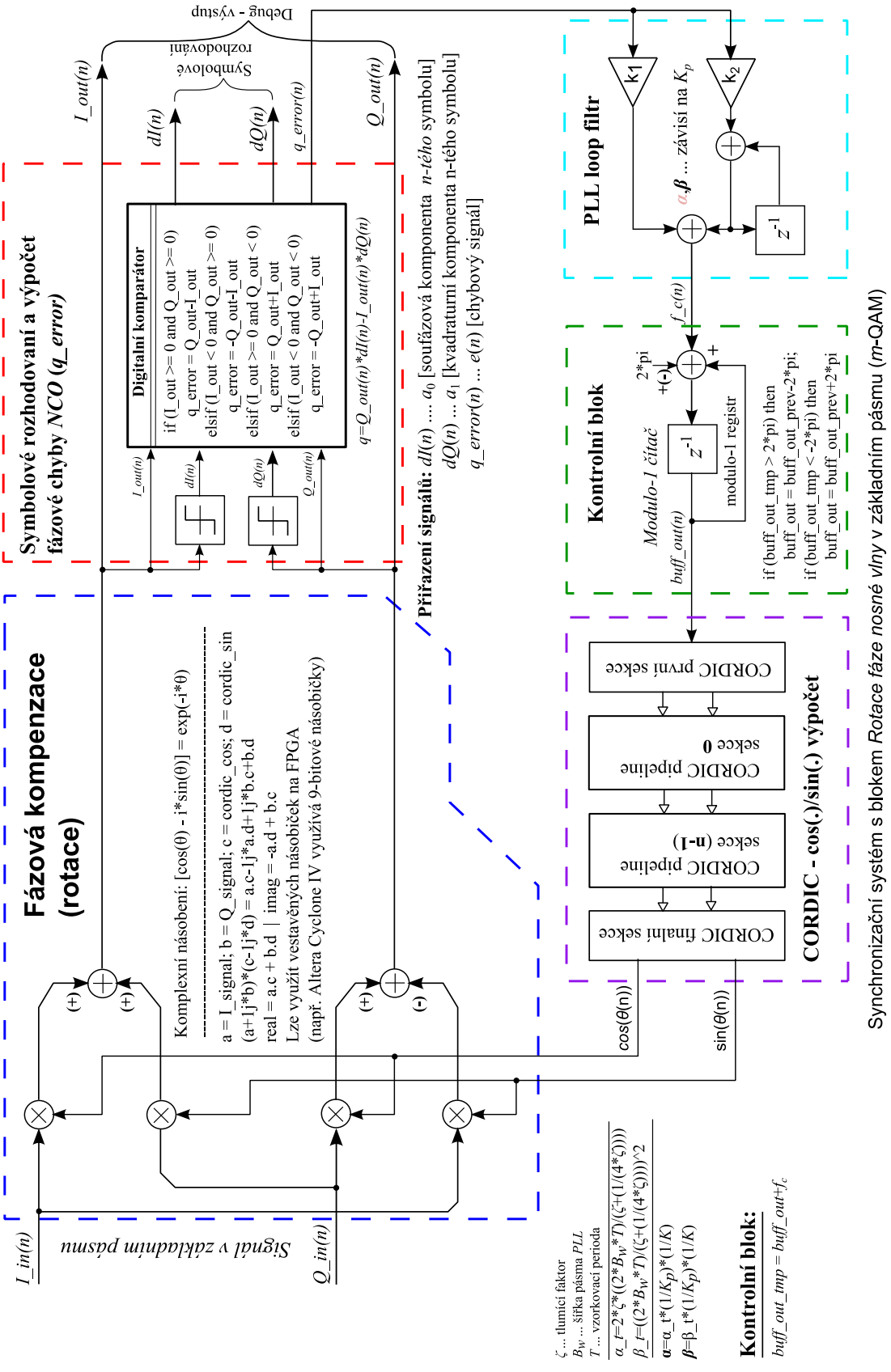
V této kapitole představuji navržené modely pro synchronizaci fáze nosné vlny, které využívají *ML* detektor (řízený úrovní - DD). Byly vytvořeny celkem dva modely. První model je postaven na modifikované *Costasově* smyčce a je určen pro QPSK nebo *m*-QAM. Jednoduchou modifikací fázového detektoru lze uvedený koncept použít i pro BPSK. Druhý navržený model využívá blok *Rotace fáze nosné vlny* po přeložení signálu do základního pásma. Tento model je určen především pro QPSK resp. *m*-QAM modulace. Navržené modely jsou určeny pro efektivní implementaci na hradlovém poli FPGA v pevné řádové čárce. Modely se skládají z následujících částí:

1. Blok násobiček pro kompenzaci ofsetu fáze nosné vlny
2. Fázový detektor postavený na metodě maximální věrohodnosti (*ML*)
3. *PLL* loop filtr
4. Blok přímé digitální syntézy – *DDS* (Kontrolní blok + CORDIC)

Navržené modely jsou zobrazeny na obrázku č. 5.12 resp. 5.13.



Obr. 5.12 Synchronizační systém nosné vlny na bázi modifikované Costasovy smyčky (QPSK resp. *m*-QAM; naznačena možnost modifikace pro BPSK)



Obr. 5.13 Synchronizační systém nosné vlny s blokem *Rotace fáze nosné vlny* v základním pásmu (QPSK resp. *m*-QAM)

Blok násobiček slouží pro kompenzaci fáze nosné vlny a přeložení signálu z mezifrekvenčního kmitočtu v případě konceptu modifikované Costasovy smyčky. Pro komplexní násobení je nutno použít minimálně čtyř násobiček. Celkové množství ale závisí na konkrétní implementaci resp. využité šířce slova. Například FPGA *Altera Cyclone IV* využívá bloky 9 bitových integrovaných násobiček. Fázový detektor využívá pro vytvoření chybového signálu výraz z rovnice (5.20), který je efektivně navržen jako digitální komparátor. V případě modifikovaného konceptu Costasovy smyčky je možné zaměnit blok digitálního komparátoru QPSK resp. *m*-QAM za blok komparátoru pro BPSK. V tomto případě se jedná o standardní Costasovu smyčku, jejíž podrobný rozbor byl proveden v kapitole č. 2. Parametry chybového signálu *e(k)* jsou dále modifikovány pomocí filtru fázového závěsu, který je stejně jako v případě symbolové synchronizace řešen jako proporcionálně integrační filtr *IIR*. Proporcionální konstantu K_p musí být opět vypočtena pro zajištění správné funkce použitého detektoru s využitím *S*-křivky. Z rovnice (5.24) vyplývá, že $K_p = 2KA^2$. Detektor řízený úrovní lze zjednodušit a modifikovat výraz (5.22) do podoby

$$\hat{a}_0(k) = sgn\{x'_i(kT_s)\}\ a\ \hat{a}_1(k) = sgn\{x'_q(kT_s)\}. \tag{5.25}$$

Dojde pouze ke změně měřítka a proporcionální konstanta bude mít tvar $K_p = 2KA$.

Blok přímé digitální syntézy *DDS* je složen ze dvou částí. První část pojmenovaná jako *Kontrolní blok* implementuje *modulo*-1 čítač, který provádí akumulaci fázové chyby po průchodu PLL loop filtrem (signál ve schématech označený jako *f_c*). V případě konceptu modifikované *Costasovy smyčky* bude mít výstupní signál *buff_out*, který je argumentem funkcí *cos*(*buff_out*) resp. *sin*(*buff_out*), podobu

$$buff\_out = NCO\_STEP + buff\_out + f\_c, \tag{5.26}$$

kde *NCO_STEP*

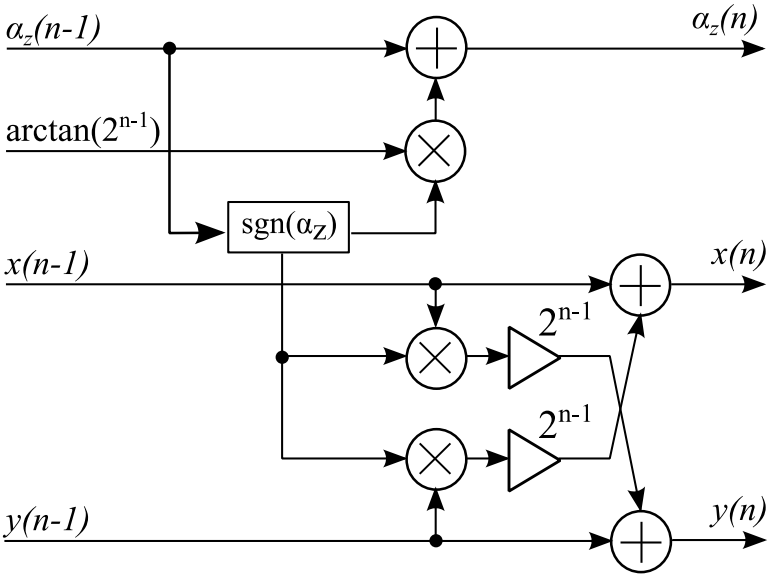
$$NCO\_STEP = 2\pi \frac{f_{IF}}{F_s}, \tag{5.27}$$

a F_s je vzorkovací frekvence [Hz]; f_{IF} je mezifrekvenční kmitočet [Hz]. V případě, že dojde k zavěšení PLL, fázový chybový signál bude nabývat přibližně nulové hodnoty. Záměrně uvádím přibližně, neboť v reálném přenosovém prostředí dochází k neustálé kompenzaci fázové chyby. Navržené modely využívají proporcionálně integračního členu a jedná se tedy o systém PLL druhého řádu. Systém druhého řádu umožňuje kompenzaci

konstantního frekvenčního ofsetu (dáno šířkou pásma *PLL*). Dynamický frekvenční ofset je možné dále kompenzovat pomocí systému třetího řádu. Tento speciální případ nastane například při satelitní komunikaci, kdy je potřeba v čase kompenzovat Dopplerovský posuv (není konstantní v čase, ale je lineární v čase). Navržené modely jsou modulární a mohou být upraveny i pro tuto nebo jinou speciální aplikaci. Druhý model s blokem *Rotace fáze nosné vlny* provádí fázovou korekci složek signálu v základním pásmu a proto výstupní signál *buff_out* má podobu

$$buff\_out = buff\_out + f\_c. \tag{5.28}$$

Druhá část bloku *DDS* zabezpečuje vlastní výpočet funkcí *cos*(.) resp. *sin*(.) s argumentem daným signálem *buff_out*. Pro implementaci v pevné řádové čárce na *FPGA* je využito algoritmu *CORDIC*. Algoritmus *CORDIC* využívá matematických a logických operací jako je bitový posun, sčítání, odečítání a porovnání, které je možné poměrně jednoduše implementovat v digitální technice. Vestavěné násobičky algoritmus nevyužívá, výjimku představuje pouze finální sekce algoritmu, kdy je pro změnu měřítka vhodné použít jednu vestavěnou násobičku. To potom celkově zjednodušuje celou strukturu výpočtu. Pro optimální výkon je nutno použít zřetězeného zpracování, které rozdělí výpočet do *N* sekcí. Počet sekcí závisí na požadované přesnosti. Struktura jedné sekce zřetězeného zpracování algoritmu CORDIC je uvedena na obrázku č. 5.14. Hodnoty funkce *arctang*(2^{n-1}) jsou vypočteny dopředu a uloženy ve vyhledávací tabulce (*look-up table*).



Obr. 5.14 Struktura sekce zřetězeného zpracování algoritmu CORDIC

6 Simulace navržených synchronizačních struktur

V této části jsou ukázány výsledky ze simulací provedených pro navržené synchronizační struktury v prostředí programu Matlab. Jedná se o kompletní simulaci přenosového řetězce s využitím navržených struktur symbolové synchronizace a synchronizace nosné vlny (resp. její fáze s detektorem ML). Je využito signálu *QPSK* z důvodu jednoduchého rozšíření pro modulace typu *m-QAM*. První kapitola se zabývá simulací s využitím struktury symbolové synchronizace s detektorem MLTED s interpolačním filtrem. Druhá kapitola se zabývá strukturami s detektorem *ZCTED* a *GTED* s interpolačním filtrem. Je dále doplněna o simulaci pro symbolovou synchronizaci s polyfázovým filtrem pro *ZCTED* a *GTED*. Simulaci v plovoucí řádové čárce je možné rozšířit o simulaci v pevné řádové čárce pro rychlejší převod modelů do jazyka VHDL a optimální syntézu na FPGA.

6.1 Simulace přenosového řetězce s využitím detektoru MLTED pro symbolovou synchronizaci

Simulace ve zpětnovazební smyčce byla provedena pro detektor postavený na metodách maximální věrohodnosti (detektor MLTED) a tvoří společně s blokem synchronizace nosné vlny základ simulovaného přenosového řetězce. Byly vytvořeny celkem dva modely. V prvním modelu je blok symbolové synchronizace zařazen před blokem synchronizace fáze nosné vlny. Tento model se skládá ze signálového *QPSK* generátoru, kanálu s bílým šumem (AWGN kanál), přizpůsobeného filtru *SRRC*, digitálního směšovače s přímou digitální syntézou (DDS - *Direct Digital Synthesis*), bloku automatického řízení zisku (AGC), zmíněného synchronizačního systému s detektorem MLTED a bloku synchronizace fáze nosné vlny.

Signálový generátor *QPSK* produkuje 16 vzorků / symbol a využitím diferenciálního kódování. Na obou stranách přenosového řetězce je použit stejný přizpůsobený *SRRC FIR* filtr s „roll-off“ faktorem 0.5. Multiplikativní skrambler je použit pro odstranění shluku stejných symbolů. Kanál s bílým šumem je možné rozšířit o simulaci variabilního fázové, frekvenčního ofsetu a zlomkového časového zpoždění (proces převzorkování ve zlomkovém poměru). DDS je také možné simulovat v pevné řádové čárce s využitím

algoritmu CORDIC. Blok automatického řízení zisku je postaven také na fázovém závěsu a je zde zařazen z důvodu nezbytnosti jeho nasazení v reálném přijímači.

Model symbolové synchronizace s detektorem MLTED je plně postaven na schématu z obrázku č. 4.7. Je využit číslicový diferenciátor, který provádí výpočet první diference mezi po sobě jdoucími vzorky. V časové oblasti je popsán následujícím způsobem

$$y_{first\_diff}(n) = x(n) - x(n-1). \tag{6.1}$$

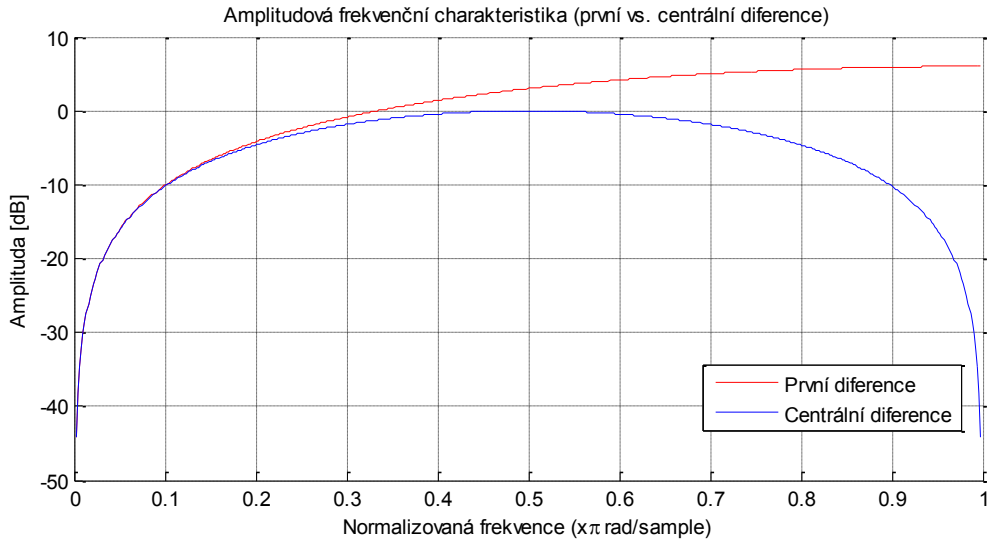
Po převedení do podoby filtru získáváme jednoduchý FIR filtr s koeficienty [1 -1]. Tento typ diferenciátoru poměrně přesně aproximuje ideální diferenci $|H_{ideal}(\omega)| = \omega$, ale může v určitých aplikacích zesilovat vysokofrekvenční šum. V této aplikaci se diferenciátor nachází až za přizpůsobeným filtrem (typu dolní propust) a z tohoto důvodu je možné tento velmi jednoduchý diferenciátor s výhodou použít pro symbolovou synchronizaci s detektorem typu MLTED. V oblasti DSP se hojně využívá diferenciátor založený na centrální diferenci, která je popsána rovnicí

$$y_{central\_diff}(n) = [x(n) - x(n-2)]/2. \tag{6.2}$$

Vytvořený filtr bude mít potom koeficienty [0.5 0 -0.5]. Nevýhodou tzv. tří bodové diference je, že pracuje v lineární oblasti pouze do $f_s/8$ a na vyšších kmitočtech je ideální diference poměrně špatně aproximována. Z tohoto důvodu doporučuji využít metodu první diference. Srovnání amplitudové frekvenční charakteristiky pro první a centrální diferenci je uvedeno na obrázku č. 6.1.

Dále je využito lineárního interpolátoru, neboť synchronizační systém pracuje se 16 vzorky / symbol. Simulace ukázaly, že 8 vzorků / symbol je minimální počet pro aplikaci lineárního interpolátoru. Chybový signál $e(k)$ je filtrován za pomoci proporcionálně integračního členu jednou za symbolovou periodu. Simulace potvrdily, že konstanta K_p (0.235) pro $E_b/N_0 = 10dB$ je použitelná v širokém rozsahu hodnot *SNR* s minimálním dopadem na celkové vlastnosti fázového závěsu. Výsledky simulace jsou uvedeny pro následující parametry PLL smyčky symbolové synchronizace:

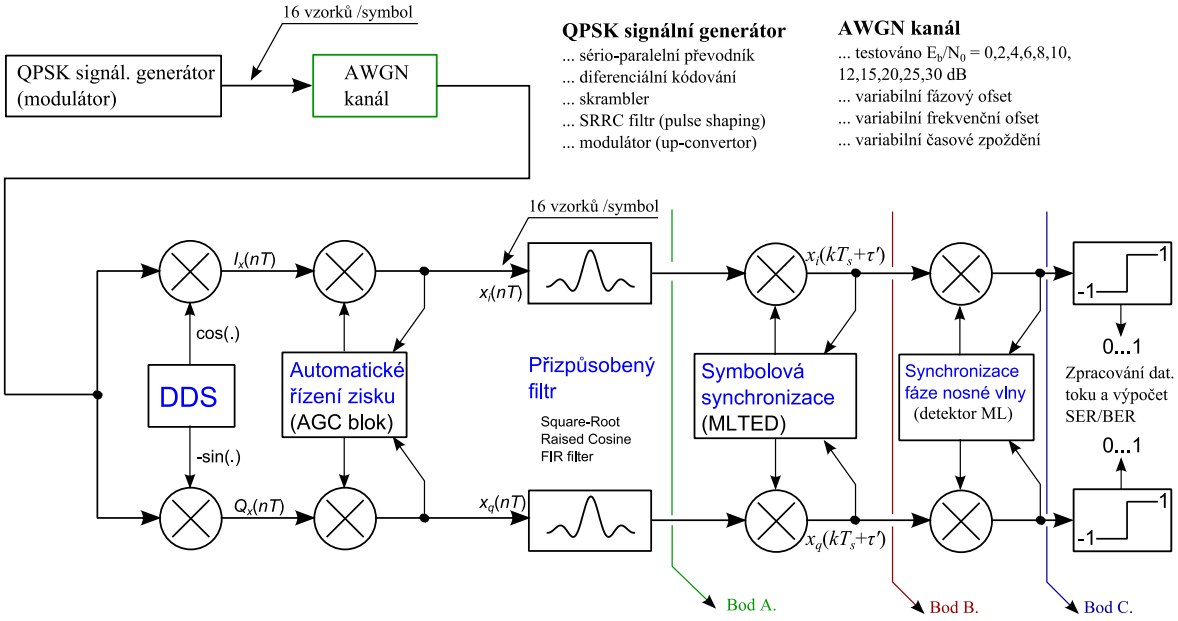
- $B_{PLL}T_s$ = 0.04, kde B_{PLL} je šířka pásma PLL pro symbolovou synchronizaci
- Přenosová funkce filtru PLL je navržena s konstantami K_1(0.00354) a K_2 (1.476e-6), tlumící faktor ζ=1/√2 (platí pro symbolovou synchronizaci).



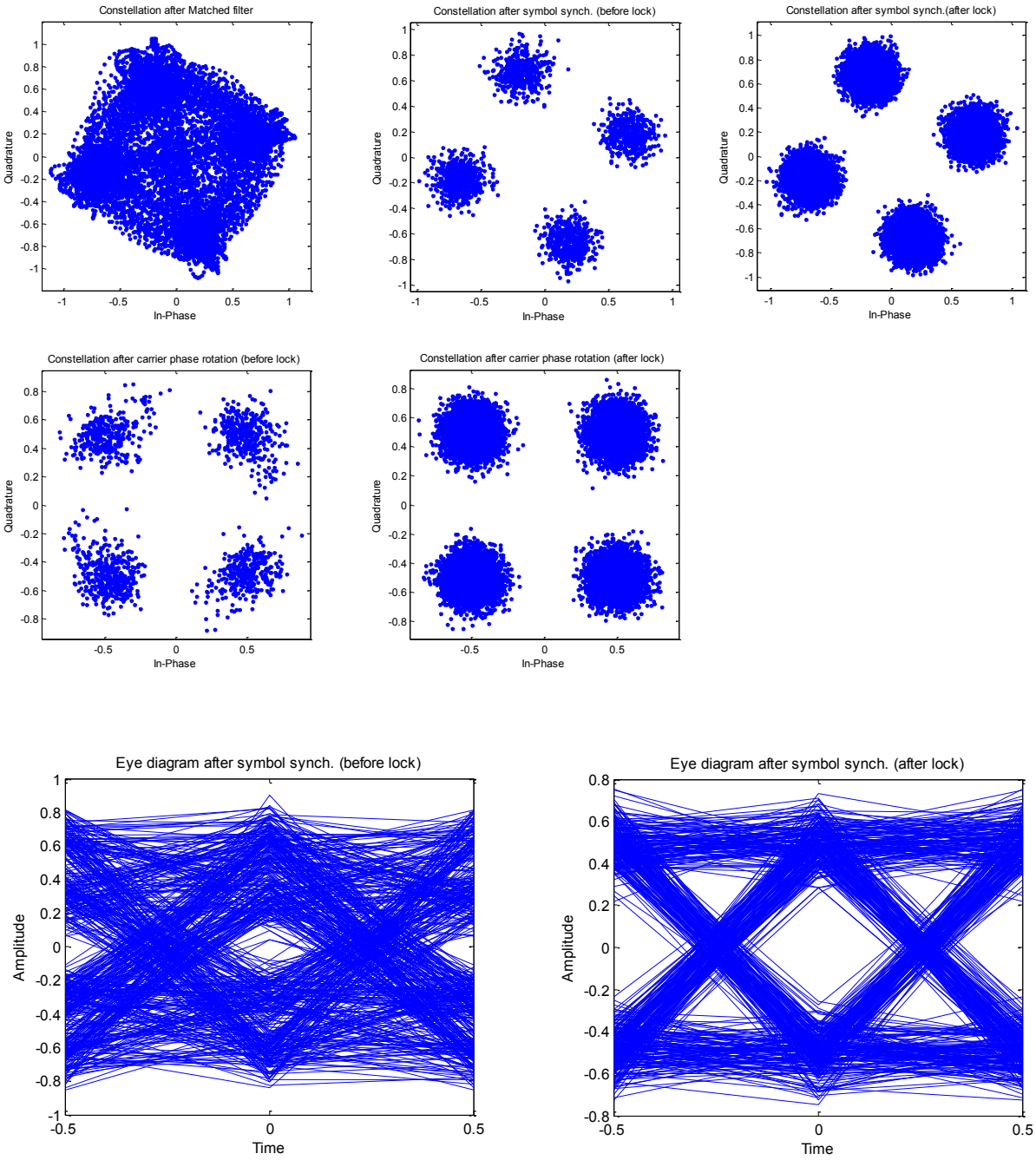
Obr. 6.1 Amplitudové frekvenční charakteristiky pro první a centrální diferenci

Protože je blok synchronizace fáze nosné vlny zařazen až za blokem symbolové synchronizace, je dle výsledků simulací vhodné volit pro blok synchronizace nosné vlny s detektorem ML šířku pásma PLL zhruba o třetinu menší než v případě symbolové. Simulační blokové schéma je zobrazeno na obrázku č. 6.2 a parametry PLL smyčky pro synchronizaci fáze nosné vlny jsou následující:

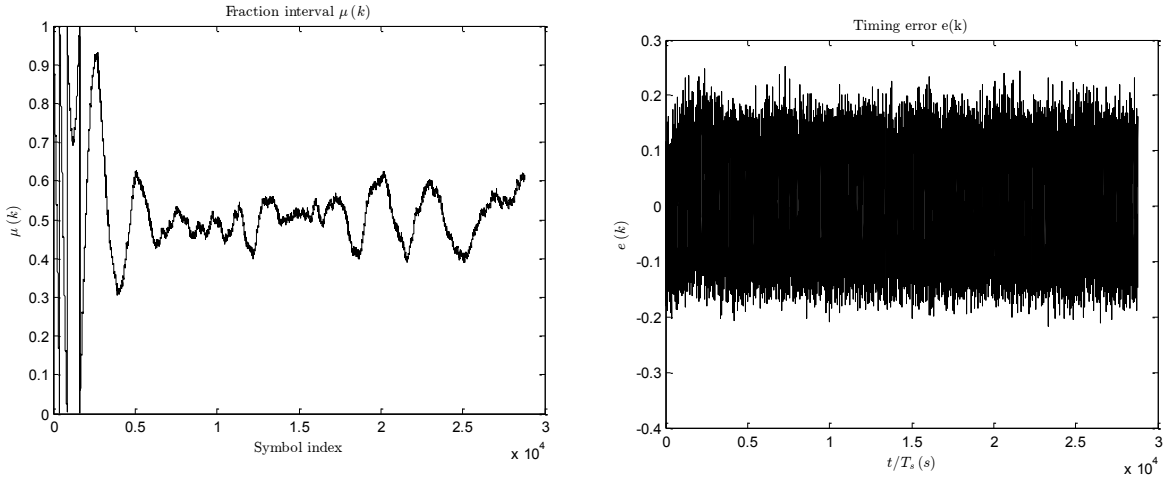
- $B_{PLL}T_s = 0.03$, kde B_{PLL} je šířka pásma PLL pro synchronizaci fáze nosné vlny
- Přenosová funkce filtru *PLL* je navržena s konstantami α(0.005) a β(1.25e-5), tlumící faktor ζ=1/√2 (platí pro synchronizaci fáze nosné vlny).



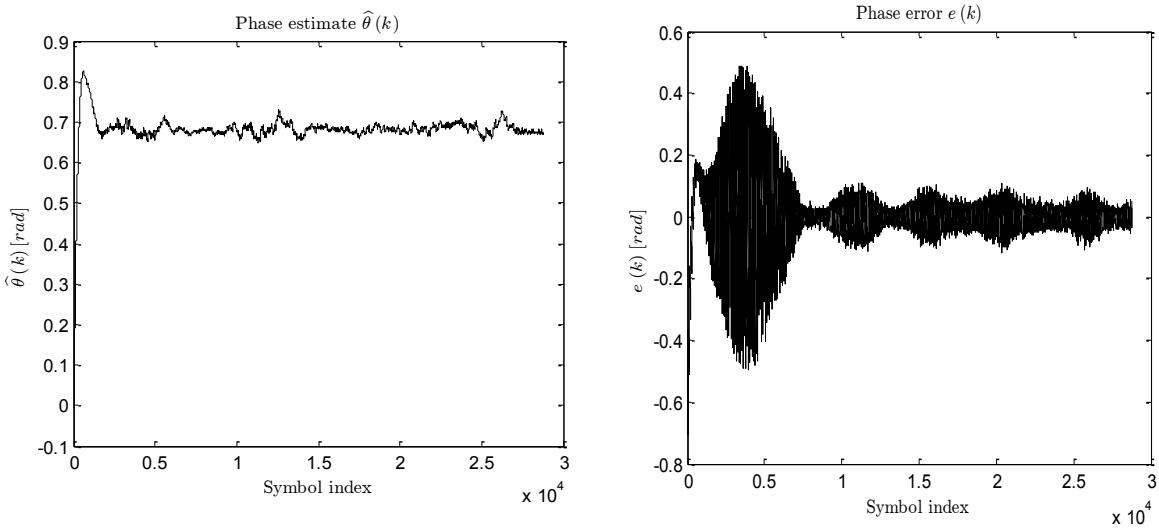
Obr. 6.2 Simulační blokové schéma pro symbolovou synchronizaci s detektorem MLTED



Obr. 6.3 Konstelační a digramy oka na straně přijímače pro E_b/N_0=8 dB a fázový ofset φ=30°. Detektor je typu MLTED. Popis z levého horního rohu dolů. Konstelační digram za přizpůsobeným filtrem (bod A.); konstelační diagram za sym. synchronizací – před zavěšením pro 50 – 2500 symbolů (bod B.); konstelační diagram za sym. synchronizací – po zavěšení pro 3000 – 20000 symbolů; konstelační diagram po synchronizaci nosné vlny – před zavěšením pro 50 – 2500 symbolů (bod C.); konstelační diagram po synchronizaci nosné vlny – po zavěšení pro 3000 – 20000 symbolů; diagram oka po symbolové synchronizaci - před zavěšením; diagram oka po symbolové synchronizaci - po zavěšení (stejný počet symbolů jako u konstelačních diagramů, zobrazena pouze soufázová složka).



Obr. 6.4 Grafy pro zlomkový interval $\mu(k)$ a chybový signál $e(k)$ pro E_b/N_0=8 dB (MLTED). Z průběhů je patrné, že k zavěšení PLL došlo zhruba po 2500 symbolových periodách.



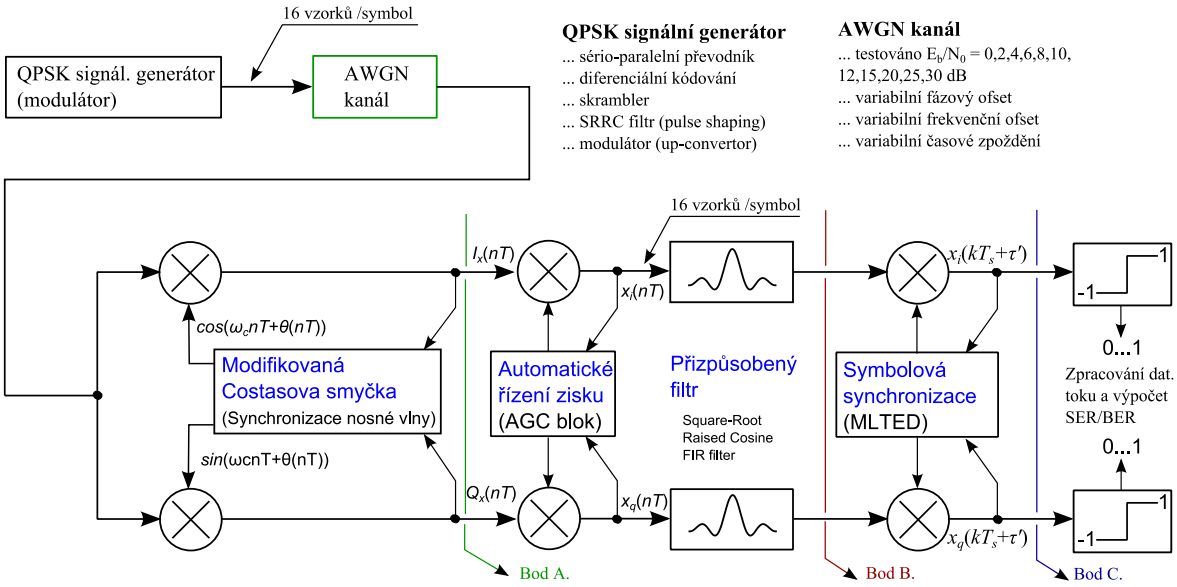
Obr. 6.5 Fázový odhad $\hat{\theta}(k)$ a fázová chyba $e(k)$ pro synchronizaci fáze nosné vlny pro E_b/N_0=8 dB. Simulovaný fázový ofset byl 0.7 radiánu (~40°). Z průběhů je zřejmé, že k zavěšení PLL došlo zhruba po 3500 symbolových periodách. Z grafu pro odhad $\hat{\theta}(k)$ dále plyne, že po zavěšení PLL $\hat{\theta}(k) \approx 0.7\,rad$, což odpovídá nastavenému fázovému ofsetu a lze tím ověřit správnou funkci PLL. V grafu $\hat{\theta}(k)$ je také zřejmý překmit jako odezva na fázový skok a smyčka představuje lehce podtlumený systém druhého řádu.

Simulační blokové schéma s modifikovanou Costasovo smyčkou (tzn. synchronizace nosné vlny je provedena před symbolovou synchronizací) je zobrazeno na obrázku č. 6.6. a parametry PLL smyčky pro synchronizaci fáze nosné vlny jsou následující:

- $B_{PLL}T_s = 0.04$, kde B_{PLL} je šířka pásma PLL pro synchronizaci fáze nosné vlny
- Přenosová funkce filtru *PLL* je navržena s konstantami α(0.0067) a β(2.23-5), tlumící faktor $\zeta=1/\sqrt{2}$ (platí pro synchronizaci fáze nosné vlny).

Parametry pro symbolovou synchronizaci jsou v tomto případě

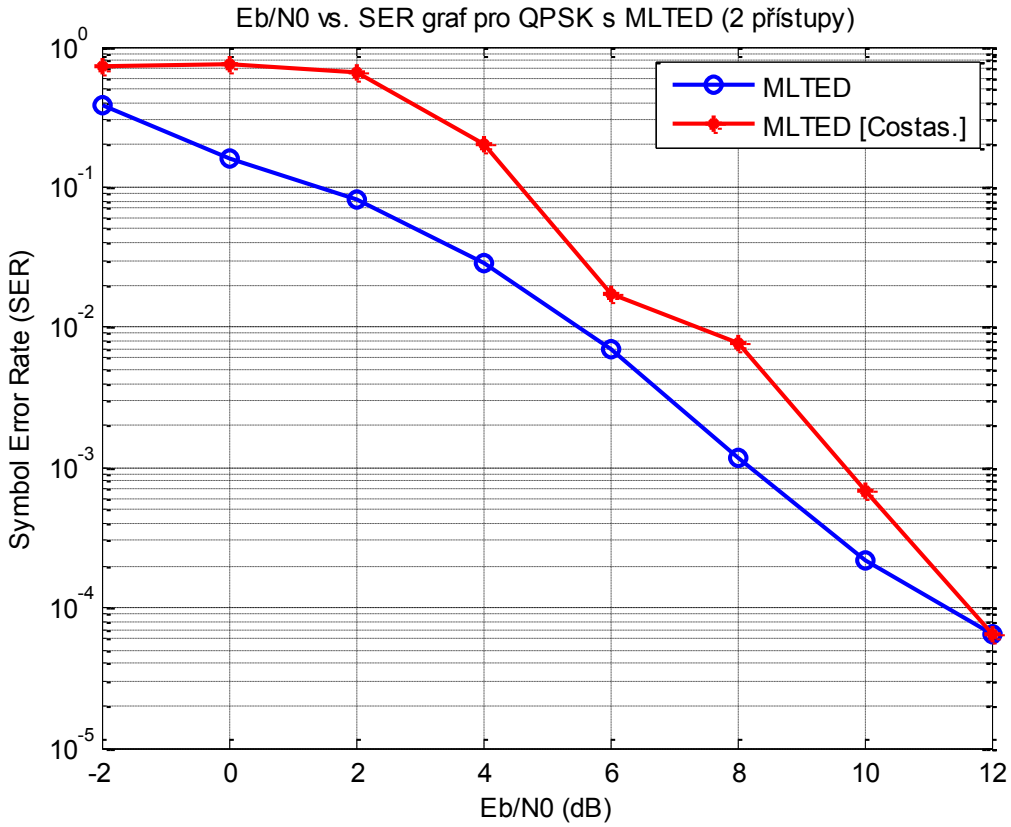
- $B_{PLL}T_s = 0.03$, kde B_{PLL} je šířka pásma PLL pro symbolovou synchronizaci
- Přenosová funkce filtru PLL je navržena s konstantami K_1(0.00132) a K_2 (2.07e-7), tlumící faktor $\zeta=1/\sqrt{2}$ (platí pro symbolovou synchronizaci).



Obr. 6.6 Simulační blokové schéma pro symbolovou synchronizaci s detektorem MLTED (využito modifikované Costasovy smyčka před symbolovou synchronizací)

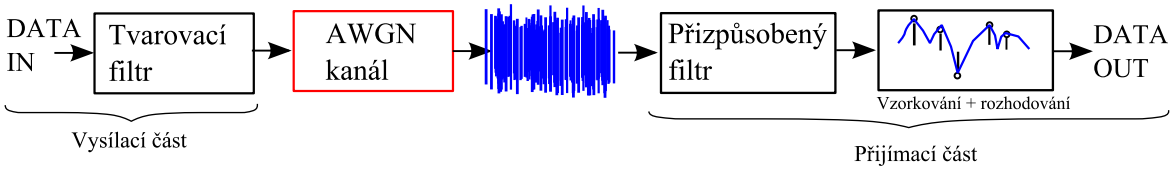
Vyhodnocení chybovosti symbolů (*Symbol Error Rate - SER*) bylo provedeno pro porovnání vlastností představených modelů s detektorem MLTED. Z výsledků simulací je patrné, že provedení symbolové synchronizace před synchronizací fáze nosné vlny je výhodné v případě malého odstupu *SNR* resp. $E_b/N_0 < 8$. Pro $E_b/N_0 > 8$ jsou výsledky simulací srovnatelné, stále je ale výhodnější využít prvního konceptu. Srovnání je uvedeno na obrázku č. 6.7. Vysvětlením je zřejmě zařazení přizpůsobeného filtru, který je nutné

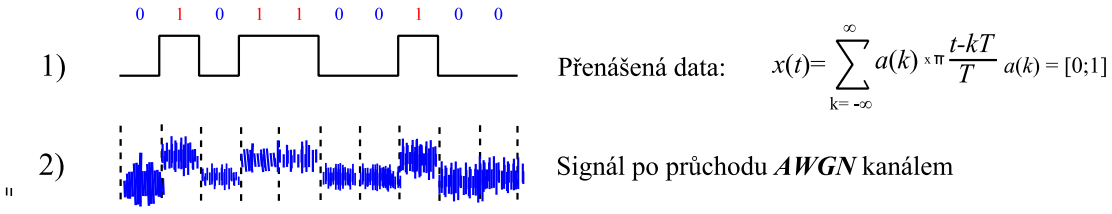
v případě druhého konceptu zařadit až za blokem synchronizace fáze nosné vlny.



Obr. 6.7 Vyhodnocení *SER* vs. E_b/N_0 pro *MLTED* a *MLTED* s modifikovanou *Costasovo* smyčkou, fázový ofset byl φ=40°, po zavěšení PLL pro 3500 – 200000 symbolů, průměrování provedeno přes 50 simulačních cyklů.

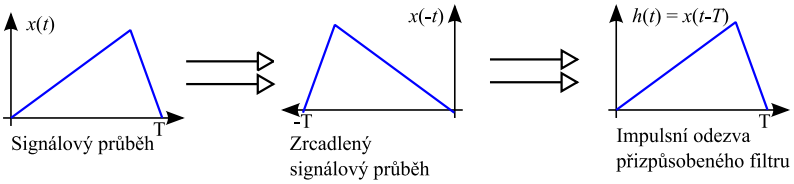
Tento filtr je nutné aplikovat na signál v základním pásmu. Impulsní odezva ideálního přizpůsobeného filtru za předpokladu komunikačního kanálu s (nekorelovaným) bílým šumem je časově reverzní komplexně sdružený signál, který je třeba detekovat. Přizpůsobený filtr tedy umožňuje provést detekci vyslaných symbolů v přijatém signálu s výrazným šumovým pozadím (resp. s malým odstupem SNR). Význam přizpůsobeného filtru v přenosovém řetězci je uveden na obrázku č. 6.8. Druhý koncept obsahuje také přizpůsobené filtry, ale ty jsou zařazeny z uvedených důvodů až za blokem synchronizace nosné vlny. Z tohoto důvodu je výhodné i při vlastní realizaci digitálního přijímače vycházet z konceptu prvního modelu, který představuje čistě diskrétní přístup. Dále tedy pro detektory ZCTED resp. GTED využívám pouze prvního principu. Na závěr kapitoly uvádím konstelační diagramy pro koncept modifikované Costasovy smyčky (obr. 6.9).





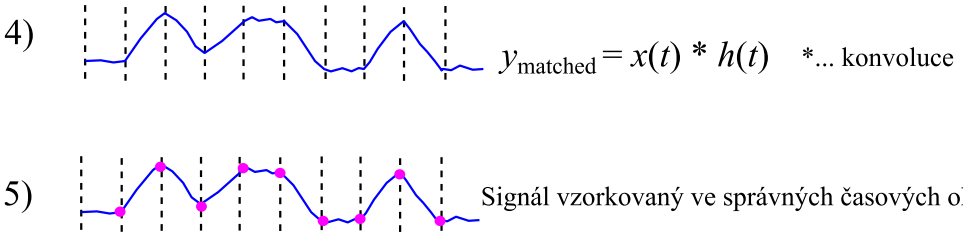
3) Pro zvýšení odstupu *SNR*, přijatý signál projde přizpůsobeným filtrem s impulsní odezvou:

$$h(t) = \begin{cases} \pi\, x(t\text{-}T) \text{ pro } 0 \le t \le T \\ 0 \text{ jinde} \end{cases}$$

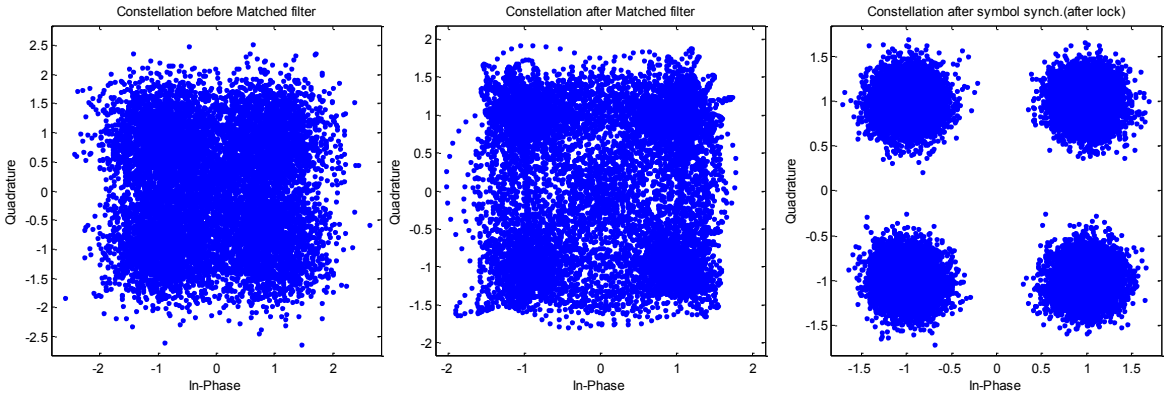


➔ impulsní odezva $h(t)$ je časově zpožděná odezva zrcadleného obrazu originálního signálu

➔ z důvodu symetrie, časově reverzní komplexně sdružená odezva $h(-t)$ je v právě $h(t)$



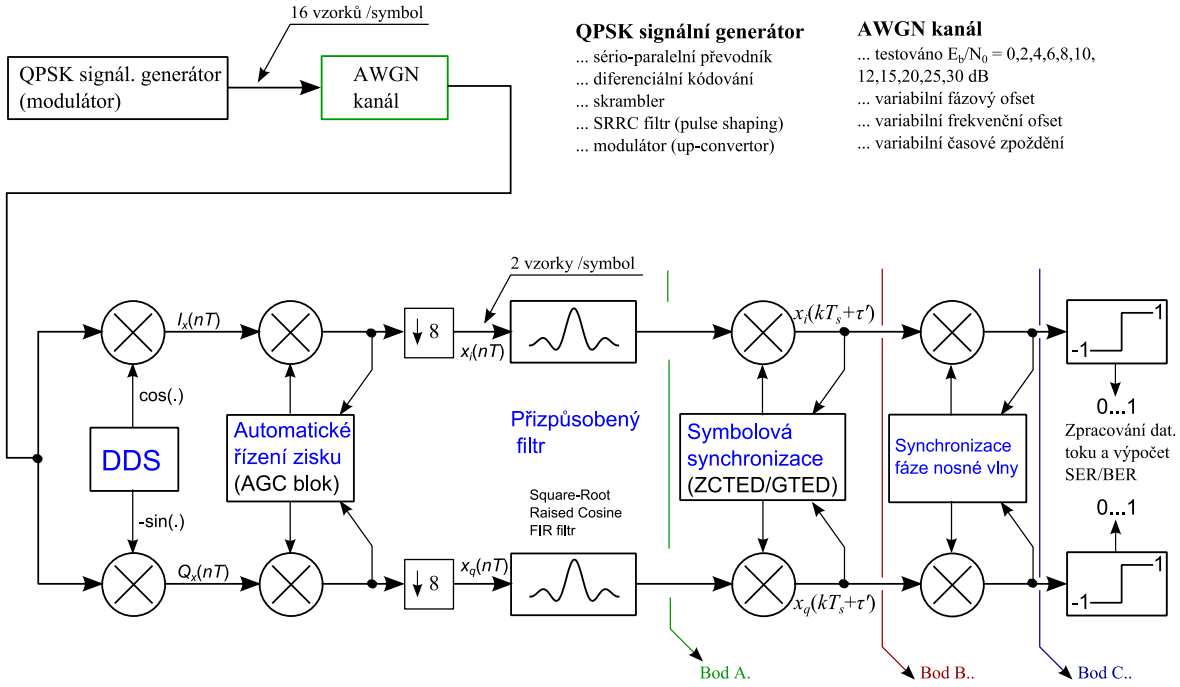
Obr. 6.8 Význam přizpůsobeného filtru v přenosovém řetězci



Obr. 6.9 Konstelační digramy na straně přijímače pro E_b/N_0=8 dB a fázový ofset φ=30°. Detektor je typu *MLTED* [mod. *Costasova* smyčka] – před přizpůsobeným filtrem (*bod A.*), za přizpůsobeným filtrem (*bod B.)* a po provedení symbolové synchronizace (*bod C.)*

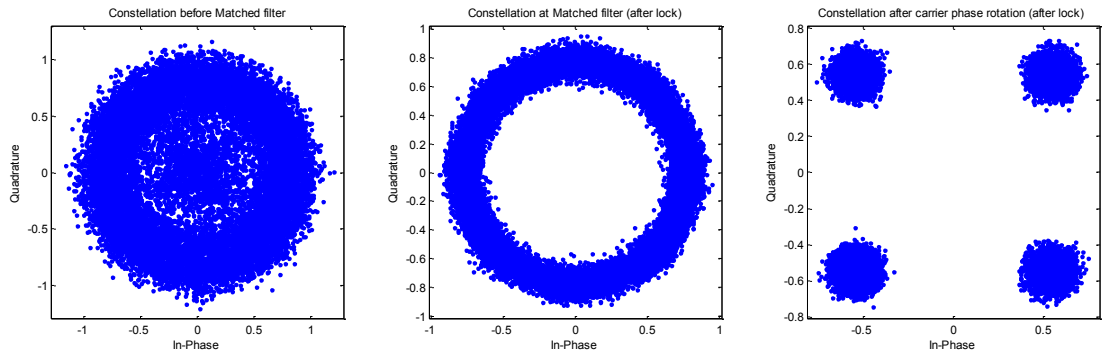
6.2 Simulace přenosového řetězce s využitím detektoru ZCTED a GTED pro symbolovou synchronizaci

Simulační blokové schéma vychází z konceptu pro MLTED (obrázek č. 6.6) s tím rozdílem, že detektor ZCTED resp. GTED pracuje se dvěma vzorky na symbol. Je potřeba provést decimaci vstupního signálu osmi. Simulační blokové schéma je zobrazeno na obrázku č. 6.10.

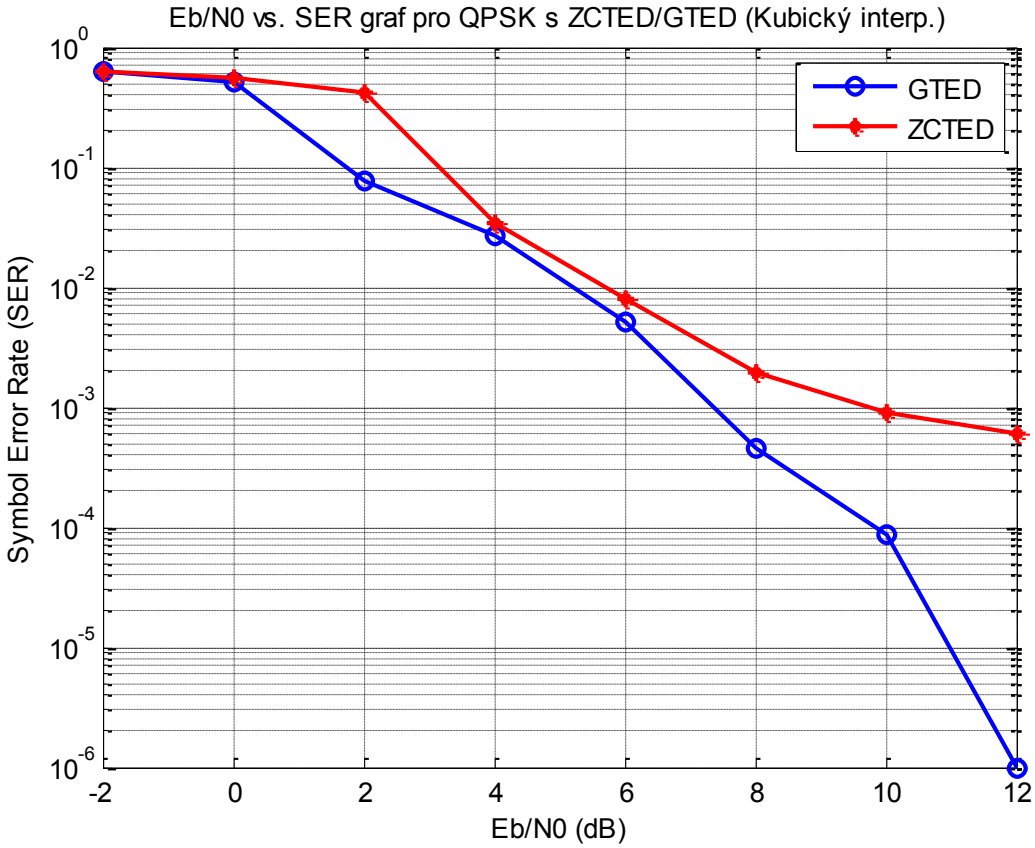


Obr. 6.10 Simulační blokové schéma pro symbolovou synchronizaci s detektorem ZCTED resp. GTED (konvenční přístup)

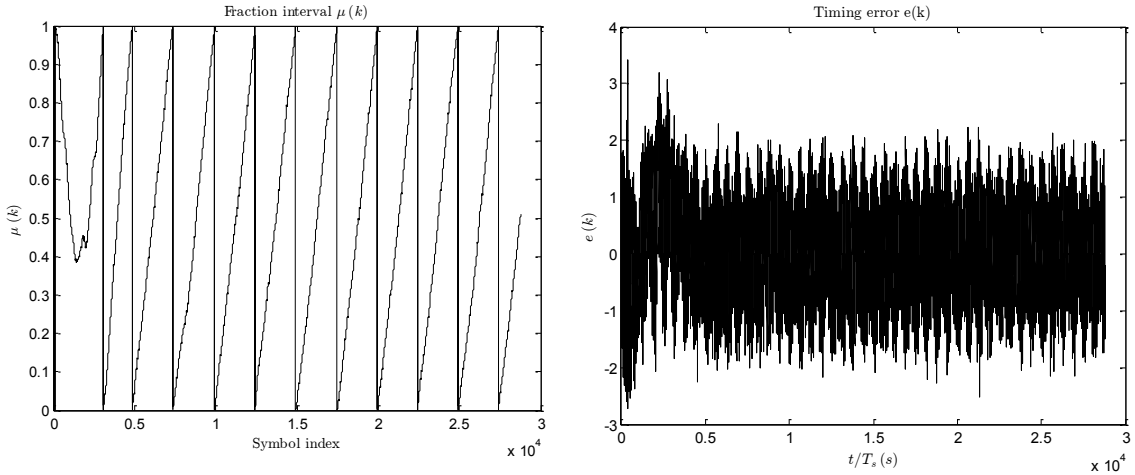
V případě detektorů ZCTED/GTED, které pracují se 2 vzorky/symbol, je nutné se zaměřit na případ, kdy vzorkovací frekvence je o určitý zlomkový interval větší / menší než 2 vzorky / symbol (obrázek 6.11).



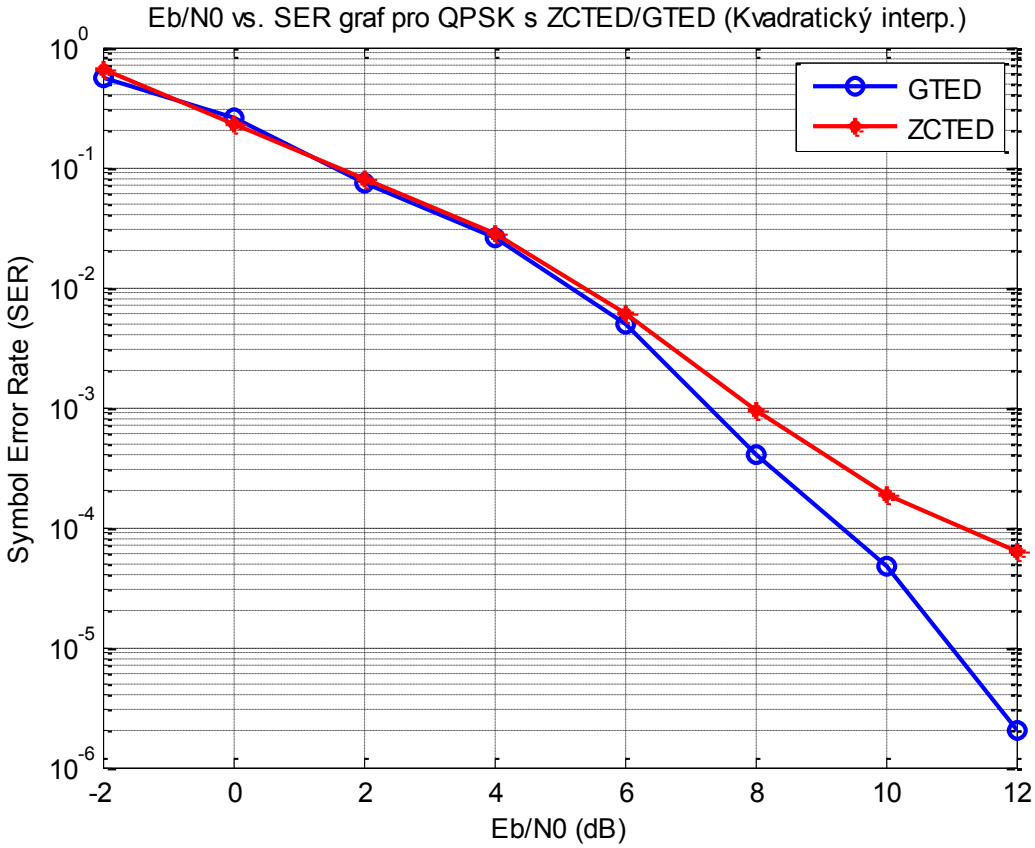
Obr. 6.11 Konstelační digramy na straně přijímače pro E_b/N_0=10 dB a fázový ofset φ=35°. Detektor je typu *GTED* – zleva - za přizpůsobeným filtrem (*bod A.*), po symbolové synchronizaci (*bod B.)* a po provedení synchronizace nosné vlny (*bod C.)*



Obr. 6.12 Vyhodnocení *SER* vs. E_b/N_0 pro *ZCTED* a GTED s kubickým interpolačním filtrem, fázový ofset byl φ=35°, po zavěšení *PLL* pro 4000 – 200000 symbolů, průměrování provedeno přes 50 simulačních cyklů.



Obr. 6.13 Grafy pro zlomkový interval $\mu(k)$ a chybový signál $e(k)$ pro E_b/N_0=8 dB (ZCTED – kvadratický interpolátor). Z průběhů je patrné, že k zavěšení PLL došlo zhruba po 2500 symbolových periodách. Vzorkovací frekvence je o 1/5000 symbolové rychlosti rychlejší než 2 vzorky/symbol (viz. pilovitý průběh $\mu(k)$)

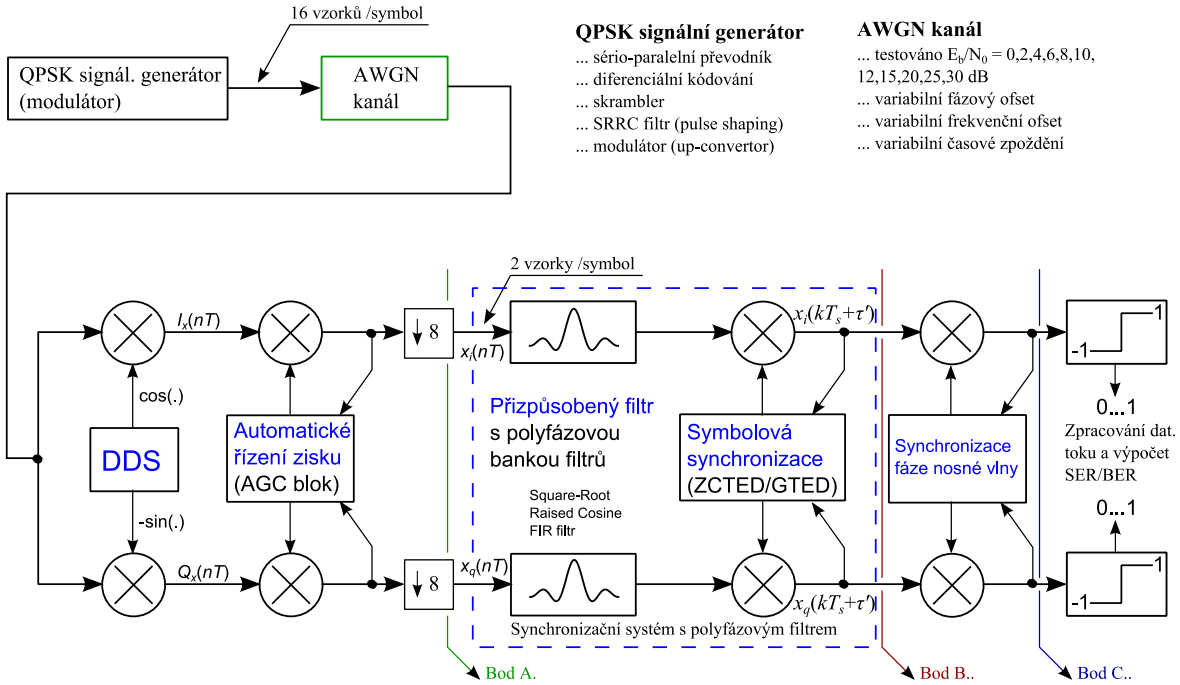


Obr. 6.14 Vyhodnocení *SER* vs. E_b/N_0 pro *ZCTED* a GTED s kvadratickým interpolačním filtrem, fázový ofset byl φ=35°, po zavěšení *PLL* pro 4000 – 200000 symbolů, průměrování provedeno přes 50 simulačních cyklů.

Vyhodnocení chybovosti symbolů (*SER*) bylo provedeno pro porovnání výkonnosti ZCTED a GTED detektorů. Simulace potvrdila, že pro větší odstup SNR (respektive pro E_b/N_0 > 6 dB) GTED předčí ZCTED. GTED je rotačně invariantní detektor, který je nezávislý na rotaci fáze nosné vlny. Využití detektoru GTED v případě, že korekce ofsetu fáze nosné vlny je provedena až za symbolovou synchronizací, je tedy velice vhodné. Výběr mezi kubickým a kvadratickým interpolačním filtrem nemá zásadní vliv na funkčnost celého synchronizačního systému. Kombinace detektoru ZCTED a kvadratického interpolátoru podle simulací dokonce při daném E_b/N_0 vykazuje nižší SER. Vždy je ale patrné, že GTED vykazuje nižší chybovost SER při daném E_b/N_0.

6.3 Simulace přenosového řetězce s využitím detektoru ZCTED a GTED pro symbolovou synchronizaci s polyfázovým filtrem

Simulace ve zpětnovazební smyčce byla provedena pro detektor ZCTED resp. GTED s využitím polyfázového filtru a tvoří společně s blokem synchronizace nosné vlny základ simulovaného přenosového řetězce. Tento polyfázový filtr slouží zároveň jako přizpůsobený a interpolační filtr. Vytvořený model se skládá opět ze signálového *QPSK* generátoru, kanálu s bílým šumem (AWGN kanál), přizpůsobeného filtru *SRRC*, digitálního směšovače s přímou digitální syntézou (DDS), bloku automatického řízení zisku (AGC), zmíněného synchronizačního systému s detektorem ZCTED resp. GTED s polyfázovým filtrem a bloku synchronizace fáze nosné vlny. Simulační model je uveden na obrázku č. 6.15.



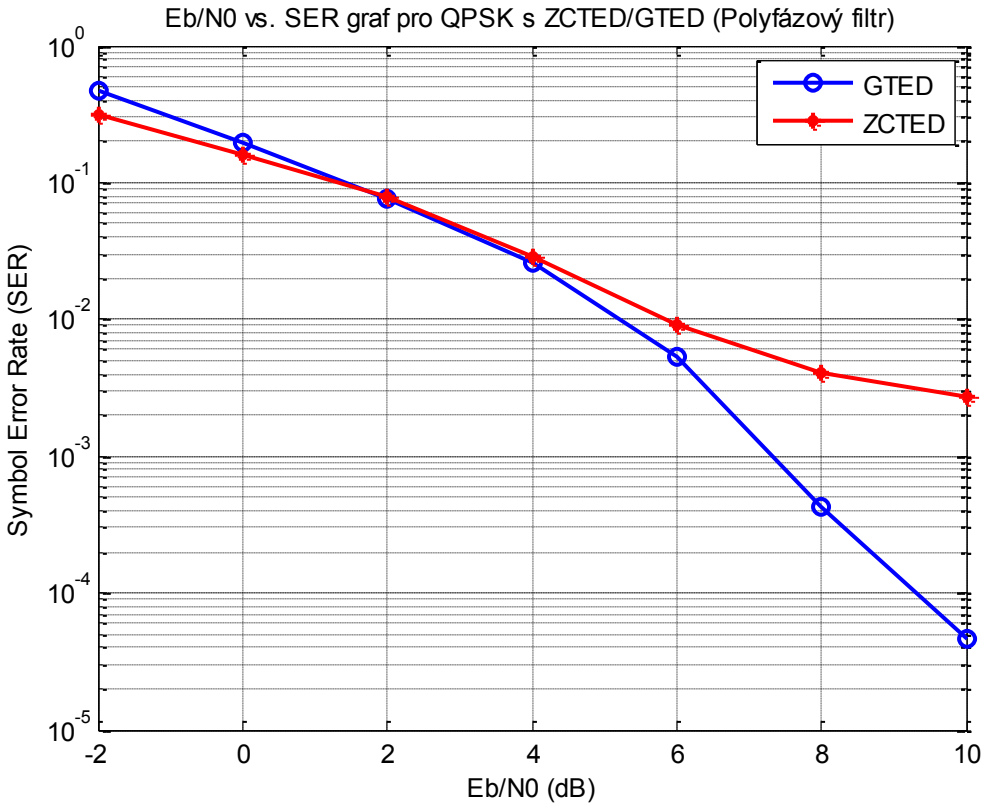
Obr. 6.15 Simulační blokové schéma pro symbolovou synchronizaci s detektorem ZCTED resp. GTED a polyfázovou bankou filtrů

Simulovaný přizpůsobený filtr má 8 sub filtrů (tedy 8 různých fází) a *roll-off* faktor je roven 0.5. Řád filtru daný v symbolech je N_{sym}= 6. Teoreticky, celkem 96 koeficientů filtru je rozděleno mezi I=8 polyfázových sub filtrů, kde každý bude 12 koeficientů za předpokladu 2 vzorků / symbol. Protože délka impulsní odezvy *SRRC* přizpůsobeného

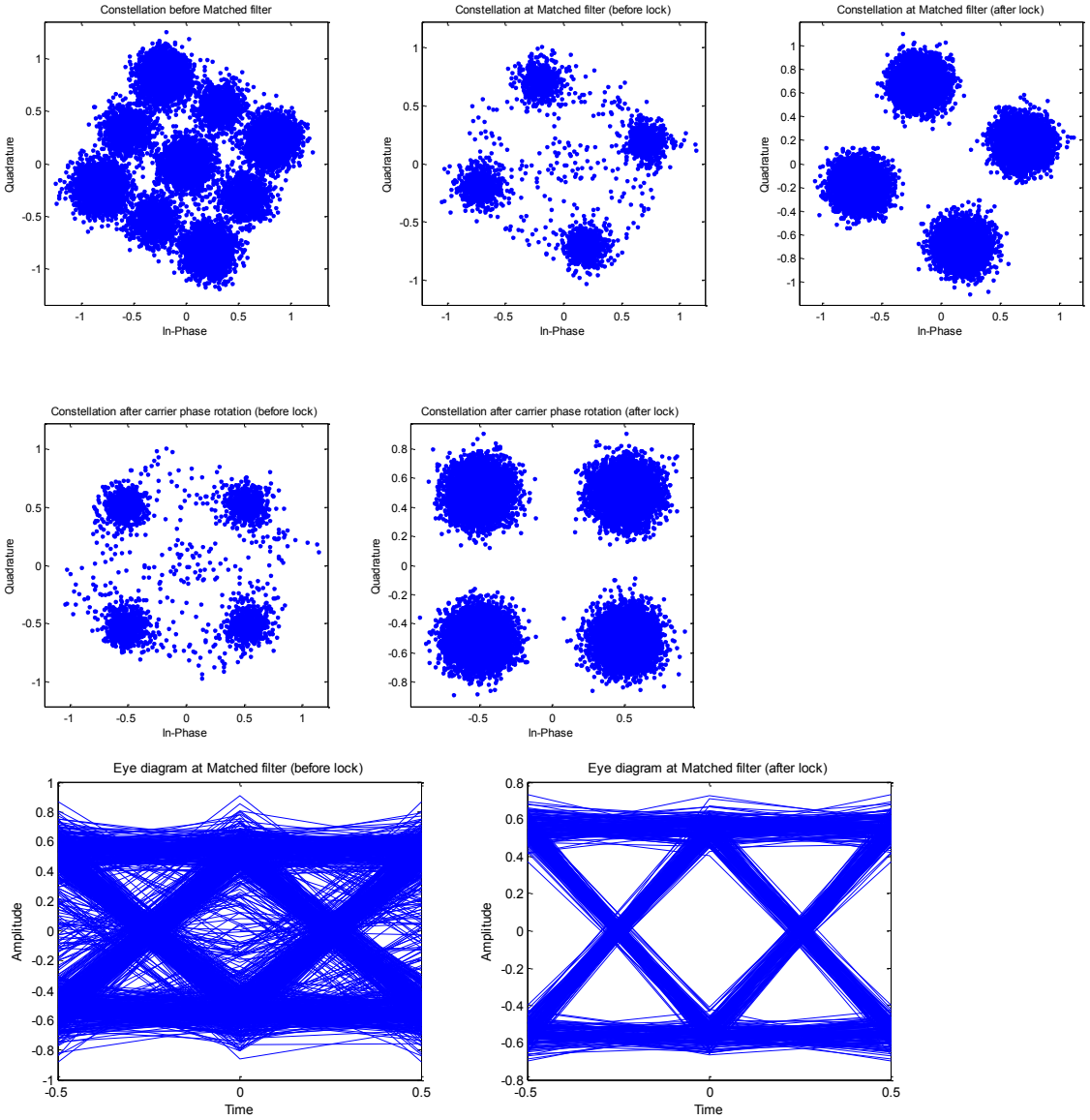
filtru je N_{sym}*$SamplesPerSymbol$+1, bude mít každý sub-filtr 13 koeficientů. Chyba časovaní symbolů je aktualizována každou symbolovou periodu. Proporciální konstanta K_p je pro ZCTED 3.95 a GTED 1. Simulace ukázali, že předchozí vypočtené konstanty K_p pro E_b/N_0=10 dB lze využít v širokém rozsahu hodnot SNR (viz. testované E_b/N_0 na obrázku č. 6.16). Parametry pro symbolovou synchronizaci jsou v tomto případě

- $B_{PLL}T_s$= 0.005, kde B_{PLL} je šířka pásma PLL pro symbolovou synchronizaci
- Přenosová funkce filtru PLL je navržena s konstantami K_1 (0.00665) a K_2 (2.2148e-05), tlumící faktor ζ=1 (platí pro symbolovou synchronizaci).

Protože korekce ofsetu fáze nosné vlny je provedena po symbolové synchronizaci, je opět lepší využít rotačně invariantní detektor jako je GTED. Vyhodnocení chybovosti symbolů (*SER*) bylo provedeno pro porovnání výkonnosti ZCTED a GTED detektorů v této aplikaci. Simulace potvrdila, že pro větší odstup SNR (respektive pro E_b/N_0 > 4 dB) GTED opět předčí ZCTED. Pro E_b/N_0 < 4 dB je již výkon srovnatelný (obrázek č. 6.16).

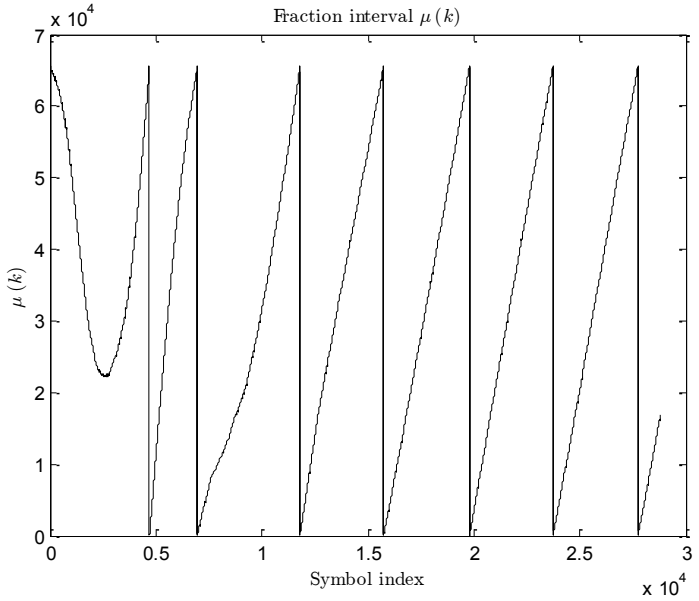


Obr. 6.16 Vyhodnocení *SER* vs. E_b/N_0 pro *ZCTED* a GTED s polyfázovým filtrem, fázový ofset byl φ=35°, po zavěšení *PLL* pro 4000 – 200000 symbolů, průměrování provedeno přes 50 simulačních cyklů.

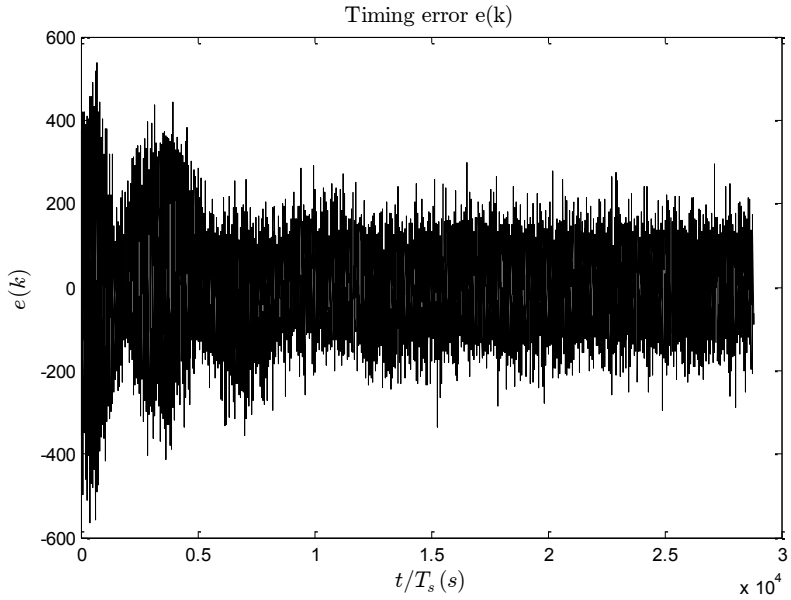


Obr. 6.17 Konstelační a digramy oka na straně přijímače pro E_b/N_0=8 dB a fázový ofset φ=35°. Detektor je typu GTED. Popis z levého horního rohu dolů. Konstelační digram před přizpůsobeným filtrem (bod A.); konstelační diagram za přizpůsobeným filtrem – před zavěšením pro 50 – 3000 symbolů (bod B.); konstelační diagram za přizpůsobeným filtrem – po zavěšení pro 4000 – 200000 symbolů; konstelační diagram po synchronizaci nosné vlny – před zavěšením pro 50 – 3000 symbolů (bod C.); konstelační diagram po synchronizaci nosné vlny – po zavěšení pro 4000 – 200000 symbolů; diagram oka za přizpůsobeným filtrem - před zavěšením; diagram oka za přizpůsobeným filtrem - po zavěšení (stejný počet symbolů jako u konstelačních diagramů, zobrazena pouze soufázová složka).

Průběh zlomkového intervalu $\mu(k)$ je zobrazen na obrázku č. 6.18 pro případ, že vzorkovací frekvence je o 1/4000 symbolové rychlosti rychlejší než 2 vzorky/symbol. Zlomkový interval má potom schodovitý průběh (po zavěšení PLL) od 0 do 1 a periodicky cykluje po 4000 symbolech. Pro rozšíření standardní simulace v plovoucí řádové čárce, byla provedena simulace v pevné řádové čárce (viz. následující obrázky č. 6.18 a 6.19), která umožňuje rychlejší přechod k VHDL modelu v pevné řádové čárce. Zlomkový interval je poté vypočten pomocí (4.77) místo (4.73).



Obr. 6.18 Zlomkový interval $\mu(k)$ pro E_b/N_0 =8 dB a fázový ofset φ=35°; GTED



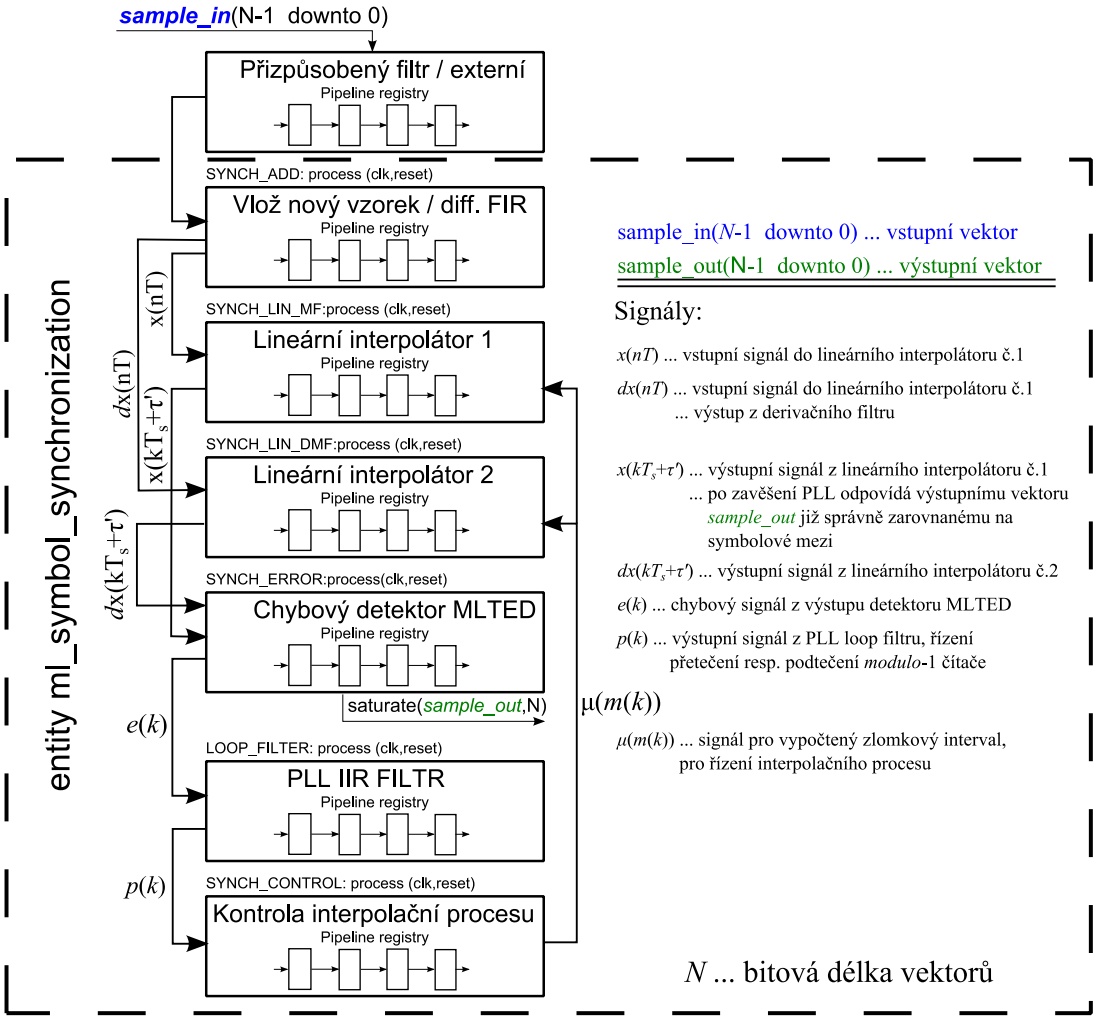
Obr. 6.19 Chyba časování $e(k)$ pro E_b/N_0 =8 dB a fázový ofset φ=35°; GTED

7 RTL simulace a syntéza na hradlovém poli FPGA

V této kapitole se věnuji RTL simulaci a následné syntéze na FPGA navržených modelů pro symbolovou synchronizaci a synchronizaci fáze nosné vlny. Jako dodatek uvádím také RTL simulaci a vlastnosti syntézy pro blok automatického řízení zisku AGC.

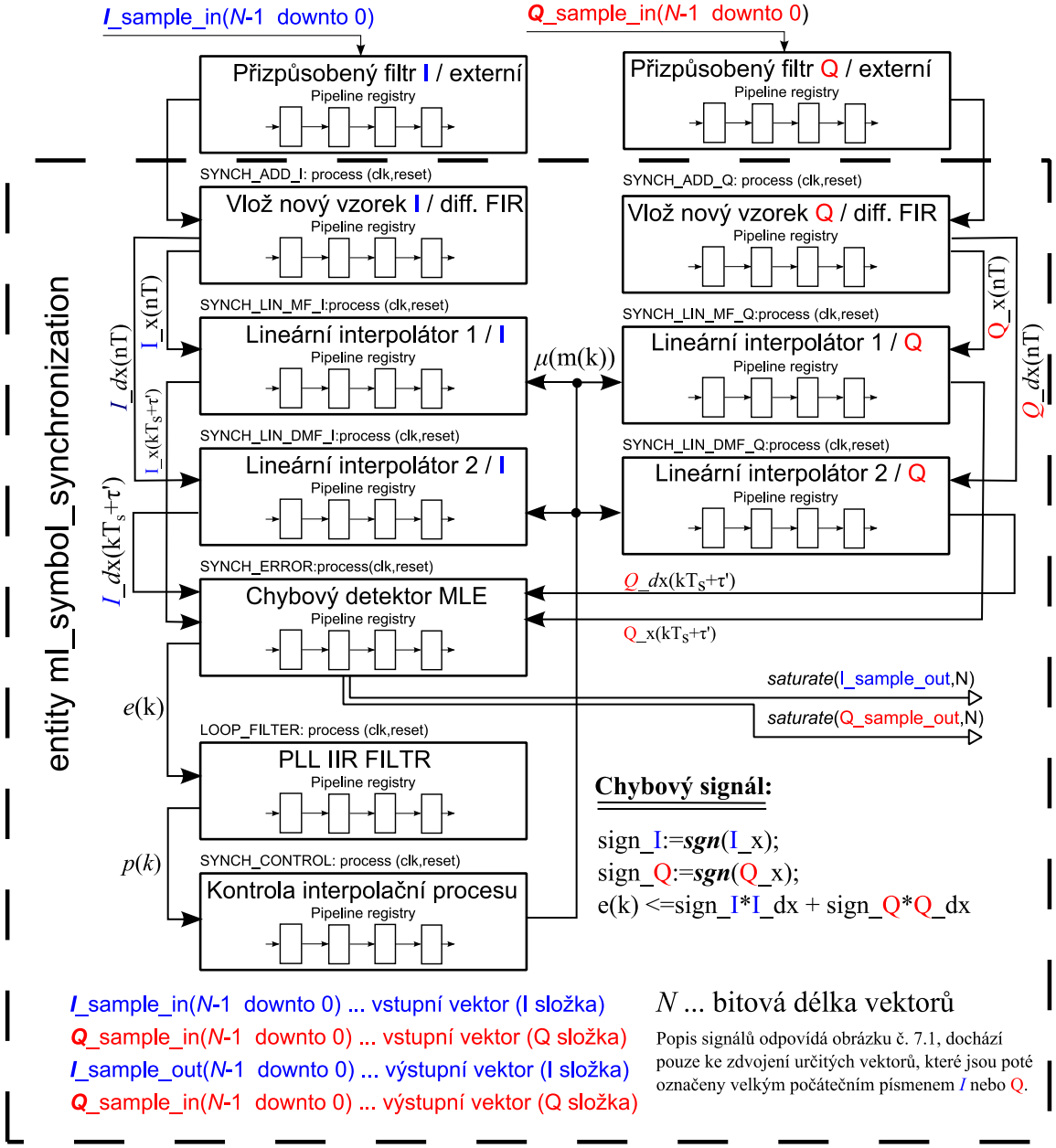
7.1 RTL simulace a syntéza modelu symbolové synchronizace s detektorem MLTED

Při návrhu synchronizačního modelu v jazyce VHDL jsem vycházel ze schématu na obrázku č. 4.7. Zřetězené zpracování bylo využito v maximální možné míře. Nejdříve uvedu na obrázku č. 7.1 strukturu VHDL modelu pro PAM signál (případně BPSK).



Obr. 7.1 Struktura VHDL modelu pro symbolovou synchronizaci s detektorem MLTED (pro signál PAM; respektive BPSK)

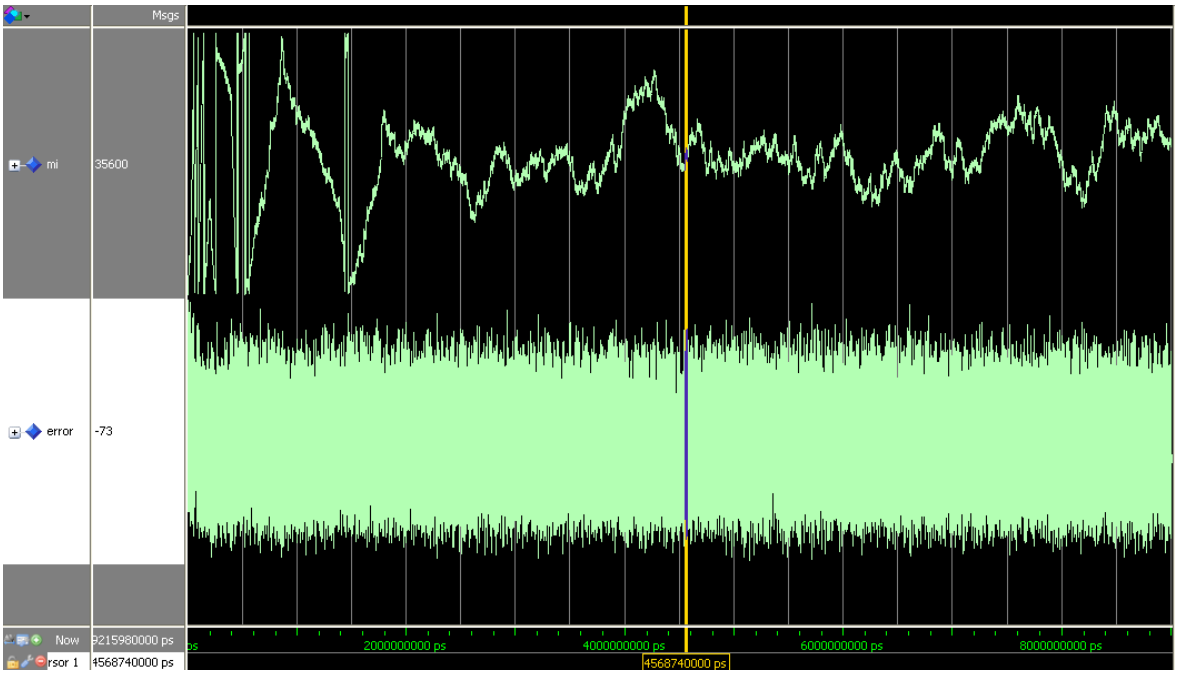
Přizpůsobený filtr není součástí synchronizačního modelu a je implementovaný jako externí modul. Je využito plně paralelní struktury z kapitoly 3.2. Lineární interpolátor č. 2 zpracovává signál z derivačního filtru, který implementuje proces nazvaný, jako *Vlož nový vzorek / diff. FIR*. Pro práci s I/Q signály (QPSK; *m*-QAM) je možné model z 7.1 relativně jednoduše rozšířit. Modifikace spočívá v přidání tří nových procesů (*Vlož nový vzorek / diff. FIR / Q, Lineární interpolátor1/ Q, Lineární interpolátor2/ Q*) a modifikaci chybového detektoru, jak uvedeno na obrázku č. 7.2. Je také nutné přidat instanci přizpůsobeného filtru pro *Q* složku signálu.



Obr. 7.2 Rozšířená struktura VHDL modelu pro symbolovou synchronizaci s detektorem MLTED (pro signál QPSK; respektive *m*-QAM)

RTL simulace byla provedena prostřednictvím programu Altera Modelsim s VHDL testovací procedurou (testbench), který načítá vstupní testovací vektory z textového souboru. Tyto testovací vektory (v pevné řádové čárce) jsou generovány z programu Matlab. Testovací procedura umožňuje export výstupních vektorů ze simulace pro další analýzu a porovnání se simulací v pevné řádové čárce v programu Matlab. Výsledky RTL simulace jsou uvedeny pro následující parametry PLL smyčky symbolové synchronizace MLTED (odpovídají simulaci v programu Matlab):

- $B_{PLL}T_s = 0.04$, kde B_{PLL} je šířka pásma PLL pro symbolovou synchronizaci
- Přenosová funkce filtru PLL je navržena s konstantami K_1(0.00354) a K_2 (1.476e-6), tlumící faktor $\zeta=1/\sqrt{2}$



Obr. 7.3 RTL simulace v pevné řádové čárce pro zlomkový interval $\mu(k)$ a chybový signál $e(k)$ pro E_b/N_0=8 dB (MLTED). Výsledky simulace odpovídají simulaci v pevné řádové čárce v programu Matlab (porovnáno s RTL simulací)

Ukázková syntéza byla provedena pro hradlové pole FPGA od společnosti Altera. Jedná se o FPGA Altera Cyclone IV (Cyclone IV EP4CE115F29C7). Tento typ FPGA se nachází na vývojovém kitu Terasic DE2-115, který ve spolupráci s rozšiřující kartou s rychlými AD/DA převodníky používám dále pro ověření funkce celého synchronizačního systému.

Syntéza je také provedena s možností korekce prvního řádu (viz. kapitola 4.3.2), kde je také v bloku řízení interpolace obecné dělení nahrazeno aproximací. V tabulce V. jsou uvedeny výsledky syntézy pro model pro PAM (BPSK) signál a v tabulce VI. rozšířený model pro I/Q signály (QPSK, *m*-QAM).

Tabulka V. Výsledky syntézy pro model symbolové synchronizace s MLTED (PAM)

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Model symbolové synchronizace s MLTED (PAM) f_{MAX}=146.7 MHz	421 (<1%) / 0	384 (<1%)	255 (<1%)	8/532 (3%)
Model symbolové synchronizace s MLTED (PAM) – korekce 1.řádu f_{MAX}=90.6 MHz	497 (<1%) /0	459(<1%)	282 (<1%)	12/532 (2%)

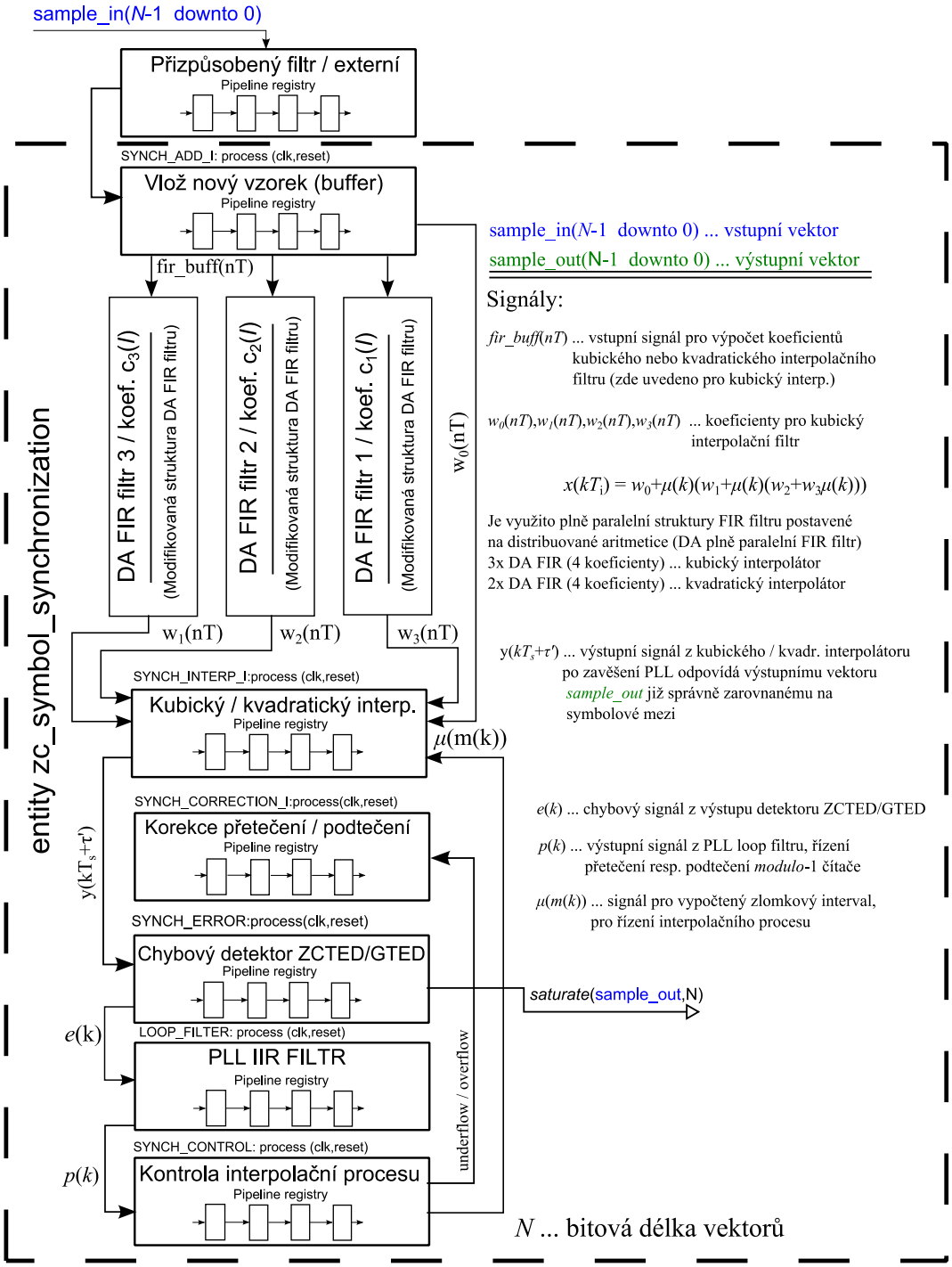
Tabulka VI. Výsledky syntézy pro model symbolové synchronizace s MLTED (I/Q)

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Model symbolové synchronizace s MLTED (I/Q) f_{MAX}=132.0 MHz	607 (<1%) / 0	545 (<1%)	381 (<1%)	16/532 (3%)
Model symbolové synchronizace s MLTED (I/Q) – korekce 1.řádu f_{MAX}=90.1 MHz	697 (<1%) / / 0	630 (<1%)	416 (<1%)	20/532 (4%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

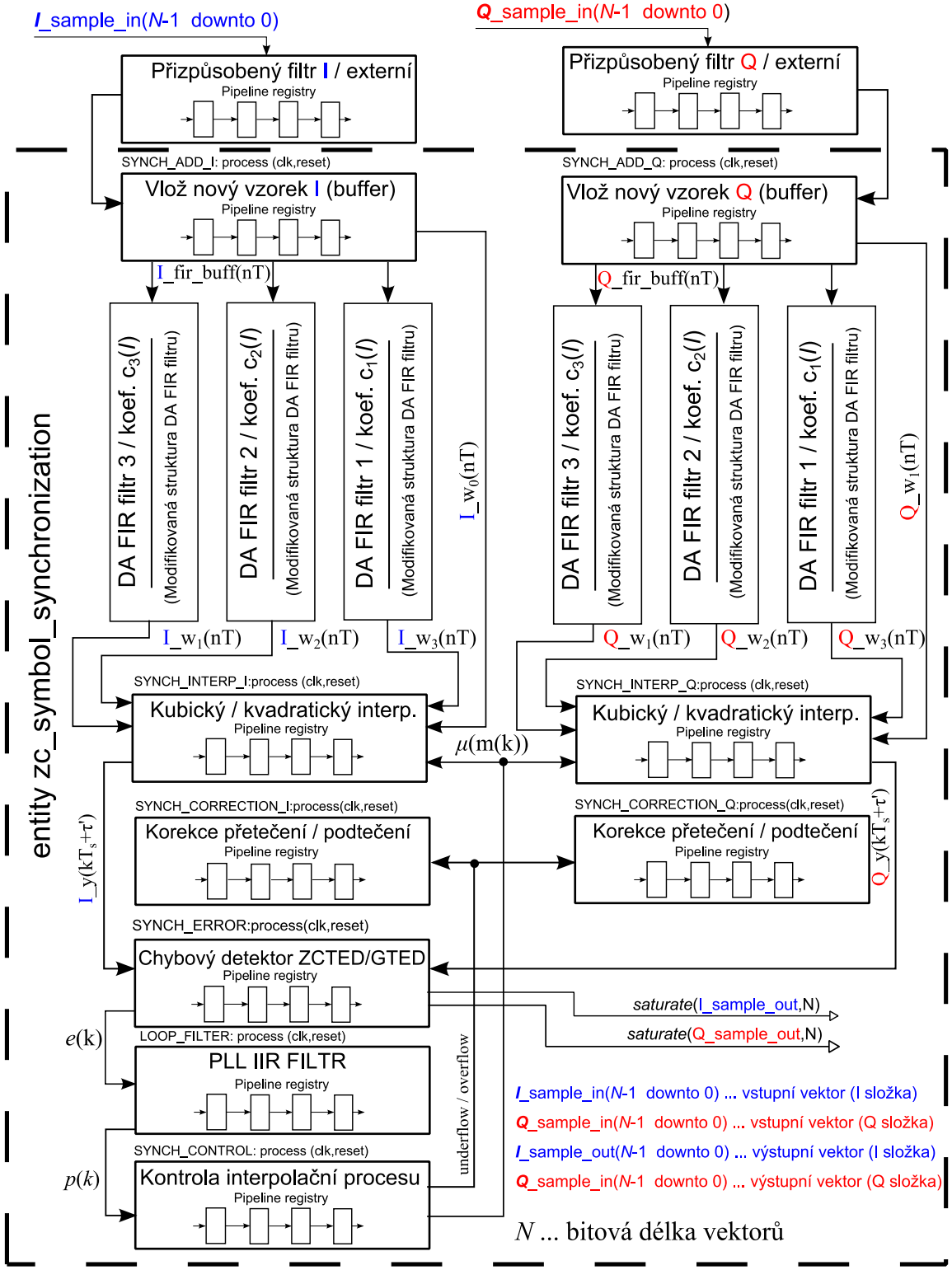
7.2 RTL simulace a syntéza modelu symbolové synchronizace s detektorem ZCTED/GTED

Při návrhu synchronizačního modelu pro detektor ZCTED/GTED v jazyce VHDL jsem vycházel ze schématu na obrázku č. 4.8. Nejdříve opět uvedu na obrázku č. 7.4 strukturu VHDL modelu pro PAM signál (případně BPSK).



Obr. 7.4 Struktura VHDL modelu pro symbolovou synchronizaci s detektorem ZCTED/GTED (pro signál PAM; respektive BPSK)

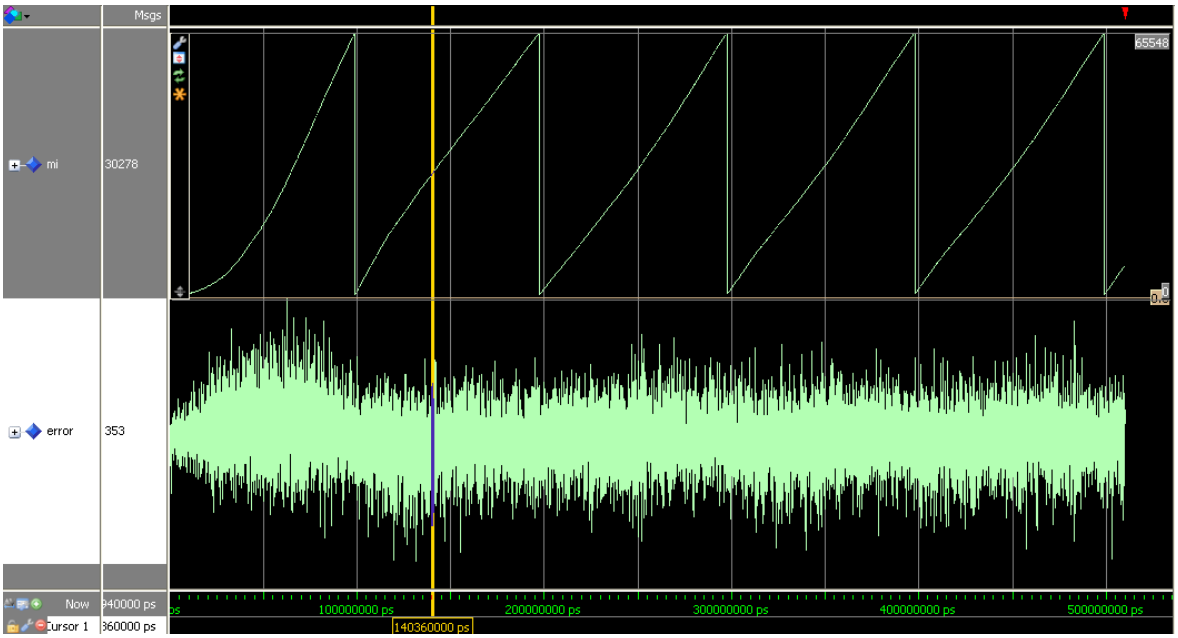
Pro výpočet koeficientů interpolačního filtru je využito struktury plně paralelného FIR filtru postavené na distribuované aritmetice (viz. kapitola 3.2 a 4.3.1). Pro práci s I/Q signály (QPSK; *m*-QAM) je rozšířený model zobrazen na obrázku č. 7.5.



Obr. 7.5 Rozšířená struktura VHDL modelu pro symbolovou synchronizaci s detektorem ZCTED/GTED (pro signál QPSK; respektive *m*-QAM)

Výsledky ukázkové RTL simulace jsou uvedeny na obrázku č. 7.6 pro následující parametry PLL smyčky symbolové synchronizace GTED (odpovídají simulaci v programu Matlab). Parametry pro symbolovou synchronizaci jsou v tomto případě:

- $B_{PLL}T_s$= 0.00625, kde B_{PLL} je šířka pásma PLL pro symbolovou synchronizaci
- Přenosová funkce filtru PLL je navržena s konstantami K_1 (0.00104) a K_2 (5.4225e-07), tlumící faktor ζ=1/ /√2



Obr. 7.6 RTL simulace v pevné řádové čárce pro zlomkový interval $\mu(k)$ a chybový signál $e(k)$ pro E_b/N_0=8 dB (GTED), je využito kvadratického interpolátoru. Vzorkovací frekvence je o 1/5000 symbolové rychlosti rychlejší než 2 vzorky/symbol. Porovnáno se simulací v programu Matlab.

Ukázková syntéza byla provedena pro synchronizační systém s detektory GTED a ZCTED. Je opět rozlišeno, zdali se jedná o synchronizační systém pro PAM signály (tabulky VII. až X.) nebo o rozšířený model se signálovými složkami I/Q (tabulky XI. až XIV.). Navíc je v tomto modelu rozlišeno, zdali je využito kvadratického nebo kubického filtru. Důsledná aplikace zřetězeného zpracování umožnila dosáhnout velice dobré maximální pracovní frekvence synchronizačního systému i v poměru k počtu obsazených logických prvků.

Tabulka VII. Výsledky syntézy pro model symbolové synchronizace s ***ZCTED (PAM)*** – ***kvadratický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kvadratický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **ZCTED** (PAM)* f_{MAX}=130.28 MHz	717 (<1%) / 0	562 (<1%)	531 (<1%)	4/532 (<1%)
*Model symbolové synchronizace s **ZCTED** (PAM) – korekce 1.řádu* f_{MAX}=90.52 MHz	779 (<1%) / / 0	626 (<1%)	545 (<1%)	8/532 (2%)

Tabulka VIII. Výsledky syntézy pro model symbolové synchronizace s ***GTED (PAM)*** – ***kvadratický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kvadratický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **GTED** (PAM)* f_{MAX}=103.46 MHz	673 (<1%) / 0	523 (<1%)	531 (<1%)	6/532 (1%)
*Model symbolové synchronizace s **GTED** (PAM) – korekce 1.řádu* f_{MAX}=90.75 MHz	732 (<1%) / / 0	587 (<1%)	545 (<1%)	10/532 (2%)

Tabulka IX. Výsledky syntézy pro model symbolové synchronizace s ***ZCTED (PAM)*** – ***kubický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kubický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **ZCTED** (PAM)* f_{MAX}=127.37 MHz	1236 (1%) / 0	936 (<1%)	1016 (<1%)	6/532 (1%)
*Model symbolové synchronizace s **ZCTED** (PAM) – korekce 1.řádu* f_{MAX}=93.18 MHz	1299 (1%) / / 0	1000 (<1%)	1032 (<1%)	10/532 (2%)

Tabulka X. Výsledky syntézy pro model symbolové synchronizace s ***GTED (PAM)*** – ***kubický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kubický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **GTED** (PAM)* f_{MAX}=101.09 MHz	1198 (1%) / 0	897 (<1%)	1016 (<1%)	8/532 (1%)
*Model symbolové synchronizace s **GTED** (PAM) – korekce 1.řádu* f_{MAX}=91.16 MHz	1251 (1%) / / 0	961 (<1%)	1032 (<1%)	12/532 (2%)

Tabulka XI. Výsledky syntézy pro model symbolové synchronizace s ***ZCTED (I/Q)*** – ***kvadratický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kvadratický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **ZCTED** (I/Q)* f_{MAX}=123.23 MHz	1175 (<1%) / 0	936 (<1%)	898 (<1%)	8/532 (2%)
*Model symbolové synchronizace s **ZCTED** (I/Q) – korekce 1.řádu* f_{MAX}=90.96 MHz	1231 (<1%) / / 0	999 (<1%)	912 (<1%)	12/532 (3%)

Tabulka XII. Výsledky syntézy pro model symbolové synchronizace s ***GTED (I/Q)*** – ***kvadratický interpolátor***

Výsledky syntézy: vstupní data – **12 bitů** konstanty – **17 bitů** **kvadratický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **GTED** (I/Q)* f_{MAX}=90.18 MHz	1099 (<1%) / 0	857 (<1%)	898 (<1%)	12/532 (3%)
*Model symbolové synchronizace s **GTED** (I/Q) – korekce 1.řádu* f_{MAX}=90.22 MHz	1147 (<1%) / / 0	921 (<1%)	912 (<1%)	16/532 (3%)

Tabulka XIII. Výsledky syntézy pro model symbolové synchronizace s ***ZCTED (I/Q)* – *kubický interpolátor***

Výsledky syntézy: vstupní data **– 12 bitů** konstanty **– 17 bitů** **kubický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **ZCTED** (I/Q)* f_{MAX}=119.19 MHz	2103 (2%) / 0	1684 (2%)	1747 (2%)	12/532 (3%)
*Model symbolové synchronizace s **ZCTED** (I/Q) – korekce 1.řádu* f_{MAX}=93.01 MHz	2156 (2%) / / 0	1748 (2%)	1761 (2%)	16/532 (3%)

Tabulka XIV. Výsledky syntézy pro model symbolové synchronizace s ***GTED (I/Q)* – *kubický interpolátor***

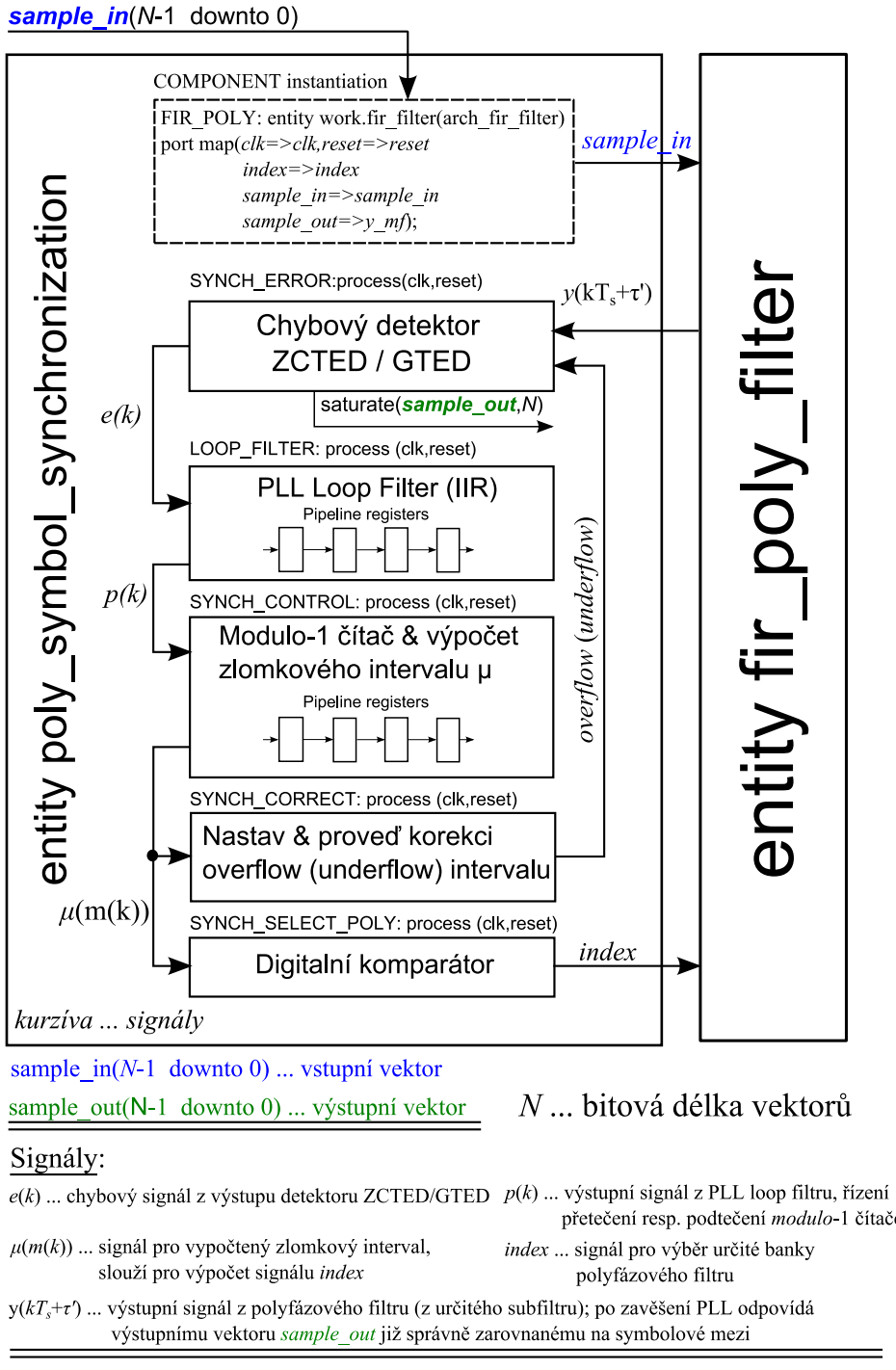
Výsledky syntézy: vstupní data **– 12 bitů** konstanty **– 17 bitů** **kubický interpolátor**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
*Model symbolové synchronizace s **GTED** (I/Q)* f_{MAX}=87.60 MHz	2023 (2%) / 0	1606 (2%)	1747 (2%)	16/532 (3%)
*Model symbolové synchronizace s **GTED** (I/Q) – korekce 1.řádu* f_{MAX}=89.67 MHz	2071 (2%) / / 0	1670 (2%)	1761 (2%)	20/532 (3%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

Z předchozích tabulek je patrné, že počet obsazených logických prvků u kubického interpolátoru je zhruba 2x větší než u kvadratického. Počet využitých registrů je přibližně roven počtu kombinačních funkcí, neboť jsou modely optimálně navrženy s využitím zřetězeného zpracování. Využití distribuované aritmetiky pro výpočet interpolačních koeficientů v rámci plně paralelních FIR filtrů umožnilo značně snížit počet obsazených integrovaných násobiček. V rámci vývoje, jsem nejprve navrhnul modely bez aplikace distribuované aritmetiky. Pro výpočet koeficientů byla využita klasická struktura FIR filtru využívající vestavěných násobiček. Po provedené optimalizaci s využitím distribuované aritmetiky se podařilo snížit počet násobiček 2.5x až 3x dle konkrétního modelu.

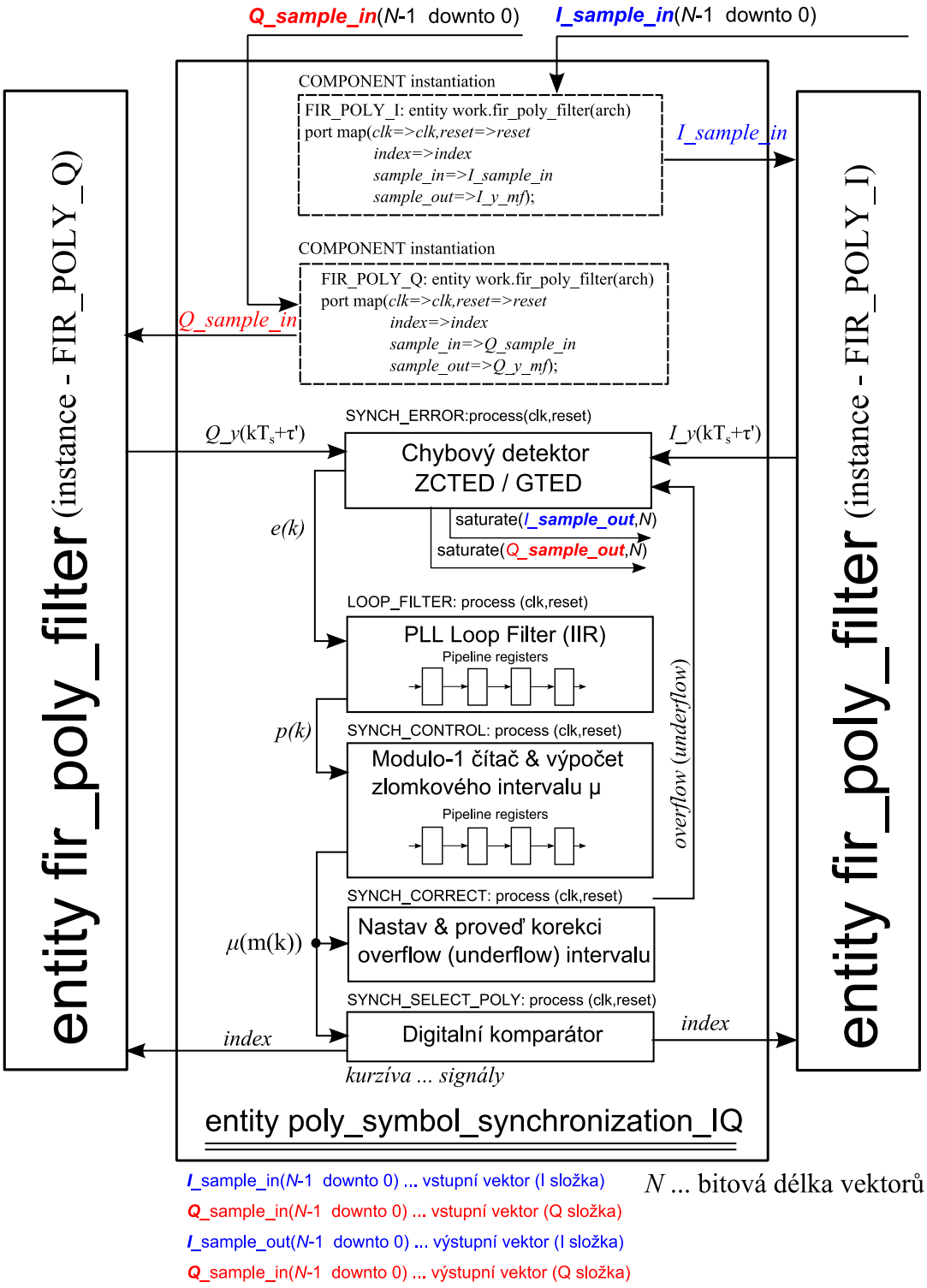
7.3 RTL simulace a syntéza modelu symbolové synchronizace s polyfázovým filtrem a detektorem ZCTED/GTED

Při návrhu synchronizačního modelu s polyfázovým filtrem a detektor ZCTED/GTED v jazyce VHDL jsem vycházel ze schématu na obrázku č. 4.25. Nejdříve opět uvedu na obrázku č. 7.7 strukturu VHDL modelu pro PAM signál (případně BPSK).



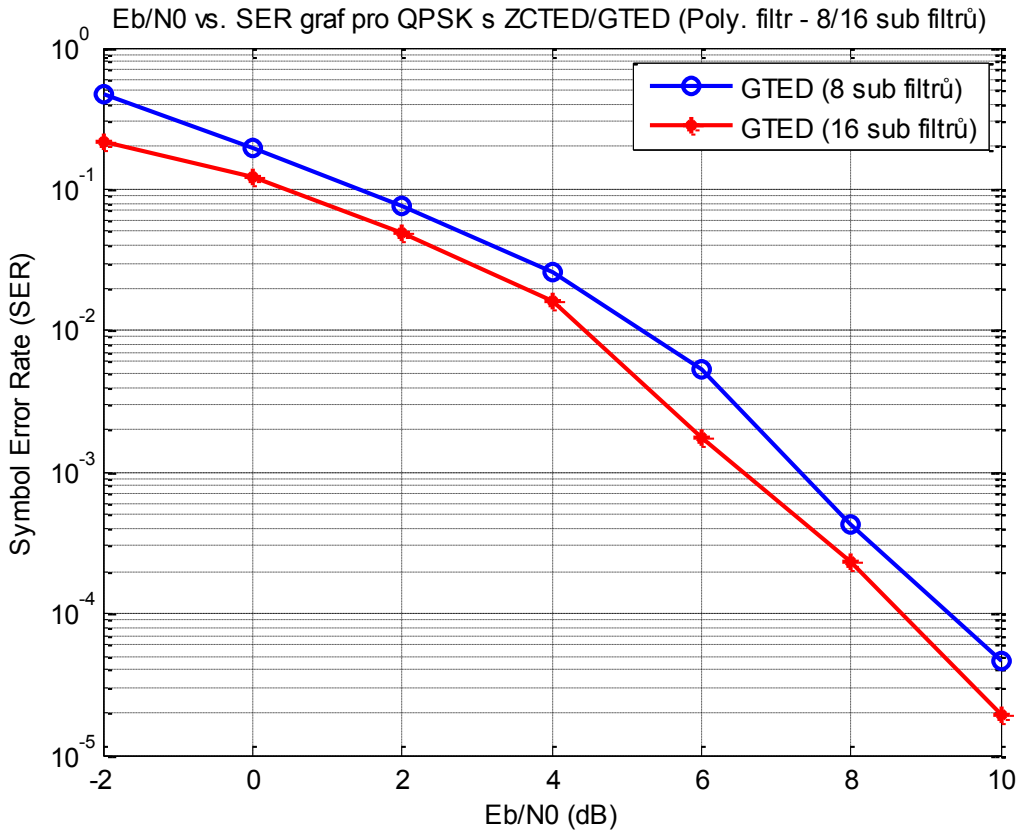
Obr. 7.7 Struktura VHDL modelu pro symbolovou synchronizaci s detektorem ZCTED/GTED a polyfázovým filtrem (pro signál PAM; respektive BPSK)

Představený model využívá polyfázového filtru zároveň jako přizpůsobeného filtru a interpolačního filtru. Polyfázový filtr využívá modifikované struktury standardního interpolačního filtru dle schématu ze 4.24. Pro práci s I/Q signály (QPSK; *m*-QAM) je rozšířený model zobrazen na obrázku č. 7.8.



Obr. 7.8 Rozšířená struktura VHDL modelu pro symbolovou synchronizaci s detektorem ZCTED/GTED a polyfázovým filtrem (pro signál QPSK; respektive *m*-QAM)

Struktura synchronizačního systému s polyfázovou bankou filtrů se odlišuje vůči konvenční struktuře ze schématu 7.4 resp. 7.5 právě vynecháním vlastního interpolačního (kvadratického nebo kubického) DA filtru a nahrazením instancí (resp. instancemi) polyfázových interpolačních filtrů. V rámci simulací v kapitole 6.3 jsem zvolil interpolační filtr, který bude mít 8 sub filtrů (tedy 8 různých fází). Tento počet dle provedených simulací vychází jako optimální a vlastnosti tohoto synchronizačního systému jsou zcela srovnatelné s konvenční strukturou využívající kubického interpolačního filtru. Je sice zřejmé, že v případě navýšení počtu sub filtrů se zmenší tzv. interpolační jitter, ale je nutné také vždy vzít v úvahu celkovou komplexnost struktury interpolačního filtru. Z tohoto důvodu jsem provedl simulaci vyhodnocení chybovosti symbolů (*SER*) pro porovnání výkonnosti detektoru GTED s 8 resp. 16 sub-filtry v rámci polyfázové banky filtrů (obrázek č. 7.9). Je využito blokové simulační schéma z obrázku 6.16.



Obr. 7.9 Vyhodnocení *SER* vs. E_b/N_0 pro GTED s 8 resp. 16 sub filtry v rámci polyfázové banky, fázový ofset byl φ=35°, po zavěšení *PLL* pro 4000 – 200000 symbolů, průměrování provedeno přes 50 simulačních cyklů. Další parametry simulace – viz. kapitola 6.3.

Z průběhů grafu na obrázku č. 7.9 je patrné, že využití polyfázové banky se 16 sub-filtry již nepřináší podstatné snížení chybovosti oproti bance s 8 sub-filtry. Provedená ukázková syntéza ovšem dále ukáže, že implementace plně paralelního DA polyfázového interpolačního filtru se 16 sub-filtry využívá značné množství logických prvků hradlového pole FPGA. Výsledky syntézy jsou uvedeny v tabulkách XV. až XVII. V tabulce XV. je také uvedena ukázková syntéza využívající integrované RAM bloky FPGA, které jsou konfigurovány jako ROM paměti (výhodou může být úspora LUT tabulek; dle FPGA).

Tabulka XV. Výsledky syntézy pro model symbolové synchronizace s polyfázovým filtrem **GTED / ZCTED *(PAM) – 8 sub-filtrů (volitelně s ROM bloky)***

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** koeficienty – **16 bitů** *Polyfázový přizpůsobený SRRC filtr* - 97 koef. / 8 sub-filtrů/ 13 koef. každý	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Polyfázová synch. (LUT verze) - ***GTED*** f_{MAX}=143.01 MHz	6000 (6%) / 0	5695 (6%)	1785 (2%)	2/532 (<1%)
Polyfázová synch. (LUT verze) - ***ZCTED*** f_{MAX}=144.93 MHz	5956 (6%) / 0	5729(6%)	1785 (2%)	2/532 (<1%)
Polyfázová synch. (**ROM** verze) - ***GTED*** f_{MAX}=93.01 MHz	3202 (2%) / / 0	2936 (2%)	1305 (2%)	2/532 (<1%)

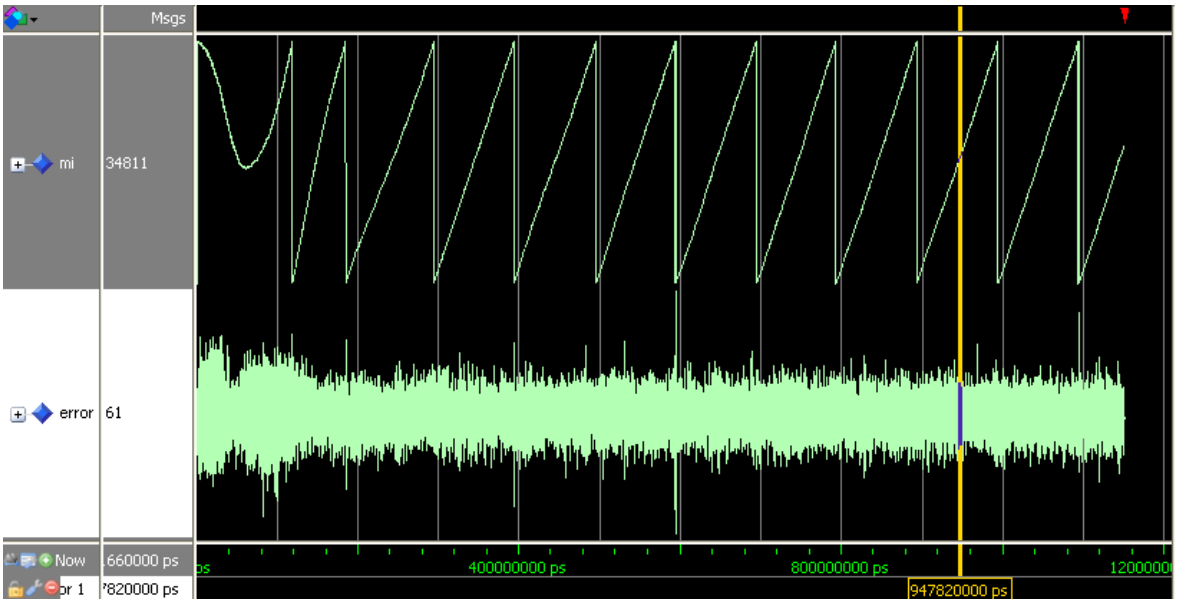
Tabulka XVI. Výsledky syntézy pro model symbolové synchronizace s polyfázovým filtrem **GTED / ZCTED *(PAM) – 16 sub-filtrů***

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** koeficienty – **16 bitů** *Polyfázový přizpůsobený SRRC filtr* - 193 koef. / 16 sub-filtrů/ 13 koef. každý	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Polyfázová synch. (LUT verze) - ***GTED*** f_{MAX}=90.5 MHz	**18941** (17%) / 0	18576 (16%)	2756 (2%)	6/532 (1%)
Polyfázová synch. (LUT verze) - ***ZCTED*** f_{MAX}=93.0 MHz	**18984** (17%) / 0	18608 (16%)	2756 (2%)	4/532 (<1%)

Tabulka XVII. Výsledky syntézy pro model symbolové synchronizace s polyfázovým filtrem **GTED / ZCTED *(I/Q) – 8 sub-filtrů***

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** koeficienty – **16 bitů** *Polyfázový přizpůsobený SRRC filtr* - 97 koef. / 8 sub-filtrů/ 13 koef. každý	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Polyfázová synch. (LUT verze) - ***GTED*** f_{MAX}=90.68 MHz	20825 (18%) / 0	20153 (18%)	5379 (5%)	8/532 (2%)
Polyfázová synch. (LUT verze) - ***ZCTED*** f_{MAX}=91.14 MHz	20877 (18%) / 0	20199 (18%)	5379 (5%)	4/532 (4%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

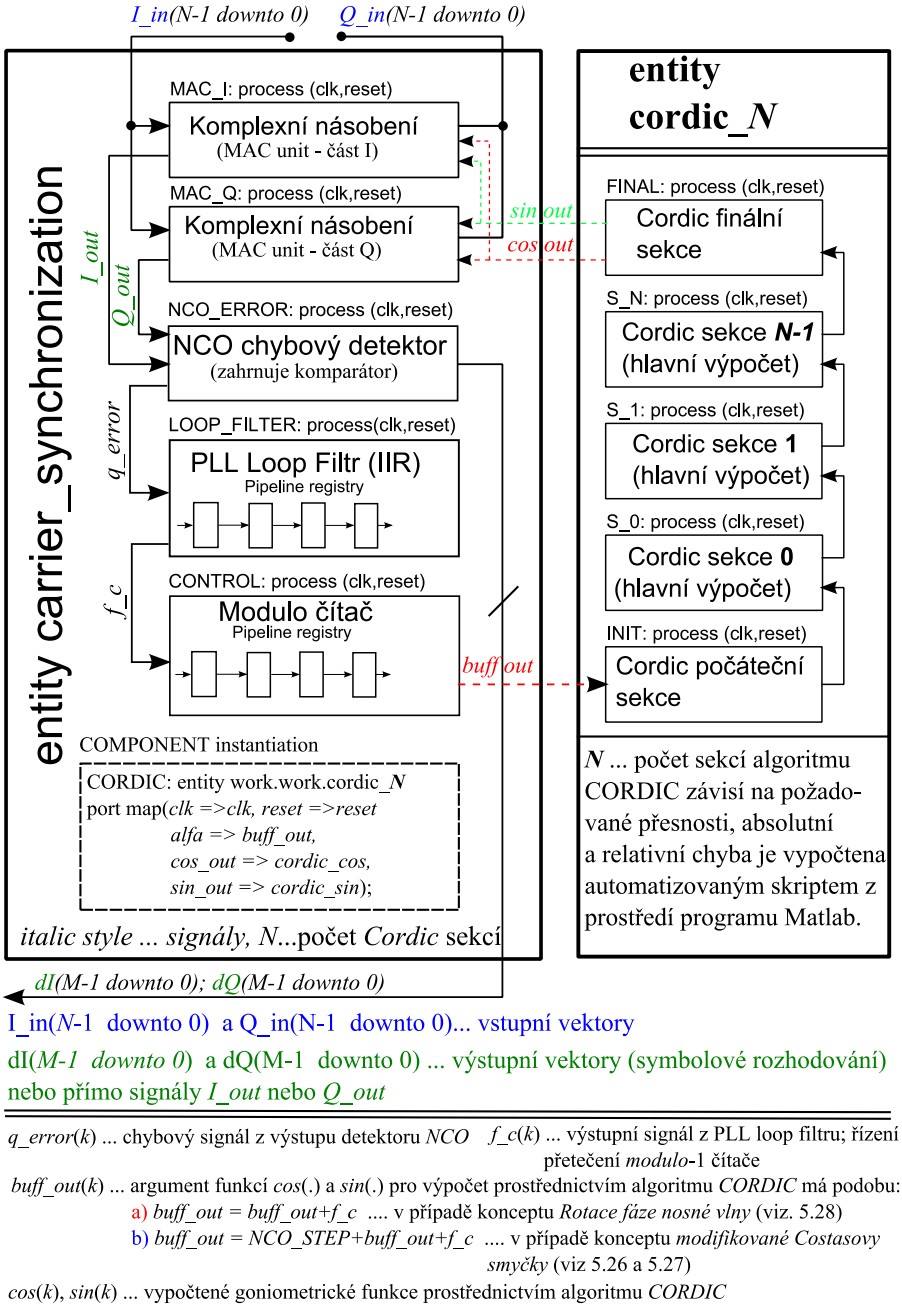


Obr. 7.10 RTL simulace v pevné řádové čárce pro zlomkový interval μ(k) a chybový signál $e(k)$ pro E_b/N_0=8 dB (GTED), je využito polyfázového filtru (8 sub filtrů). Vzorkovací frekvence je o 1/4000 symbolové rychlosti rychlejší než 2 vzorky/symbol. Porovnáno se simulací v programu Matlab.

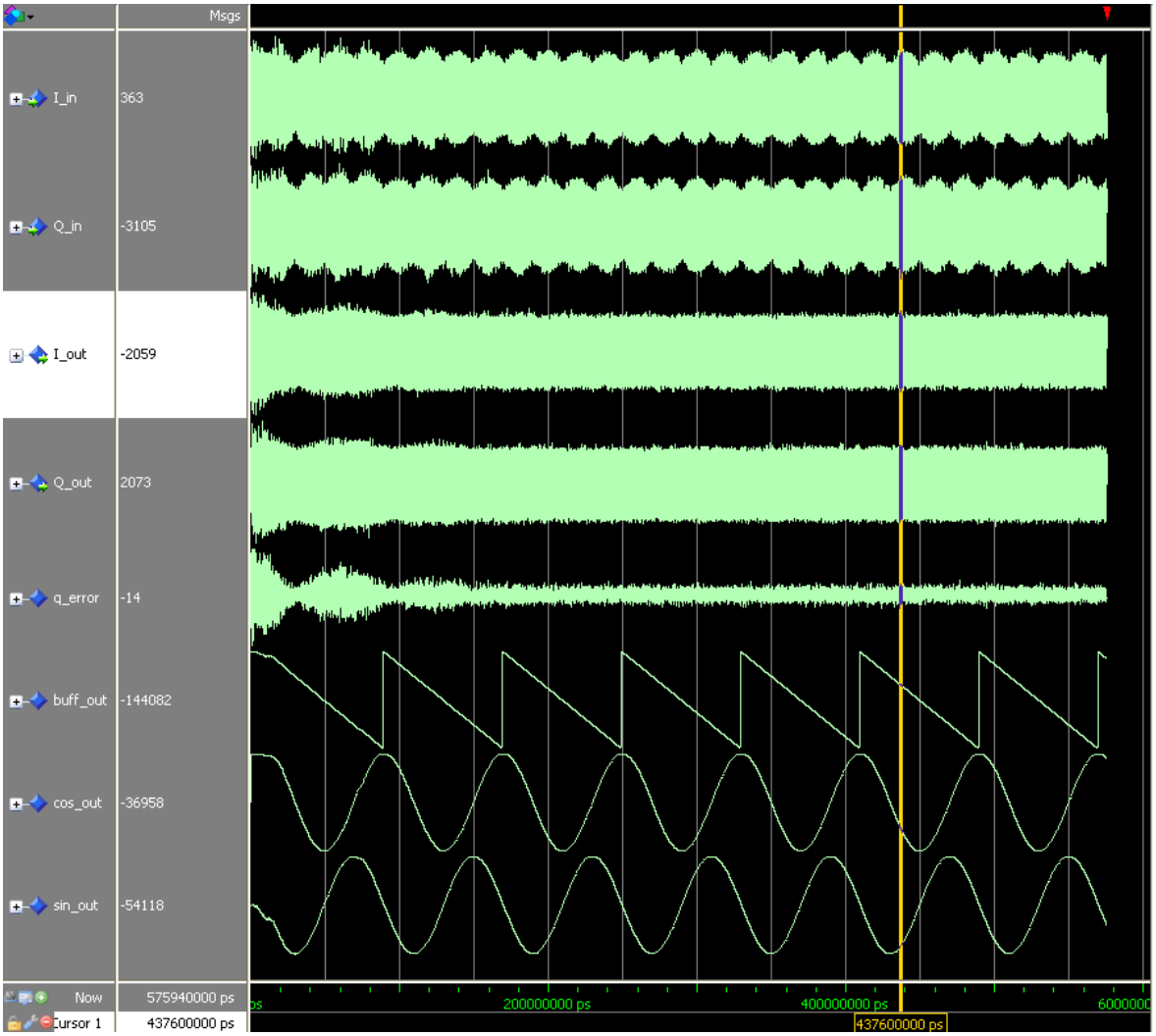
Jak již bylo uvedeno, polyfázový filtr plní zároveň úlohu přizpůsobeného filtru a interpolačního filtru zároveň. Další výhodou je snížení latence synchronizačního systému, neboť je vynechán právě vlastní interpolační filtr.

7.4 RTL simulace a syntéza modelu pro synchronizaci fáze nosné vlny

Při návrhu synchronizačního modelu pro synchronizaci fáze nosné vlny jsem vycházel ze schémat č. 5.12 a č. 5.13. Navržený model na obrázku č. 7.11 je konfigurovatelný a lze jej využít v případě konceptu *Rotace fáze nosné vlny* nebo i v případě konceptu modifikované *Costasovy smyčky*.



Obr. 7.11 Konfigurovatelná struktura VHDL modelu pro synchronizaci fáze nosné vlny, lze využít v případě konceptu *Rotace fáze nosné vlny* nebo *modifikované Costasovy smyčky.*



Obr. 7.12 Výstupy ze RTL simulace v pevné řádové čárce pro model synchro. fáze nosné vlny (koncept *Rotace fáze nosné vlny*). Počáteční fázový ofset byl 0.7 radiánu (~40°), E_b/N_0=8 dB. Z průběhů signálu *I_out* resp. *Q_out* je zřejmé, že po zavěšení *PLL* mají signály „hladký“ průběh (tzn. i správné zarovnání v I/Q diagramu).

Výsledky syntézy jsou uvedeny v tabulkách XVIII. a XIX. V tabulce XVIII. je uvedena syntéza pro model označený jako *Rotace fáze nosné vlny* a v tabulce XIX. pro modifikovanou *Costasovu smyčku*. V tabulkách je uvedena také samotná syntéza bloku pro výpočet goniometrických funkcí v rotačním módu algoritmu *Cordic*. Tento blok VHDL kódu je automaticky generován za pomocí skriptu z programu Matlab. Vstupní parametry skriptu tvoří kvantizační úroveň a počet sekcí algoritmu *Cordic*. Je možné také určit absolutní a relativní chybu výpočtu pomocí algoritmu *Cordic*. Počet sekcí algoritmu *Cordic* přímo odpovídá počtu sekcí zřetězeného zpracování (viz. obrázek č. 7.13).

Tabulka XVIII. Výsledky syntézy pro model *Rotace fáze nosné vlny*

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** *Algoritmus Cordic má 16 sekcí zřetězeného zpracování*	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Pouze algoritmus ***Cordic*** f_{MAX}=130.57 MHz	1486 (1%) / 0	1474 (1%)	774 (<1%)	4/532 (2%)
Rotace fáze nosné vlny s algoritmem ***Cordic*** f_{MAX}=96.8 MHz	1860 (2%) / 0	1851 (2%)	884 (<1%)	12/532 (2%)

Tabulka XIX. Výsledky syntézy pro *modifikovanou Costasovu smyčku*

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** *Algoritmus Cordic má 16 sekcí zřetězeného zpracování*	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Pouze algoritmus ***Cordic*** f_{MAX}=130.57 MHz	1486 (1%) / 0	1474 (1%)	774 (<1%)	4/532 (2%)
Modifikovaná Costasova smyčka s algoritmem ***Cordic*** f_{MAX}=95.8 MHz	1929 (2%) / 0	1920 (2%)	886 (<1%)	12/532 (2%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

```
if v_z(0) < 0 then
 v_x(1)<=v_x(0)+v_y(0);
 v_y(1)<=v_y(0)-v_x(0);
 v_z(1)<=v_z(0)+angles_pow(0);
else
 v_x(1)<=v_x(0)-v_y(0);
 v_y(1)<=v_y(0)+v_x(0);
 v_z(1)<=v_z(0)-angles_pow(0);
end if;
```

```
if v_z(1) < 0 then
 v_x(2)<=v_x(1)+v_y(1)/2;
 v_y(2)<=v_y(1)-v_x(1)/2;
 v_z(2)<=v_z(1)+angles_pow(1);
else
 v_x(2)<=v_x(1)-v_y(1)/2;
 v_y(2)<=v_y(1)+v_x(1)/2;
 v_z(2)<=v_z(1)-angles_pow(1);
end if;
```

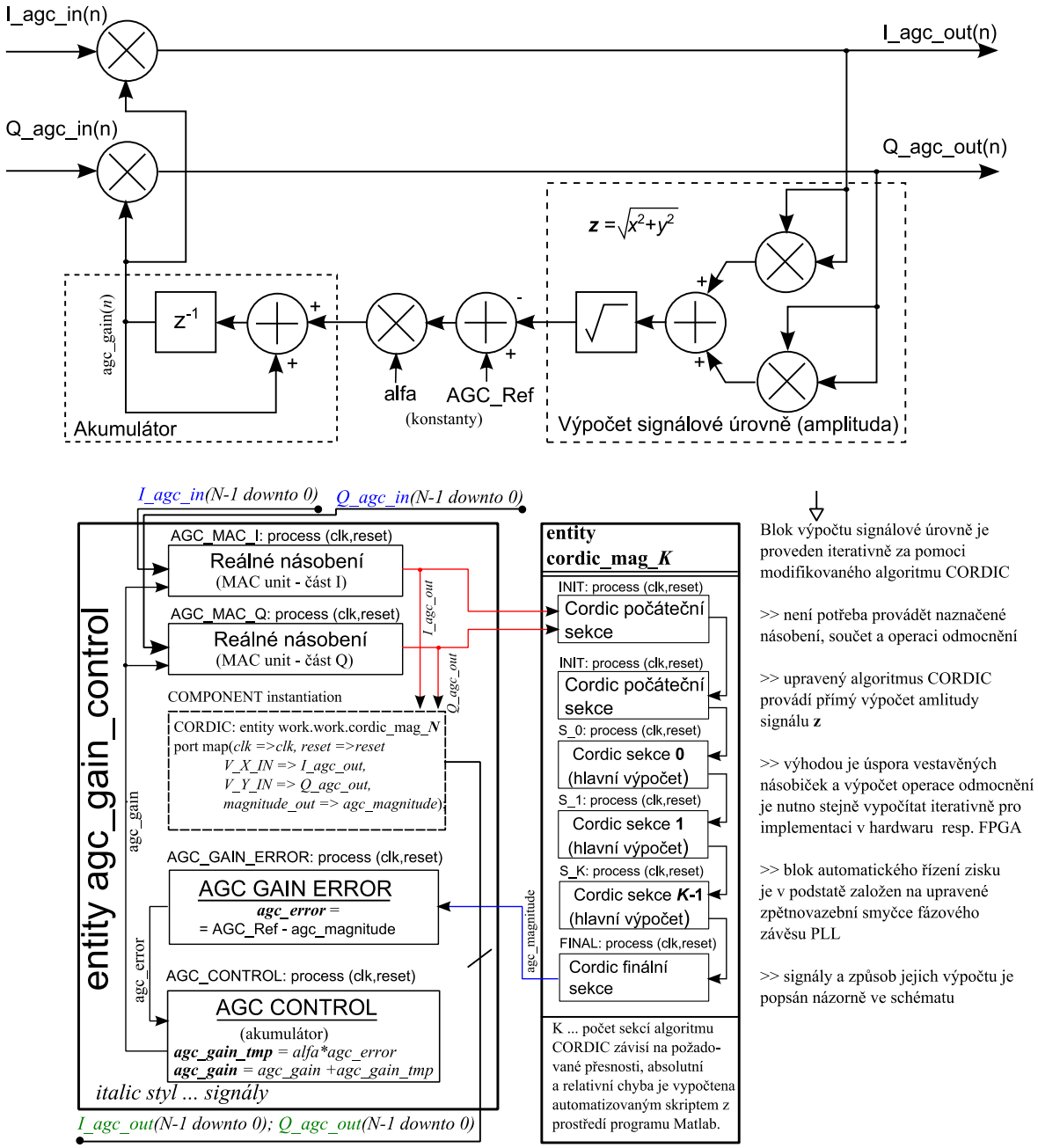
```
if v_z(2) < 0 then
 v_x(3)<=v_x(2)+v_y(2)/4;
 v_y(3)<=v_y(2)-v_x(2)/4;
 v_z(3)<=v_z(2)+angles_pow(2);
else
 v_x(3)<=v_x(2)-v_y(2)/4;
 v_y(3)<=v_y(2)+v_x(2)/4;
 v_z(3)<=v_z(2)-angles_pow(2);
end if;
```

v_x(0)<=to_signed(-CONST_ONE_INT_VALUE,CONST_V_X_Y); -- např. 65536 při 14 bitech
v_y(0)<=(others=>'0'); -- 0
v_z(0)<=alfa; -- *argument goniometrické funkce*
constant ***angles_pow***: angles_pow_type :=(51471,30385,16054,8149,4090,2047,1023,511,255,127,63,31,15,7,3,1);

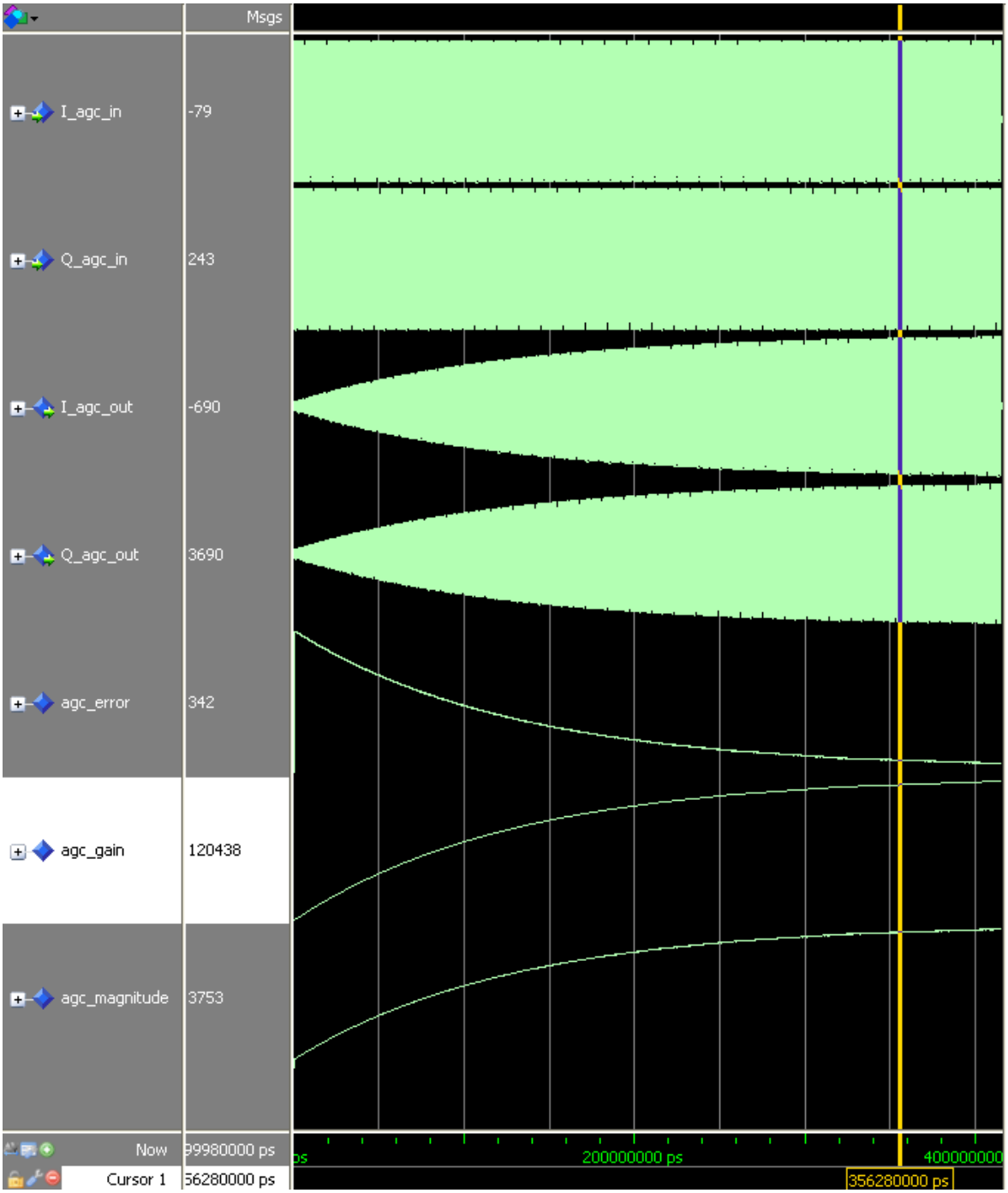
Obr. 7.13 Výpis VHDL kódu pro první 3 sekce zřetězeného zpracování *Cordic*

7.5 RTL simulace a syntéza bloku automatického řízení zisku AGC

Implementaci (digitálního) bloku automatického řízení zisku AGC je nutné provést pro správnou funkci symbolové synchronizace a synchronizace fáze nosné vlny (viz. konstanta K_p). Z tohoto důvodu uvádím na obrázku č. 7.14 navržené blokové schéma i VHDL strukturu systému AGC. Je využito zpětnovazební smyčky postavené na upravené struktuře PLL.



Obr. 7.14 Blokové schéma a VHDL struktura navrženého digitálního systému AGC



Obr. 7.15 Výstupy ze simulace v pevné řádové čárce pro model automatického řízení zisku AGC. Amplituda vstupní signálů byla nastavena na 500 a referenční hodnota na 4096. Časová konstanta je funkcí parametru *alfa* a úrovně vstupního signálu (jedná se o systém prvního řádu). Průběh signálu *agc_error* se musí ustálit na nulové hodnotě a signál *agc_magnitude* na referenční hodnotě (zde 4096).

Výsledky syntézy jsou uvedeny v tabulce XX. V tabulce je také uvedena samotná syntéza bloku pro výpočet amplitudy signálu $z = \sqrt{x^2 + y^2}$ s využitím algoritmu CORDIC [27]. Tento blok VHDL kódu je opět automaticky generován za pomocí skriptu z programu Matlab. Vstupní parametry skriptu tvoří kvantizační úroveň a počet sekcí algoritmu CORDIC, který v tabulce XX. označuji jako *CORDIC magnitude*. Počet sekcí algoritmu *CORDIC magnitude* přímo odpovídá počtu sekcí zřetězeného zpracování (obrázek č. 7.16).

Tabulka XX. Výsledky syntézy pro model automatického řízení zisku AGC

Výsledky syntézy: vstupní data – **12 bitů** konstanty - **16 bitů** *Algoritmus Cordic magnitude má 14 sekcí zřetězeného zpracování*	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
Pouze algoritmus ***Cordic magnitude*** f_{MAX}=128.55 MHz	1342 (1%) / 0	1320 (1%)	538 (<1%)	0/532 (0%)
Automatického řízení zisku AGC **s *Cordic magnitude*** f_{MAX}=85.98 MHz	1557 (1%) / 0	1547 (1%)	592 (<1%)	8/532 (2%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

```
if v_x(0) < 0 then
 v_x(1)<=v_x(0)+v_y(0);
 v_y(1)<=v_y(0)-v_x(0);
else
 v_x(1)<=v_x(0)-v_y(0);
 v_y(1)<=v_y(0)+v_x(0);
end if;
```

```
if v_x(1) < 0 then
 v_x(2)<=v_x(1)+v_y(1)/2;
 v_y(2)<=v_y(1)-v_x(1)/2;
else
 v_x(2)<=v_x(1)-v_y(1)/2;
 v_y(2)<=v_y(1)+v_x(1)/2;
end if;
```

```
if v_x(2) < 0 then
 v_x(3)<=v_x(2)+v_y(2)/4;
 v_y(3)<=v_y(2)-v_x(2)/4;
else
 v_x(3)<=v_x(2)-v_y(2)/4;
 v_y(3)<=v_y(2)+v_x(2)/4;
end if;
```

```
v_x(0)<=I_agc_out;
v_y(0)<=Q_agc_out;
```

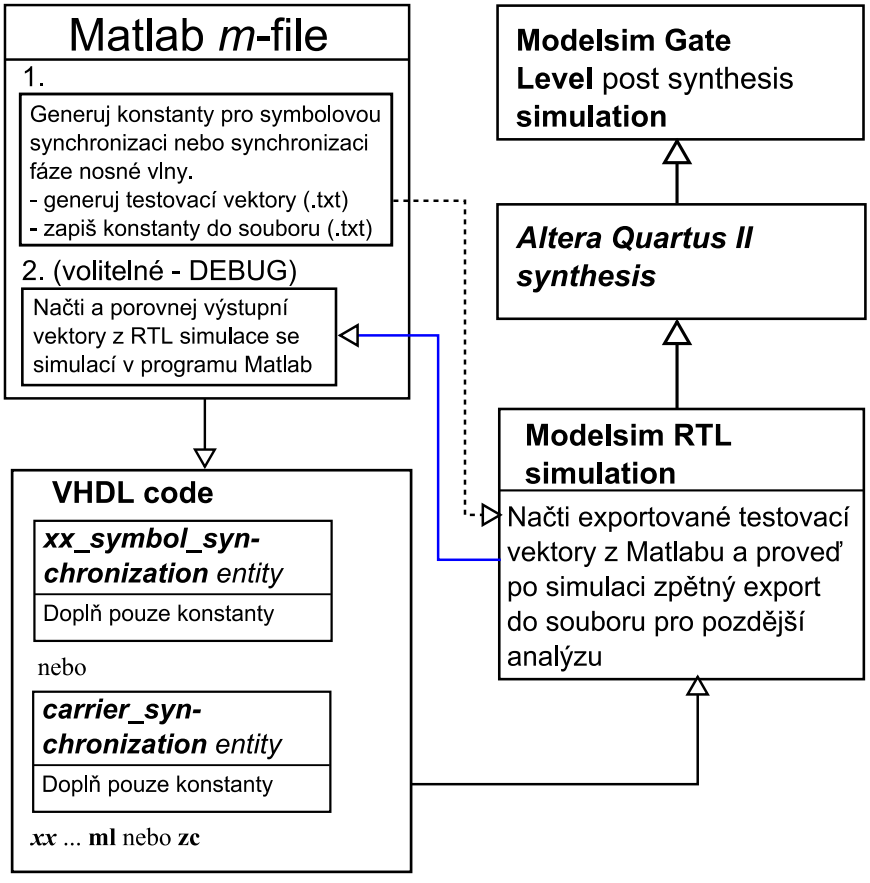
magnitude_out <= v_y(***k-1***); **--** ***k*** == počet sekcí algoritmu ***Cordic mag***.

Obr. 7.16 Výpis VHDL kódu pro první 3 sekce zřetězeného zpracování *Cordic magnitude*

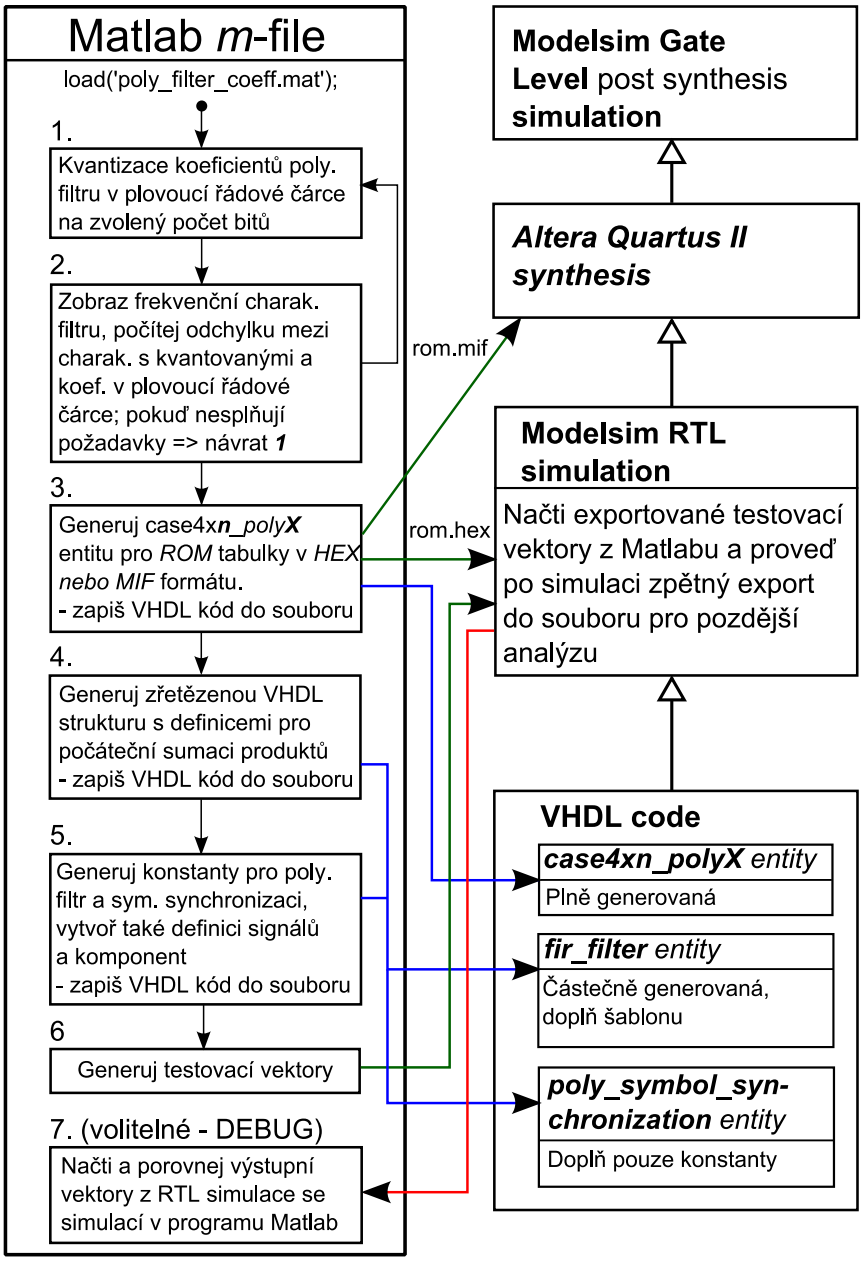
7.6 Vývojový cyklus

Vývojový cyklus začíná simulací navržené synchronizační struktury v prostředí programu Matlab. V případě konvenční struktury pro symbolovou synchronizaci (MLTED,ZCTED nebo GTED) nebo synchronizaci fáze nosné vlny je vývojový cyklus uveden na obrázku č. 7.17. Zde jsou exportovány pro RTL simulaci jen konstanty a testovací vektory. Vývojový cyklus se liší pro případ symbolové synchronizace využívající polyfázový filtr. Tento filtr je součástí synchronizačního modelu a z tohoto důvodu je návrhový cyklus složitější (obrázek č. 7.18).

Provedení úpravy příslušného entity je velice rychlé a trvá pouze několik minut. VHDL kód je vždy doplněn o konstanty nebo generované bloky kódu. Exportované vektory v pevné řádové čárce umožňují testovacímu programu (testbench) provést RTL simulaci v programu Modelsim. Testbench umožňuje také zpětný export výstupních vektorů do textového souboru, které lze potom analyzovat v prostředí programu Matlab.



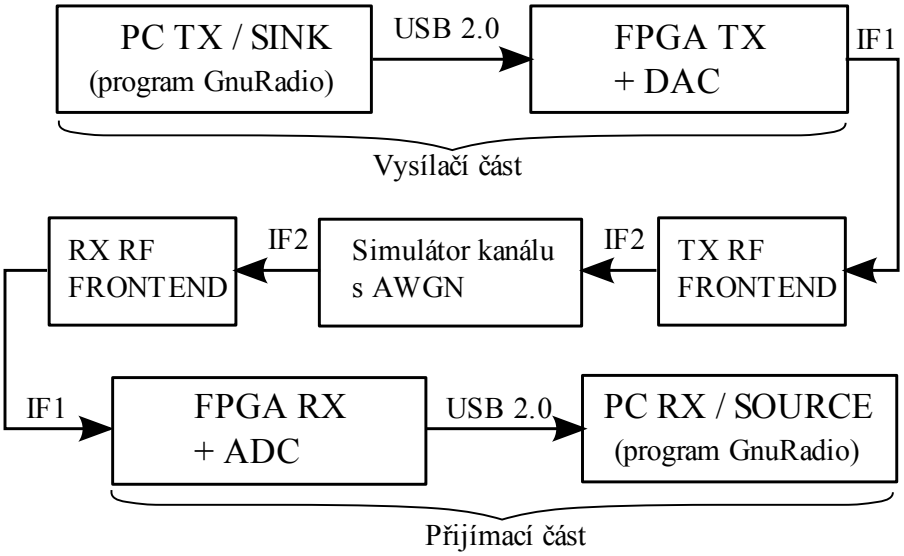
Obr. 7.17 Vývojový cyklus pro (konvenční) symbolovou nebo synchronizaci nosné vlny



Obr. 7.18 Vývojový cyklus pro symbolovou synchronizaci s polyfázovým filtrem

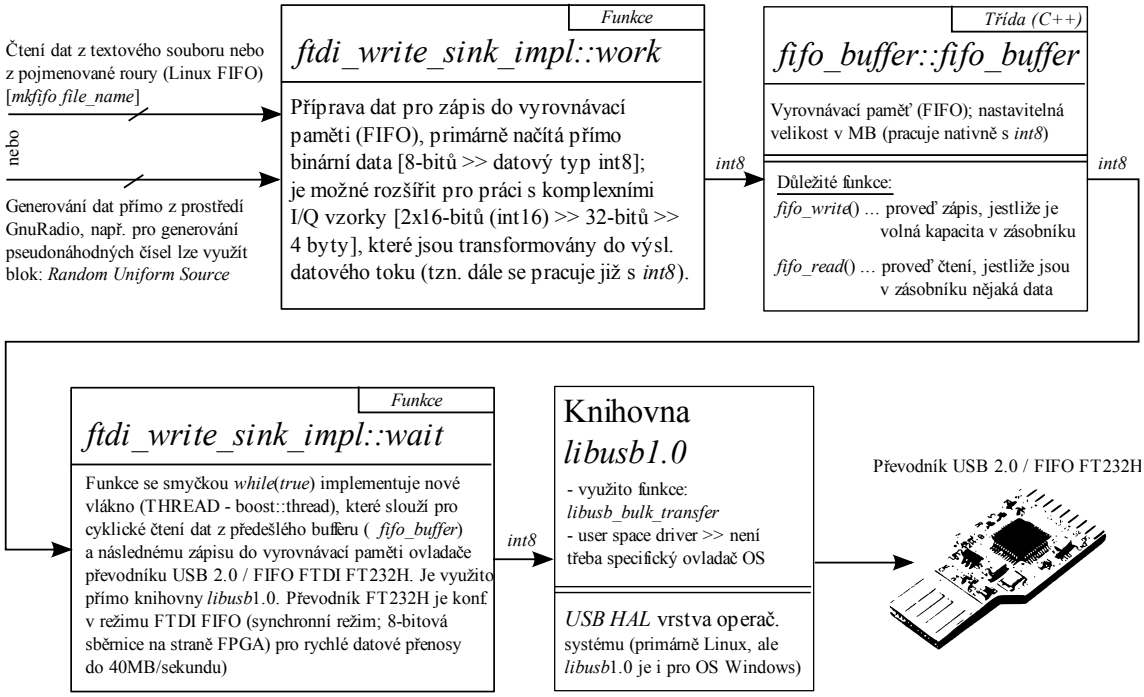
8 Navržená experimentální platforma softwarově definovaného rádia

Navržená hybridní experimentální platforma slouží především pro ověření a testování uvedených synchronizačních algoritmů, ale lze ji využít i jako SDR platformu pro všeobecné použití. Slovo hybridní v předchozí větě znamená, že signálové zpracování je rozděleno mezi PC a FPGA. Je využito modulové koncepce, kterou je možno upravit podle konkrétních požadavků. Blokové schéma je uvedeno na obrázku č. 8.1. Implementované schéma využívá modulační schéma QPSK podobně jako v případě simulačního procesu.



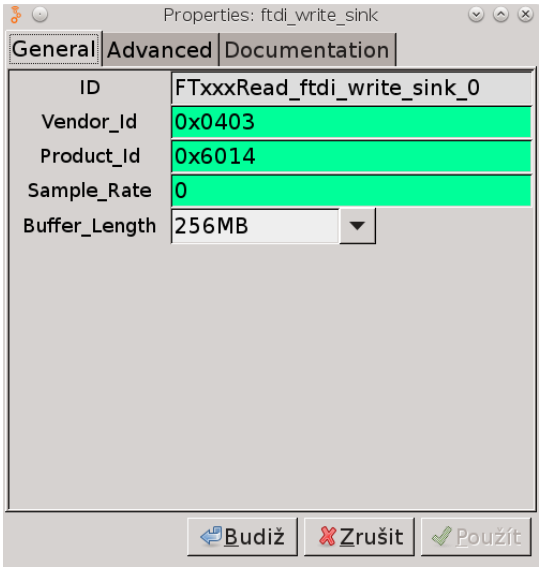
Obr. 8.1 Blokové schéma hybridní experimentální platformy SDR

Vysílací část zahrnuje zdrojový blok pro prostředí *GnuRadio*, který zajišťuje přenos dat a přístup k ovladačům převodníku na sběrnici USB 2.0. Převodník pracuje v režimu FIFO a na straně FPGA implementuje 8 bitovou paralelní sběrnici. Na straně hradlového pole se dále nachází vlastní vyrovnávací paměť (FIFO), signálové zpracování datového toku (diferenciální kódování, symbolové mapování, NRZ kodér, atd.), blok filtrace bázového signálu (SRRC filtr) a blok digitálního směšovače. Simulátor kanálu s AWGN je modelován prostřednictvím specializovaného měřicího přístroje *NoiseCom UFX-BER*. K dispozici je také analogový RX frontend, který obsahuje integrovaný tuner s řízením přes USB 2.0. Přijímač je v podstatě duální k vysílací části a liší se především synchronizační sekcí (zahrnuje určitý model symbolové synchronizace a synchronizaci fáze nosní vlny). Podrobný popis jednotlivých sekcí je uveden na obrázku č. 8.2, 8.3, 8.4, 8.5, 8.6 a 8.7.

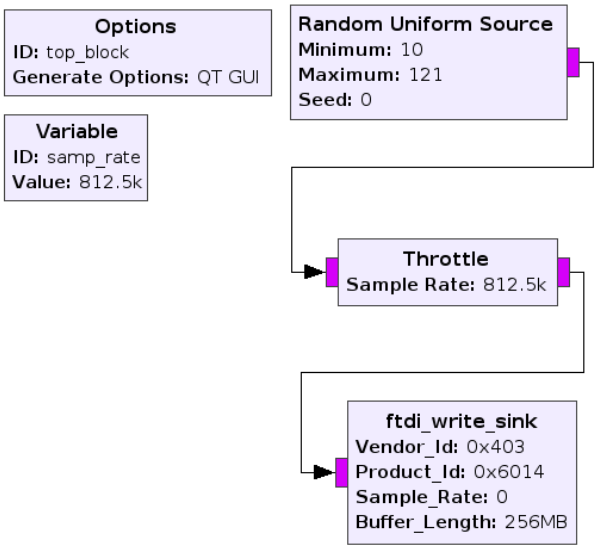


ftdi_write_sink_impl - třída v jazyce *C++*, která implementuje blok v prostředí programu GnuRadio [blok je konzumentem dat (sink)]

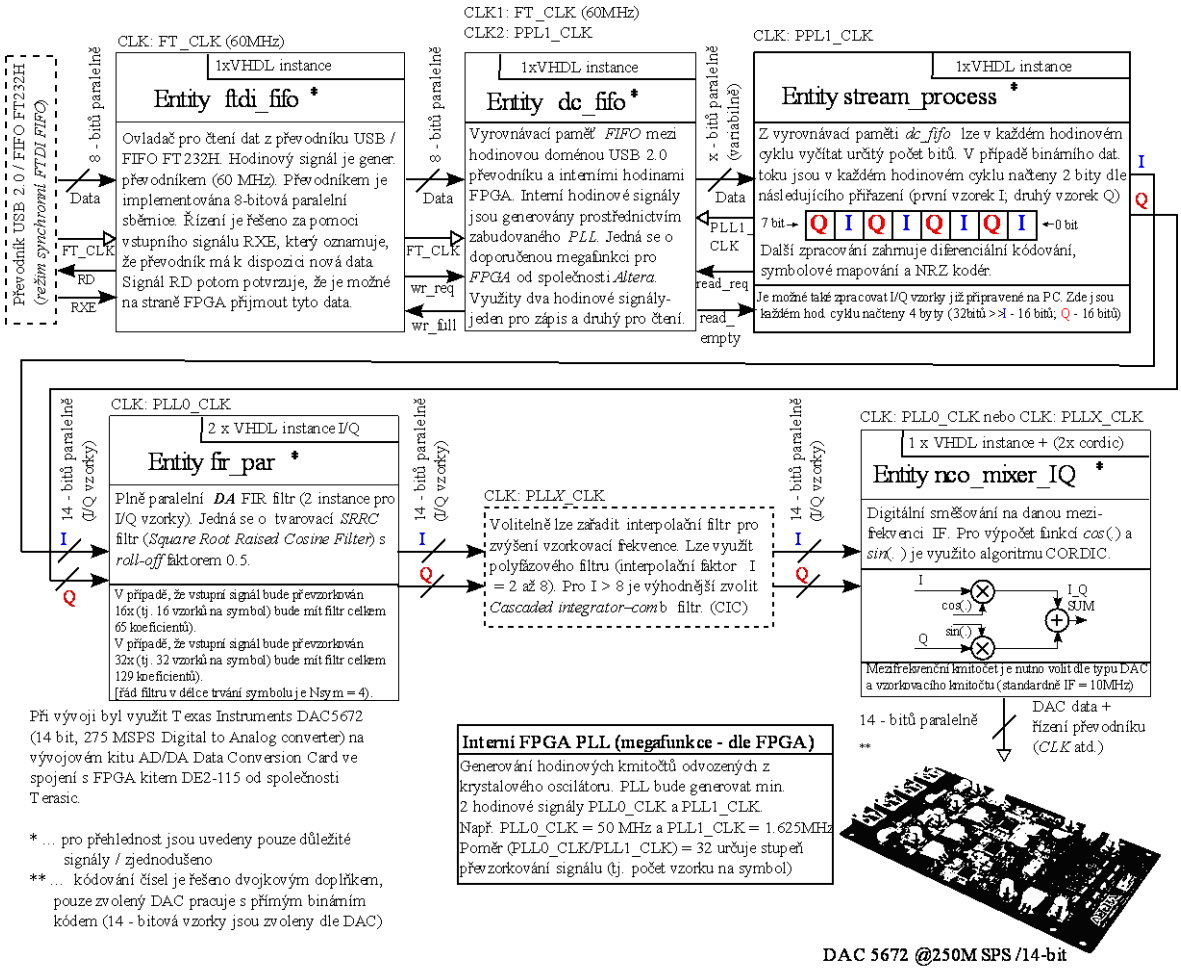
Navržený GnuRadio blok (*ftdi_write_sink*)



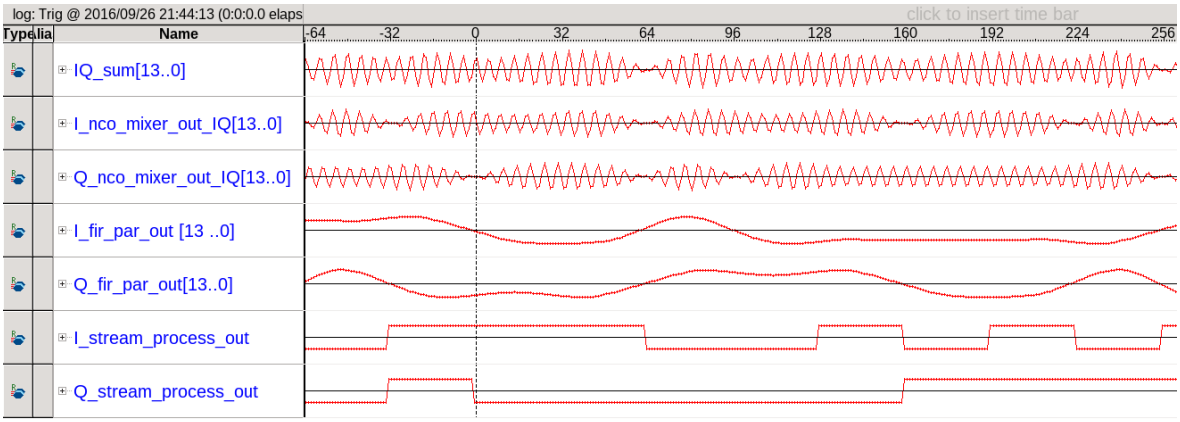
Ukázkový řetězec v prostředí *GnuRadio* pro generování pseudo náhodných dat s navrženým blokem *ftdi_write_sink*



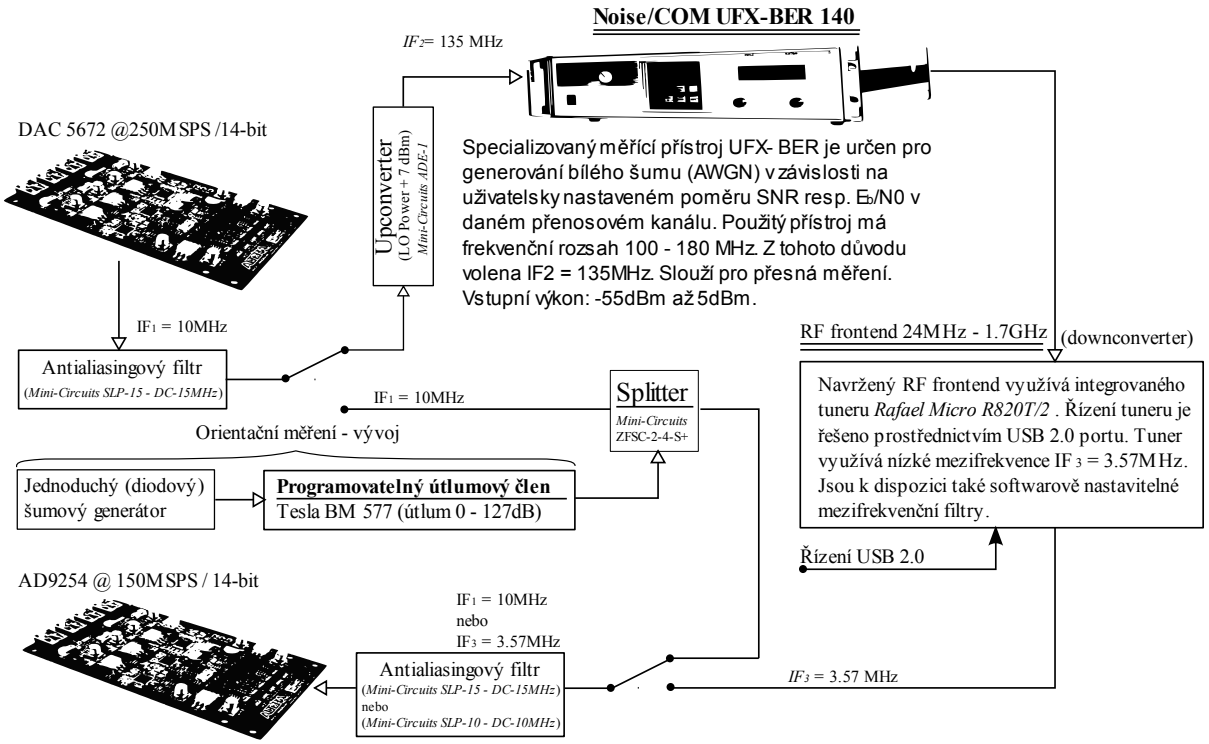
Obr. 8.2 Experimentální platforma SDR – *PC část TX*



Signály z výstupů jednotlivých sekcí navrženého modulátoru (popsáno logicky dle názvů jednotlivých sekcí od spodu směrem k převodníku DAC)

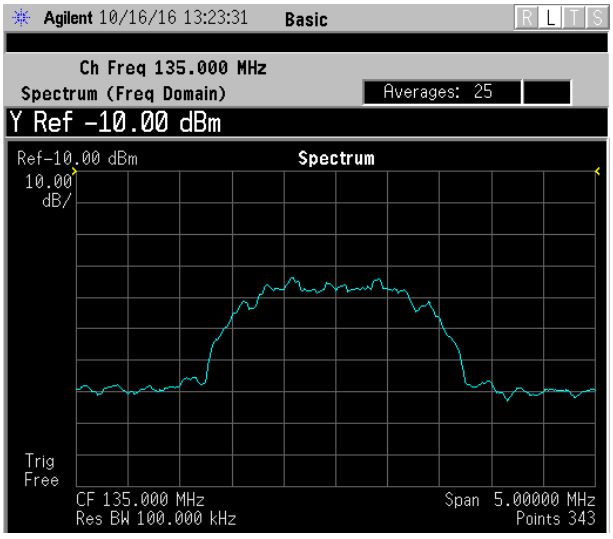


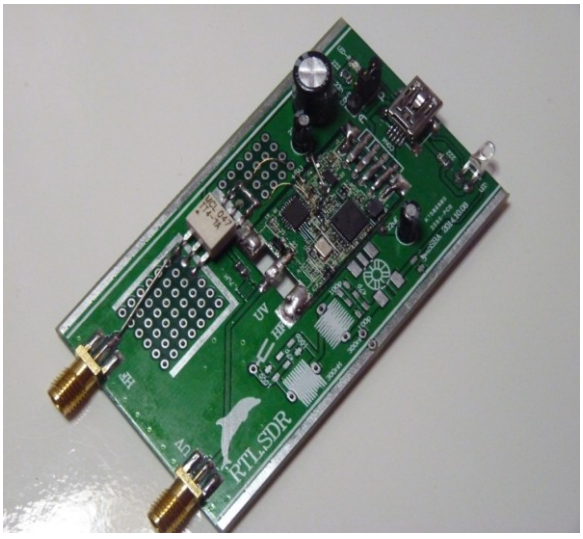
Obr. 8.3 Experimentální platforma SDR – *FPGA část TX*



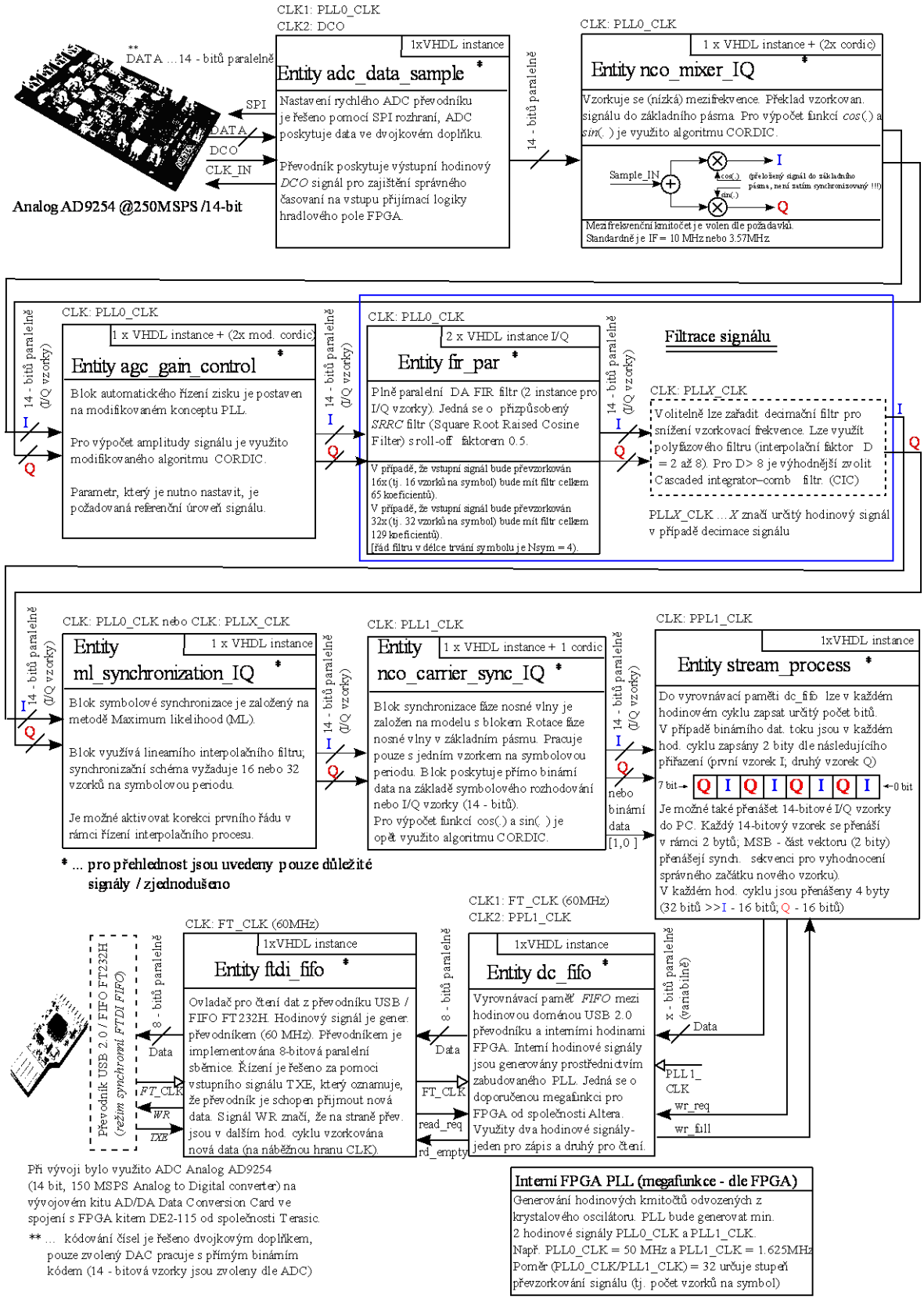
Spektrum signálu QPSK IF_2 = 135MHz
v_p = 3.2 Mbit/s (E_b/N_0 = 10 dB)

Navržený RX frontend s integrovaným tunerem R820T/T2

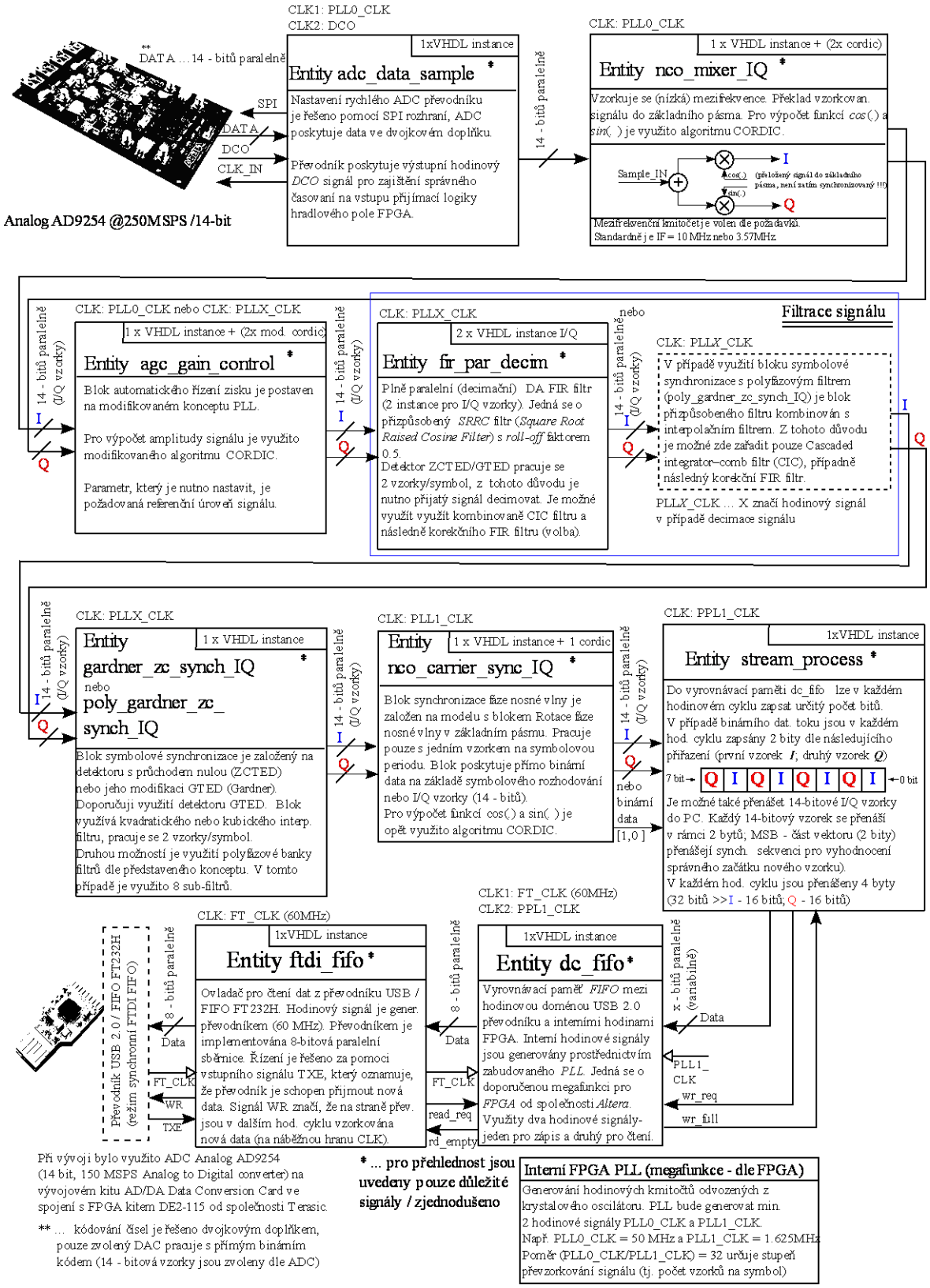




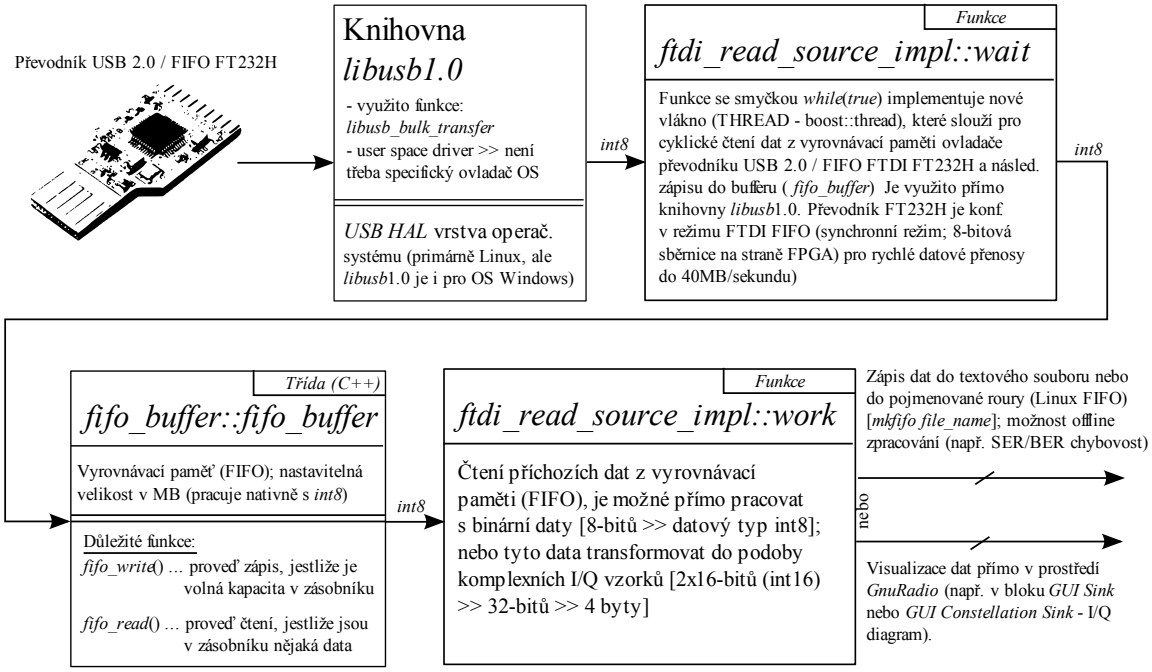
Obr. 8.4 Simulace přenosového AWGN kanálu s využitím specializovaného měřicího přístroje *NoiseCom UFX-BER 140*



Obr. 8.5 Experimentální platforma SDR – *FPGA část RX* (využito modelu symbolové synchronizace s detektorem MLTED)



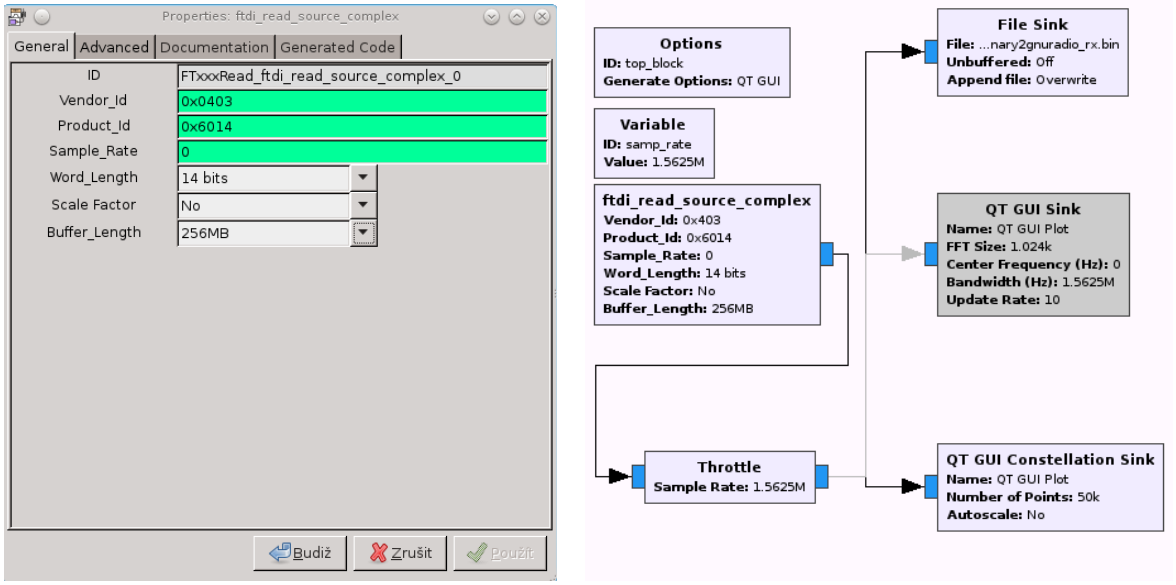
Obr. 8.6 Experimentální platforma SDR – *FPGA část RX* (využito modelu symbolové synchronizace s detektorem ZCTED/GTED). S výhodou je možné také využít polyfázové banky filtrů podle představeného modelu z kapitoly 4.4 (volba).



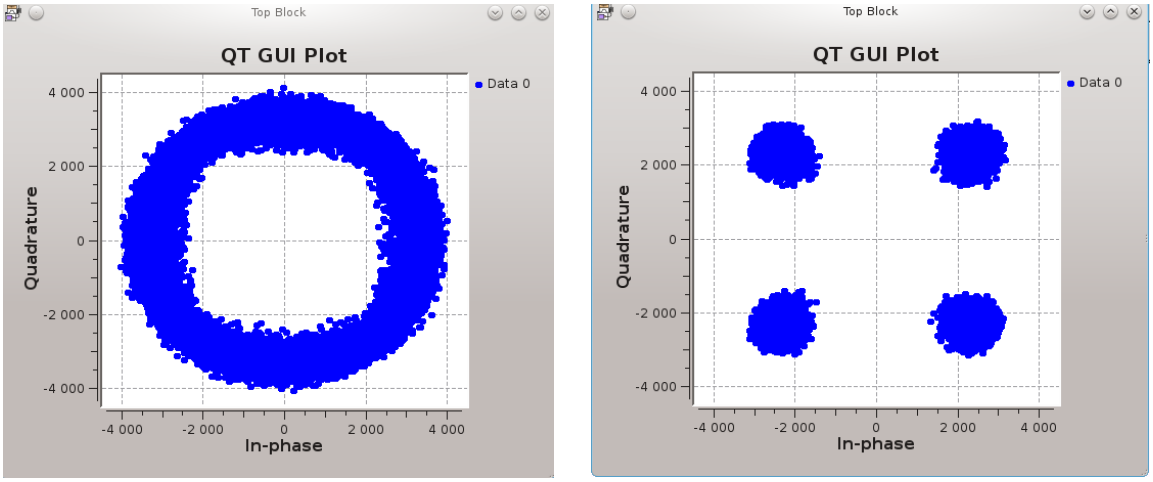
ftdi_read_source_impl - třída v jazyce *C++*, která implementuje blok v prostředí programu GnuRadio [blok je konzumentem dat (source)]

Navržený GnuRadio blok (*ftdi_read_source_complex*)

Ukázkový řetězec v prostředí *GnuRadio* pro čtení dat (příchozích RX FPGA) s navrženým blokem *ftdi_read_source_complex*

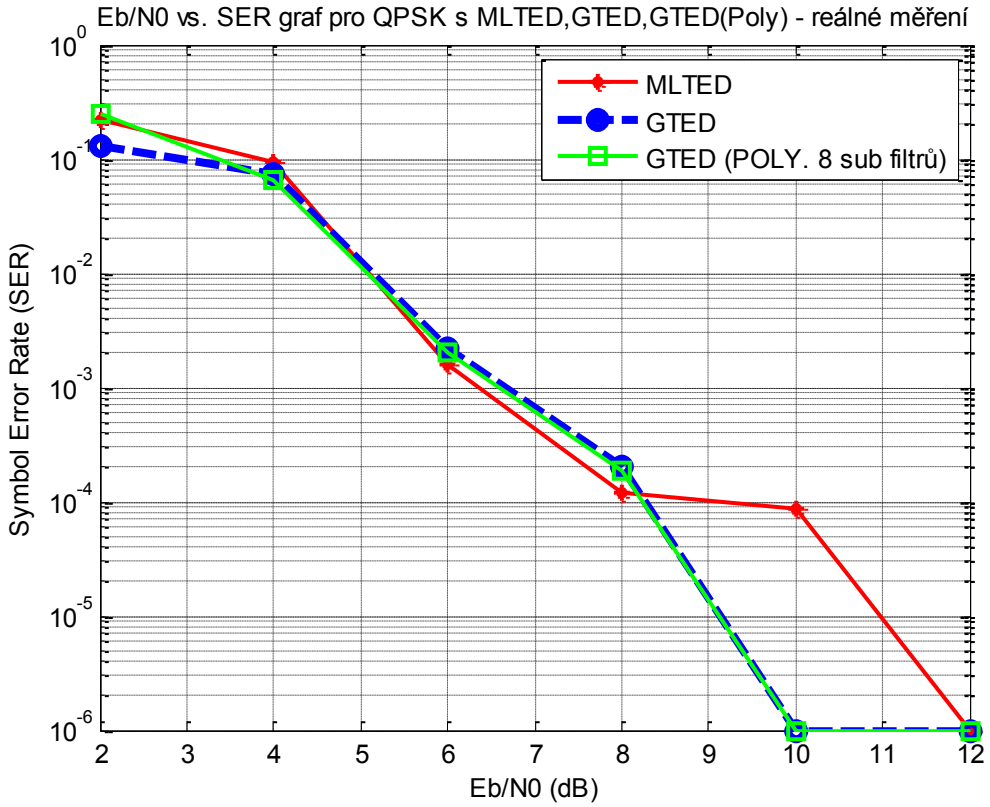


Obr. 8.7 Experimentální platforma SDR – *PC část RX*



Obr. 8.8 Konstelační digramy na straně realizovaného experimentálního SDR přijímače pro E_b/N_0=12 dB. Detektor je typu *MLTED* (32 vzorků/symbol) – zleva – po provedení symbolové synchronizace, po synchronizaci fáze nosné vlny. Vizualizace – *GnuRadio*.

Vyhodnocení chybovosti symbolů bylo provedeno pro ověření vlastností detektorů symbolové synchronizace v reálném přenosovém prostředí (obrázek č. 8.9).



Obr. 8.9 Reálné vyhodnocení *SER* vs. E_b/N_0 pro detektory *MLTED*, GTED a GTED s polyfázovou bankou filtrů, synchronizace fáze nosné vlny provedena po symbolové synchronizaci, průměrování probíhalo po částech – 1000 měřících cyklů / 72 symbolů, při vyhodnocení se synchronizuji na určitý vložený stále se opakující vzorek symbolů.

Pro porovnání byly vybrány detektory symbolové synchronizace MLTED, GTED a GTED s polyfázovou bankou filtrů. Měření chybovosti SER probíhá offline v programu *Matlab* na základě získaných resp. uložených vzorků z prostředí *GnuRadio*. Chybovost je poté vyhodnocována po částech a data jsou statisticky zpracována. Výsledky jsou srovnatelné, pouze detektor MLTED vychází trochu hůře pro $E_b/N_0 > 6$. Dále uvádím výsledky kompletní syntézy pro představenou experimentální platformu SDR v následující tabulce.

Tabulka XXI. Výsledky syntézy pro navrženou experimentální platformu SDR

Výsledky syntézy: vstupní data – **14 bitů** Konstanty symbolové syn. a synchronizace fáze nosné vlny – **17 bitů**	**Cyclone IV EP4CE115F29C7 (115.200 LEs)** *Vývojové prostředí Altera Quartus 14.1*			
	LUTs/RAM MEM. bitů	Počet kombinačních funkcí	Počet registrů	Počet využitých integrovaných násobiček
FPGA modulátor QPSK TX (vysílací strana) f_{MAX}=90.03 MHz	37.211 (33%) / 131.072 (3%)	33.113 (29%)	36.163 (32%)	8/532 (2%)
Demodulátor QPSK RX (detektor MLTED) f_{MAX}=85.8 MHz	35.723 (31%) /524.848 (13%)	29.923 (26%)	32.241 (28%)	44/532 (8%)
Demodulátor QPSK RX (detektor GTED), kubický interpolační filtr f_{MAX}=78.04 MHz	37.680 (33%) /525.078 (13%)	31.316 (27%)	34.053 (30%)	44/532 (8%)
Demodulátor QPSK RX (detektor GTED), s polyfázovým filtrem (+decimační CIC filtr) f_{MAX}=91.5 MHz	32.470 (33%) /525.078 (13%)	29.050 (26%)	32.174 (28%)	30/532 (6%)

Altera Quartus II parametry syntézy: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

Z výsledků uvedené finální syntézy je patrné, že nejmenší počet logických prvků FPGA obsadí demodulátor s detektorem symbolové synchronizace GTED využívající polyfázového filtru. Tento systém symbolové synchronizace využívá zároveň přizpůsobeného filtru jako interpolačního filtru. To představuje velkou výhodu, neboť pro snížení vzorkovací frekvence před symbolovou synchronizací lze využít CIC filtr (*Cascaded integrator–comb filter*) a není nutno tedy aplikovat další přizpůsobený FIR filtr.

Detektor MLTED s lineárním interpolačním filtrem má zase výhodu, že pracuje s velkým počtem vzorků na symbolovou periodu (obvykle 16 nebo 32) a decimace signálu před symbolovou synchronizací často není nutná (dle konkrétní implementace). Pro úplnost uvádím podobu experimentálního pracoviště pro testování navržené platformy SDR na obrázku č. 8.10.

Obr. 8.10 Experimentální pracoviště pro vývoj platformy SDR. Měřící pracoviště zahrnuje signálový vektorový analyzátor *Agilent E4406a* (DC – 4.4GHz) a speciální měřicí přístroj *NoiseCom UFX-BER 140* (simulátor AWGN kanálu na základě nastavené hodnoty E_b/N_0 nebo C/N). Vlastní SDR je implementováno na dvou vývojových deskách od firmy *Terasic DE2-115* (*FPGA Altera Cyclone IV*) s rozšiřující rychlou kartou ADC/DAC převodníků - AD/DA Data Conversion Card).

9 Výsledky a přínos práce

V úvodu práce je stručně shrnuta základní problematika využití fázového závěsu, a to zejména z pohledu synchronizace fáze nosné vlny a symbolové synchronizace. Pro potřeby návrhu efektivních synchronizačních systémů fáze nosné vlny jsem nejprve provedl rozbor diskrétní podoby Costasovy smyčky. V rámci rozboru jsem především řešil kompenzaci fázového ofsetu a konstantního frekvenčního ofsetu. V rámci symbolové synchronizace jsem se nejprve zaměřil na popis obecného zpětnovazebního modelu. Na základě odvození chyby časování symbolů v diagramu oka jsem vytvořil základní modely využívající detektor maximální věrohodnosti a „*early-late*“ detektor. Dále jsem stanovil základní požadavky na synchronizační model pracující s pevnou vzorkovací periodou *T*, který využívá interpolačního filtru. Tento ideový model s interpolátorem se stal základem pro následně navržené systémy symbolové synchronizace a je optimální pro implementaci v hardwaru resp. FPGA.

Z důvodu značného využití digitálních filtrů v oblasti synchronizačních systémů jsem provedl návrh efektivních struktur FIR filtrů s možností implementace v určitém HDL jazyce a následnou syntézou na hradlovém poli FPGA. Vytvořené struktury zahrnují sériový model, plně paralelní model a paralelní polyfázové modely využitelné jako decimační nebo interpolační filtry. Navržené struktury jsou postaveny na distribuované aritmetice. Plně paralelní model (včetně odvozených polyfázových struktur) je velice flexibilní a škálovatelný, a proto byl vybrán pro další modifikace související s oblastí synchronizace. Sériový model nenalezl většího uplatnění. Důležitou aplikací byl interpolační filtr s *Farrow* strukturou, kde jsou zařazeny zároveň dva nebo tři paralelní filtry dle typu interpolačního filtru (kvadratický resp. kubický interpolátor). Modifikovaná struktura interpolačního polyfázového filtru byla také využita při návrhu specifického systému symbolové synchronizace. Automatizovaný vývojový cyklus dále umožňuje podstatně zkrátit čas návrhu konkrétního filtru. Z výše uvedených skutečností plyne, že se podařilo navrhnout optimální struktury s těmito podstatnými vlastnostmi:

- optimální využití logických prvků FPGA
- jednoduchá přenositelnost díky využití standardních knihoven
- velmi dobrý výkon i ve srovnání s komerčními IP bloky
- úspora vestavěných násobiček z důvodu využití distribuované aritmetiky
- možnost modifikace struktur pro specifické účely (problematické s IP bloky)

Při návrhu efektivních modelů symbolové synchronizace pro syntézu na FPGA jsem nejprve provedl odvození detektoru MLTED na základě pravděpodobnostní teorie maximální věrohodnosti. Provedl jsem analýzu využití tohoto detektoru pro případ známé i neznámé symbolové posloupnosti pro přijatý signál s modulací QPSK, který lze jednoduše rozšířit pro modulace typu *m*-QAM. Soubor detektorů jsem ještě rozšířil o odvozený detektor průchodu nulou (ZCTED) a jeho čistě NDA rozšíření nazvané Gardnerův detektor (GTED). Vlastnosti detektorů ve zpětnovazební smyčce PLL jsem ověřil na základě simulací *S*-křivek a navrhnul jsem metodu pro jejich experimentální měření. Odvozené detektory nelze většinou přímo implementovat v pevné řádové čárce v hardwaru, z tohoto důvodu jsem provedl jejich analýzu a navrhnul možné úpravy. Na základě uvedených detektorů jsem již vytvořil zpětnovazební modely symbolové synchronizace s PLL.

V rámci jednotlivých modelů symbolové synchronizace jsem se především zaměřil na vlastní interpolační filtr a řízení interpolačního procesu. Speciálně navržené struktury interpolačních filtrů zahrnují lineární, kvadratický a kubický interpolátor. Kvadratický interpolátor bylo možné využít na základě rozboru jeho impulsní odezvy a následně provedených úprav. Kvadratický a kubický filtr jsou navrženy s využitím speciální struktury plně paralelního DA FIR filtru, jak již bylo výše uvedeno. Řízení interpolačního procesu prostřednictvím *modulo*-1 čítače umožňuje efektivním a relativně jednoduchým způsobem řídit celý interpolační proces. Uvedený algoritmus řízení bez využití obecného dělení je možno také dobře implementovat. Na základě získaných poznatků s návrhem modelů využívající určitý typ interpolačního filtru jsem vytvořil model symbolové synchronizace s využitím bank polyfázového filtru. Tento model zahrnuje modifikovanou strukturu polyfázového interpolačního filtru. Tento polyfázový filtr plní funkci přizpůsobeného a interpolačního filtru zároveň. Výsledkem je především celkové snížení latence synchronizačního systému a možná úspora logických prvků dle komplexnosti návrhu.

Synchronizace fáze nosné vlny těsně souvisí se symbolovou synchronizací. Z tohoto důvodu jsem navrhnul možná uspořádání celého synchronizačního systému, která jsem následně v rámci simulačního procesu návrhu experimentálně ověřil. Fázový detektor synchronizace nosné vlny jsem se rozhodl řešit opět na základě metod maximální věrohodnosti. První navržený model s tímto detektorem představuje modifikaci Costasovy smyčky pro *m*-QAM signály. Druhý navržený model s blokem *Rotace fáze nosné vlny* v základním pásmu představuje čistě diskrétní přístup a je určený pro zařazení do

synchronizačního systému za blokem symbolové synchronizace. Blok přímé digitální syntézy je založen na škálovatelném algoritmu CORDIC.

V rámci simulačního procesu jsem ověřil funkci a vlastnosti navržených modelů. V prostředí programu Matlab jsem vytvořil kompletní přenosové řetězce. Tento řetězec zahrnuje modulátor, model kanálu a koherentní demodulátor, kde jsou právě zařazeny vytvořené bloky symbolové synchronizace a synchronizace fáze nosné vlny. Je využito modulace QPSK. V rámci simulace s detektorem MLTED pro symbolovou synchronizaci jsem nejprve ověřil funkci lineárního interpolačního filtru. Na základě simulací jsem zjistil, že je výhodné pro interpolační filtr využit tzv. první diference, neboť poměrně přesně aproximuje ideální diferenci. Dále jsem vyhodnocoval vliv zařazení bloku symbolové synchronizace před blokem synchronizace fáze nosné vlny a naopak. Vyhodnocení chybovosti symbolů bylo provedeno pro porovnání vlastností představených modelů. Z výsledků simulací vyplývá, že provedení symbolové synchronizace před synchronizací fáze nosné vlny je především výhodné v případě malého odstupu *SNR* resp. $E_b/N_0 < 8$. Chybovost je menší zhruba o 20% až 30% než v případě konceptu modifikované Costasovy smyčky. Doporučuji tedy nejprve provést symbolovou synchronizaci a následně zařadit blok synchronizace fáze nosné vlny. Vyhodnocení chybovosti bylo provedeno také pro porovnání vlastností ZCTED a GTED detektorů. Simulace potvrdila, že pro větší odstup SNR respektive pro $E_b/N_0 > 6$ dB GTED předčí ZCTED. Detektor GTED je rotačně invariantní detektor, který je nezávislý na rotaci fáze nosné vlny. Symbolové synchronizace s polyfázovým filtrem také potvrdila, že je výhodnější využít detektor GTED místo ZCTED. Výběr mezi modifikovaným kvadratickým a kubickým filtrem neměl zásadní vliv na funkčnost celého synchronizačního systému.

Na základě simulačních modelů jsem provedl návrh jednotlivých synchronizačních modelů v jazyce VHDL. Pro každý model jsem provedl RTL simulaci a následnou ukázkou syntézu na FPGA. Určitý model symbolové synchronizace byl nejprve navržen pro práci s PAM signály a následně provedeno rozšíření pro I/Q signály (QPSK; *m*-QAM). Na základě provedené syntézy model symbolové synchronizace s detektorem MLTED obsadil nejmenší počet logických prvků FPGA. Model využívá lineárního interpolačního filtru. Z tohoto důvodu je nutno počítat s tím, že tento synchronizační systém ale pro svoji správnou funkci potřebuje z výstupu přizpůsobeného filtru větší počet vzorků na symbolovou periodu. Zjištěný optimální počet je 16 nebo 32 vzorků/symbol. Modely

symbolové synchronizace s detektorem ZCTED resp. GTED obsadí zhruba dvojnásobný počet logických zdrojů proti modelu s MLTED. Uvedené navýšení je vzhledem k počtu logických prvků současných obvodů FPGA ovšem zanedbatelné. Zásadní vliv na počet obsazených prvků mají především plně paralelní struktury FIR filtrů. V případě syntézy symbolové synchronizace s polyfázovým filtrem je vhodné volit banku filtrů s 8 sub-filtry, neboť vlastní polyfázový filtr má zásadní dopad na využití zdrojů v rámci syntézy. V případě využití decimačního CIC filtru ovšem tento přístup při celkové syntéze demodulátoru může obsadit nejmenší počet logických zdrojů FPGA, neboť polyfázový filtr kombinuje zároveň funkci interpolačního a přizpůsobeného filtru. Pro správnou funkci synchronizačních algoritmů byl nakonec vytvořen model automatického řízení zisku, který pro výpočet amplitudy signálu využívá speciálně upravenou verzi algoritmu CORDIC.

Jako prostředek pro ověření navržených modelů jsem navrhnul experimentální platformu softwarově definovaného rádia. Tato hybridní struktura SDR využívá pro zpracování číslicového signálu částečně PC a následně logických zdrojů FPGA. Platforma obsahuje vysílací část, analogový RF frontend a přijímací část, kde jsou implementovány navržené synchronizační modely. Data jsou z/do PC přenášena prostřednictvím sériového rozhraní USB 2.0. Velkou výhodou je možnost rychlé vizualizace v jakémkoli bodě synchronizačního řetězce v prostředí programu *GnuRadio*. Modely jsem otestoval v reálném přenosovém prostředí s využitím speciální měřicího přístroje *NoiseCom UFX-BER*, který umožňuje simulovat reálný AWGN kanál.

10 Závěr

Cílem této práce bylo navrhnout efektivní synchronizační algoritmy a provést jejich optimalizaci tak, aby bylo možné je využít v rámci konceptu SDR částečně implementovaného na hradlovém poli FPGA. Prostředkem pro splnění cílů byl rozbor diskrétního fázového závěsu, rozbor vhodných synchronizačních metod, návrh jednotlivých synchronizačních modelů a jejich následná optimalizace. Všechny tyto cíle práce byly splněny, neboť nově navržené synchronizační modely splňují požadavky na efektivní a rychlou implementaci na FPGA. Synchronizační modely se podařilo optimalizovat z hlediska využití logický prvků, vestavěných násobiček a paměťových bloků. Důležitým cílem práce byla možnost rychlé úpravy navrženého synchronizačního modelu pro konkrétní přijímač. Tento požadavek je také splněn, neboť je využito automatizovaných skriptů pro generování kódových bloků a následně navržené automatické testovací procedury umožňují podstatně zkrátit vývojový cyklus. Synchronizační algoritmy byly doplněny o snadno modifikovatelné struktury FIR filtrů, které bylo možné začlenit do synchronizačních řetězců. Inovativním řešením je především využití metod distribuované aritmetiky v rámci interpolačních filtrů, celkové řízení interpolačního procesu a především návrh struktury polyfázového filtru pro symbolovou synchronizaci. Přínosem práce jsou také výsledky a závěry z podrobného simulačního procesu. Srovnání navržených modelů včetně jejich popisu v jazyce VHDL umožňuje zvolit synchronizační schéma přesně podle požadavků dané aplikace. Jako prostředek pro ověření navržených modelů vznikla unikátní hybridní platforma SDR, která je určena pro ověření synchronizačních algoritmů a má široké možnosti v oblasti vizualizace přenášených dat. Velkou výhodou platformy je její přenositelnost, neboť je využito pouze standardních VHDL knihoven.

Další směr výzkumu v oblasti synchronizačních algoritmů a jejich efektivní realizace na FPGA by mohly směřovat do oblasti přenosových systému využívající frekvenční ortogonální multiplex (OFDM). Chtěl bych se zde především zaměřit na možnost aplikace metod maximální věrohodnosti pro symbolovou synchronizaci s využitím pilotního symbolu.

Literatura:

[1] Stephens, R.: *Phase-Locked Loops For Wireless Communications*. Kluwer Academic, New York, 2002, ISBN 0-792-37602-1.

[2] Kuo S., Lee H.: *Real-Time Digital Signal Processing*. JOHN WILEY & SONS, Chichester, 2001, ISBN 0-470-84137-0.

[3] Tretter A.S.: *Communication System Design Using DSP Algorithms*. Springer, New York, 2008, ISBN 978-0-387-74885-6.

[4] Mengali, U., Andrea A.: *Synchronization Techniques for Digital Receivers*. Plenum, New York, 1997, ISBN 0-306-45725-3.

[5] Reed, J.: *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall, New York, 2002, ISBN 0-13-081158-0.

[6] Rice M.: *Digital Communications: A Discrete-Time Approach.* Prentice Hall, New York, 2009, ISBN 0-13-030497-1.

[7] Altera Corporation, San Jose USA. *ALTPLL (Phase-Locked Loop) IP Core User Guide*.Dostupné[Online]:https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_altpll.pdf

[8] Andrea, A., Luise, M.: *Design and analysis of jitter-free clock recovery scheme for QAM systems*, IEEE Transactions on Communications, roč. 41, č. 5, s. 299-406, Březen 1996.

[9] MCClellan, J, Parks, T.: *A unified approach to the design of optimum FIR linear-phase digital filters*, IEEE Transactions on Circuit Theory, roč. 20, č. 6, s. 697-701, Listopad 1973

[10] Rorabaugh, C. B.: *Digital Filters Designer's Handbook*. McGraw-Hill Book Company, Inc., NewYork, 1993.

[11] White, S.: *Applications of distributed arithmetic to digital signal processing: a tutorial review*, ASSP Magazine, IEEE, roč. 6, č. 3, s. 4-19, 1989.

[12] Altera Corporation, San Jose USA. *Internal Memory (RAM and ROM) User Guide* (datasheet). [Online] Dostupné: www.altera.com/literature/ug/ug_ram_rom.pdf

[13] Altera Corporation, San Jose USA. *Memory Blocks in Cyclone IV Devices* (datasheet). [Online] Dostupné: http://www.altera.com/literature/hb/cyclone-iv/cyiv-51003.pdf

[14] Meyer-Baese, U., Botella, G.,Romeroa, D., Kumm M.: *Optimization of High Speed Pipelining in FPGA-based FIR Filter Design using Genetic Algorithm*, SPIE Defense Security + Sensing, 2012.

[15] Harris, F.: *Multirate Signal Processing for Communication Systems*, Prentice Hall, Upper Saddle River, New York, 2004.

[16] Russell, B. Millar.: *Maximum Likelihood Estimation and Inference: With Examples in R, SAS and ADMB*, JOHN WILEY & SONS, Chichester, 2011, ISBN 978-1-119-97771-1.

[17]Meyer-Baese, U.*: Digital Signal Processing with Field Programmable Gate Arrays*. Springer, Berlin Heidelberg, 2007, ISBN 978-3-540-72612-8.

[18] GARDNER, F. *A.*: *BPSK/QPSK Timing-Error Detector for Sampled Receivers*, IEEE Transactions on Communications, 1986, roč. 34, č. 5, s. 423-429.

[19] GAEDDERT, J., VOLOS, H.I., Cormier D., Reed, J.H.: *Multi-rate Synchronization of Digital Receivers in Software-Defined Radios*. In Proceeding of the SDR 07 Technical Conference and Product Exposition, Denver (USA), 2007, s. 195-200.

[20] GARDNER, F.: *Interpolation in digital modems – Part I: Fundamentals*, IEEE Transactions on Communications, 1993, roč. 41, č. 3, s. 501-507.

[21] Press, WH, Teukolsky, SA, Vetterling, WT, Flannery.: *Vandermonde Matrices – Numerical Recipes*. The Art of Scientific Computing New York: Cambridge University Press. ISBN 978-0-521-88068-8

[22] Erup L., Gardner F., Harris.: *Interpolation in digital modems – Part II: Implementation and performance*, IEEE Trans. on Communications, roč. 41, č. 6, s. 998-1008, Červen 1993.

[23] AWAN, M., KOCH, P. *Combined matched filter and arbitrary interpolator for symbol timing synchronization in SDR receivers*. IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Vídeň, 2010, s. 153 – 156.

[24] DICK, CH., HARRIS, F., RICE, M.: *Synchronization in Software Radios - Carrier and Timing Recovery Using FPGAs*. In IEEE Symposium on Field-Programmable Custom Computing Machines. Napa Valley, CA (USA), 2000, s. 195 - 204.

[25] Luise M., Reggiannini R.: *Carrier frequency recovery in all-digital modems for burst-mode transmissions*. IEEE Trans. on Communications 1995, roč. 45, č. 4, s. 524-532, Červen 1993.

[26] Kay S.M.: *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[27] Andraka R.: *A survey of CORDIC algorithms for FPGA based computers* Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, s. 191-200, New York, USA, 1998.

Seznam obrázků

Seznam tabulek

Seznam zkratek a specifických názvů

ADC	označení pro analogově digitální převodník
ASIC	označení pro zákaznický integrovaný obvod
ARM	označení architektury procesorů
AGC	označení pro systém automatického řízení zisku
AWGN	označuje přenosový kanál s bílým šumem (s normálním rozložením)
BPSK	označení pro druh digitální modulace
CIC	označení pro typ digitálního filtru (*Cascaded integrator–comb filter*)
CORDIC	označuje obecně algoritmus používaný pro jednoduchý výpočet goniometrických a dalších matematických funkcí
CPU	označení pro centrální procesorovou jednotku
CLK	označení pro hodinový cyklus v digitální elektronice
DAC	označení pro digitálně analogový převodník
DAD	označuje určitou skupinu synchronizačních detektorů (závislost na datových přenášených symbolech)
DD	označuje určitou skupinu synchronizačních detektorů (detektor s využitím rozhodovací úrovně)
DDS	označení pro přímou digitální syntézu
DVB-T/2	označení pro pozemský systém digitální televize
DSP	označení pro digitální zpracování signálu
DA	označení pro distribuovanou aritmetiku (matematická metoda)
DPLL	označení pro digitální fázový závěs
FPGA	označení pro typ programovatelného hradlové pole
FIFO	označení pro frontu (First In, First Out - první dovnitř, první ven)
FIR	označení pro skupinu digitálního filtrů
FSM	označení pro konečný stavový automat
GTED	odvozený detektor na bázi detektoru průchodu nulou
HDL	označuje skupinu jazyků určených k popisu číslicových systémů
ISI	označení pro mezisymbolové interference
IP	označení pro opakovaně použitelnou logiku, která je duševním vlastnictvím
IIR	označení pro skupinu digitálních filtrů
LUT	značení pro (programovatelnou) vyhledávací tabulku
LNA	označení pro nízkošumové zesilovače na vstupu přijímače
LSB	označení pro nejméně významný bit čísla
m-QAM	označení pro druh digitální modulace (s m stavy)
MAC	označení pro výpočet resp. jednotku, která provádí násobení a následné přičtení do akumulátoru
MLTED	detektor využívající metod maximální věrohodnosti
ML	označení pro metody maximální věrohodnosti (z matematické statistiky)
NDA	označuje určitou skupinu synchronizačních detektorů (nezávislost na datových přenášených symbolech)
OFDM	označení pro ortogonální multiplex s frekvenčním dělením
PAM	označení pro pulzně amplitudovou modulaci
PCI-E	označení pro rychlou systémovou sběrnici
PLL	označení pro fázový závěs
PC	označení pro osobní počítač

QPSK	označení pro druh digitální modulace
RTL	označení pro syntézu na úrovni meziregistrových přenosů
RAM	označení pro paměť s přímým přístupem, umožňující čtení i zápis
ROM	označení pro paměti konstant u kterých jsou data trvale uložena
SRRC	označení pro filtr, který zaručuje nulové mezisymbolové interference
SDR	označení pro softwarově definované rádio
TED	detektor synchronizační chyby
USB	označení pro rychlou univerzální sériovou sběrnici
VCO	označení pro napětím řízený oscilátor
VCC	označení pro napětím řízený hodinový generátor
VHDL	označení pro jazyk vyšší úrovně pro popis číslicových systémů
ZCTED	detektor průchodu nulou

Seznam symbolů

$\mathbf{E_b/N_0}$	poměr potřebné energie na přenesení 1 bitu informace ku spektrální výkonové hustotě šumu
C	výkon přijímaného užitečného signálu na vstupu rádiového přijímače
$\mathbf{C/N_0}$	poměr výkonů užitečného signálu ku spektrální výkonové hustotě šumu
$\mathbf{N_0}$	spektrální výkonová hustota šumu
I	intenzita signálu
$\mathbf{P_a}$	výstupní výkon výkonového zesilovače
BER	bitová chybovost
SER	symbolová chybovost
SNR	označení pro odstup signálu od šumu
T	vzorkovací perioda
$\mathbf{T_s}$	symbolová perioda
I/Q	soufázová a kvadraturní složku signálu
$\boldsymbol{\theta_e}$	fázová chyba
$\boldsymbol{\Delta\omega}$	frekvenční ofset
$\boldsymbol{\Delta\theta}$	fázový rozdíl
$\boldsymbol{r_p}$	autokorelační funkce
$\boldsymbol{\tau}$	neznámé časové zpoždění
$\bar{\boldsymbol{\tau}}$	předpokládaná hodnota (odhad) časového zpoždění

Tištěné přílohy práce:

- seznam publikací
- seznam uplatněných funkčních vzorků a softwaru
- seznam grantů
- zkrácený profesní životopis
- seznam elektronických příloh práce na přiloženém DVD

Seznam publikací:

a) Články ve sbornících (recenzovaných)

[1] Fiala P.: *Platforma pro příjem a monitorování digitálního televizního vysílání*. Stať ve sborníku. Elektrotechnika a informatika, Nečtiny 2010.

[2] Fiala P.: *Řídicí systém mikropočítače pikosatelitu PilsenCube.* Stať ve sborníku. Elektrotechnika a informatika, Nečtiny 2011.

[3] Fiala P.: *Metody symbolové synchronizace pro experimentální softwarově definované rádio.* Stať ve sborníku. Elektrotechnika a informatika, Nečtiny 2012.

[4] Fiala P. *Computer embedded system for PilsenCUBE picosatellite.* Příspěvek ve sborníku. 4th European CubeSat Symposium, Belgie 2012

[5] Linhart R.; Fiala P.; Voborník A.: *Selected Subsystems in PilsenCUBE Satellite.* Přednáška. Pico and Nano Satellite Workshop 2011. Univerzita Würzburg.

[6] Masopust, J.; Veřtát, I.; Voborník, A.; Pokorný, M.; Mráz, J.; Linhart, J.; Fiala, P.; Hrubec, M., Kavalír, T.; Štemberová, O.: *PilsenCUBE - Picosatellite project at the University of West Bohemia in Pilsen.* Přednáška. Pico and Nano Satellite Workshop 2011. Univerzita Würzburg.

[7] Masopust, J.; Veřtát, I.; Voborník, A.; Pokorný, M.; Mráz, J.; Linhart, R.; Fiala, P.; Hrubec, M., Kavalír, T.; Štemberová, O.; Kraus, V.: *Komunikační a navigační systémy pikosatelitu PilsenCUBE.* Přednáška. Pravidelné setkání zájemců o mikrovlnnou techniku, Praha 2012.

[8] Masopust, J.; Veřtát I.; Voborník, A.; Pokorný, M.; Mráz, J.; Linhart, J.; Fiala, P.; Hrubec, M., Kavalír, T.; Štemberová, O.; Kraus, V.: *Projekt experimentálního pikosatelitu PilsenCUBE.* Přednáška. Přednáška. Pravidelné setkání zájemců o mikrovlnnou techniku, Praha 2012.

[9] FIALA, P. *Efektivní implementace FIR filtru v jazyce VHDL pro softwarově definované rádio*. In *Elektrotechnika a informatika 2013. Část 2., Elektronika.* Plzeň: Západočeská univerzita, 2013. s. 25-28. ISBN: 978-80-261-0232-8

[10] FIALA, P., VOBORNÍK, A. *Embedded microcontroller system for PilsenCUBE picosatellite*. In *2013 IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS).* Brno: Československá sekce IEEE (The Institute of Electrical and Electronics Engineers), 2013. s. 131-134. ISBN: 978-1-4673-6133-0

[11] MASOPUST, J., VEŘTÁT, I., FIALA, P., LINHART, R., MRÁZ, J., POKORNÝ, M., VOBORNÍK, A. *Pikosatelit PilsenCUBE, jeho radiokomunikační a ADCS systémy*. In *14. výjezdní interdisciplinární seminář Nečtiny 2013.* Plzeň: Západočeská univerzita, 2013. ISBN: 978-80-261-0214-4

[12] MASOPUST, J., LINHART, R., VOBORNÍK, A., VEŘTÁT, I., POKORNÝ, M., MRÁZ, J., FIALA, P., KAVALÍR, T. *Pozemní stanice pikosatelitu PilsenCUBE na ZČU - FEL v Plzni*. In *Radiokomunikace 2013.* Pardubice: Unit, 2013. s. 255-263. ISBN: 978-80-905345-2-0

[13] FIALA, P., LINHART, R. *Efficient VHDL Implementation of Symbol Synchronization for Software Radio based on FPGA*. In *Proceedings of the 2014 IEEE 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*. Varšava: IEEE (The Institute of Electrical and Electronics Engineers), 2014. s. 318-321. ISBN: 978-1-4799-4558-0

[14] FIALA, P., LINHART, R. *High performance VHDL FIR filter structure for symbol timing system implemented on FPGA*. In *2014 22nd Telecommunications Forum (TELFOR)*. Běloherad: Telecommunications Society Belgrade, Serbia, 2014. s. 477-480. ISBN: 978-1-4799-6190-0

[15] FIALA, P., LINHART, R. *High performance polyphase FIR filter structures in VHDL language for Software Defined Radio based on FPGA*. In *2014 International Conference on Applied Electronics*. Plzeň: University of West Bohemia, 2014. s. 83-86. ISBN: 978-80-261-0276-2 , ISSN: 1803-7232

[16] LINHART, R., FIALA, P. *Measuring RF Circuits Response Using Software Defined Radio System*. In *2014 International Conference on Applied Electronics*. Pilsen: University of West Bohemia, 2014. s. 185-188. ISBN: 978-80-261-0276-2, ISSN: 1803-7232

[17] LINHART, R., FIALA, P. *Measuring RF Circuits Response Using Software Defined Radio System*. In *2014 International Conference on Applied Electronics*. Pilsen: University of West Bohemia, 2014. s. 185-188. ISBN: 978-80-261-0276-2 , ISSN: 1803-7232

[18] STIFTER, J., FIALA, P., MASOPUST, J., LINHART, R., MRÁZ, J., HLOUŠEK, P., BOUZEK, S., JIRÁK, M., MAJOR, J. *Monitorování technických parametrů zvukové složky vysílání dle doporučení EBU R128*. In *Radiokomunikace 2014*. Pardubice: UNIT s.r.o. Pardubice, 2014. s. 165-178. ISBN: 978-80-905345-4-4

[19] STIFTER, J., FIALA, P. *Monitorování technických parametrů zvukové složky televizního vysílání*. Praha, 2014.

[20] FIALA, P., LINHART, R. *High efficient carrier phase synchronization for SDR using CORDIC implemented on an FPGA*. In *Proceedings of Papers : 2015 23rd Telecommunications Forum (TELFOR 2015)*. Piscataway: IEEE, 2015. s. 512-515. ISBN: 978-1-5090-0055-5

b) Články v impaktovaných časopisech

[21] FIALA, P., LINHART, R. *Symbol synchronization for SDR using a polyphase filterbank based on an FPGA*. *Radioengineering*, 2015, roč. 24, č. 3, s. 772-782. ISSN: 1210-2512

Seznam uplatněných funkčních vzorků a softwaru:

(1) FIALA. P.: *dvbMonitor*. Funkční vzorek – software. 2010

(2) FIALA, P.,Voborník A.: *Deska řídícího počítače pikosatelitu PilsenCube*. Funkční vzorek. 2011.

(3) FIALA, P. *Efektivní implementace symbolové synchronizace v jazyce VHDL pro softwarově definovaný přijímač na FPGA*. 2013.

(4) FIALA, P. *Software pro zobrazování multimediálních materiálů na platformě Linux*. 2013.

(5) FIALA, P. *Softwarová implementace QPSK demodulátoru pro DSP procesor Texas Instruments*. 2013.

(6) LINHART, R., FIALA, P. *Zobrazovač síťově sdílených prezentačních materiálů*. 2013.

(7) STIFTER, J., FIALA, P., BOUZEK, S. *Aplikace pro monitorování vnímané hlasitosti zvukové složky TV vysílání*. 2014.

(8) FIALA, P. *Experimentální platforma pro softwarově definované rádio založená na procesoru TI Omap L-138 / TMS320C6748*. 2014.

(9) FIALA, P. *Generátor VHDL struktur FIR filtrů optimalizovaných pro syntézu na FPGA*. 2014.

(10) FIALA, P. *Softwarová implementace koherentního QPSK přijímače ve VHDL pro syntézu na FPGA*. 2014.

Grantová činnost:

(1) Řešitel: FRVŠ G1/570/2012 - *Zavedení praktické laboratorní výuky do předmětu Diplomový seminář 1*

(2) Spoluřešitel: GAČR 102/09/0455 - *Energeticky úsporná platforma pro experimentální výzkum na bázi pikosatelitů*

(3) Spoluřešitel: TA04011295 - *Širokoúhlý systém pro rentgenové zobrazování s detektorem Timepix* (projekt s mezinárodní účastí)

(4) Spoluřešitel: - *Smart@Fire – Inteligentní hasičské obleky* (evropský projekt; řešena komunikace přes RF moduly)

(5) Spoluřešitel: SGS 2011/2012 (interní grantový projekt) - *Komunikační rozhraní pro experimentální softwarové rádio*

(6) Spoluřešitel: SGS 2013/2014 (interní grantový projekt) - *RF frontend pro experimentální softwarové rádio*

Zkrácený profesní životopis

Ing. Pavel Fiala – narozen v roce 1984, v roce 2004 absolvoval studium na 8 letém gymnáziu v Plzni, v roce 2010 získán titul Ing. na FEL ZČU v Plzni, od téhož roku doktorské studium programu Elektronika na FEL ZČU v Plzni. Od začátku roku 2014 zaměstnancem Regionálního inovačního centra elektrotechniky v Plzni (RICE).

Odborné působení v oblastech:

radioelektronika (zpracování číslicového signálu v radiotechnice, technologie softwarově definovaného rádia, synchronizační metody), televizní technika (číslicové zpracování signálu, digitální televizní systémy – DVB-T), modelování a simulace (modelování a algoritmizace v prostředí Matlab), číslicové systémy (návrh a simulace rozsáhlých systémů číslicového zpracování v jazyce VHDL, DSP procesory)

Výuka v předmětech

Základy sdělovací techniky, Základy elektroniky

Grantové zaměření

Oblast softwarově definovaného rádia, číslicové zpracování na signálových procesorech a FPGA, od 2011 směrování do oblasti satelitní techniky v návaznosti na GAČR (102/09/0455) projekt: *Energeticky úsporná platforma pro experimentální výzkum na bázi pikosatelitů* a dále projekt TAČR (TA04011295): *Širokoúhlý systém pro rentgenové zobrazování s detektorem Timepix.*

Seznam elektronických příloh práce na DVD

1) Adresář „***Matlab_simulace***“ (uvedeny pouze důležité skripty) – [*m-file*]
    - *QPSK_chain_simulation.m* – simulace přenosového kompletního přenosového řetězce s MLTED / ZCTED / GTED
    - *QPSK_chain_simulation_polyphase.m* – simulace přenosového kompletního přenosového řetězce s ZCTED / GTED (polyfázový filtr)
    - *DA_fir_fpga_par.m* – generátor struktur plně paralelních FIR filtrů
    - *DA_fir_fpga_par_polyphase.m* – generátor polyfázových struktur FIR filtrů (interpolační / decimační struktury)
    - *cordic_sin_cos.m* – generátor CORDIC algoritmu pro výpočet funkcí *cos*(.) a *sin*(.)
    - *cordic_magnitude_sqrt.m* – generátor CORDIC algoritmu pro výpočet amplitudy signálu $z = \sqrt{x^2 + y^2}$
    - *cordic_division.m* – generátor CORDIC algoritmu pro výpočet obecného dělení
    - *ber_ser_evaluation.m* – skript pro offline vyhodnocení SER/BER při reálném měření, data získána z programu GnuRadio
    - *costas_loop_simulation.m* – simulace Costasovy smyčky
2) Adresář „***GnuRadio***“ – obsahuje navržené bloky pro převodník *FT232H* (TX/RX) v režimu FIFO (*ftdi_tx_rx_gnuradio_blocks_1.0.tar.gz)* – [*C++*] a další obslužné skripty včetně grafických diagramů (*.grc*) pro *Gnuradio Companion*.
3) Adresář „***VHDL_FIR_filters***“ – obsahuje navržené struktury VHDL FIR filtrů pro implementaci na FPGA. Adresář obsahuje popsané paralelní, sériové a polyfázové struktury včetně testovací procedury (testbench) a projektu v programu Modelsim.
4) Adresář „***VHDL_synchronization***“ – obsahuje navržené synchronizační struktury v jazyce VHDL pro symbolovou synchronizaci a synchronizaci fáze nosné vlny. Pro každý model je k dispozici testovací procedura a projekt v programu Modelsim. Podrobný popis adresářů uveden v souboru README.TXT na DVD.
5) Adresář „***FPGA_mod_demod***“ – obsahuje navržený kompletní modulátor QPSK(TX) a demodulátor QPSK (RX) pro SDR s detektorem symbolové synchronizace MLTED nebo GTED. Obsahuje projekt pro syntézu na FPGA v prostředí *Altera Quartus* (projekty jsou určeny pro FPGA Altera Cyclone IV).