VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

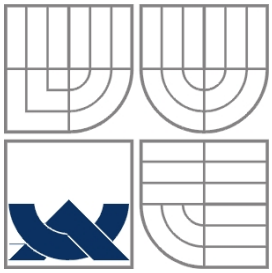
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

# METODY KOMPENZACE NESYMETRIÍ KVADRATURNÍHO DEMODULÁTORU

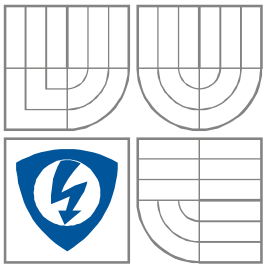
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. KAREL POVALAČ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

# METODY KOMPENZACE NESYMETRIÍ KVADRATURNÍHO DEMODULÁTORU

METHODS FOR QAUDRATURE MODULATOR IMBALANCE COMPENSATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Karel Povalač
AUTHOR

VEDOUCÍ PRÁCE — Ing. Roman Maršálek, Ph.D.
SUPERVISOR

BRNO 2008

# LICENČNÍ SMLOUVA
# POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

**1. Pan/paní**

Jméno a příjmení: Bc. Karel Povalač
Bytem: Horní Lideč 61, 756 12
Narozen/a (datum a místo): 10. ledna 1984 ve Vsetíně

(dále jen „autor“)

a

**2. Vysoké učení technické v Brně**

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

## Čl. 1
## Specifikace školního díla

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

   ☐ disertační práce
   ☒ diplomová práce
   ☐ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

   Název VŠKP: Metody kompenzace nesymetrií kvadraturního demodulátoru
   Vedoucí/ školitel VŠKP: Ing. Roman Maršálek, Ph.D.
   Ústav: Ústav radioelektroniky
   Datum obhajoby VŠKP: _____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

---

- hodící se zaškrtněte

## Článek 2

## Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   ☒ ihned po uzavření této smlouvy
   ☐ 1 rok po uzavření této smlouvy
   ☐ 3 roky po uzavření této smlouvy
   ☐ 5 let po uzavření této smlouvy
   ☐ 10 let po uzavření této smlouvy
   (z důvodu utajení v něm obsažených informací)

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

## Článek 3

## Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 30. května 2008

…….. ……

Nabyvatel Autor

# Abstrakt

Kvadraturní modulátor (demodulátor) je používán ve vysílací (přijímací) části mnoha zařízení. Nežádoucí parametry mohou ovlivňovat amplitudu, fází nebo stejnosměrný offset modulátoru (demodulátoru). Kompenzování těchto nesymetrií bylo hlavním úkolem práce. Nejprve v prostředí MATLAB vznikly simulace těchto metod a dále byly zkoumány jejich výsledky. Následovala implementace těchto metod na programovatelný logický obvod pomocí programu Xilinx ISE. K tomuto účelu byla využita vývojová deska V2MB1000 s analogovým modulem Memec P160. V poslední fázi byly výsledky simulací podloženy praktickým měřením.



# Abstract

Quadrature modulator (demodulator) is used in transmitting (receiving) part of many devices. Unwanted imbalance can influence amplitude, phase or DC offset of modulator (demodulator). Compensation of imbalance was a main subject of thesis. Simulations of these methods were created in MATLAB and results were compared. Basics of methods were implement on programmable logic field by program Xilinx ISE. Development kit V2MB1000 with analogue board Memec P160 was chosen for this purpose. In the last part were compare simulation results with practical measurement.


# Bibliografická citace

POVALAČ, K. *Metody kompenzace nesymetrií kvadraturního demodulátoru.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 54 s. Vedoucí diplomové práce Ing. Roman Maršálek, Ph.D.

# Prohlášení

Prohlašuji, že svou diplomovou práci na téma Metody kompenzace nesymetrií kvadraturního demodulátoru jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 30. května 2008 .....
podpis autora

# Poděkování

Děkuji vedoucímu diplomové práce Ing. Romanu Maršálkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

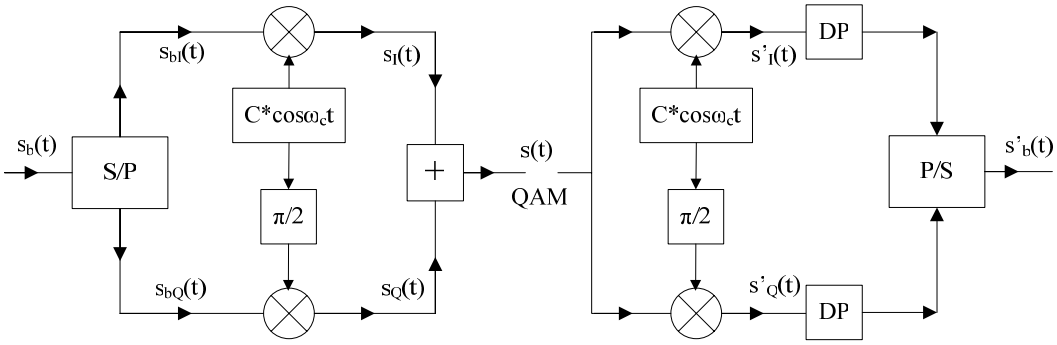
V Brně dne 30. května 2008 .....
podpis autora

# Obsah

# Seznam obrázků

# Úvod

V mnoha komunikačních systémech dnešní doby je používáno vícestavových modulací, jejichž základem je kvadraturní modulátor na vysílací straně a kvadraturní demodulátor na straně přijímací. Základní principielní schéma je na Obr. 1, kde lze vidět, že signál nejprve vstupuje do bloku sériově-paralelního převodníku, který signál rozdělí do *I* a *Q* větve.




**Obr. 1 Kvadraturní modulátor a demodulátor**

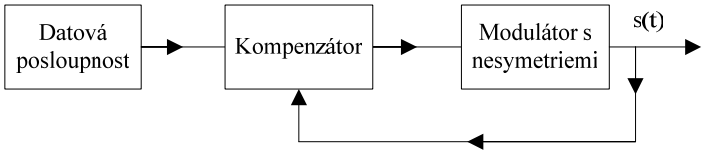
V každé této větvi dochází pak k ovlivňování nosné signálem na kmitočtu $\omega_c$. Jedna větev (*I*) je násobena nosným signálem $C\cos(\omega_C t)$ a druhá větev (*Q*) je násobena nosnou $C\sin(\omega_C t)$. Vznikne tak soufázová složka $s_I(t)$ a kvadraturní složka $s_Q(t)$. Výsledný modulovaný signál vznikne sečtením těchto dvou složek

$$s(t) = s_I(t)\cos(\omega_C t) + s_Q(t)\sin(\omega_C t). \qquad (1)$$

Strana demodulátoru je nejprve tvořena opětovným násobením nosným signálem, stejným jako v modulátoru. Vzniknou tak i složky na dvojnásobném kmitočtu nosného signálu, které následně potlačují zapojené filtry dolní propusti (2). Poté je výsledný signál převeden z paralelních dat na sériová.

$$\begin{aligned} s'_I(t) &= \left[s_I(t)\cos(\omega_c t) + s_Q(t)\sin(\omega_c t)\right]\ C\cos(\omega_c t) = \\ &= \frac{1}{2}Cs_I(t) + \frac{1}{2}Cs_I(t)\cos(2\omega_c t) + \frac{1}{2}Cs_Q(t)\sin(2\omega_c t) \end{aligned} \qquad (2)$$

V praxi mohou nastávat situace, které se mírně rozchází od ideálního kvadraturního modulátoru a demodulátoru. Jedná se zejména o nesymetrické vlastnosti signálů v *I* a *Q* větvích. Úkolem této diplomové práce je shromáždit informace a najít možnou metodu, která by eliminovala nesymetrie v obou větvích. Následně pak provést simulace metody v prostředí MATLAB a získané výsledky zhodnotit. Poté implementovat některou z metod na programovatelné logické pole FPGA Virtex 2 pomocí programu Xilinx ISE. Metody, které zde budou popsány se budou držet blokového uspořádání dle Obr. 2.




**Obr. 2 Obecné blokové uspořádání kompenzace nesymetrií modulátoru**

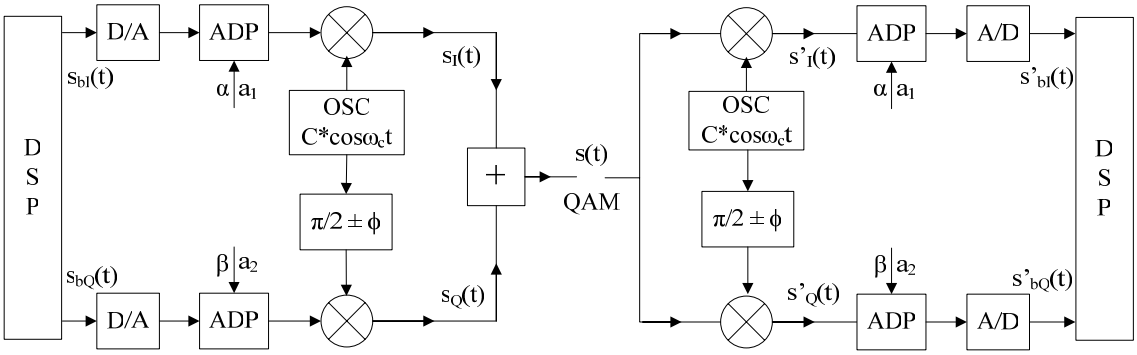
Vytvořená bitová posloupnost dat nejprve vstupuje do kompenzátoru, který již signál upraví tak, aby po průchodu modulátorem s nesymetriemi byl výstupní signál vykompenzován.

# 1. Model systému a jeho nesymetrií

Nyní bychom mohli přistoupit k základnímu popisu nesymetrií v obvodu. Budeme vycházet z blokového uspořádání na Obr. 3, které již zobrazuje reálnější představu modulátoru a demodulátoru.

## 1. 1. Kvadraturní modulátor

Signálové složky *I* a *Q* vychází přímo z bloku DSP, který je tvořen např. signálovým procesorem, a jsou nejdříve převedeny na analogový signál (D/A převodníkem). Poté prochází blokem aktivní dolní propusti (ADP), kde už může být signál v jednotlivých větvích postižen nesymetriemi. Jedná se zejména o nesymetrické zesílení v těchto větvích, které je zde označeno symboly α a β. Dále pak může ve filtru vzniknout stejnosměrný napěťový offset, který různě napěťově posune signály v obou větvích. Tento stejnosměrný posun je označován symboly $a_1$ a $a_2$.




**Obr. 3 Reálný model kvadraturního modulátoru a demodulátoru**

Při fázovém posunu nosného signálu o úhel $\frac{\pi}{2}$ dojde k odchylce od této hodnoty. Tato odchylka je v blokovém schématu označena jako $\phi$. Velikost odchylky je sice malá, ale nelze ji považovat za úplně nulovou. Lze tedy definovat koeficient zesílení γ a nerovnováhu zesílení ε:

$$\gamma = \frac{\alpha}{\beta} \quad \text{a} \quad \varepsilon = \gamma - 1. \tag{3}$$

Hodnoty typické pro jednotlivé koeficienty jsou 2–3% pro nesymetrii zesílení a 2–3° pro fázovou nerovnováhu. Stejnosměrné napěťové posunutí může být definováno v rozmezí 2–3%, které jsou brány z celého rozsahu. Těmito zmiňovanými parametry je popsána „nedokonalost“ modulátoru nebo demodulátoru a hlavním úkolem tedy je, aby se nalezl způsob, jak adaptivně vyvažovat tyto nežádoucí parametry.

K tomu, aby bylo možné modulátor a zároveň demodulátor vhodně nasimulovat v MATLABu, je výhodné vycházet z maticového popisu jednotlivých veličin. Ty budou nyní zapsány v příslušném tvaru. Pro práci v MATLABu je to velmi vhodný způsob, jak poměrně

výhodně a efektivně pracovat s těmato veličinami. Je možné nejdříve zapsat sloupcový vektor signálu $s_b(t)$ ve tvaru

$$s_b(t) = (s_{b1}(t), s_{b2}(t))^T . \quad (4)$$

Jednotlivé složky $s_b(t)$ odpovídají signálům $s_{bI}$ a $s_{bQ}$ v soufázové a kvadraturní větvi modulátoru.

Výsledný kvadraturně modulovaný signál lze získat po provedení této maticové rovnice

$$\boldsymbol{s}(t) = \boldsymbol{M}(\boldsymbol{s}_b(t) + \boldsymbol{a}), \quad (5)$$

kde matice $\boldsymbol{M}$ je definována jako [1]

$$\boldsymbol{M} = \begin{bmatrix} \alpha\cos\left(\frac{\phi}{2}\right) & \beta\sin\left(\frac{\phi}{2}\right) \\ \alpha\sin\left(\frac{\phi}{2}\right) & \beta\cos\left(\frac{\phi}{2}\right) \end{bmatrix}. \quad (6)$$

Popisuje tedy nesymetrii zesílení a fázový posun složek v obou větvích. Sloupcový vektor $\boldsymbol{a}$, popisující napěťový posun, je pak jednoduše zapsán

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}. \quad (7)$$

V tomto případě byla fázová chyba rozdělena souměrně mezi jednotlivé větve *I* a *Q*, což nemusí být ve skutečnosti vhodné. Nesymetrické rozdělení pak bude ukázáno později.

## 1. 2. Kvadraturní demodulátor

Druhá část schématu na Obr. 3 zobrazuje demodulátor na přijímací straně. Stejně jako v modulátoru zde nastává nejprve vynásobení signálu nosnou ve větvích *I* a *Q*. Opět může dojít k pootočení fáze od ideální hodnoty $\frac{\pi}{2}$ (pootočení může být následek průchodu komunikačním kanálem nebo nedokonalostí demodulátoru). Rovněž může opět dojít k nesymetrickému zesílení v obou větvích demodulátoru, které má za následek průchod aktivními filtry. Tento průchod může také vytvořit stejnosměrný napěťový offset signálu. Maticově lze demodulátor popsat podobnou rovnicí jako v modulátoru

$$\boldsymbol{s}_d(t) = \boldsymbol{D} \cdot \boldsymbol{s}_r(t) + \boldsymbol{b}, \quad (8)$$

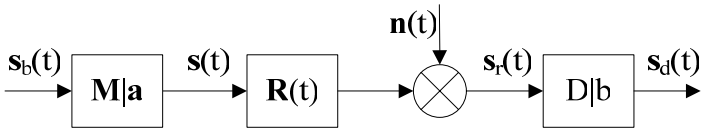
kde matice $\boldsymbol{D}$ a $\boldsymbol{b}$ jsou podobné maticím $\boldsymbol{M}$ a $\boldsymbol{a}$, ale s rozdílnými parametry.

## 1. 3. Celkový maticový model

Všechny předchozí popsané maticové operace ukazuje model systému na Obr. 4. Navíc může ještě dojít k tomu, že v kanále vznikne další fázový posun a lze jej popsat maticí (9), kde úhel $\theta$ udává zmiňovaný fázový posun.

$$\boldsymbol{R}(t) = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) \\ \sin(\theta(t)) & \cos(\theta(t)) \end{bmatrix} \quad (9)$$

Pro zjednodušení byla v celé práci považována matice ***R(t)*** za konstantu rovnou jednotkové matici ***I***, pokud je fázová chyba rovna nule. Vyplývá to z toho, že pokud je $\theta$ rovno nule, v matici budou hodnoty funkce sinus nulové a cosinus bude roven jedné.




**Obr. 4 Model maticového řešení systému**

Celkově tedy může být signál popsán spolu s kanálovým šumem další maticovou rovnicí

$$\boldsymbol{s}_d(t) = \boldsymbol{H}(t)\cdot\boldsymbol{s}_b(t) + \boldsymbol{c}(t) + \boldsymbol{m}(t), \qquad (10)$$

kde použité matice ***H(t)***, ***c(t)*** a ***m(t)*** vyjadřují

$$\boldsymbol{H}(t) = \boldsymbol{D}\cdot\boldsymbol{R}(t)\cdot\boldsymbol{M}\,,\quad \boldsymbol{c}(t) = \boldsymbol{H}(t)\cdot\boldsymbol{a} + \boldsymbol{b} \quad \text{a} \quad \boldsymbol{m}(t) = \boldsymbol{D}\cdot\boldsymbol{n}(t). \qquad (11)$$

Tyto maticové operace zahrnují tedy i různé rušivé vlivy na signál (zejména matice ***n(t)*** popisuje přídavný šum) a uvažují zmiňované druhy nesymetrií kvadraturního modulátoru a demodulátoru.

# 2. Kompenzace nesymetrií ve vysílači

Doposud byl popsán modulátor s demodulátorem a byly zmíněny jaké nesymetrie se vyskytují v těchto zapojeních. Nyní budou popsány metody, pomocí kterých se tyto nedostatky mohou snížit, nebo v ideálním případě adaptivně kompenzovat. Pro vhodnější implementaci bude opět využíváno maticových úprav a rovnic.

## 2. 1. Metoda dle Caverse [1]

### 2. 1. 1. Nesymetrický tvar

Bylo vycházeno z původní rovnice (5), která matematicky popisuje modulátor. Tato kompenzace je založena na tom, že členy, které způsobují zmiňované nežádoucí vlastnosti, jsou kompenzovány inverzními maticemi těchto členů. Tuto myšlenku lze vyjádřit maticovou rovnicí

$$\boldsymbol{s}_c(t) = \boldsymbol{C}\cdot\boldsymbol{s}_b(t) + \boldsymbol{f}\ , \qquad (12)$$

kde v nejlepším případě je matice ***C*** rovna inverzní matici ***M*** $\left(\boldsymbol{C} = \boldsymbol{M}^{-1}\right)$. Podobně pak platí výraz pro matici ***f*** $\left(\boldsymbol{f} = -\boldsymbol{a}\right)$. Hlavním úkolem metody je, aby se iteračními kroky došlo k přesné (podobné) kompenzační matici C a f. Metoda má k těmto hodnotám samozřejmě směřovat po co nejmenším počtu kroků. Blokově se dá tato metoda zakreslit tak, jak je uvedeno na Obr. 5. Proti původnímu obecnému schématu zde přibyl blok s obálkovým detektorem, který řídí kompenzátor. Podrobněji bude tento princip popsán dále.

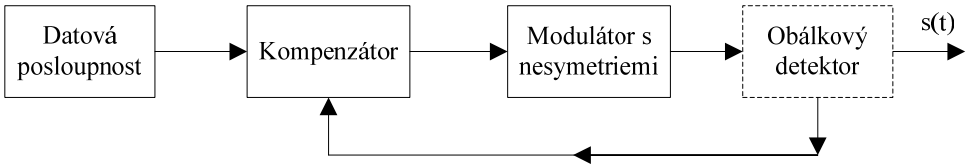
Pokud se dosadí parametry z matice (6), mělo by se pak jednat o symetrický modulátor. Je také možné použít úvahu, že celková chyba fázové nesymetrie je přesunuta do větve *Q* modulátoru. Což lze zapsat jako

$$\boldsymbol{M} = \begin{bmatrix} \alpha & \beta\sin(\phi) \\ 0 & \beta\cos(\phi) \end{bmatrix}. \qquad (13)$$

Inverzní matice pak má tvar

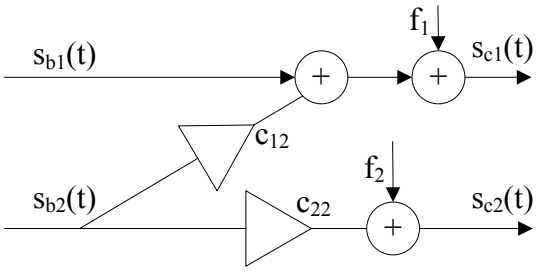
$$\boldsymbol{M}^{-1} = \alpha^{-1} \cdot \begin{bmatrix} 1 & -\tan(\phi) \\ 0 & \gamma\sec(\phi) \end{bmatrix}. \qquad (14)$$

Funkce sekans se dá zapsat i ve tvaru pomocí funkce cosinus $\sec(\phi) = \frac{1}{\cos(\phi)}$. Nyní tento způsob zápisu už odpovídá nesymetrickému popisu. V literatuře [1] je odkazováno i na symetrickou adaptivní metodu přizpůsobení (článek se však nepodařilo získat), v tomto případě však byla použita metoda využívající nesymetrický zápis.




**Obr. 5 Blokové uspořádání kompenzace nesymetrií pro metodu dle Caverse**

Tato kompenzace nesymetrie byla navržena pro zpracování signálu pomocí DSP, proto je v této metodě kladen důraz i na jednoduchost a rychlost výpočtu. Tomu pak odpovídá určitý počet strojových cyklů potřebných pro výpočet. Metoda se dá mírně zjednodušit, když se vynechá dělení konstantou $\alpha$ ve vzorci (14). Toto zjednodušení má pak za následek odchylku od přesného výsledku o několik jednotek procent [1]. Takové přizpůsobení je zakresleno např. na Obr. 6.




**Obr. 6 Model kompenzátoru**

Tato úprava spočívá v podmínce, kdy se musí nastavovat čtyři koeficienty dokud nedojde k vyváženosti v obou větvích nebo pokud nevymizí napěťový offset na výstupu kvadraturního modulátoru. Schéma na Obr. 6 přesně odpovídá vynásobení matice $\boldsymbol{C}$ s maticí $\boldsymbol{s_b(t)}$ a přičtení matice $\boldsymbol{f}$, která zde eliminuje napěťový offset.

### 2. 1. 2. Přizpůsobení nesymetrie fáze a zesílení

Nyní přichází na řadu přizpůsobení zesílení a fázové nesymetrie. Tyto dvě nesymetrie jsou spolu více svázány. Maticový zápis kompenzace může vycházet ze vztahu (14), kdy lze matici $\boldsymbol{C}$ vyjádřit:

$$\boldsymbol{C} = \begin{bmatrix} 1 & -\tan(\phi) + e_p \\ 0 & \gamma\sec(\phi) + e_g \end{bmatrix}, \qquad (15)$$

kde $e_p$ a $e_g$ jsou chyby přizpůsobení. K těmto hodnotám matice $\boldsymbol{C}$ by se výše zmíněná metoda měla dopracovat po několika iteračních krocích. Protože se jedná o nesymetrickou kompenzaci je výhoda v tom, že matice obsahuje na svých pozicích i hodnoty 0 a 1. Dopočítávají se prvky matice s indexy $c_{12}$ a $c_{22}$. Výsledná složená kompenzační matice pro kvadraturní modulátor má tvar

$$MC = \beta \begin{bmatrix} 1+\varepsilon & (1+\varepsilon)e_p + e_g \sin(\phi) \\ 0 & 1+\varepsilon+e_g \cos(\phi) \end{bmatrix}. \quad (16)$$

Jako testovací signál byly použity signály [1] se čtyřmi různými fázemi a amplitudou $A$. Signály jsou postupně přiváděny do kompenzátoru a poté prochází modulátorem. Zmiňované signály jsou

$$\begin{aligned} \boldsymbol{s}_m^{\langle 1\rangle} &= [0, A]^T & \boldsymbol{s}_m^{\langle 2\rangle} &= [A, 0]^T \\ \boldsymbol{s}_m^{\langle 3\rangle} &= \frac{[A, A]^T}{\sqrt{2}} & \boldsymbol{s}_m^{\langle 4\rangle} &= \frac{[-A, A]^T}{\sqrt{2}} \end{aligned} \quad (17)$$

Přizpůsobení je u konce v případě, kdy všechny čtyři fáze dávají stejnou hodnotu výstupu $s_e$ z obálkového detektoru. Hlavní výhodou této metody je to, že stačí znát pouze obálku signálu. Chybové signály jsou pak získány jako párové hodnoty

$$\begin{aligned} f_g(e_g, e_p) &= G'_e \left\| s_q^{\langle 1\rangle} \right\| - G'_e \left\| s_q^{\langle 2\rangle} \right\| \\ f_p(e_g, e_p) &= G'_e \left\| s_q^{\langle 3\rangle} \right\| - G'_e \left\| s_q^{\langle 4\rangle} \right\| \end{aligned}. \quad (18)$$

Jedná se zde o rozdíl norem jednotlivých získaných výstupů testovacích signálů z modulátoru. Proměnná $G'_e$ udává zesílení celé smyčky.

Přibližně je možné stanovit i chybové hodnoty $e_g$ a $e_p$ z následujícího vyjádření

$$\begin{aligned} f_g(e_g, e_p) &\cong G'_e \cdot A \cdot e_g \\ f_p(e_g, e_p) &\cong G'_e \cdot A \cdot e_p \end{aligned}. \quad (19)$$

Toto platí pro asymetrický kompenzátor. Dále lze z jednotlivých iterací průběžně zpřesňovat druhý sloupec matice $\boldsymbol{C}$, který je nejdůležitější při následné kompenzaci. Koeficienty matice $\boldsymbol{C}$ odpovídají zápisu

$$\begin{aligned} c_{12}(k+1) &= c_{12}(k) - \delta \cdot f_p(e_g(k), e_p(k)) \\ c_{22}(k+1) &= c_{22}(k) - \delta \cdot f_g(e_g(k), e_p(k)) \end{aligned}. \quad (20)$$

Následující koeficient je vždy počítán z předchozí hodnoty, od které se odečítá chybová hodnota násobená $\delta$, což je velikost kroku přiblížení k požadované hodnotě. Tento krok (viz. [1] )musí vyhovovat podmínce

$$0 < \delta \cdot G'_e \cdot A < 2, \quad (21)$$

aby byla zachována konvergence ke správné hodnotě. Změnou velikosti kroku metody se ovlivňuje zejména počet nutných iterací. Při příliš malé hodnotě kroku se ke správnému výsledku nemusí algoritmus ani dopracovat, i když je směr postupu správný. Naopak velká hodnota může způsobit, že algoritmus diverguje. Podrobněji budou výsledky simulací popsány v následujících kapitolách.

## 2. 2. Metoda dle Zhu [4]

### 2. 2. 1. Teoretický popis metody

Podobně jako v předchozím případě bude nyní uveden základní popis jiné metody, která vede k adaptivnímu přizpůsobení nesymetrií vysílače. Je zde vycházeno z podobného maticového popisu. Kvadraturní modulátor je tak vyjádřen vztahem

$$\boldsymbol{s}_s(n) = \boldsymbol{M}\boldsymbol{s}_m(n) + \boldsymbol{a}, \tag{22}$$

kde sloupcový vektor $\boldsymbol{s_m}(n) = [I_m(n) \quad Q_m(n)]^T$ představuje ideální vstup modulátoru bez nesymetrií. Výstupem modulátoru je pak sloupcový vektor $\boldsymbol{s}_s(n) = [I_s(n) \quad Q_s(n)]^T$, který je již ovlivněn jednotlivými nesymetriemi. Napěťový offset je definován sloupcovým vektorem $\boldsymbol{a} = [a_I \quad a_Q]^T$. Jednotlivé indexy $_I$ a $_Q$ vždy odpovídají aktuálním příslušným hodnotám v *I* a *Q* větvi modulátoru. Nesymetrie modulátoru jsou definovány podobnou maticí, jako v předchozí metodě

$$\boldsymbol{M} = \begin{bmatrix} 1 & -\alpha_m \sin(\phi_m) \\ 0 & \alpha_m \cos(\phi_m) \end{bmatrix}. \tag{23}$$

Lze říci, že kompenzátor je umístěn před modulátorem. Průběh kompenzace tedy může být charakterizován rovnicí, kde se uplatní jednak modulátor se svými nesymetriemi a v další řadě také kompenzátor, který má nesymetrie eliminovat. Rovnici lze zapsat ve tvaru

$$\tilde{\boldsymbol{s}}_m(n) = \boldsymbol{M}(\boldsymbol{P}(n)\boldsymbol{s_m}(n) + \boldsymbol{d}(n)) + \boldsymbol{a}. \tag{24}$$

Matice ***P(n)*** představuje kompenzační matici pro nesymetrii zesílení a fáze

$$\boldsymbol{P}(n) = \begin{bmatrix} p_{11}(n) & p_{12}(n) \\ p_{21}(n) & p_{22}(n) \end{bmatrix} \tag{25}$$

a sloupcový vektor ***d(n)*** představuje kompenzaci napěťového offsetu $\boldsymbol{d} = [d_I(n) \quad d_Q(n)]^T$. Pokud budou splněné podmínky, může v ideálním případě dojít k úplné kompenzaci, kdy $\boldsymbol{P}(n) = \boldsymbol{M}^{-1}$ a $\boldsymbol{d}(n) = -\boldsymbol{M}^{-1}\boldsymbol{a}$. V takovém případě pak bude $\tilde{\boldsymbol{s}}_m(n)$ nabývat hodnoty $\boldsymbol{s}_m(n)$. Což tedy odpovídá hodnotám ideálního modulátoru.

### 2. 2. 2. Adaptivní algoritmus pro odhad nesymetrií

Hlavním smyslem metody je, aby se postupnými iteračními kroky došlo ke správným hodnotám kompenzačních matic ***P(n)*** a ***d(n)***. Metoda je vhodná zejména pro signály s konstantní velikostí obálky. Základ algoritmu je popsán rovnicí, která se snaží minimalizovat chybovou funkci *J(n)*. Funkce popisuje kolísání amplitudy vzhledem k jisté konstantě $R_2$. Funkce se dá vyjádřit vztahem

$$J(n) = E\left[\left(\| \tilde{\boldsymbol{s}}(n) \|^2 - R_2\right)^2\right]. \tag{26}$$

Vlnovkou nad symbolem ***s*** je označeno to, že signál již prošel modulátorem s nesymetriemi. Symbolem *E* je označována očekávaná hodnota a konstanta $R_2$ závisí pouze na vstupních datových symbolech signálu $\boldsymbol{s}_m(n)$ a je v [4] definována jako

$$R_2 = \frac{E\left[\ \left\|\boldsymbol{s}_m(n)\right\|^4\ \right]}{E\left[\ \left\|\boldsymbol{s}_m(n)\right\|^2\ \right]}. \qquad (27)$$

Konstanta R2 nabývala v simulacích hodnotu 2, což bylo způsobeno velikostí signálu ideálního modulátoru.

Pro nalezení kompenzačních koeficientů ***P(n)*** a ***d(n)*** byl použit stochastický gradientní algoritmus. Kompenzační koeficienty byly poskládány do vektoru $\boldsymbol{p}(n) = \left[p_{11}(n) \quad p_{12}(n) \quad p_{21}(n) \quad p_{22}(n) \quad d_I(n) \quad d_Q(n)\right]^T$.

Nyní je možné vypočítat jeden iterační krok. Je nutné ještě definovat velikost iteračního kroku, který je označován $\mu$. Následující hodnota ***p(n + 1)*** je dána vztahem

$$\boldsymbol{p}(n+1) = \boldsymbol{p}(n) - \mu \cdot e(n)\boldsymbol{x}_t(n). \qquad (28)$$

Kde *e(n)* představuje vztah $e(n) = \left\| \tilde{\boldsymbol{s}}(n) \right\|^2 - R_2$ a sloupcový vektor $\boldsymbol{x}_t$ je $\boldsymbol{x}_t(n) = \left[\tilde{I}_m^2(n) \quad \tilde{I}_m(n)\tilde{Q}_m(n) \quad \tilde{Q}_m(n)\tilde{I}_m(n) \quad \tilde{Q}_m^2(n) \quad \tilde{I}_m(n) \quad \tilde{Q}_m(n)\right]^T$. Zde symboly $\tilde{I}_m(n)$ a $\tilde{Q}_m(n)$ značí signály v *I* a *Q* větvi po průchodu modulátorem.

Velikostí nastaveného kroku $\mu$ se ovlivňuje rychlost konvergence metody. Tento krok je nutné zvolit vyzkoušením několika hodnot a poté podle získaných výsledků nastavovat tento parametr.

Po vykonání několika těchto iteračních kroků tedy postupně dojde k ustálení hodnot ve vektoru ***p(n)***. První čtyři složky vektoru jsou pak následně použity na kompenzaci nesymetrie fáze a zesílení. Zbylé dvě složky kompenzují napěťový offset.

Tato metoda se zdá být jednodušší, než metoda prof. Caverse. V závěru této práce bude uvedeno malé srovnání všech popsaných metod.

## 2. 3. Metoda dle Helda [2]

### 2. 3. 1. Teoretický popis metody, odhad kompenzačních koeficientů

Jako třetí způsob pro snížení nesymetrií kvadraturního modulátoru byla vybrána metoda publikovaná v [2]. Dle zmiňovaného článku byla tato metoda použita pro eliminaci nesymetrií u OFDM WLAN přijímačů. Bude zde uvedeno jen několik hlavních informací, týkajících se kompenzačního algoritmu. Základ výpočtů spočívá v nalezení kompenzačních koeficientů z předchozích vysílaných (resp. přijatých) datových symbolů. Po nalezení těchto koeficientů dochází pak k následné kompenzaci modulátoru.

Model nesymetrií, který používá tato metoda je podobný jako u předchozích dvou. Je definován amplitudovou nesymetrií *K*. Tato nesymetrie má však symetrické vyjádření v podobě dvou faktorů $K_I$ a $K_Q$ pro *I* a *Q* větev modulátoru. Fázová chyba mezi *I* a *Q* větví je značena $\varphi_{err}$. V maticové rovnici (29), která je uvedena v [2], není brán v úvahu offsetový posun. Což u předchozích metod bylo. Vliv nesymetrií na signál je dán následujícím vztahem:

$$\begin{bmatrix} s_I[k] \\ s_Q[k] \end{bmatrix} = \begin{bmatrix} K_I & 0 \\ -K_Q \cdot \sin\varphi_{err} & K_Q \cdot \cos\varphi_{err} \end{bmatrix} \cdot \begin{bmatrix} s_I'[k] \\ s_Q'[k] \end{bmatrix}. \quad (29)$$

Symboly $s'_I$ a $s'_Q$ značí původní signál. Výsledkem této maticové rovnice pak je výstupní signál nedokonalého modulátoru.

Nyní bude přistoupeno k výpočtu odhadu kompenzačního koeficientu amplitudové nesymetrie. Odhad tohoto koeficientu i kompenzačního koeficientu fázové nesymetrie je tedy založen na sledování předchozích vyslaných (resp. přijatých) datových symbolů. To se následně projevuje i v matematickém zápisu odhadu. K nalezení kompenzačního koeficientu amplitudové nesymetrie $K_{est}$ mohou vést podle [2] tři algoritmy:

$$K_{est,A} = \frac{\sum_{k=1}^{L} |s_Q[k]|}{\sum_{k=1}^{L} |s_I[k]|}, \quad (30)$$

$$K_{est,B} = \sqrt{\frac{\sum_{k=1}^{L} |s_Q^2[k]|}{\sum_{k=1}^{L} |s_I^2[k]|}}, \quad (31)$$

$$K_{est,C} = \frac{\sum_{k=1}^{L} |s_Q[k] - s_Q[k+1]|}{\sum_{k=1}^{L} |s_I[k] - s_I[k+1]|}. \quad (32)$$

Symbolem $L$ je značena délka okna, která udává počet vzorkovaných symbolů použitých k výpočtu koeficientu. Délka tohoto okna je také úměrná přesnosti odhadu koeficientu. V simulacích a následně pak i při VHDL implementaci byla délka $L$ volena v rozmezí několika stovek vzorků. Dle informací, uvedených v [2], je nejvhodnější použít odhad $K_{est,B}$ (31), protože bylo dosaženo stále stejných výsledků pro různé pozice okna. Hodnota koeficientů $K_{est,A}$ a $K_{est,C}$ byla závislá na pozici okna. Výsledky simulací poté ukáží správnost výběru koeficientu.

Koeficient fázové nesymetrie musí odstranit z maticové rovnice (29) fázovou chybu, která je zastoupena výrazy $-\sin\varphi_{err} \cdot s_I'[k]$ a $\cos\varphi_{err} \cdot s_Q'[k]$. Odhad koeficientu $P_{est}$ je prováděn ve stejně dlouhém okně $L$ z vyslaných (resp. přijatých) datových symbolů $s_I$ a $s_Q$.

$$P_{est} = \frac{\sum_{k=1}^{L} \left( s_I[k] \cdot s_Q[k] \right)}{\sum_{k=1}^{L} s_I^2[k]} \quad (33)$$

Po nalezení těchto dvou koeficientů je možno přistoupit ke korekci amplitudové a fázové nesymetrie

### 2. 3. 2. Korekce nesymetrií

Nejprve je tedy přistoupeno ke kompenzaci amplitudy, ke které slouží koeficient (31). Matematický zápis, uvedený v [2] je:

$$\begin{aligned} w_I[k] &= \frac{1}{K_{est}} \cdot s_I[k] \\ w_Q[k] &= s_Q[k] \end{aligned} \quad . \qquad (34)$$

Kde $s_I$ a $s_Q$ jsou signály, které již byly znehodnoceny nesymetriemi.

Následuje poté využití koeficientu (33), který slouží k potlačení fázové nesymetrie. Zápis je obdobný jako v předchozím případě:

$$\begin{aligned} w_I[k] &= s_I[k] \\ w_Q[k] &= \frac{1}{\sqrt{1-P_{est}^2}}\left[s_Q[k] - P_{est} \cdot s_I[k]\right] \end{aligned} \qquad (35)$$

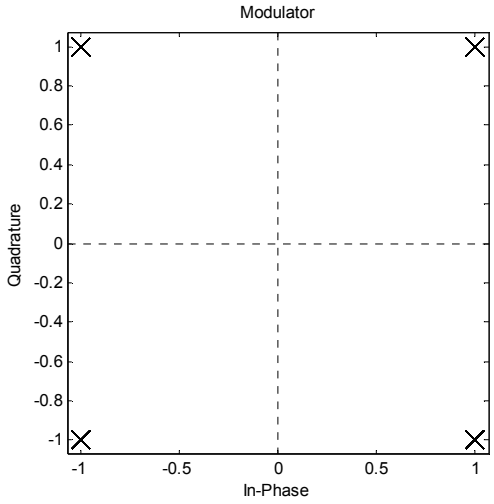
Po provedení těchto úprav dojde tedy k vykompenzování modulátoru. Výsledky simulací této metody budou pak popsány dále. Ze získaných informací je patrné, že metoda by mohla být i jednodušeji implementována do FPGA, než předchozí dvě metody.

# 3. Simulace v prostředí MATLAB

Zmíněné programovací prostředí bylo vybráno z důvodů vhodného matematického popisu celé metody. Maticové operace jsou zde prováděny poměrně jednoduše a výstupem jednotlivých úkonů může být téměř libovolný graf, v tomto případě nejčastěji asi konstelační diagram signálu, z kterého je na první pohled znatelné jednotlivé působení kterékoliv z nesymetrií. Ať už pootočením diagramu, způsobené fázovým posunem, nebo změnou velikosti jednotlivých vektorů signálu (nesymetrie zesílení v každé větvi modulátoru) nebo pak v posledním případě stejnosměrné posunutí celého diagramu, zapříčiněné stejnosměrným offsetem signálu.

## 3. 1. Popis implementace metody dle Caverse

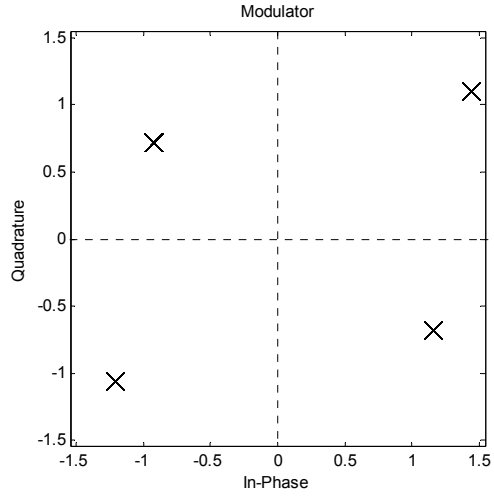
Pro zmiňovanou metodu je nejdříve důležité si definovat jednotlivé parametry nesymetrií modulátoru nebo demodulátoru. Zejména se jedná o vztahy (3), který popisuje koeficient zesílení a vztah (7), který udává napěťový offset signálu. Byla vytvořena funkce, která generuje datový signál kvadraturního modulátoru. Ten může být pro začátek ideální, tedy tvořen datovými symboly $\pm 1 \pm j$. Konstelační diagram kvadraturně modulovaného signálu ukazuje Obr. 7.




**Obr. 7 Konstelační diagram ideálního kvadraturního modulátoru**

Vzniklé ideální rozložení bodů konstelačního diagramu modulátoru je následně degradováno nejdříve symetrickým tvarem matice ***M***, která odpovídá vztahu (6), a poté pak stejnosměrně posunut dle zadaného offsetu. Tuto operaci vyjadřuje rovnice (5), která popisuje vznik kvadraturně modulovaného signálu pomocí maticového násobení a sčítání. Takto vytvořený výsledný signál může mít konstelační diagram např. jako na Obr. 8. Pokud jsou předem známy jednotlivé hodnoty nesymetrií, jedná se při kompenzaci pouze o inverzní operace a je tak možné poměrně rychle získat opět ideální tvar rozložení koncových bodů vektorů symbolových prvků.

V tomto případě byl v algoritmu použit nesymetrický tvar kompenzátoru, kdy má matice ***C*** tvar dle rovnice (14). Následně se výstup kompenzátoru přivede na vstup modulátoru, čímž se zjistí, jakých změn dostál výstupní signál modulátoru. Dojde tedy ke zpětnému posunutí bodů konstelačního diagramu na své původní pozice – vykompenzování kvadraturního modulátoru.




**Obr. 8 Konstelační diagram modulátoru s nesymetrií**

### 3. 1. 1. Implementace adaptivního přizpůsobení zesílení a fáze

Vzhledem k tomu, že se jedná o nechtěné „vady“ modulátoru, nemusí být předem hodnoty jednotlivých nesymetrií známy a nebo se mohou jednotlivé parametry i s časem měnit. Proto musí být kompenzování vytvořeno adaptivně, aby mohl kompenzátor reagovat na případné změny signálu.

Nejdříve bylo nutné definovat jednotlivé parametry obálkového detektoru. Jednalo se zejména o hodnotu $B_e$ (pracovní bod obálkového detektoru), která má v ideálním případě nabývat nulové hodnoty. Dále pak zesílení, které je značeno $G_e$. Aby toto zesílení nemělo vliv na signál vystupující z kvadraturního modulátoru, byla použita hodnota 1. V dalším kroku dochází k definování testovací signálů $\boldsymbol{s}_m$ podle vztahů (17), které se liší hodnotou fáze. Amplituda $A$ byla nastavena na velikost rovnu 1. Výchozí hodnoty kompenzačních matic $\boldsymbol{C}$ a $\boldsymbol{f}$ byly nastaveny na hodnoty $\boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ a $\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

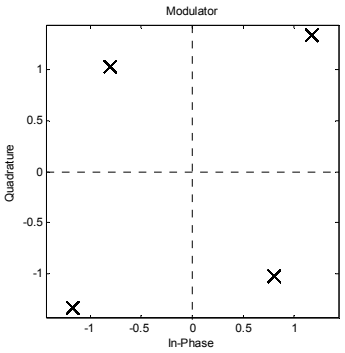
V této části je nejdříve nutné, aby proběhly čtyři stejné cykly, které se vždy liší jiným testovací signálem $\boldsymbol{s}_m$. Testovací signál je násoben kompenzačními maticemi a dále pak prochází modulátorem. Podle vztahů (18) jsou vypočítány rozdíly výstupů modulátorů mezi jednotlivými testovacími signály. Při úplném vykompenzování by tyto rozdíly měly být nulové. Protože se jedná o nesymetrickou kompenzaci jsou jako nové iterační hodnoty vypočítány koeficienty $c_{12}$ a $c_{22}$ matice $\boldsymbol{C}$ podle vztahů (20). V těchto vztazích figuruje zejména velikost nastaveného kroku metody $\delta$. Podmínky pro jeho vhodnou velikost říká vztah (21). Byly zkoušeny různé hodnoty tohoto kroku, což dokazují i výsledky simulací na Obr. 15 a Obr. 16. Pro rychlejší konvergenci iterací této metody byla nastavována velikost $\delta$ na hodnotu 0,5.

Vyskytlo se i několik problémů, které nebyly blíže specifikovány v článku [1]. Jedná se zejména o problém s kompenzací fáze. Fáze signálu byla totiž kompenzována jen zčásti. Výsledný „vykompenzovaný“ signál byl natočen o polovinu fázového úhlu $\phi$. Bylo tedy nutné do algoritmu doplnit část, která „pootočila“ konstelační diagram o zbývajících $\frac{\phi}{2}$ stupňů. S touto úpravou pak metoda fungovala již spolehlivě. Nyní bychom tedy mohli přistoupit k zobrazení jednotlivých získaných výsledků.

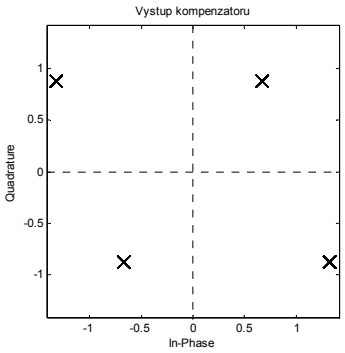
### 3. 1. 2. Výsledky simulace pro modulaci QPSK

Mezi první důležité nastavení simulace rozhodně patří nastavení nesymetrií modulátoru. Nyní bude zvoleno nesymetrické zesílení v jednotlivých větvích modulátoru. Jedna větev bude mít zesílení rovnu 1 a druhá větev např. 1,2. Fázový posuv signálu byl zvolen např. $\frac{\pi}{10} rad$ (18°). Tyto hodnoty byly záměrně zvoleny větší než jsou předpokládané hodnoty v reálném použití. Účelem byla lepší demonstrace činnosti kompenzátoru. Stejnosměrné offsetové posunutí signálu nyní není uvažováno. Krok metody byl $\delta = 0{,}5$. Jednotlivé signálové prvky jsou mapovány pomocí QPSK, konstelační digram na výstupu modulátoru ukazuje Obr. 9. Jde vidět rozdíly od ideálního mapování signálu QPSK. Dalším

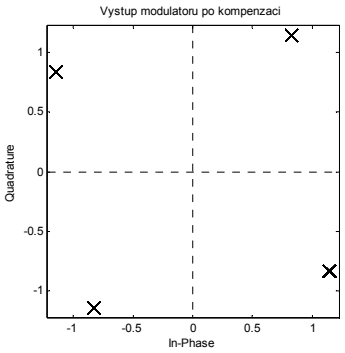
obrázkem konstelačního diagramu můžeme zobrazit výstup již adaptovaného kompenzátoru. Což ukazuje Obr. 10. Z těchto dvou diagramů lze vidět, že při jejich „sečtení“ by mělo dojít k vykompenzování signálu.




**Obr. 9 Nesymetrie kvadraturního modulátoru při *α =1,0; β = 1,2* a** $\phi = \frac{\pi}{10}$



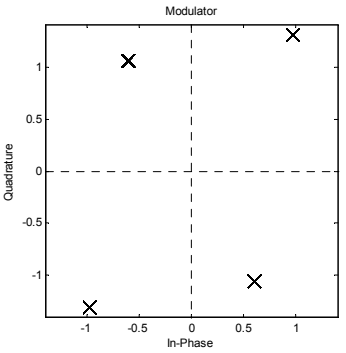

**Obr. 10 Výstup přizpůsobivého kompenzátoru**



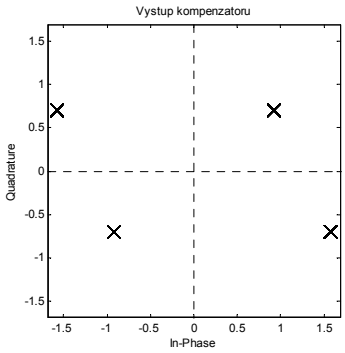

**Obr. 11 Výstup modulátoru po kompenzaci – neúplná kompenzace fázové nesymetrie**

Lze si také všimnout, že v tomto případě nedojde k úplnému fázovému vykompenzování, což je nedostatek této metody. Situace je zobrazena na Obr. 11. Problém byl vyřešen tím, že se provedlo další fázové dokompenzování, které již upravilo velikost fáze na správnou hodnotu. Výsledný konstelační diagram má poté ideální tvar, jako na Obr. 7.

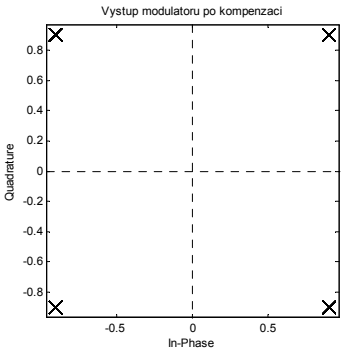
Dále můžeme zkusit, jak kompenzuje tato metoda nesymetrie, které mají symetrické rozložení. V každé větvi kvadraturního modulátoru bude signál zesilován nebo zeslabován jinak. Zesílení resp. zeslabení ve větvi I bude *α = 0,8* a větev Q bude zesílena *β = 1,2*. Fázový posuv může zůstat stejný jako v předchozím případě, tedy $\frac{\pi}{10} rad$ . Krok metody byl $\delta = 0{,}5$ . Výsledný konstelační diagram modulátoru ukazuje Obr. 12.




**Obr. 12 Nesymetrie modulátoru při *α = 0,8; β = 1,2* a** $\phi = \frac{\pi}{10}$




**Obr. 13 Výstup přizpůsobivého kompenzátoru**



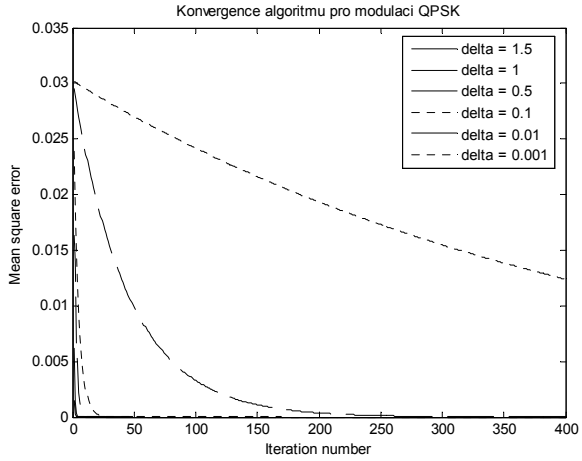

**Obr. 14 Výstup modulátoru po kompenzaci**

Z výstupu kompenzátoru jde již poznat, že nedojde v tomto případě k úplnému vykompenzování kvadraturního modulátoru. Což následně potvrzuje i Obr. 14, na kterém lze vidět, že k částečné kompenzaci opět došlo, ale jednotlivé body diagramu neleží v ideálních bodech $\pm 1 \pm j$ . Poloha těchto bodů se pohybuje kolem hodnot $\pm 0{,}9 \pm j \cdot 0{,}9$ .

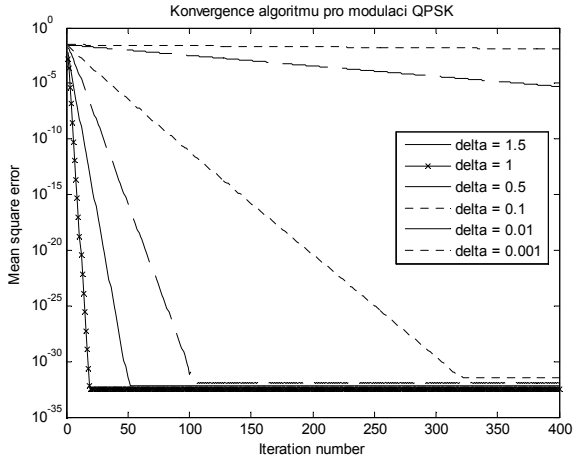
Z provedených simulací tedy vyplývá, že metoda ideálně kompenzuje v případě, kdy se jedná o nesymetrický tvar parametrů modulátoru. V jiných případech dochází do jisté míry také ke kompenzaci, ale výsledek není úplně ideální.

Následující výsledky simulací ukazují vliv velikostí nastaveného kroku metody na konvergenci metody ke správným hodnotám. Nastavení parametru velikosti kroku bylo měněno v hodnotách $\delta = \{1{,}5; 1; 0{,}5; 0{,}1; 0{,}01; 0{,}001\}$. Všechny tyto hodnoty tedy vyhovují podmínce (21). Velikosti nesymetrií modulátoru byly ve větvi I $\alpha = 0{,}8$ a ve větvi Q $\beta = 1{,}2$. Fázový posuv zůstává $\frac{\pi}{10} rad$ (18°).

Výstupem této simulace je závislost střední kvadratické chyby na počtu iteračních kroků. Je patrné, že čím menší je velikost kroku tím více kroků je potřeba, aby metoda došla ke správnému cíli. Toto všechno potvrzuje i Obr. 15 a Obr. 16. Tyto dva grafy se liší pouze měřítkem na svislé ose. V logaritmickém měřítku lze vidět, jak malá je odchylka metody. Naopak v lineárním měřítku je názornější zobrazení toho, že pro velikost kroku 0,001 je počet iteračních kroků stále malý na to, aby metoda došla ke správné hodnotě a aby se dala střední kvadratická chyba považovat v ideálním případě za téměř nulovou.




**Obr. 15 Konvergence algoritmu pro modulaci QPSK (lineární měřítko osy y)**

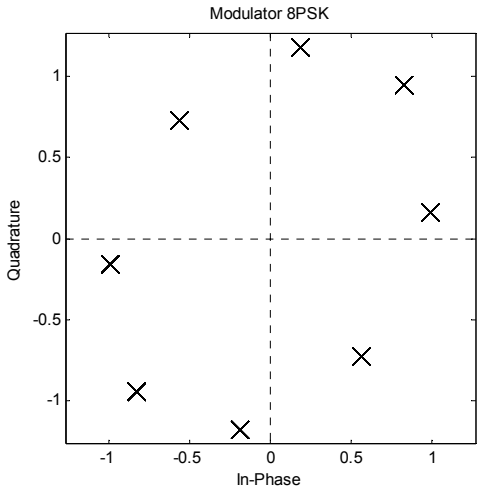



**Obr. 16 Konvergence algoritmu pro modulaci QPSK (logaritmické měřítko osy y)**

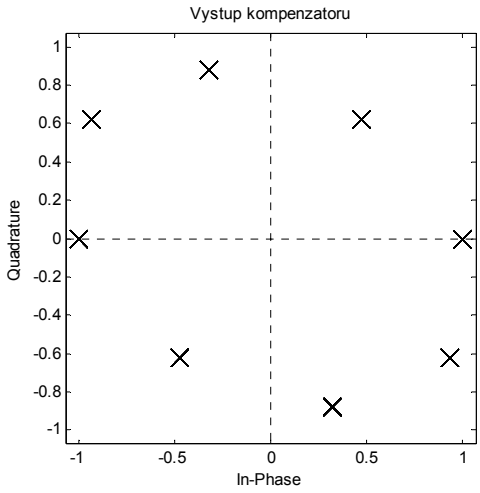
Získané grafy byly porovnány s grafy v [1] a dosažené výsledky si odpovídají. Jediná výhrada se vztahuje k tomu, že v článku [1] není uvedeno, jaké bylo nastavení simulace, které odpovídají tyto výsledky. Z grafů je patrné, že metoda nejrychleji konvergovala pro hodnotu kroku $\delta = 1$, poté byla hodnota $\delta = 0{,}5$ a jako třetí až hodnota $\delta = 1{,}5$. Pro kroky $\delta = 0{,}01$ a $\delta = 0{,}001$ metoda ještě nedorazila ke správné hodnotě. Řešením by bylo zvýšení počtu iteračních kroků.

### 3. 1. 3. Výsledky simulace pro modulaci 8PSK

Protože je tato metoda vhodná pro signály s konstantní velikostí amplitudy byla metoda zkoušena i s modulací 8PSK.

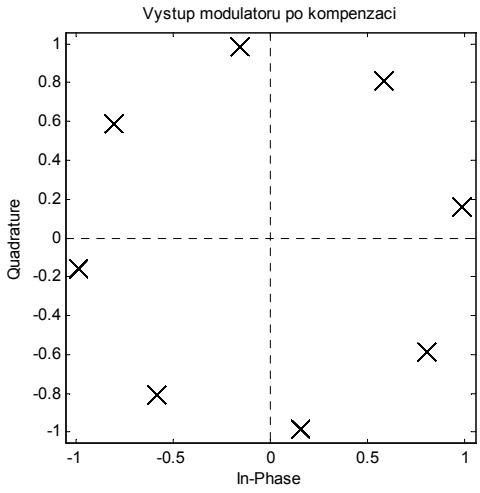



**Obr. 17 Modulátor 8PSK s nesymetriemi $\alpha = 0,8$; $\beta = 1,2$ a $\phi = \frac{\pi}{10}$**

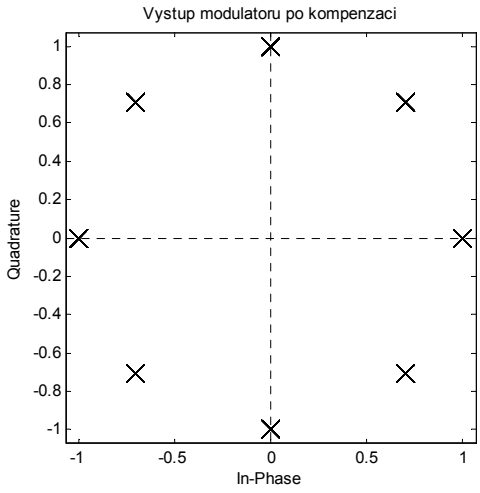



**Obr. 18 Výstup přizpůsobivého kompenzátoru**

Nejdříve byl vytvořen modulátor 8PSK a na něj pak byly aplikovány stejné nesymetrie jako v předchozích případech. Na Obr. 17 je tedy výstup nekompenzovaného modulátoru. Simulace byla nastavena stejně jako v předchozích případech. Ve větvi I byla $\alpha = 1$ a větev Q byla zesílena $\beta = 1,2$. Fázový posuv byl $\frac{\pi}{10} rad$. Krok metody zůstal na hodnotě $\delta = 0,5$. Výstup adaptivního kompenzátoru ukazuje Obr. 18, kdy jednotlivé body diagramu jsou v téměř opačných pozicích než jsou body konstelačního diagramu modulátoru.




**Obr. 19 Výstup modulátoru 8PSK po kompenzaci bez odstranění fázového pootočení**




**Obr. 20 Výstup modulátoru 8PSK po kompenzaci a odstranění fázového pootočení**

Pokud se tedy kompenzuje s tímto rozložením bodů kompenzátoru, dojde opět k tomu, že není zcela vykompenzováno fázové posunutí. Tento nedostatek ukazuje Obr. 19. Po další dodatečné úpravě dochází k úplné kompenzaci i fázového posunutí a tento ideální stav zobrazuje konstelační diagram na Obr. 20.

Z prováděných simulací je tedy patrné, že popisovaná metoda trpí nedostatky v tom, že neumí úplně vykompenzovat modulátor, který v obou větvích zesiluje (resp. zeslabuje)

signál různě. Dojde tak jen k částečnému vykompenzování. Natavením velikosti kroku metody se dá měnit rychlost konvergence metody ke konečným hodnotám a taky počet nutných iteračních kroků k nalezení správných hodnot.

## 3. 2. Popis implementace metody dle Zhu

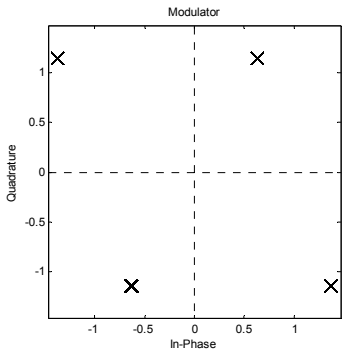
Do značné míry se základní nastavení této metody podobá metodě předchozí. Nejdříve je potřeba definovat jednotlivé nesymetrie modulátoru a demodulátoru. Poté se vytvoří ideální modulátor, který je následně degradován maticí (23). Výsledný výstupní signál modulátoru udává rovnice (22). Konstelační diagram signálu z výstupu modulátoru může být podobný, jako je na Obr. 8.

### 3. 2. 1. Implementace adaptivního přizpůsobení

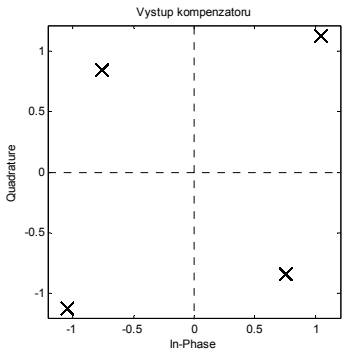
Metoda dle Dr. Zhu vyžaduje mimo jiné i počáteční nastavení vektoru ***p(n)***. Toto je poměrně důležitý výchozí parametr metody, od kterého se poté odvíjí i získané výsledky. V základním cyklu je nejdříve opakovaně prováděn výpočet konstanty $R_2$ (27), která má pro modulaci QPSK stálou hodnotu 2. Poté následuje výpočet podle maticové rovnice (24), kde dochází k vyjádření výstupního signálu modulátoru z ideálního signálu QPSK. Ve vztahu se tak přímo uplatní jak matice ***M*** a ***a***, představující nesymetrie, tak nově vzniklé matice ***P*** a ***d***, které mají kompenzovat tyto nesymetrie kompenzovat. Do jaké míry se kompenzace zdařila popisuje odchylka *e(n)*. V poslední fázi pak zbývá vypočítat následující hodnotu ***p(n+1)*** podle rovnice (28) s krokem $\mu$. Popsaný cyklus probíhá vícekrát za sebou a jeho výsledkem by mělo být postupné snižování chyby *e(n)* a tím také přibližování k ideálním hodnotám vykompenzování. Právě v tomto místě spočívá nevýhoda této iterační metody. Protože nalezené (lokální) minimum může vyhovovat podmínce, kdy se bude chyba *e(n)* blížit k nule, ale nebude to přitom globální minimum funkce, které zaručuje ideální vykompenzování. V článku [4] byla zmínka o zkoušení této metody s modulací 8PSK. Nyní budou popsány výsledky při simulaci s čtyřstavovou modulací QPSK a poté i 8PSK.

### 3. 2. 2. Výsledky simulace pro modulaci QPSK

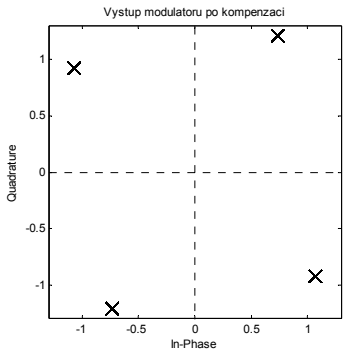
Bylo by vhodné použít podobné nastavení, jako u metody prof. Caverse, aby bylo možné nějakým způsobem metody srovnat. Zesílení v obou větvích modulátoru je zde definováno konstantou $\alpha = 1{,}2$. Fázový posuv vzniklý v modulátoru je $\frac{\pi}{10} rad$. Stejnosměrný offsetový posuv byl nastaven na hodnotu 0. Počáteční bod iterací byl zvolen s ohledem na ideální kompenzační hodnoty, které úplně vykompenzují nesymetrie modulátoru. Velikost kroku metody byla $\mu = 0{,}01$. V tomto případě jsou ideální kompenzační vypočtené hodnoty vektoru $\boldsymbol{p}$ rovny $\boldsymbol{p}(n) = \begin{bmatrix} 1 & 0{,}32492 & 0 & 0{,}87622 & 0 & 0 \end{bmatrix}^T$. Jako počáteční hodnoty byly zvoleny $\boldsymbol{p}(1) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$, což jsou poměrně značně blízké hodnoty.




**Obr. 21 Konstelační diagram výstupu modulátoru**



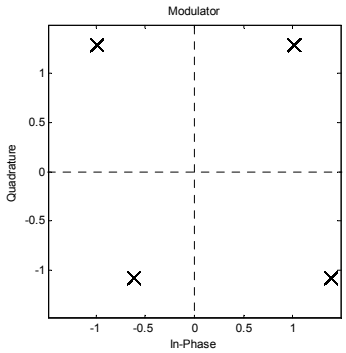

**Obr. 22 Konstelační diagram na výstupu kompenzátoru**



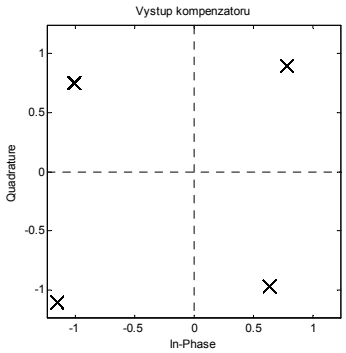

**Obr. 23 Výstup modulátoru po kompenzaci**

Konstelační diagram na Obr. 21 ukazuje, jak je modulátor ovlivňován jednotlivými nesymetriemi. Výstup kompenzátoru ukazuje Obr. 22 a hned na první pohled je patrné, že v tomto případě nedojde ke správnému vykompenzování, což následně potvrzuje výstup modulátoru po kompenzaci na Obr. 23. V tomto případě nastal problém s tím, že iterační proces sice našel jisté minimum, protože velikost chyby se pohybuje v řádu $10^{-16}$, ale sklouzl do minima lokálního, nikoliv globálního.

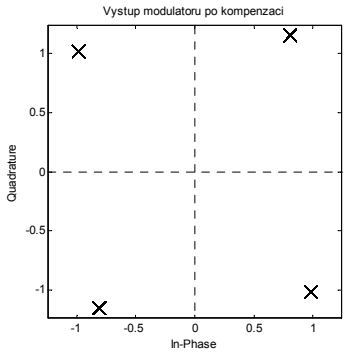
Postupným testováním metody bylo zjištěno, že nejvíce je metoda náchylná na větší hodnoty fázové zkreslení. Kompenzace nesymetrie fáze činí metodě větší problémy, než kompenzování zbylých dvou nesymetrií. V následující simulaci bylo zadáno poněkud menší fázové posunutí o velikosti $\frac{\pi}{20} rad$. Nesymetrie zesílení byla $\alpha = 1,2$ a offsetový posun modulátoru byl pro větev *I 0,2* a pro větev *Q 0,1*. Krok metody byl $\mu = 0{,}01$. V tomto případě již dochází k podstatně lepšímu vykompenzování, což ukazují i Obr. 24, 25 a 26.




**Obr. 24 Konstelační diagram výstupu modulátoru**



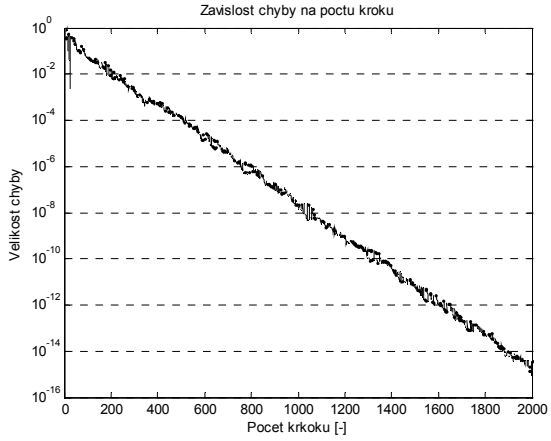

**Obr. 25 Konstelační diagram na výstupu kompenzátoru**



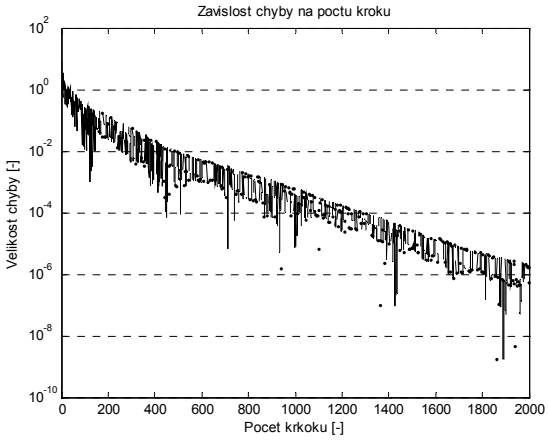

**Obr. 26 Výstup modulátoru po kompenzaci**

Lze pozorovat, že dva koncové body konstelačního diagramu na výstupu kompenzovaného modulátoru jsou kompenzovány téměř bezchybně a zbylé dva body jsou kompenzovány hůře. Ke kompenzaci tedy do jisté míry dochází, ale tyto výsledky nejsou zcela uspokojivé. Z prováděných simulací vyplývá, že metoda je schopná kompenzovat jen velmi malé nesymetrie modulátoru. Problém metodě činí zejména fázová nesymetrie. Pokud se nastaví nesymetrie fáze na nulovou hodnotu, dochází už ke správnému vykompenzování modulátoru. To bude ukázáno s modulací 8PSK, která byla použita i v [4].

Iterační kroky metody se opírají o výpočet odchylky (chyby). Právě velikost této odchylky má být minimální. Tato chyba je počítána v každém iteračním kroku a jedná se o jediné číslo, které má poté korigovat šest členů vektoru. Právě v tomto kroku může vznikat problém, protože jednomu členu vektoru může vyhovět např. jeho zvýšení hodnoty, ale druhému členu toto zvýšení naopak uškodí, protože ho vzdálí od ideální hodnoty. Kolísání této chyby v závislosti na počtu iterací ukazuje Obr. 27.




**Obr. 27 Závislost velikosti chyby na počtu iteračních kroků pro modulaci QPSK**

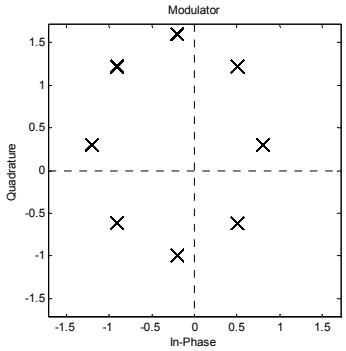



**Obr. 28 Závislost velikosti chyby na počtu iteračních kroků pro modulaci 8PSK**

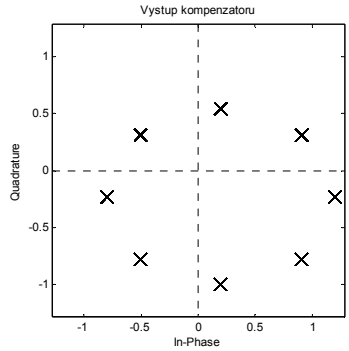
Velikost chyby se skokově mění v jednotlivých iteračních krocích, ale celkově má velikost chyby s rostoucím počtem kroků klesající charakter. V grafu klesá chyba až k hodnotám $10^{-16}$.

### 3. 2. 3. Výsledky simulace pro modulaci 8PSK

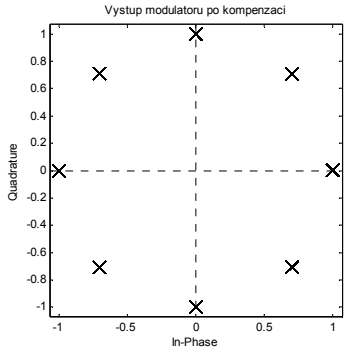
Nyní budou zobrazeny výsledky simulací při použité modulaci 8PSK. Už bylo zmíněno dříve, že tato metoda je vhodná pro modulace s konstantní velikostí amplitudy (do této kategorie spadá i modulace QPSK). Nejlépe tato metoda funguje právě s modulací 8PSK. Důkazem toho mohou být následující konstelační diagramy.




**Obr. 29 Konstelační diagram výstupu modulátoru 8PSK**




**Obr. 30 Konstelační diagram na výstupu kompenzátoru**

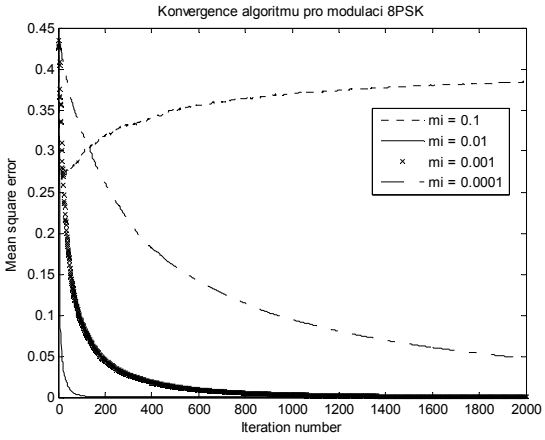



**Obr. 31 Výstup modulátoru po kompenzaci modulace 8PSK**

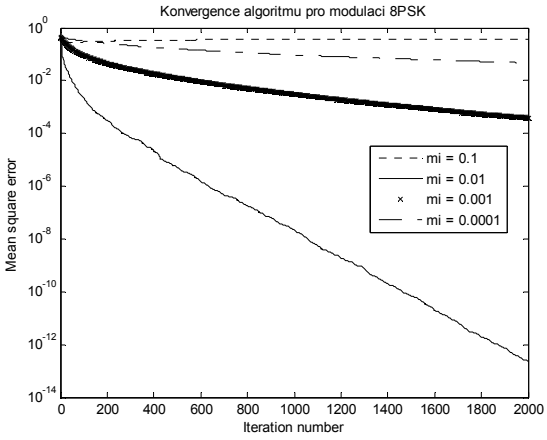
Při nastavené nesymetrii zesílení $\alpha = 1{,}3$ a offsetového posunu modulátoru pro větev *I -0,2* a pro větev *Q 0,3*, dochází k úplnému vykompenzování. Iterační krok metody byl nastaven na $\mu = 0{,}01$. Fázová nesymetrie byla v tomto případě nulová. Jako počáteční bod iteračních kroků byl zvolen $\boldsymbol{p}(1) = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}^T$. Ideální kompenzační hodnoty byly

$\boldsymbol{p}(n) = [1 \quad 0 \quad 0 \quad 0{,}76923 \quad 0{,}2 \quad -0{,}23077]^T$. K těmto hodnotám iterační proces postupně dorazil. Závislost odchylky metody na počtu iteračních kroků ukazuje Obr. 28. Průběh grafu je podobný předchozí simulaci. V tomto případě není rychlost snižování odchylky tak velká. Dochází také k větším rozdílům mezi jednotlivými iteračními kroky. V této simulaci však dochází k úplnému vykompenzování modulátoru, což se předchozím případě nedalo říci. To je dáno také tím, že nyní nebyla kompenzována fázová nesymetrie.

V poslední simulaci bude zjištěno, jakým způsobem je vhodné volit velikost iteračního kroku $\mu$ u této metody. Simulace byla nastavena podobně, jako v předchozích případech, tzn. nesymetrie zesílení $\alpha = 1{,}3$ a offsetový posun modulátoru pro větev *I -0,2* a pro větev *Q 0,3*. Průběžně byla měněna velikost iteračního kroku v hodnotách $\mu \in \{0{,}1; 0{,}01; 0{,}001; 0{,}0001\}$. Výsledkem jsou grafy střední kvadratické odchylky od ideálních kompenzačních hodnot v závislosti na počtu iteračních kroků. Na Obr. 32 je zobrazena tato závislost v lineárním měřítku osy *y* a na Obr. 33 je měřítko osy *y* logaritmické.




**Obr. 32 Konvergence algoritmu pro modulaci 8PSK (lineární měřítko osy y)**

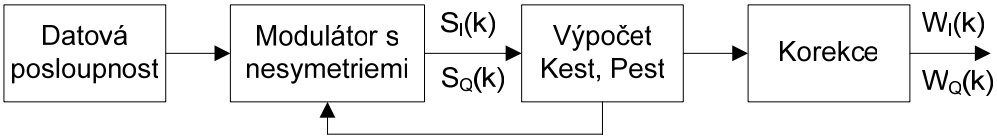



**Obr. 33 Konvergence algoritmu pro modulaci 8PSK (logaritmické měřítko osy y)**

Z grafů lze vypozorovat, že pro hodnotu $\mu = 0{,}1$ metoda diverguje a nedojde k vykompenzování. Krok metody má příliš velkou hodnotu. Naopak pro nejmenší krok $\mu = 0{,}0001$ metoda sice konverguje správně, ale příliš pomalu. Bylo by zapotřebí většího počtu iteračních kroků. Jako nejvhodnější velikost $\mu$ se jeví hodnota 0,01, protože v tomto případě metoda konverguje nejrychleji a zároveň dosažená velikost střední kvadratické odchylky je nejmenší, což je nejlépe vidět na Obr. 33.

## 3. 3. Popis implementace metody dle Helda

Ve srovnání s předchozími simulovanými metodami má tato metoda několik nesporných výhod. V první řadě lze říci, že celkový algoritmus metody je jednodušší a tím pádem by mohl být i snáze implementovatelný jednak pomocí prostředí MATLAB a poté i následně do FPGA. Nyní bude tedy shrnuto, jak byla metoda naprogramována v MATLABu a budou zobrazeny její výsledky. Na Obr. 34 se nachází blokové zobrazení této metody. Narozdíl od předchozích simulovaných metod se liší zejména zařazením korekce nesymetrií za blok modulátoru.

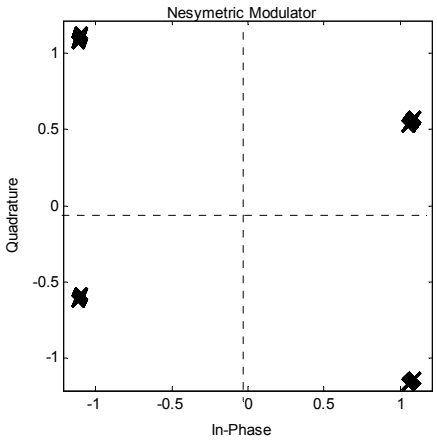



**Obr. 34 Blokové uspořádání pro metodu dle Helda**

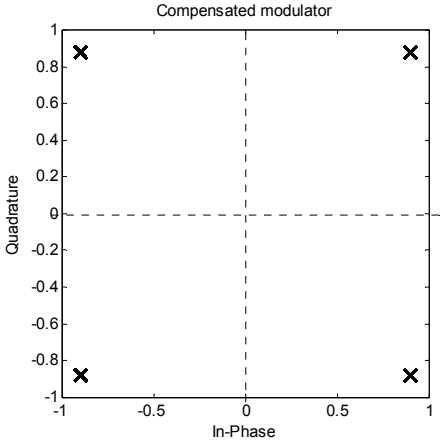
Definování jednotlivých nesymetrií bylo omezeno jen na nesymetrii amplitudy a fáze. Offset zde nebyl brán v úvahu. Nejdříve byly náhodně generovány jednotlivé datové symboly pro *I* a *Q* větev modulátoru. Maticový zápis dalšího kroku odpovídal rovnici (29), která definuje působení nesymetrií na ideální signál. Vstupní parametry simulace byly nastaveny tak, že fázový posuv byl $\varphi = \frac{\pi}{10} rad$, amplitudová nesymetrie ve větvi *I* byla $K_I = 1{,}1$ a ve větvi *Q* byla $K_Q = 0{,}9$. Obr. 35 ukazuje výsledný konstelační diagram, který vytvořily takto definované nesymetrie. Dále následoval blok pro výpočet koeficientů amplitudové (31) a fázové (33) nesymetrie. Posledním krokem algoritmu pak bylo vykompenzování jednotlivých datových symbolů obou větví modulátoru podle rovnic (34) a (35). Následně pak bylo možné zobrazit výsledky kompenzace.

### 3. 3. 1. Výsledky simulace pro modulaci QPSK

Pro možné srovnání všech tří metod byla i tato simulace podobně nastavena, což dokazují i zvolené hodnoty v předchozím odstavci. Pomocí této metody nelze eliminovat nesymetrii offsetu, což ovšem nebrání srovnání výsledků, protože i v předešlých metodách byl offset nastaven na 0. Z výsledného konstelačního diagramu, který ukazuje Obr. 36, je patrné, že došlo k vykompenzování fázové nesymetrie – přičemž bylo odstraněno pootočení konstelačního diagramu.




**Obr. 35 Konstelační diagram výstupu modulátoru QPSK s nesymetriemi ($\varphi = \pi/10\ rad$ ; $K_I = 1{,}1$; $K_Q = 0{,}9$)**




**Obr. 36 Konstelační diagram výstupu modulátoru QPSK po kompenzaci**

Lze však vidět, že amplitudová nesymetrie není úplně vykompenzována. Jednotlivé symboly neleží přesně v bodech o souřadnicích [±1, ±1], ale v bodech [±0,9; ±0,9].

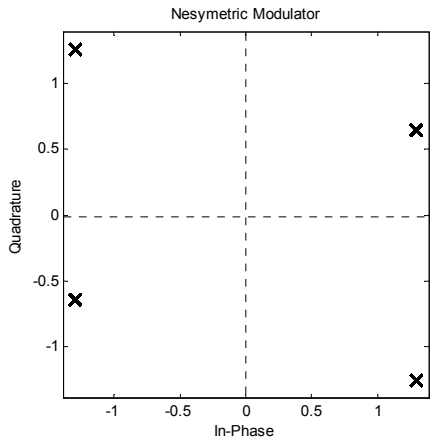
Během mnohonásobného testování této metody bylo zjištěno, že nejlepších kompenzačních výsledků bylo dosaženo v případech, kdy se pro výpočet kompenzačních koeficientů použila tzv. tréninková sekvence. Tato posloupnost byla charakteristická tím, že

její 4 datové symboly neměly náhodný charakter, ale pravidelně se opakovaly. Jako sekvence byla zkoušena např:

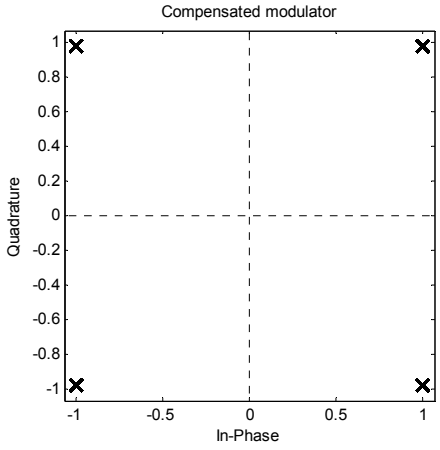
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1,1 | -1,1 | 1,-1 | -1,-1 | 1,1 | -1,1 | 1,-1 | -1,-1 | … |

Postup kompenzace byl takový, že nejdříve byla vyslána tréninková sekvence, pomocí které se určily kompenzační koeficienty $K_{est}$ a $P_{est}$. Následně byla pomocí těchto koeficientů prováděna kompenzace jednotlivých datových symbolů. Tato úprava byla provedena, protože při úplně náhodném charakteru datové posloupnosti docházelo k tomu, že se měnila zejména velikost kompenzačního koeficientu fáze $P_{est}$. To následně vedlo k horšímu vykompenzování, což bylo pozorovatelné v konstelačním diagramu.

V případech, kdy byly vstupní parametry simulace nastaveny tak, že jedna z amplitudových nesymetrií byla rovna *1* (např. fázový posuv $\varphi = \frac{\pi}{10} rad$, amplitudová nesymetrie ve větvi *I* byla $K_I = 1{,}3$ a ve větvi *Q* byla $K_Q = 1$), jsou výsledky kompenzace podstatně lepší. Tento stav modulátoru ukazuje Obr. 37, kde jsou patrné definované nesymetrie. Na výsledném konstelačním digramu po kompenzaci nesymetrií (Obr. 38) lze vidět, že dochází k ideálnímu vykompenzování, kdy se jednotlivé body diagramu nachází v souřadnicích [±1, ±1].




**Obr. 37 Konstelační diagram výstupu modulátoru s nesymetriemi ( $\varphi = \pi/10\ rad$ ; $K_I = 1{,}3$; $K_Q = 1$)**




**Obr. 38 Konstelační diagram výstupu modulátoru QPSK po kompenzaci**

### 3. 3. 2. Adaptivní kompenzace

Princip této adaptivní funkce byl takový, že byly vždy po nějaké době znovu shromažďovány jednotlivé datové symboly, které sloužily pro výpočet nových kompenzačních koeficientů. Problém s touto funkcí nastával v okamžiku, kdy nebyla vysílána tréninková sekvence, která by zaručovala nejvyšší přesnost přiblížení metody ke správným koeficientům. Docházelo tak opět k mírnému zhoršování výsledného konstelačního diagramu na výstupu modulátoru.

Řešením tedy bylo vždy zkusit vyslat krátkou tréninkovou sekvenci, určenou pro výpočet nových hodnot $K_{est}$ a $P_{est}$. Poté je vhodné pomocí těchto nových koeficientů provádět stejným způsobem kompenzaci.

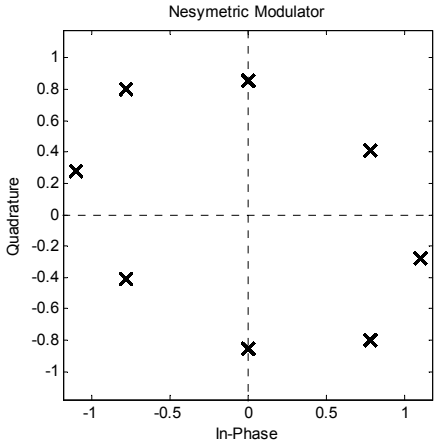
Při testování této metody bylo také zjištěno, že mnohem vhodnější je použít pro kompenzování amplitudové nesymetrie mírně modifikovaný vztah:

$$\begin{aligned} w_I[k] &= K_{est} \cdot s_I[k] \\ w_Q[k] &= s_Q[k] \end{aligned}, \qquad (36)$$

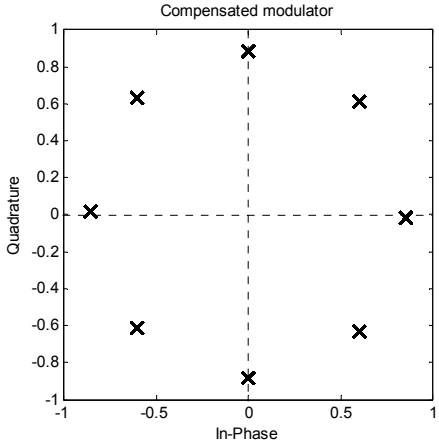
který se od (34) liší tím, že pro výpočet $w_I[k]$ není požito podílu $s_I[k]$ a $K_{est}$, ale jejich součin. Tato změna se potvrzovala při všech testovaných možnostech.

### 3. 3. 3. Výsledky simulace pro modulaci 8PSK

Zde budou zobrazeny výsledky simulace s použitou modulací 8PSK, tak, jak bylo vyzkoušeno i u předchozích dvou metod. Nastavená hodnota amplitudové nesymetrie pro větev *I* byla $K_I = 1{,}1$ a pro větev *Q* byla $K_Q = 0{,}9$. Hodnota fázové nesymetrie byla stejná jako v předchozích případech $\varphi = \frac{\pi}{10} rad$. Výsledný konstelační diagram s těmito nesymetriemi je zobrazen na Obr. 39. Lze vidět, k jaké deformaci diagramu dochází u modulace 8PSK. Dále je zobrazen na Obr. 40 výstup vykompenzovaného modulátoru. Je patrné, že došlo téměř k ideálnímu vykompenzování. Jednotlivé body v konstelačního diagramu by měly ležet na kružnici s poloměrem *1* a se vzájemným odstupem *π/4 rad*. Opět se zde projevilo neúplné dekompenzování amplitudové nesymetrie, kdy poloměr této kružnice je cca *0,9*. Fáze je vykompenzována dostatečně.




**Obr. 39 Konstelační diagram výstupu modulátoru 8PSK s nesymetriemi ($\varphi = \pi/10\ rad$ ; $K_I = 1{,}1$; $K_Q = 0{,}9$)**




**Obr. 40 Konstelační diagram výstupu modulátoru 8PSK po kompenzaci**

Z těchto získaných výsledků je možné tuto metodu považovat za vhodnou i pro modulaci 8PSK. Nevýhodou této metody proti metodám předchozím může být absence vykompenzování stejnosměrného offsetu. Naopak jedna z výhod simulované metody může spočívat v její jednoduchosti, což se pak následně může projevit i v možnosti jejího implementování na FPGA.

# 4. Implementace na FPGA

Každá ze zkoumaných metod trpí při specifických nastaveních určitými nedostatky a také naopak v některých případech se chová lépe, než ostatní metody. Důležité tedy bylo zvolit kompromis mezi uspokojivým vykompenzováním modulátoru a náročností celkového výpočtu algoritmu metody. Na tyto ohledy bylo nutné brát zřetel, protože dalším krokem

řešení zadaného problému bylo implementovat některou z metod pomocí programovacího jazyka VHDL (**V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage) na obvod FPGA (**F**ield **P**rogrammable **G**ate **A**rrays).

Nejdříve bylo nutné seznámit se základní syntaxí a způsobem zápisu v jazyce VHDL, který se značně liší od klasických programovacích jazyků. Většina základních informací byla čerpána z literatury [8]. Byly zde vysvětleny základní pojmy týkající se jazyka VHDL. Důležitým pomocníkem byla i kniha [7], v které se nacházejí vytvořené příklady.

Jako programovací nástroj bylo zvoleno prostředí Xilinx ISE. Program umožňuje zvolit několik druhů programování, např. pomocí schématického kreslení jednotlivých bloků nebo přímé psaní VHDL kódu. Pro větší srozumitelnost byl zvolen přímý zápis VHDL kódu. Nástrojem pro simulování získaných výsledků byl vybrán ModelSim, který umožňoval mnohem lepší zobrazení a srovnávání výstupních dat než interní program ISE Simulator. V následujících podkapitolách budou uvedeny a popsány jednotlivé použité bloky při VHDL syntéze. Získané výsledky budou jednak výstupy ze simulačního prostředí ModelSim a následně poté i reálné hodnoty získané měřením na vektorovém signálovém analyzátoru. Nezbytný bude i základní popis desky Virtex 2 V2MB1000 spolu s analogovým modulem Memec P160, na které byly metody implementovány a následně měřeny.

## 4. 1. Prostředí Xilinx ISE

Postup programování v prostředí Xilinx je možno rozdělit do několika částí, podle kterých bylo při implementování postupováno. Jednoduché vysvětlení těchto kroků bude uvedeno nyní.

Nejdříve byl tedy psán samostatný VHDL kód, který měl za úkol vytvářet stejné algoritmy, požité u simulací v prostředí Matlab. K dosažení správné funkčnosti byly požívány některé z dostupných CORE bloků, které byly vytvářeny pomocí tzv. IP CORE generátoru. Mezi tyto použité bloky např. patřil DCM (Digital Clock Manager), DSS (Direct Digital Synthesizer), floating-point blok nebo FIR filtr. Po správném definování vstupních a výstupních portů pak bylo možné pokročit do další části a provést základní simulaci v programu ModelSim. V tomto prostředí se zobrazovaly jednotlivé časové průběhy všech signálů použitých v projektu. Dalo se vytvořit i několik základních operací nad těmito signály a také volit různá zobrazení výstupních hodnot. Testováním a simulováním tak bylo možné ladit program nebo případně odstraňovat syntaktické chyby.

Další částí, která je nutná pro správnou implementaci, bylo provedení celkové syntézy naprogramované funkce. Zde se mohly vyskytnout některé problémy v podobě toho, že obvod není realizovatelný. V nastavení procesu syntézy je možnost zvolit optimalizaci na rychlost, při níž obvod funguje s nejvyšším možným taktovacím kmitočtem, nebo optimalizaci na plochu. V tom případě by obvod již nemusel být tak rychlý, ale bylo by více užito vhodnějšího rozmístění jednotlivých bloků na čipu. Při testování bylo využíváno obou nastavení. Pokud syntéza proběhla správně, bylo možné zobrazit celkové schéma složené z jednotlivých logických bloků tak, jak bylo naprogramováno ve zdrojovém kódu. To mohla být do jisté míry také kontrola, kdy bylo možné sledovat propojení jednotlivých bloků. Nutné bylo taky definovat propojení vstupních a výstupních signálů programu s jednotlivými

vývody programovatelného pole. Ve vytvořeném souboru (*.ucf) bylo také nutné správně definovat taktovací hodinový signál FPGA, od kterého se pak odvíjí celkové časování obvodu.

Po úspěšné syntéze bylo možné přistoupit k dalšímu kroku, kterým je návrh implementace na čip programovatelného pole. Zde se opět může vyskytnout několik problémů s tím, že např. obvod může být realizovatelný, ale vzniklé zapojení se nevleze na použitý čip nebo ho není možné správně fyzicky propojit. Projevit se může i to, že při složitějším obvodu mají jednotlivé signály větší zpoždění a musí se tak několik hodinových taktů čekat, až signál dorazí na potřebný vstup nebo výstup bloku. Problémů s konečnou implementací se vyskytovala celá řada. Postupnými kroky tak bylo nutné měnit vytvořené zapojení nebo nastavení jednotlivých bloků, aby implementace úspěšně proběhla.

Jedním z posledních kroků před konečným naprogramováním FPGA bylo vytvoření programovacího souboru. Ten může být nastaven pro naprogramování PROM paměti nebo přímé programování FPGA. K počítači se následně připojí JTAG kabel, pomocí kterého se vytvořené soubory zavádějí do testovací desky. Úspěšné zdolání všech těchto kroků vede k tomu, že vytvořený program je implementován v FPGA a následně je možné ověřovat správnou funkci např. měřením na výstupních portech. Zde byl v jednotlivých krocích popsán postup vytvoření a implementování VHDL kódu v prostředí Xilinx. Následovat bude popis použitých funkčních CORE bloků a algoritmů testovaných metod.

### 4. 1. 1. Použité IP CORE bloky

Pro vytváření některých složitějších funkcí bylo možné použít již vytvořených bloků, které bylo nutné správně definovat a nastavit. K tomuto nastavovaní slouží tzv. IP CORE generátor. Po správném nastavení požadovaných parametrů pak dojde k vygenerování zdrojového kódu bloku. V hlavním programu se pak jen správně definují vstupní a výstupní proměnné. Správná funkčnost a nastavení každého bloku byla pak následně pozorována při simulaci.

**FLOATING - POINT blok**

Při realizaci kterékoliv ze zkoumaných tří metod byl vždy řešen problém s výpočty v desetinné čárce. Problém nastává tehdy, když se mají realizovat binární výpočty desetinných čísel. Nejprve byl zkoušen zápis binárních čísel v tzv. Q formátu. Kdy je použit první bit jako znaménkový. Poté následovala část před a za desetinnou čárkou. Zápis může být *Qm.n*, kdy bylo celkem použito *m* bitů před a *n* bitů za desetinnou čárkou. Na závěr se připočítával ještě jeden znaménkový bit. Pokud se tedy chce vyjádřit binárně ve formátu *Q1.6* číslo např. *1,90625* vznikne binární zápis *01.111010*. První bit představuje kladné znaménko, druhý bit je část před desetinnou čárkou a zbylých 6 bitů tvoří desetinnou část s hodnotu:

$$1.111010 \Rightarrow 1\cdot 2^{0}+1\cdot 2^{-1}+1\cdot 2^{-2}+1\cdot 2^{-3}+0\cdot 2^{-4}+1\cdot 2^{-5}+0\cdot 2^{-6}=\underline{1,90625}$$

Problém s reprezentací desetinných čísel v tomto formátu se projevil např. při vícenásobných součinových nebo podílových operacích. Navíc jen stěží by byla realizovatelná odmocnina tohoto binárního desetinného čísla. Jedno z řešení se naskytlo použití bloku Floating - Point, který pracuje s 32-bitovými čísli (Single Precision). První bit tak představuje opět znaménko,

následuje 8 bitů, které tvoří exponent, a 23 bitů mantisy. Výsledné číslo vyjádřené ve floating-point se vypočítá:

$$x = s \cdot 2^e \cdot m . \qquad (37)$$

Kde *s* má hodnotu *+1* pro kladná čísla (znamínkový bit je 0) a hodnotu *-1* pro záporná čísla (znamínkový bit je 1). Hodnota *e* představuje číslo *exponent – 127*. Mantisa je značena písmenem *m*.

**Tab. 1 Tvar čísla floating-point**

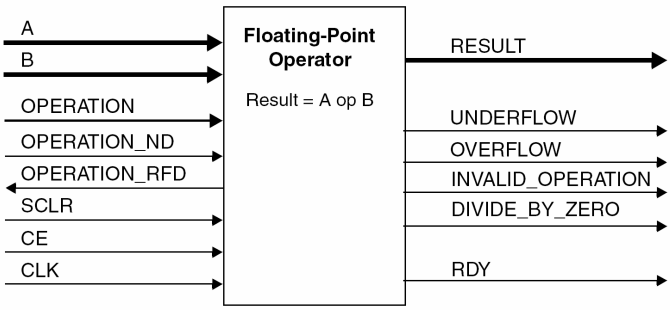
| *s* | *e* exponent 8 bitů | | | | | | | | *m* mantisa 23 bitů | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | | | | | | | | 23 | | | | | | | | | | | | | | | | | | | | | | | | 0 |

Nastavení bloku Floating-Point probíhalo dle pokynů v CORE generátoru. Nejdříve se zvolil druh operace, kterou má blok provádět. Na výběr bylo z několika možností:

- součet
- rozdíl
- součin
- podíl
- odmocnění
- porovnání
- převod z Fixed na Float
- převod z Float na Fixed
- převod z Float na Float

Všechny tyto operace probíhaly s čísly ve tvaru s plovoucí čárkou. Výhodné bylo i použití zpětného převodu z pohyblivé desetinné čárky na pevnou. Po nastavení typu operace bylo možné zvolit tvar vstupních čísel. Na výběr byly tři možnosti přesnosti: 32 bitů (single precision), 64 bitů (double precision) a uživatelsky definovaný počet bitů. Toto nastavení definovalo tvar vstupních i výstupních hodnot proměnných. Pro všechny výpočty prováděné s plovoucí desetinnou čárkou byla nastavena přesnost single precision. Zpětný převod z pohyblivé desetinné čárky na pevnou byl nastaven na délku 12 bitů. Tři bity tvořily část před desetinnou čárkou a znaménko. Devět bitů definovalo desetinnou část. Toto nastavení bylo zvoleno z toho důvodu, protože bylo předpokládáno, že nesymetrie budou dosahovat desetinných čísel v rozmezí cca od -2 do +2.

Na Obr. 41 lze vidět všechny vývody, které floating-point blok nabízí a které bylo možno aktivovat při nastavování. Vždy byly zapojeny vstupní a výstupní vývody spolu s hodinovým signálem. Jako pomocný řídící signál byl v některých případech volen vstup CE. Ostatní vývody nebyly využity.




**Obr. 41 Znázornění vývodů bloku floating-point**

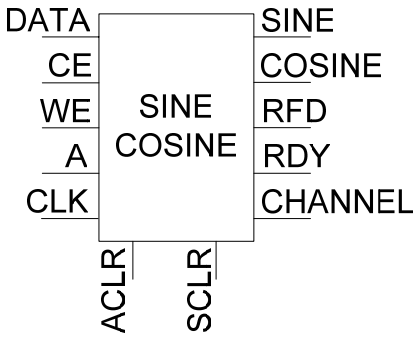
Posledním bodem nastavení byla volba druhu architektury daného bloku. Důraz mohl být kladen na nízké zpoždění bloku a jeho vyšší rychlost. Pro blok, který představoval násobičku, bylo možné vytvořit tuto funkci pomocí speciálních 18 bitových násobiček, které obsahovalo FPGA Virtex 2. Tak bylo možné v programovatelném poli ušetřit logické jednotky (LUT – LookUp Table). U zbylých funkcí bylo vždy nutné použít logické jednotky LUT. Nastavení parametru LATENCY definovalo počet hodinových taktů mezi aktuálními platnými daty na vstupu a odpovídajícími daty na výstupu bloku. Toto zpoždění se postupně nastavovalo podle požadavků, které vznikaly při implementaci. Zpoždění pro jednotlivé bloky bylo nastaveno:

- násobení 2 cykly
- dělení 12 cyklů
- odmocnina 10 cyklů
- rozdíl 3 cykly
- součet 3 cykly
- převod float-fixed 6 cyklů

Těmito parametry bylo nastavování bloku u konce a posledním úkonem bylo vygenerování zdrojového kódu definovaného bloku.

**DIRECT DIGITAL SYNTHESIZER (DDS) blok**

Jednalo se o blok, který byl použit pro vytvoření nosného signálu funkce sinus a cosinus kvadraturního modulátoru. Blok tedy vytváří „vysokofrekvenční“ nosný signál pro modulátor. Nastavení parametrů bloku probíhalo opět v několika krocích. Nejprve se definovalo, jaké signály se mají generovat.

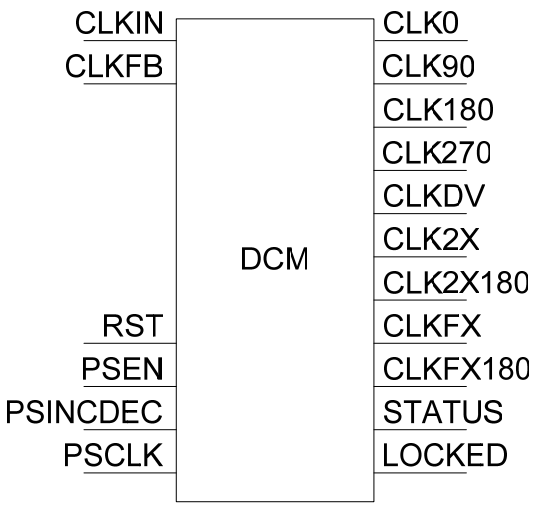



**Obr. 42 Znázornění vývodů bloku DDS**

V tomto případě bylo nastaveno generování sinu a cosinu současně. Frekvence, s kterou byl časován blok DDS, byla nastavena na 100 MHz (příp. 50 MHz). Přesnost generování tohoto signálu byla nastavena na 0,3 Hz. Výstupní signál byl pak nastaven tak, aby měla nosná frekvenci 500 kHz. Nastavení fázového posuvu bylo ponecháno na nulové hodnotě. V dalších oknech nastavení byla volba a aktivace jednotlivých ovládacích vývodů bloku DSS. V tomto případě bylo použito nejjednodušší uspořádání, kdy byl zapojen pouze hodinový vstup CLK a dva výstupy (SINE, COSINE), na kterých byl generován výstupní periodický signál. Zpoždění výstupu celého bloku bylo 3 hodinové cykly. Šířka výstupních signálů byla nastavena na 12 bitů. Jedna perioda výstupního signálu tak bude tvořena 100 diskrétními vzorky, což představuje dostatečně „jemné“ vzorkování pro tento harmonický signál.

**DIGITAL CLOCK MANAGER (DCM) blok**

Programovatelné pole Virtex 2 obsahuje přímo v sobě několik bloků DCM. Jedná se o zapojení, které je schopné vytvářet ze vstupního signálu o určité frekvenci výstupní signál s jinou frekvencí. Blok byl v projektu využit pro generování hodinového signálu pro D/A převodníky, umístěné na desce Memec P160, a hodinového signálu pro celé FPGA. Jako vstupní byl brán signál ze 100 MHz krystalového oscilátoru, umístěného na desce V2MB1000 (viz. Obr. 58).

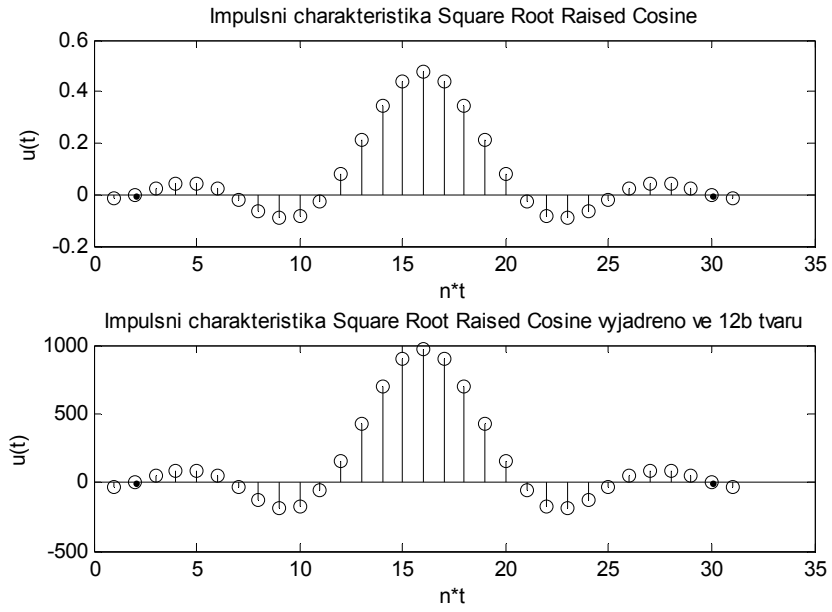



**Obr. 43 Znázornění vývodů bloku DCM**

Při implementaci byl nejdříve zkoušen hodinový signál s frekvencí 100 MHz pro celé FPGA. S rostoucím počtem použitých prvků a vytížením většího množství logických jednotek programovatelného pole musel být kmitočet celého obvodu snížen na hodnotu 50 MHz. Jinak nebylo možné provést implementaci se zdárným koncem. Se sníženým kmitočtem již byla implementace možná. Byla tedy nastavena dělička signálu na hodnotu 2, která dělila vstupní externí 100 MHz signál. Zpětná vazba DCM bloku byla použita interní a její hodnota byla 1. Na Obr. 43 lze vidět všechny vývody, které umožňuje DCM blok zapojit. Výstupní signály jsou brány i s různými fázovými posuvy. Pro nastavení hodinové frekvence D/A převodníku na desce Memec P160 byly využity piny CLK0 a CLK180, které generovaly signál s frekvencí 100 MHz a se vzájemným fázovým posuvem o 180°. Na desce bylo tak zvlášť řízeno vzorkování D/A převodníku a zápis do vstupního registru převodníku. Dále byl zapojen vývod CLKDV, na kterém se vyskytoval signál s frekvencí, která odpovídala nastavení v děličce (50 MHz). Tento signál byl tedy brán jako hodinový pro celé FPGA. Výstup CLKFB byl nastaven jako otevřený výstup. Ostatní z vývodů nebylo nutné zapojovat a tedy i nastavovat.

## DISTRIBUTED ARITHMETIC FIR filtr

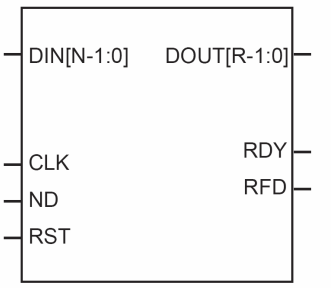
Blok, který představuje FIR filtr, tedy filtr s konečnou impulsní charakteristikou, byl zapojen na výstup obou větví kvadraturního modulátoru. Celkem byly použity dva bloky, přičemž oba byly určeny ke snížení mezisymbolových interferencí.




**Obr. 44 Impulsní odezva Square Root Raised Cosine filtru**

Impulsní odezva filtru byla zvolena ve tvaru Square Root Raised Cosine. To umožňuje použít stejný filtr na přijímací straně se stejnou impulsní odezvou. Přijímací strana byla při měření tvořena vektorovým analyzátorem, který měl volbu zapojení tohoto filtru, což bude popsáno později. Filtr tohoto typu lze charakterizovat několika parametry. Jednak vzorkovací frekvencí signálu, který se má filtrovat. Dále vzorkovací frekvencí filtru, která musí být vyšší než je vzorkovací frekvence signálu a zároveň musí být jejím celistvým násobkem. Poměr těchto dvou frekvencí tedy udává počet vzorků filtru na jeden vzorek signálu. Následně je definován tzv. rolloff faktor, který udává rozšíření přenosové funkce filtru vzhledem k filtru, který ideálně potlačuje mezisymbolové interference. Posledním parametrem je zpoždění, které filtr zavádí. Toto zpoždění odpovídá polovině délky jeho impulsní charakteristiky. Z důvodů omezených možností na volné logické jednotky v FPGA a náročnost realizace tohoto filtru byla zvolena celková délka impulsní odezvy na 31 vzorků. Odezva filtru a jednotlivé vzorky byly vytvořeny v MATLABu. Vzorkovací frekvence byly nastaveny tak, že jednomu vzorku filtrovaného signálu odpovídalo 5 vzorků filtru. Rolloff faktor byl nastaven na hodnotu 0,22. Výsledná charakteristika je zobrazena na Obr. 44.

Nastavení bloku s FIR filtrem bylo provedeno dle základních instrukcí v CORE generátoru. Na výběr bylo několik módů, které mohl blok realizovat. Vybrána byla první varianta filtru Single-Rate, která odpovídala nastavení, kdy je vstupní vzorkovací frekvence rovna výstupní vzorkovací frekvenci. Počet kanálů filtru byl ponechán na hodnotě 1. Jedná se o filtr, který má dle Obr. 44 symetrickou impulsní odezvu a na její vyjádření je potřeba znaménkového bitu.




-35, 2, 50, 86, 89,
45, -36, -128, -187, -170,
-53, 161, 433, 702, 899,
**971**,
899, 702, 433, 161, -53,
-170, -187, -128, -36, 45,
89, 86, 50, 2, -35

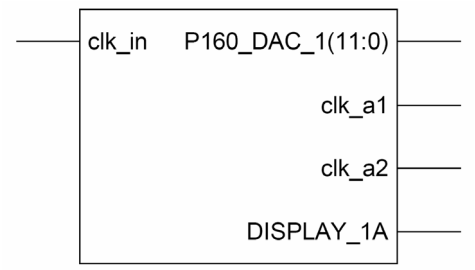
**Obr. 45 Znázornění vývodů bloku FIR filtru a 12 bitových koeficientů impulsní odezvy filtru**

Tyto parametry byly nastaveny v dalších možnostech bloku. Počet vzorků celé odezvy byl tedy 31. Jednotlivé koeficienty impulsní odezvy byly definovány jako 12 bitové s tím, že první bit tvoří znaménko. Proto je i v Obr. 44 uvedena druhá charakteristika, která je tvarově stejná, ale koeficienty odpovídají 12 bitovému dekadickému vyjádření. Přesné hodnoty koeficientů jsou uvedeny v Obr. 45. Uloženy byly samostatně v souboru s příponou *.coe, který byl pak následně načten CORE generátorem. Na Obr. 45 jsou také znázorněny jednotlivé vývody bloku FIR filtru v módu Single-Rate. Vstupní data byla tedy 12 bitová, celkovému nastavení pak odpovídala výstupní šířka dat 29 bitů. Obvod zaváděl zpoždění o délce 9 hodinových cyklů. Blok fungoval tak, že vždy na platný signál ND (New Data) načetl filtr vstupní data. Poté se čekalo na aktivaci výstupu RDY (Ready), který signalizoval, že na výstupu DOUT jsou platná data.

Tím zde byly popsány použité funkční bloky v programu. Následně bude popsáno implementování konkrétní metody s použitím i těchto zmíněných bloků.

### 4. 1. 2. Implementace metody dle Helda

Základem této metody je algoritmus, který byl popsán v kapitole 2. 3. Implementace ve VHDL je poněkud odlišná, protože napsaný program není vykonáván postupně za sebou podle toho, jak je napsán v kódu programu, ale probíhá současně. Vše je řízeno hodinovým signálem, případně jeho náběžnou (resp. sestupnou) hranou.

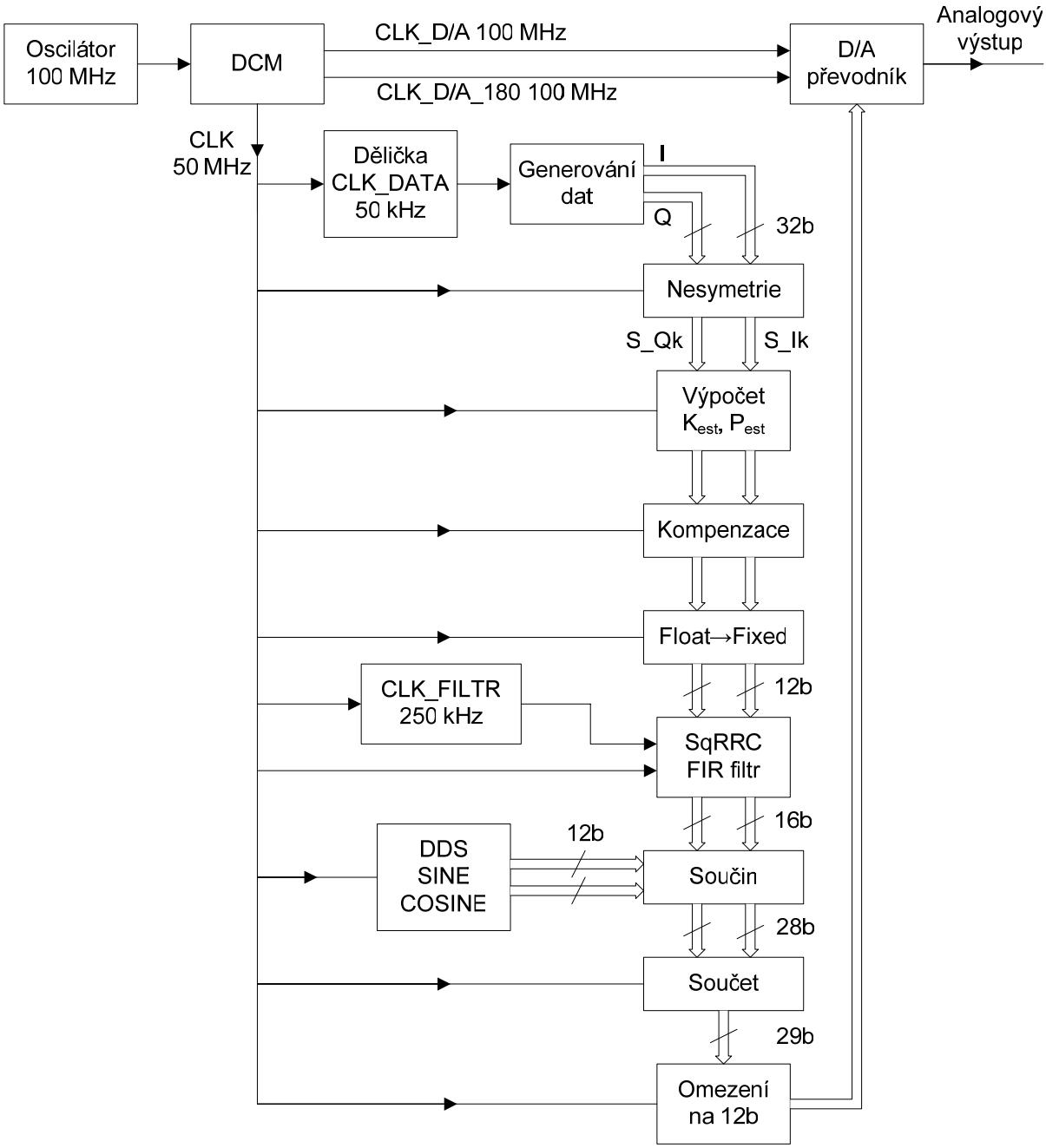



**Obr. 46 Znázornění vývodů celkového algoritmu**

Na Obr. 46 je znázorněna schématická značka s jednotlivými vývody, které tvoří vstup a výstupy algoritmu. Podle uvedeného obrázku vstupuje pouze hodinový signál o frekvenci 100 MHz do CLK_IN. Naopak výstupní jsou dva hodinové signály vzájemně posunuté o 180°, které řídí převodník D/A na analogové desce. Výstup DISPLAY_1A slouží pouze pro kontrolu funkce běžícího programu. Což signalizuje blikající LED. Hlavním datovým výstupem je tedy 12 bitový signál P160_DAC, který je následně přiváděn na datový vstup D/A převodníku.

Důležitým blokem, který realizuje většinu výpočtů této metody je blok Floating-point, který byl zapojen pro realizaci součinu, součtu, podílu, rozdílu, odmocniny a převodu z Floatin-point na Fixed-point. Aby byly ušetřeny volné logické jednotky na programovatelném poli, byl základní výpočet algoritmu metody vytvořen tak, že byl použit vždy jen jeden blok, který vytváří např. funkci součinu, aby obsloužil postupně všechny výpočty, kde se součin čísel nachází. Jedná se tedy o sdílení jednotlivých hardwarových prostředků. Celý realizovaný algoritmus metody je zobrazen na Obr. 47.

Toto blokové uspořádání využívá logické jednotky v FPGA téměř na 100%. Funkce algoritmu je patrná z obrázku. Nejdříve je signál z krystalového oscilátoru o frekvenci 100 MHz přiveden do bloku DCM, který jednak časuje hodinový signál pro D/A převodník a dále pak snižuje kmitočet na 50 MHz, kterým je taktován zbytek všech členů na programovatelném poli. Pro generovaní datových signálů slouží kmitočet 50 kHz, který vzniká v děličce. S touto frekvencí jsou generována data pro $I$ a $Q$ větev. Následuje „přidání“ nesymetrií a výpočet jednotlivých kompenzačních koeficientů $K_{est}$ a $P_{est}$. Poté je prováděna kompenzace za pomocí získaných koeficientů. Dále je proveden převod z 32 bitového čísla ve tvaru Floating na 12 bitové číslo ve tvaru Fixed point. Od bloku generování dat po blok převodu čísel jsou prováděny matematické operace v pohyblivé desetinné čárce s 32 bitovými čísly. Tyto výpočty jsou náročné na počet využitých logických bloků v FPGA. Proto byl program zapsán tak, aby jeden blok pro danou operaci sloužil pro všechny výpočty, kde byla tato operace použita. Na vstupy bloku byly postupně přiváděny různé vstupní hodnoty a s odpovídajícími časovými intervaly byly odebírány hodnoty výstupní.



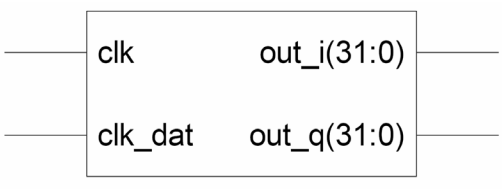

**Obr. 47 Blokové schéma implementované metody dle Helda**

Po získání desetinného čísla ve tvaru s pevnou desetinnou čárkou byl jako další blok použit Square Root Raised Cosine filtr, který byl zařazen pro každou větev zvlášť. Filtr byl jednak taktován s frekvencí 50 MHz, ale zároveň bylo třeba, aby jednomu datovému bitu odpovídalo 5 vzorků filtru. Což řídil zvlášť signál s frekvencí 250 kHz. Počet těchto vzorků byl dán poměrem vzorkovací frekvence filtru ku vzorkovací frekvenci filtrovaného signálu. Pro tyto hodnoty byla vygenerována impulsní odezva z Obr. 44. Výstupní signál z obou filtrů měl šířku 29 bitů, ale pro další operace bylo využito jen 16 bitů. Následoval součin signálu s generovanou funkcí sinus a cosinus v obou větvích modulátoru. Po součinu byly větve *I* a *Q* sečteny a vznikl signál s šířkou 29 bitů. Tato šířka byla dále omezena na 12 bitů, protože převodník D/A byl právě 12 bitový. Do převodníku tedy již vstupoval vykompenzovaný

signál z modulátoru s nosnou frekvencí 500 kHz. Velikost výstupního analogového signálu se pohybovala v rozmezí od 1,5 do 3,5 V, tak, jak je uvedeno v Tab. 3.

Dále pak budou popsány a zobrazeny výsledky takto naprogramované kompenzační metody.

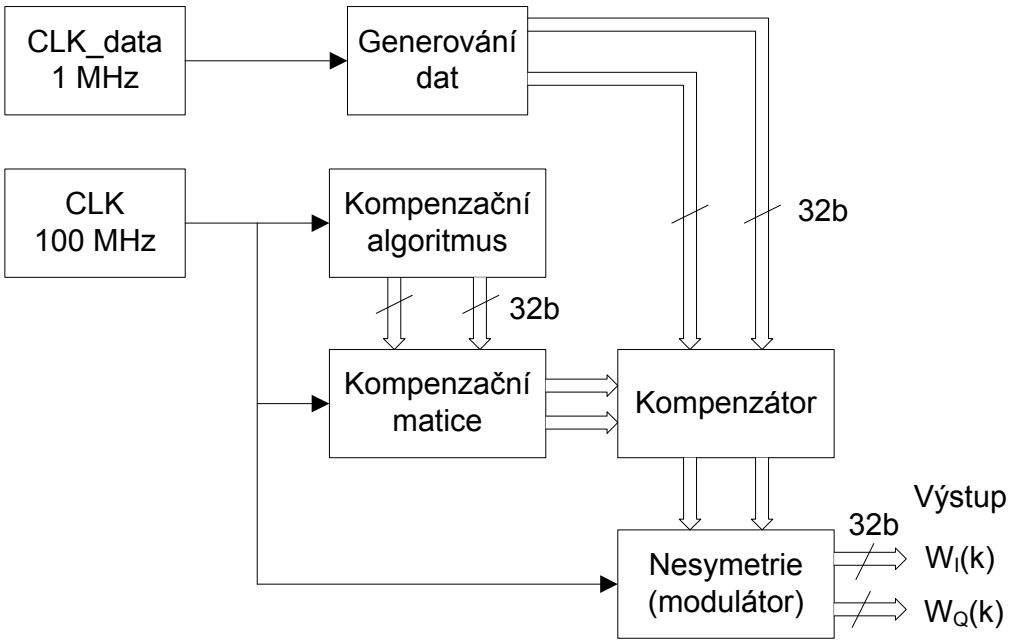
### 4. 1. 3. Implementace metody dle Caverse




**Obr. 48 Znázornění vývodů celkového algoritmu**

Základ metody tvoří algoritmus popsaný v kapitole 2. 1. Zpracování výpočtu bylo prováděno jen na takové úrovni, kdy hlavním výsledkem měly být simulace v programu ModelSim. Hardwarová implementace na desku s programovatelným polem nebyla realizována. Metoda je totiž náročnější na jednotlivé výpočty a operace. Celý algoritmus by tak realizovatelný na programovatelné logické pole Virtex 2. Vstupní a výstupní vývody jsou vidět na Obr. 48.

K provádění většiny operací bylo opět použito bloků pracujících s plovoucí desetinnou čárkou. Většina proměnných byla tedy 32 bitová, což samo o sobě představovalo velké využití logických bloků programovatelného pole na pokrytí těchto proměnných. Blokové uspořádání implementované metody lze vidět na Obr. 49. Schéma je zakresleno poměrně jednoduše, protože není podrobněji rozkreslen blok, který představuje kompenzační algoritmus a ve kterém se skrývá nejvíce operací.




**Obr. 49 Blokové schéma implementované metody dle Caverse**

Datový signál byl generován s frekvencí 1 MHz, ostatní funkční bloky jsou řízeny kmitočtem 100 MHz. Velkou část výpočtů tedy prováděl blok kompenzačního algoritmu, ve kterém bylo použito podobné sdílení hardwarových prostředků, jako v předchozí metodě. Výsledkem bylo nalezení jednotlivých prvků kompenzační matice, které následně slouží k tomu, aby došlo

k předkreslení signálu v kompenzátoru. Za kompenzátorem pak následoval modulátor s nesymetriemi. Při správném předzkreslení signálu by měl být výstupní signál z modulátoru již vykompenzovaný. Výstupní signály jsou stále ve tvaru s plovoucí desetinnou čárkou. Hodnoty těchto signálů byly vyexportovány do souboru a dále zpracovány v MATLABu, kde byly zobrazeny do konstelačního diagramu. Nebyly zapojeny další bloky, jako generátor funkce sinus a cosinus nebo převod na pevnou desetinnou čárku. Převodník na analogový signál nebylo v simulaci také potřeba zapojovat.

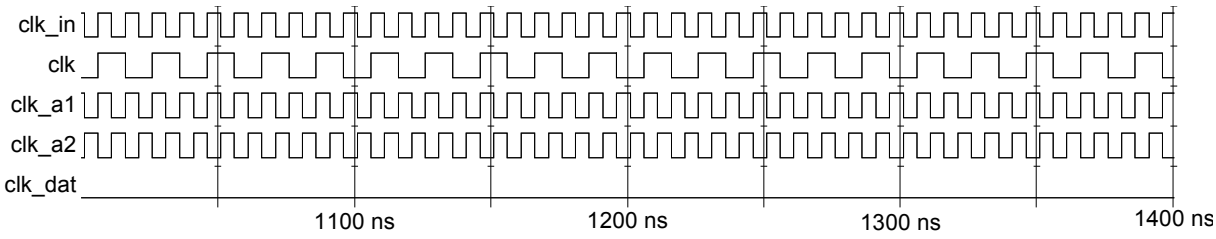
Výsledky ze simulačního programu budou zobrazeny v následující kapitole.

## 4. 2. Simulace v ModelSim

Jednotlivé simulace byly prováděny v programu ModelSim, který umožňoval pokročilejší nastavení získaných výstupních dat z naprogramovaného kódu než původní iterní simulační program v prostředí Xilinx ISE. Pro „oživení“ každé funkce nebo použitého bloku byly provedené simulace velmi důležité. Vznikala tak stále kontrola stavu výstupních dat. Bylo snazší odhalovat vzniklé chyby algoritmu a následně je odstraňovat. Získaná data bylo také možné exportovat do souboru a dále po úpravě s nimi pracovat např. v MATLABu. Což bylo využito při zobrazování konstelačních diagramů výsledných hodnot jednotlivých metod. Dají se tak srovnat dosažené výsledky jednotlivých simulací prováděných v MATLABu s výsledky, které byly dosaženy v průběhu výpočtu na FPGA.

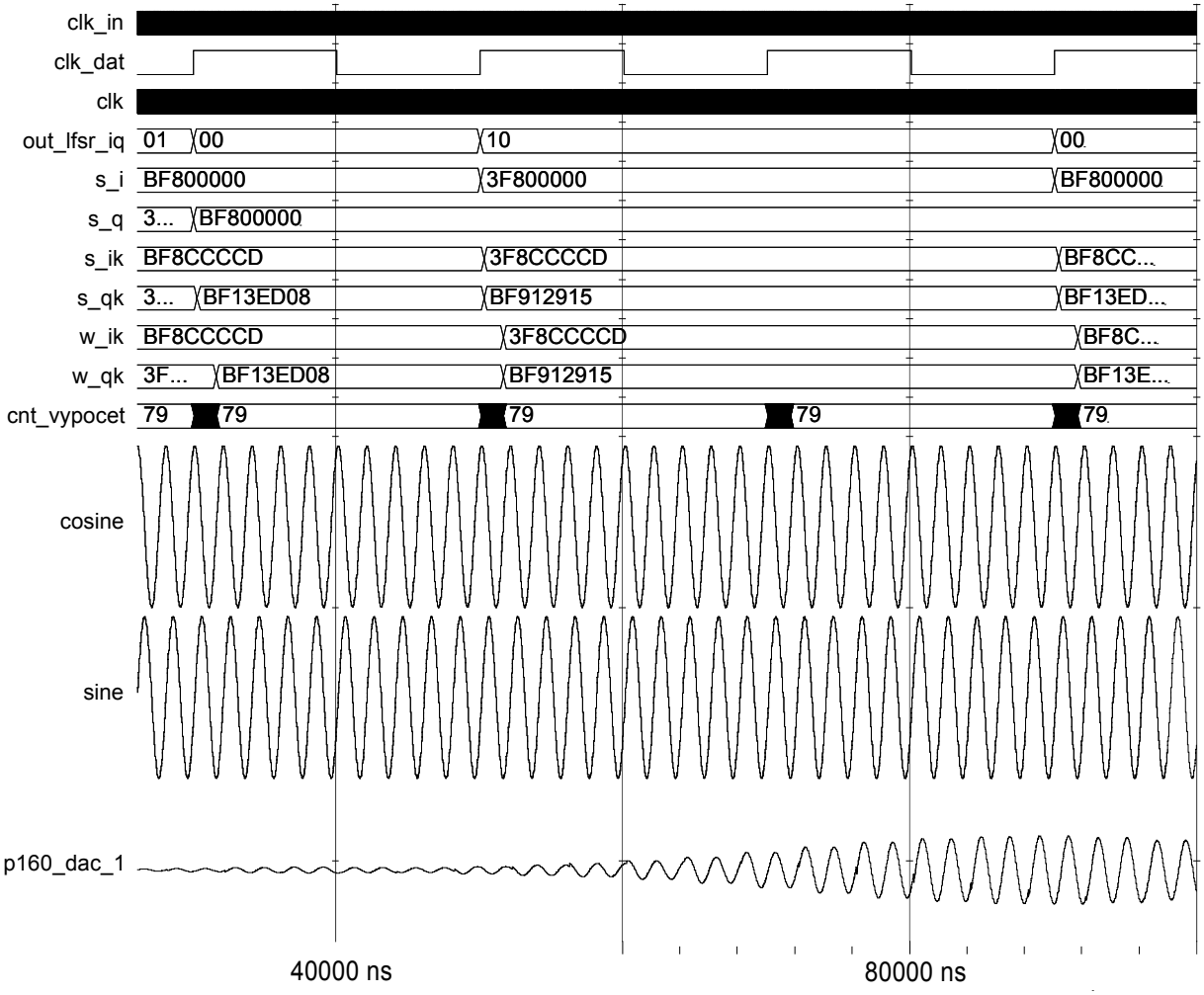
### 4. 2. 1. Výsledky metody dle Helda

Vstupní parametry, které bylo možné u této metody měnit, byly nastaveny na stejné hodnoty jako při simulacích v MATLABu, a to především kvůli srovnání jednotlivých výsledků. U této metody byla nastavována amplitudová nesymetrie ve větvi $I$ $K_I = 1,1$ a ve větvi $Q$ byla $K_Q = 0,9$ . Fázový posuv byl $\varphi = \frac{\pi}{10} rad$ . Po vykompenzování by tedy měl mít signál podobně vlastnosti, jak bylo uvedeno v kapitole 3. 3. 1. Ve výstupních časových průbězích je zobrazeno několik signálů. V prvé řadě je vidět hlavní hodinový signál CLK_IN s frekvencí 100 MHz, od kterého se odvíjí všechna činnost. V DCM bloku je tedy tento signál zpracován a vytvořen 50 MHz hodinový signál (CLK), kterým jsou řízeny všechny bloky v FPGA, a 100 MHz vzorkovací signál pro D/A převodník (CLK_A1 a CLK_A2). Všechny tyto průběhy jsou zobrazeny v Obr. 50. Průběhy CLK_A1 a CLK_A2 jsou mezi sebou vzájemně posunuty o 180°. Signál CLK_DAT má zde v celém průběhu logickou hodnotu 0. To je dáno tím, že frekvence tohoto signálu je „pouze“ 50 kHz a v tomto krátkém časovém úseku, který je vidět v grafu, není zobrazena žádná změna.



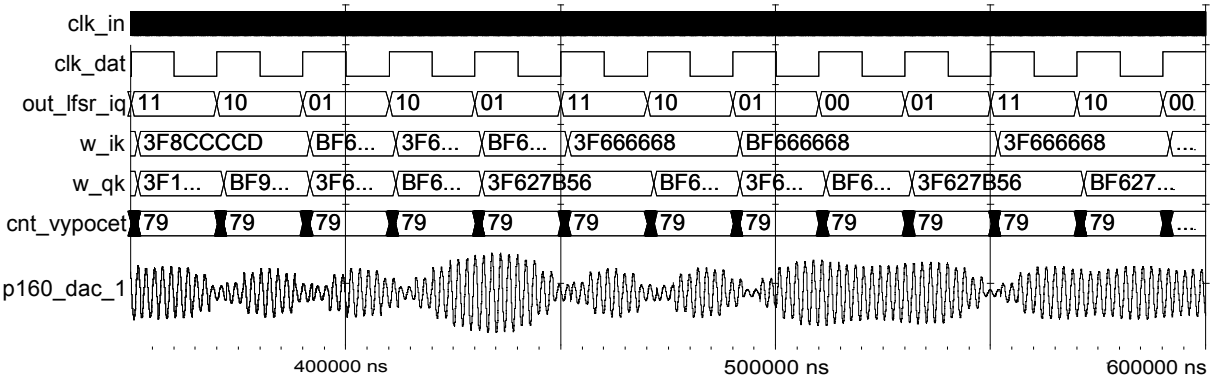

**Obr. 50 Časové průběhy hodinových signálů získaných simulací**

Na dalším Obr. 51 jsou opět zobrazeny hodinové signály a pro zobrazení datových výstupů byly vybrány signály z *I* a *Q* větve modulátoru. Všechny jsou vyjádřeny ve tvaru floating-point, který je dlouhý 32 bitů. Pro kratší zápis bylo zvoleno zobrazení v hexadecimálním formátu. Signály *s_i* a *s_q* nabývají ideálních hodnot 1 a -1. Výstupy z LFSR registrů a tedy i generované náhodné datové bity jsou zobrazeny signálem *out_lfsr_iq*. Dále pak *s_ik* a *s_qk* reprezentují vyjádření s nesymetriemi a *w_ik* a *w_qk* pak představují vykompenzovaný signál. Proměnná *cnt_vypocet* ukazuje, v kterém stavu se nachází celý algoritmus výpočtu. Lze vidět, že pro každé nové datové bity proběhne jednou celý výpočet s délkou 79 hodinových cyklů. Tato hodnota je zobrazena i v grafu jako konečný stav, kdy se ukončil výpočet algoritmu pro nová data. V průbězích je zachycen počáteční stav celého obvodu, což jde vidět na průběhu *p160_dac_1*, kde dochází k „nábíhání“ FIR filtru Square Root Raised Cosine. Výstupní signál je tak zpožďován. Tento výstupní signál je následně přiveden na převodník. V simulátoru ModelSim lze přepnout druh zobrazení hodnot jednotlivých signálů. Proto právě úplný výstup *p160_dac_1* je reprezentován analogovým tvarem signálu. Je to stejný tvar jako ten, který by se pak následně měl vyskytovat za převodníkem D/A. Podobně lze také vidět i analogové reprezentace průběhů *sine* a *cosine*, které jsou generovány blokem DSS.



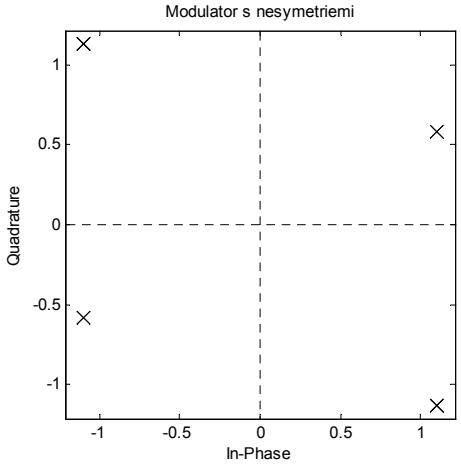

**Obr. 51 Časové průběhy datových signálů získaných simulací**

Situace, kdy je signál modulátoru již vykompenzován je zobrazena na Obr. 52. Jedná se pouze o zachycení časového průběhu po odvysílání alespoň 20 datových bitů, pomocí kterých se kompenzace vypočítává. Z výsledného datového průběhu je patrné, že signál je modulován.

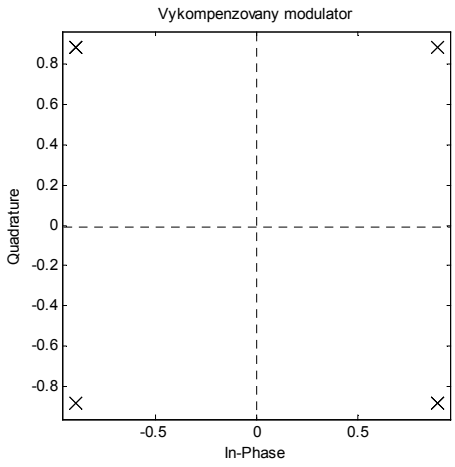



**Obr. 52 Časový průběh datových signálů získaných simulací po kompenzaci**

Data ve tvaru floating-point mohla být vyexportována do souboru a následně pak zobrazena do konstelačního diagramu. Jsou zobrazeny hodnoty signálů s nesymetriemi (*s_ik* a *s_qk*) a signály po vykompenzování, kterým odpovídají *w_ik* a *w_qk*. Což je zobrazeno na následujících obrázcích konstelačních diagramů. Nejprve lze vidět signál s použitými nesymetriemi a poté výstupní hodnoty kompenzačního algoritmu. Při srovnání těchto výsledků se simulací, která byla provedena samostatně v programu MATLAB (což je zachyceno na Obr. 35 a Obr. 36) si tyto výsledky odpovídají a i numerické hodnoty vycházejí stejně. Z toho se dá usuzovat, že naprogramovaná metoda ve VHDL dosahuje totožných výsledků.




**Obr. 53 Konstelační diagram výstupu modulátoru QPSK s nesymetriemi ($\varphi = \pi/10\ rad$ ; $K_I = 1{,}1$; $K_Q = 0{,}9$)**



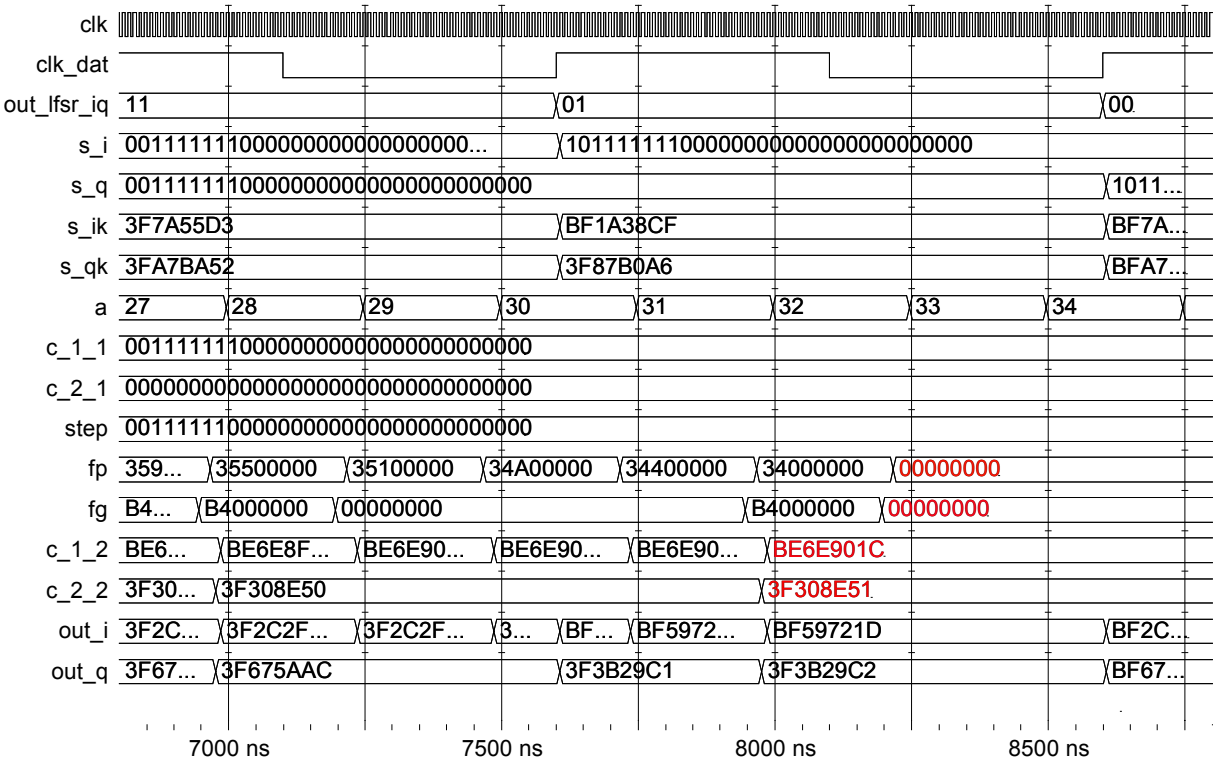

**Obr. 54 Konstelační diagram výstupu modulátoru po kompenzaci**

Po odzkoušení všech parametrů, které byly testovány i při simulacích v MATLABu bylo dále přistoupeno k samotné implementaci na FPGA s cílem získat podobné reálné výsledky, které by korespondovaly oběma simulacím. Tyto výsledky naměřené vektorovým signálovým analyzátorem (VSA) jsou uvedeny v kapitole 4. 5.

### 4. 2. 2. Výsledky metody dle Caverse

Tato metoda byla pouze implementována do VHDL kódu, ale nebylo pak již přistupováno k přímé realizaci na FPGA. Pro srovnání dosažených výstupních hodnot byly za kompetentní považovány výsledky ze simulace v ModelSim. Porovnávány byly výstupní konstelační diagramy kompenzační metody pro stejná nastavení nesymetrií, které byly uvedeny v kapitole 3. 1. 2. Simulace byla tedy nastavena s krokem přiblížení algoritmu 0,5. Nesymetrie zesílení pro jednu větev modulátoru $\alpha = 0,8$ a pro druhou $\beta = 1,2$. Fázový posuv byl stejně jako v předchozích případech roven $\varphi = \pi/10\ rad$ . Také byly srovnány výsledné koeficienty kompenzační matice, ke kterým konvergoval algoritmus vytvořený ve VHDL.

Nejprve jsou na Obr. 55 zobrazeny časové průběhy hodinových a datových signálů, které se vyskytovaly v programu. Hlavní hodinový signál s frekvencí 100 MHz, kterým byly řízeny všechny bloky je označen CLK. Hodinový signál, se kterým byla generována nová data byl CLK_DAT s frekvencí 1 MHz. Datový signál *out_lfsr_iq* ukazuje generované logické hodnoty pro *I* a *Q* složku modulátoru. Signály *s_i* a *s_q* pak reprezentují adekvátní hodnoty 1 a -1 ve tvaru floating – point. Logické hodnotě 1 odpovídá i hodnota signálů 1, logické úrovni 0 odpovídá hodnota signálů -1. Další signály, které jsou označeny *s_ik* a *s_qk*, představují již složky s nesymetriemi. Symbolem *a* je označena proměnná, která čítá jednotlivé iterační kroky metody. Proměnné *c_1_1*, *c_2_1*, *c_1_2* a *c_2_2* představují jednotlivé koeficienty kompenzační matice. Koeficienty *c_1_2* a *c_2_2* jsou měněny v jednotlivých iteračních krocích algoritmu. Proměnná *step* ukazuje velikost nastaveného kroku metody (0,5). Proměnné *fp* a *fg* pak představují velikost, o kterou se změní jednotlivé koeficienty *c_1_2* a *c_2_2* tak, jak bylo popsáno u této metody v kapitole 2. 1. 2 ve vztahu (20). Výstupní proměnné *out_i* a *out_q* odpovídají vykompenzovanému výstupu modulátoru.



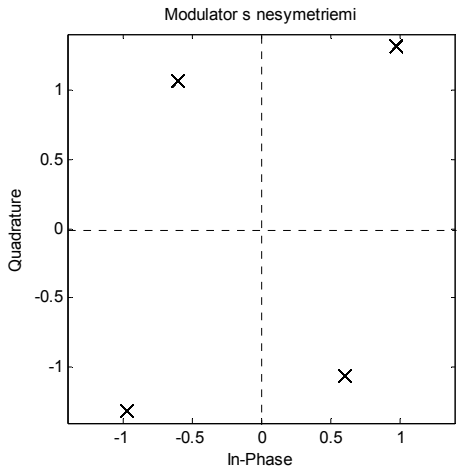

**Obr. 55 Časový průběh ze simulace metody dle Caverse**

Z časových průběhů je dále patrné, kdy metoda nalezne konečné hodnoty maticových koeficientů *c_1_2* a *c_2_2*. Tyto okamžiky jsou označeny červeně. Hodnoty odchylek *fp* a *fg* jsou v těchto okamžicích nulové, proto se podle vztahu (20) již nijak nemění maticové koeficienty. Z grafů na Obr. 15 a Obr. 16 je patrné, že pro nalezení těchto koeficientů bylo při podobně nastavené simulaci v MATLABu potřeba cca 30 až 40 iteračních kroků, což odpovídá i tomuto případu. Proměnná *a*, ukazující počet iteračních kroků, měla v tomto případě hodnotu 32. Při srovnání hodnot kompenzačních koeficientů, získáných simulací v MATLABu, s hodnotami, které vyšly při simulaci ve VHDL, lze říci, že jejich rozdíly jsou malé.

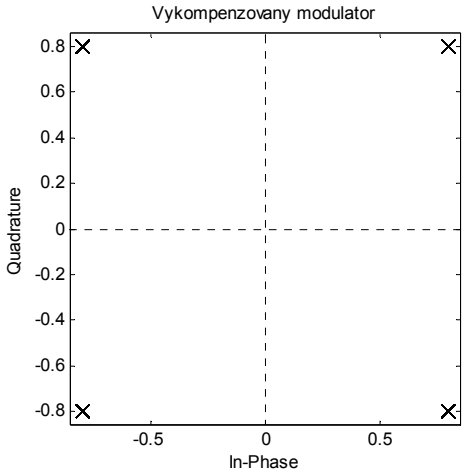
**Tab. 2 Srovnání kompenzačních koeficientů**

| | MATLAB | VHDL |
|---|---|---|
| c_1_1 | -0,3249 | -0,2997 |
| c_2_2 | 0,7009 | 0,6896 |

Konkrétní hodnoty jsou uvedeny v Tab. 2. Hodnoty v obou případech se liší o dvě setiny. Tyto rozdíly pak dokazují i mírnou odchylku v získaných konstelačních diagramech. Na Obr. 56 se nachází konstelační diagram ze simulace ve VHDL, který odpovídá signálům *s_ik* a *s_qk*. Další Obr. 57 představuje konstelační diagram vykompenzovaného modulátoru a odpovídá signálům *out_i* a *out_q*. Při srovnání s výsledky na Obr. 12 a Obr. 14 jsou pozorovatelné jen malé rozdíly v diagramu výstupu modulátoru. Výstupní signál je fázově vykompenzován správně. Vykompenzování amplitudy by mělo být větší, aby bylo dosaženo ideálních hodnot $\pm 1 \pm j \cdot 1$.



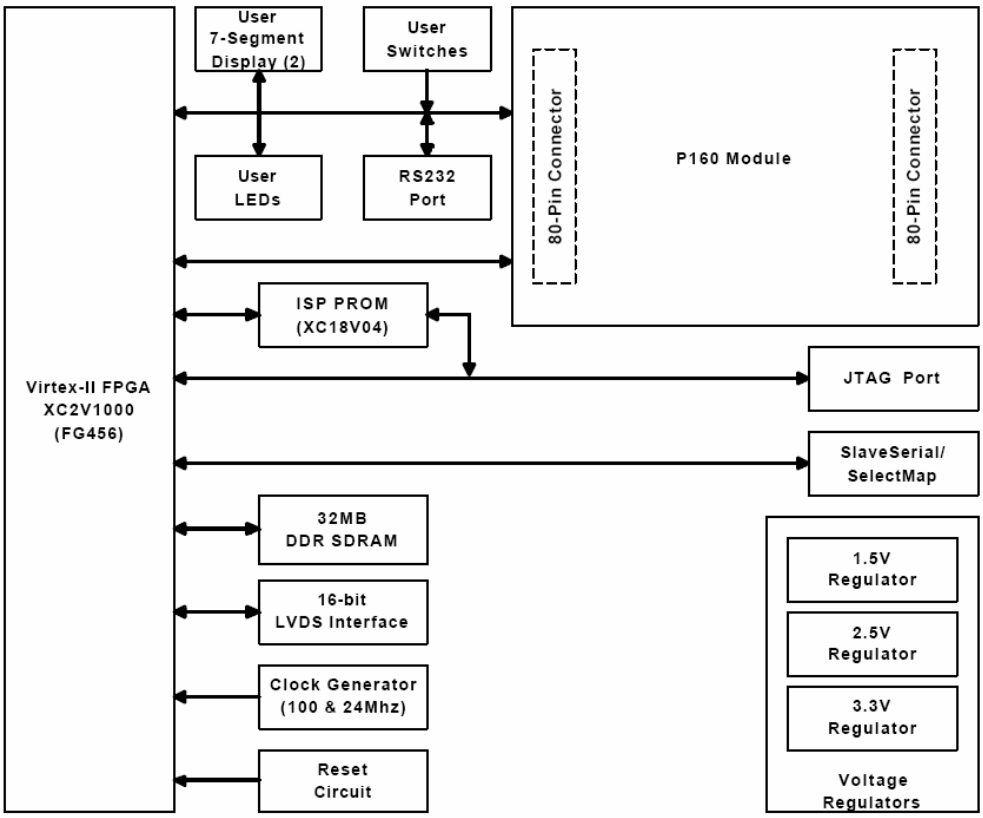

**Obr. 56 Konstelační diagram kvadraturního modulátoru s nesymetriemi $\alpha = 0{,}8$; $\beta = 1{,}2$ a $\varphi = \pi/10\ rad$**




**Obr. 57 Konstelační diagram výstupu modulátoru po kompenzaci**

## 4. 3. Základní popis desky Virtex 2 V2MB1000

Jedná se o vývojový kit, který je osazen FPGA od firmy Xilinx a patří do rodiny Virtex 2. Přesné označení programovatelného pole je XC2V1000-4FG456C, což značí i druh patice s 456 vývody.




**Obr. 58 Blokové schéma desky Virtex 2 V2MB1000, převzato z [9]**

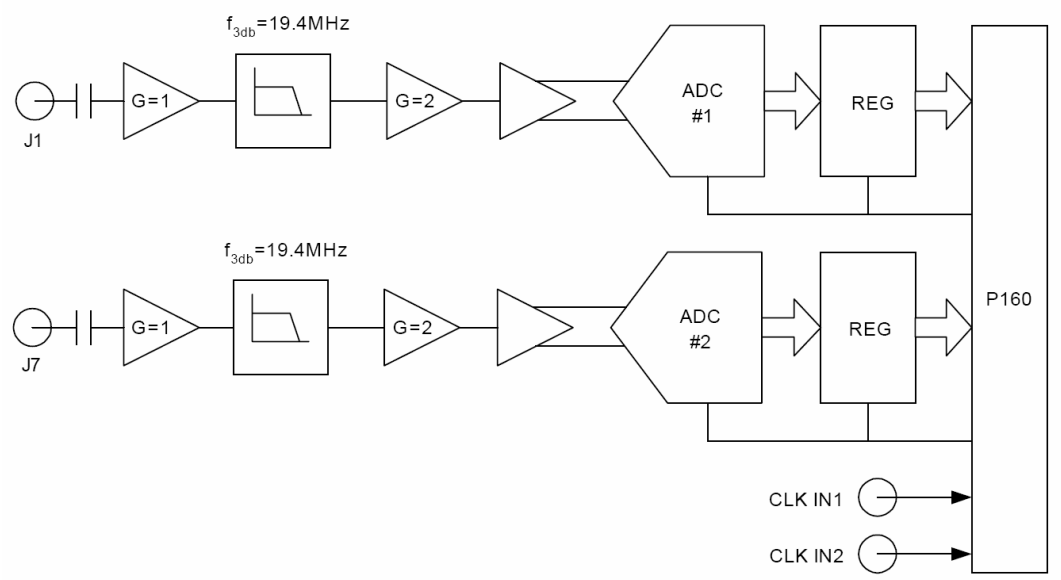
Jedná se o platformu určenou i pro návrh pomocí tzv. IP jader a různých volitelných modulů. Deska je tak vhodná i pro DSP aplikace. Konkrétní popis všech prvků obsazených v programovatelném poli XC2V1000 je uveden v [11]. Bylo nutné při implementaci počítat s jistým omezením v počtu volných logických jednotek v FPGA, což následně vedlo na „úspornější“ změnu zápisu VHDL kódu. Kromě klasických slice bloků se v FPGA nachází i např. 40 bloků 18-bitových násobiček a 8 DCM bloků.

Pro základní uživatelské programování se na desce nachází pomocné a testovací části v podobě uživatelských LED, sedmi segmentového displeje nebo tlačítek a DIP přepínačů. Blokové schéma se nachází na Obr. 58. Deska je napájena ze tří napěťových stabilizátorů na 1,5; 2,5 a 3,3 V. Hodinový signál pro celou desku je vytvářen pomocí 100 MHz nebo 24 MHz oscilátoru, řízeného dvěma krystaly. Další možností je přivést hodinový signál z nějakého externího zdroje. V našem případě byl zvolen hodinový kmitočet 100 MHz, který byl pak následně upravován DCM blokem. Na desce je také osazena 16 bitová 32 MB DDR paměť. Resetovací obvod desky umožňuje generování resetovacího pulzu pomocí tlačítka. Pro komunikaci s okolím je na desce vytvořeno několik portů. Jednak sériový port RS232 a dále vysílací a přijímací 16 bitový LVDS port. Pro naprogramování paměti ISP PROM nebo přímou konfiguraci FPGA slouží JTAG konektor, který je pak propojen přes JTAG kabel do USB portu počítače. Dva 80-pinové konektory na desce slouží k připojení analogového modulu P160. Pomocí toho konektoru je tak přímo propojeno FPGA se vstupy A/D a D/A převodníků, které se nachází na modulu. Podrobnější popis bude uveden v následující kapitole.

## 4. 4. Základní popis analogového modulu Memec P160

Jak už bylo zmíněno dříve, jedná se o analogové rozhraní, které je zapojeno přes konektor P160 přímo s vývody FPGA. Rozhraní tak slouží k přivádění resp. odvádění vstupních resp. výstupních dat z programovatelného pole.

Jedná se o modul, kde jsou umístěny dva nezávislé 12 bitové A/D převodníky (53 Msps) se střídavou AC vazbou. Na vstupech převodníků je filtr typu dolní propust s mezním kmitočtem 19,4 MHz, sloužící k omezení vstupního signálu a dodržení vzorkovacího teorému. Vstupní obvody tak mohou sloužit k převodu analogového signálu na 12 bitové data určené pro FPGA. Popisované zapojení ukazuje Obr. 59.



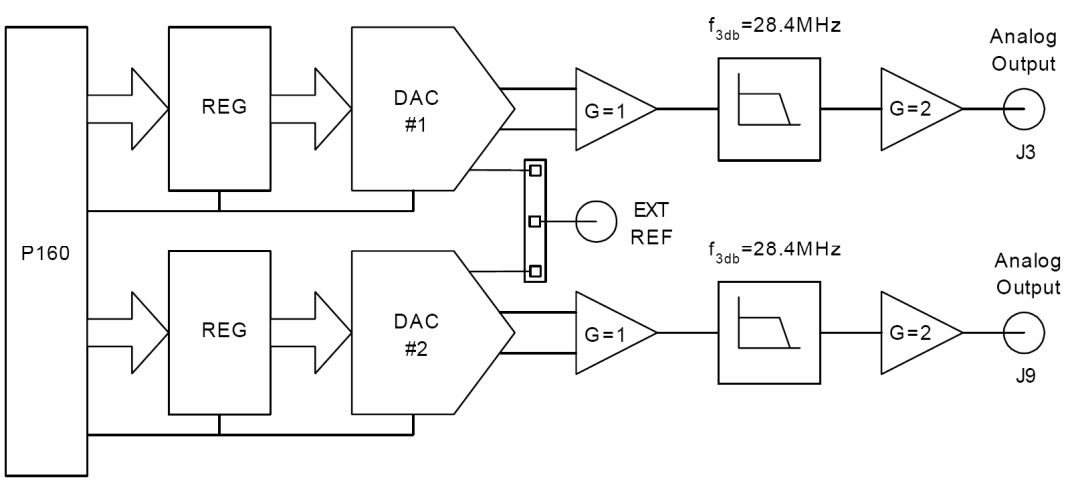

**Obr. 59 Blokové schéma zapojení A/D převodníků modulu Memec P160, převzato z [10]**

Výstupní signály z FPGA pak mohou být přivedeny na D/A převodníky, které z digitálních dat vytvoří opět analogový signál. Při implementaci bylo nutné MSB znaménkový bit negovat, aby byla dodržena správná znaménková forma signálu. Převodníky jsou na desce použity opět dva a jsou 12 bitové s rychlostí 165 Msps. Výstupní analogový signál má maximální rozkmit 2 $V_{p\text{-}p}$. Tento rozkmit je souměrný kolem hodnoty 2,5 V výstupního napětí. V Tab. 3 jsou pro názornost uvedeny odpovídající hodnoty výstupního napětí D/A převodníků na vstupní digitální slova. Výstupní signál má tedy neustále nenulovou střední hodnotu.

**Tab. 3 Datová slova D/A převodníků**

| Vstupní data 12b | Velikost výstupního napětí [V] |
|---|---|
| 1111 1111 1111 | 3,5 |
| 1000 0000 0000 | 2,5 |
| 0000 0000 0000 | 1,5 |

Datová slova vstupují do DAC přes 12 bitové registry, které jsou řízeny stejným hodinovým signálem. Ten je přiváděn přímo z FPGA přes konektor P160. Na výstupech D/A převodníků je ještě zařazen filtr v podobě dolní propusti s mezním kmitočtem 28,4 MHz. Popisované údaje jsou patrné i z blokového uspořádání na Obr. 60.




**Obr. 60 Blokové schéma zapojení D/A převodníků modulu Memec P160, převzato z [10]**

V následující tabulce je uvedeno konkrétní propojení jednotlivých vývodů FPGA s D/A převodníkem, které bylo použito při programování obvodu.

**Tab. 4 Rozhraní mezi FPGA a D/A převodníkem**

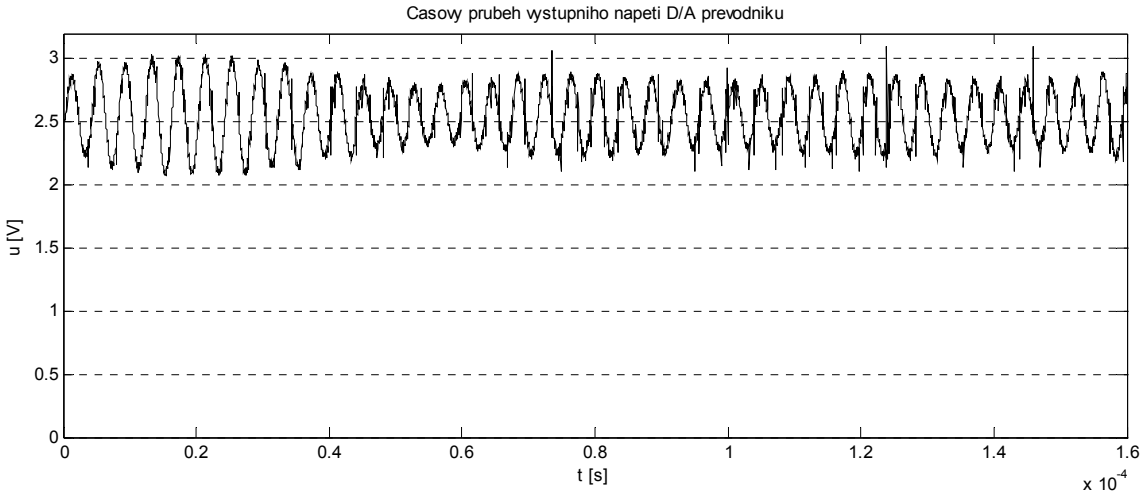
| Jméno signálu | Vývod FPGA | Popis |
|---|---|---|
| DB1 | V6 | 11. vstupní bit D/A (MSB) |
| DB2 | AB5 | 10. vstupní bit D/A |
| DB3 | V7 | 9. vstupní bit D/A |
| DB4 | AA6 | 8. vstupní bit D/A |
| DB5 | V8 | 7. vstupní bit D/A |
| DB6 | AB6 | 6. vstupní bit D/A |
| DB7 | V9 | 5. vstupní bit D/A |
| DB8 | AA7 | 4. vstupní bit D/A |
| DB9 | U9 | 3. vstupní bit D/A |
| DB10 | AB7 | 2. vstupní bit D/A |
| DB11 | U10 | 1. vstupní bit D/A |
| DB12 | AA8 | 0. vstupní bit D/A (LSB) |
| CLK | AA5 | D/A clock (náb. hrana) |
| CL2 | U13 | Registr clock |

Čísla těchto vývodů byla zapsána do *.ucf souboru, který bylo nutné vytvořit při samotné implementaci na FPGA. Byl tedy používán jeden D/A převodník, ze kterého vycházel výsledný výstupní signál z modulátoru.

## 4. 5. Naměřené výsledky

V této kapitole budou shrnuty výsledky, které byly naměřeny pomocí digitálního osciloskopu a signálového analyzátoru. Vždy byl pozorován signál na výstupu D/A převodníku analogového modulu Memec P160. Pro toto měření byl použit osciloskop Tektronix TDS 3054B. Do programovatelného logického pole byl nahrán vytvořený program. Ihned po naprogramování bylo pomocí blikající LED diody signalizováno, že program běží. Výstupní signál, připojený na osciloskop s vypnutou střídavou vazbou, je zobrazen na Obr.

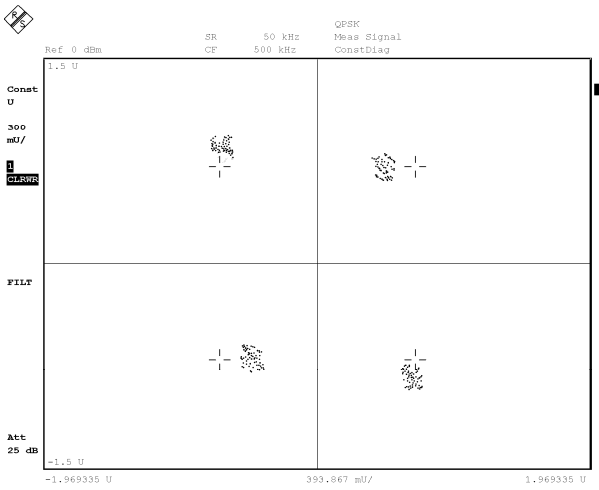
61. Lze vidět modulovaný signál, který se mění v hodnotách od cca 2 Vdo 3 V. Rozsah výstupního napětí, které dokáže D/A převodník vytvořit bylo od 1,5 V do 3,5 V, což potvrzuje Tab. 3. Mírné zvlnění amplitudy signálu může být způsobeno použitým FIR filtrem.



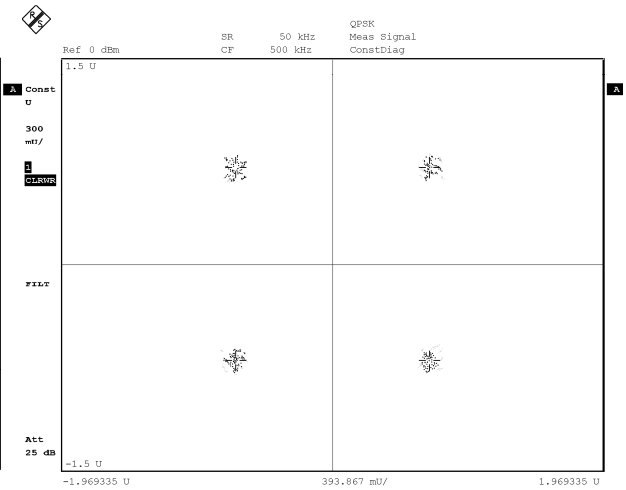

**Obr. 61 Časový průběh výstupního signálu zobrazeného osciloskopem**

Průběh je srovnatelný se simulací v programu ModelSim, která je znázorněna na Obr. 52. Mírné problémy se objevovaly při správném zasynchronizování signálu tak, aby byl bez zákmitů. Tímto měřením bylo vždy ověřeno, zda je na výstupu generován správný signál a dále následovalo připojení výstupu D/A převodníku na vstup vektorového analyzátoru. Pro měření byly požívány signální analyzátor Rohde & Schwarz, typ FSQ3 a spektrální analyzátor Sony Tektronix 3086. Mezi hlavní nastavení patřilo druh zvolené modulace, což byla QPSK. Nosná vlna byla na frekvenci 500 kHz. Šířka pásma neboli span byl nastaven na hodnotu 200 kHz. Mezi poslední důležité parametry nastavení patřila symbolová rychlost. Její hodnota byla 50 kHz.

Pro odstranění mezisymbolových interferencí byl v modulátoru použit FIR filtr s impulsní odezvou tytu Root Square Raised Cosine. Poté byl zkoušen i filtr s impulsní odezvou Raised Cosine. Na obou vektorových analyzátorech bylo možné nastavit na přijímací straně demodulátoru různé typy přijímacích filtrů a tím ovlivnit výsledné výstupní charakteristiky. Pro srovnání výsledků byl nejprve upraven algoritmus programu tak, aby se na výstupu převodníku vyskytoval signál modulátoru s nesymetriemi. Velikost těchto nesymetrií byla nastavena stejně jako v předchozích dvou simulacích. Amplitudová nesymetrie ve větvi *I* byla $K_I = 1,1$ a ve větvi *Q* byla $K_Q = 0,9$. Fázová nesymetrie nabývala hodnoty $\varphi = \pi/10\ rad$. Tyto nastavené hodnoty potvrzuje konstelační diagram na Obr. 62, který ukazuje data zachycené signálovým analyzátorem. Jak jde vidět, tvar diagramu je velmi podobný diagramům, které vznikly při simulacích a jsou uvedeny na Obr. 53 a Obr. 35.

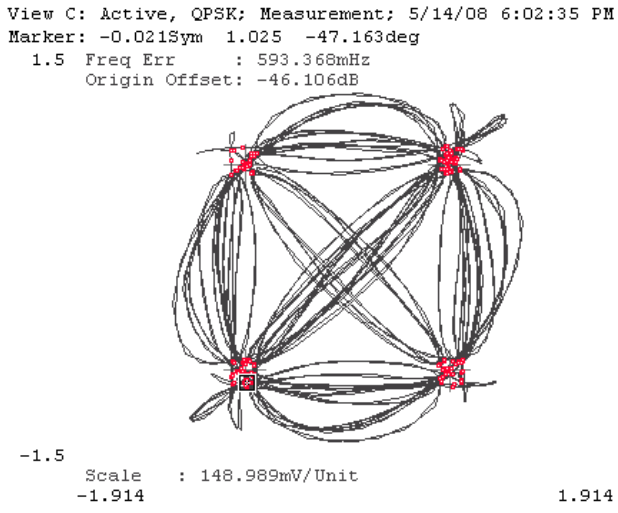



**Obr. 62 Naměřený konstelační diagram modulátoru s nesymetriemi $K_I = 1{,}1$; $K_Q = 0{,}9$ a $\varphi = \pi/10\ rad$**




**Obr. 63 Naměřený konstelační diagram vykompenzovaného modulátoru**

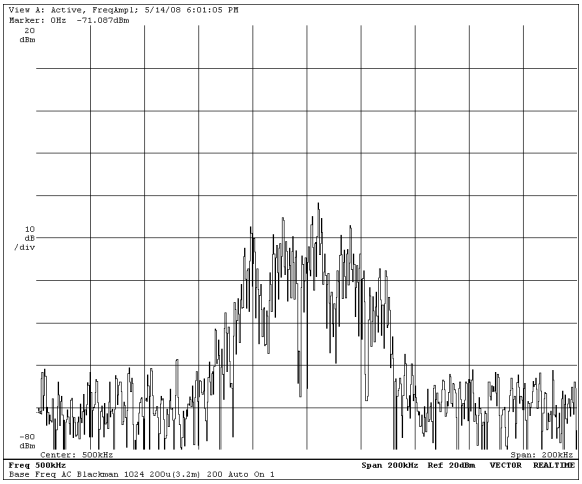
V naměřeném diagramu se pro každý stav modulace vyskytuje více rozptýlených bodů, což je částečně způsobeno délkou impulsní odezvy použitého filtru. V tomto případě bylo použito pro definování impulsní charakteristiky 31 bodů. Při použití více bodů na popsání charakteristiky by měl být i rozptyl menší. Při této realizaci na FPGA Virtex 2 to ovšem již nebylo možné, neboť vytvoření složitějšího FIR filtru kladlo vyšší nároky na množství použitých strukturálních prvků hradlového pole (slice).



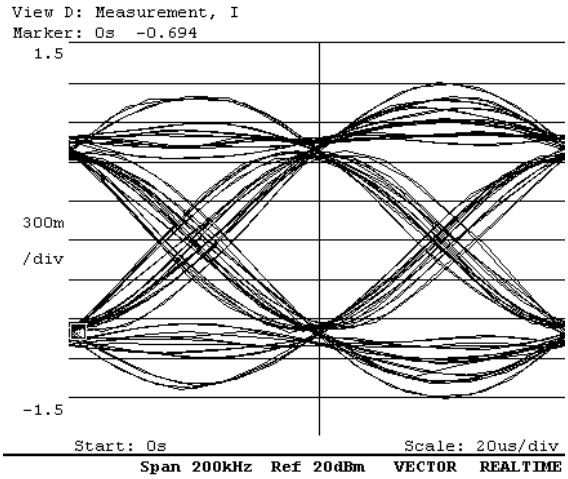

**Obr. 64 Konstelační diagram vykompenzovaného modulátoru**

Vzhledem k tomu, že byly použity dva filtry, kdy v každé větvi modulátoru byl jeden, musela být ponechána impulsní odezva s 31 vzorky. Správná funkce celého kompenzačního algoritmu byla následně dokázána změřeným konstelačním diagramem na Obr. 63. Opět je patrný rozptyl bodů v jednotlivých stavech modulace. Ideální poloha bodů je značena křížky. Získaný diagram koresponduje s výsledky předchozích simulací, které jsou zachyceny na Obr. 54 a Obr. 36. IQ diagram, ve kterém jsou zakresleny i jednotlivé přechody mezi stavy modulace je uveden na Obr. 64. Z diagramu je také patrný vliv použitého filtru, který má za následek tvar jednotlivých cest mezi stavy modulátoru.

Dále bylo pomocí spektrálního analyzátoru získáno frekvenční spektrum výstupního signálu QPSK modulátoru. Jeho střední kmitočet byl nastaven na frekvenci 500 kHz s šířkou pásma (span) 200 kHz, což potvrzuje i Obr. 65. Tvar spektra odpovídal předpokladu. Úroveň nosné dosahovala hodnoty kolem -30dBm. Šířka užitečného pásma byla kolem 50 kHz.




**Obr. 65 Frekvenční spektrum výstupního signálu modulátoru**




**Obr. 66 Diagram oka pro I větev modulátoru**

Jako poslední získaný výsledek měření je na Obr. 66 uveden diagram oka pro *I* větev modulátoru. Z diagramu je patrné, že dochází k mezisymbolovým interferencím, protože cesty mezi jednotlivými přechody se neprotínají v jediných bodech. Tyto přeslechy by patrně byly menší při použití Square Root Raised Cosine filtru s delší impulsní odezvou. Získáním těchto výstupních hodnot bylo měření ukončeno.

# Závěr

Řešení zadaného problému probíhalo v několika fázích. Nejprve bylo třeba zpracovat informace o modelování kvadraturních modulátorů a demodulátorů a získat také představu o jednotlivých nežádoucích vlastnostech skutečného zapojení. Tyto vlastnosti jsou popisovány tzv. nesymetriemi. Úkolem tedy bylo, najít metodu nebo metody, které dokáží eliminovat tyto nežádoucí vlivy. Následovala pak implementace jednotlivých metod v MATLABu a provedení jejich simulací. V dalších krocích bylo nutné se seznámit s programovacím jazykem VHDL a prostředím Xilinx ISE. Pomocí tohoto nástroje byla prováděna implementace metod na programovatelné hradlové pole. Pro kontrolu sloužily pomocné simulace programem ModelSim. V poslední fázi samotné realizace bylo nutné naučit se ovládat vývojový kit V2MB1000, který byl osazen programovatelným logickým polem Virtex 2, s analogovým modulem Memec P160. Měřením pomocí osciloskopu a vektorového signálového analyzátoru měly být potvrzeny výsledky získané simulací.

Metody pro implementaci v MATLABu byly vybrány tři. První byla uvedena v článku Caverse [1], druhá poté byla metoda dle Zhu [4] a třetí byla metoda dle Helda [2]. První dvě metody byly vhodné zejména pro modulace s konstantní velikostí amplitudy a byly simulovány s modulacemi QPSK a 8PSK. Výsledky ukazují funkčnost obou metod, nicméně se projevovaly některé jejich nedostatky, jako např. možnost uvíznutí v lokálním minimu. Třetí metoda dle Helda byla na první pohled jednodušší při implementaci. Simulace byly také prováděny pro modulaci QPSK i 8PSK. Každá z metod byla podrobně teoreticky popsána a byly nastíněny jednotlivé důležité pasáže při implementaci. Výhodou použitého prostředí MATLAB může být i to, že většina operací je prováděna maticově.

Pro implementaci v prostředí Xilinx v jazyce VHDL byly vybrány dvě metody. První zvolenou byla metoda dle Helda a druhá byla dle Caverse. Výsledky ze simulačního programu ModelSim byly porovnávány s výsledky simulací v MATLABu. Pro samotnou realizaci na desku V2MB1000 byla vybrána jen metoda dle Helda a po nastavení všech potřebných částí desky bylo přistoupeno k měření pomocí osciloskopu a vektorového analyzátoru. Získané praktické výsledky tak mohly být srovnány s teoretickými z předchozích simulací.

Výsledky simulací metody prof. Caverse ukazují, že metoda se potýká s problémy při kompenzaci fázové nesymetrie. Bylo nutné provést úpravu, která provádí pootočení jednotlivých bodů konstelačního diagramu o úhel, který je roven $\phi/2$. Tato metoda konverguje ke správnému vykompenzování v případě, kdy se jedná o nesymetrické rozložení jednotlivých nesymetrií, což je popsáno v kapitole 3. 1. 2. Významný parametr, který ovlivňuje rychlost konvergence metody, je krok $\delta$. Mezi další výsledky simulace patří i závislosti střední kvadratické odchylky na počtu iteračních kroků (Obr. 15). Z tohoto grafu také vyplývá, jak je vhodné volit velikost kroku. Vzhledem k rychlosti konvergence metody se jeví jako vhodná volba $\delta$ v rozmezí 1 až 0,5. Toto odpovídá i uvedeným výsledkům v článku Caverse [1]. V kapitole 3. 1. 3 jsou dále zobrazeny výsledky kompenzace pro modulaci 8PSK. V tomto případě je dosahováno podobných výsledků a z tvaru jednotlivých

konstelačních diagramů je patrný i průběh celé kompenzace. Je také nutné poznamenat, že u této metody je poměrně složitá část při kompenzaci offsetu modulátoru. Následná implementace ve VHDL potvrdila předchozí poznatky. Krok metody $\delta$ byl nastaven na hodnotu 0,5. I ze simulací v ModelSim vyplynulo, že metoda při tomto nastavení provedla 32 iteračních kroků k nalezení správných kompenzačních koeficientů, což odpovídá i výsledkům simulací v MATLABu. Bližší popis je uveden v kapitole 4. 2. 2.

Druhá metoda dle Zhu [4] byla zkoušena s podobnými nastavenými parametry. I u této metody se projevilo, že při kompenzování fázové nesymetrie vznikají problémy s neúplným dokompenzováním. Vůči předchozí metodě se zde projevovala poměrně značná závislost na přesnosti odhadu prvního bodu iteračních kroků. Také se stávalo, že metoda při hledání minimální odchylky nalezla jiné minimum. Což ovšem nebyly správné hodnoty pro vykompenzování. Nejlépe metoda kompenzuje v případě nulového fázového posunutí. V tomto případě je kompenzace téměř ideální, což lze vidět i na výsledcích při použité modulaci 8PSK. Pro tuto modulaci dosahuje metoda lepších výsledků, než s modulací QPSK. Z jednotlivých různých nastavení simulací byla zjištěna ideální velikost iteračního kroku na hodnotu $\mu = 0{,}01$. Počet nutných iteračních kroků pro konvergenci algoritmu je větší, než u první metody dle Caverse. Na druhou stranu se celkově metoda jeví jednodušší pro implementaci a realizaci, než metoda dle Caverse.

Třetí metoda dle Helda byla založena na nalezení kompenzačních koeficientů z předchozích odvysílaných datových symbolů. K výpočtu bylo využíváno vždy alespoň 20 odeslaných datových symbolů. Metoda nebyla navržena pro kompenzování stejnosměrného offsetu, což ostatní metody byly. V případech, kdy se amplitudová nesymetrie vyskytuje jen v jedné větvi, dochází k ideálnímu vykompenzování. V ostatních případech byla kompenzace fáze i amplitudy velmi dobrá. Výsledky byly shrnuty v kapitole 3. 3. 1. Metoda byla vhodná i pro realizaci automatické adaptivní kompenzace. Při testování metody bylo zjištěno, že pro kompenzování amplitudové nesymetrie bylo vhodnější použít mírně modifikovaný vztah (36). Implementace algoritmu v jazyce VHDL je blokově zachycena na Obr. 47, kde bylo využíváno postupného sdílení jednotlivých hardwarových prostředků, aby byly ušetřeny volné logické jednotky hradlového pole. Celý postup s nastavením všech použitých bloků FPGA je popsán kapitolou 4. 1. 1. Data získané simulací programem ModelSim přesně odpovídala výstupním hodnotám v MATLABu. Tyto údaje byly dále potvrzeny při realizaci s hradlovým polem Virtex 2. Pro zjednodušení nebyl kompenzován reálný analogový IQ modulátor, ale jeho model implementovaný v obvodu FPGA s nosnou frekvencí 500 kHz a symbolovou rychlostí 50 kHz. Pomocí signálového analyzátoru byly naměřeny konstelační diagramy, které tvarově odpovídaly simulacím. Potvrdily schopnost metody ke kompenzování amplitudových a fázových nesymetrií QPSK modulátoru. Docházelo však k rozptýlení jednotlivých bodů konstelačního diagramu. Důvodem tohoto rozptýlení mohl být malý počet koeficientů Square Root Raised Cosine filtru. Omezení však bylo ze strany náročnosti celého algoritmu, kdy bylo využito 98% možných strukturálních prvků hradlového pole - slice bloků.

Každá ze zkoumaných metod má své přednosti, ale i nedostatky. Při srovnání výsledků, které byly zveřejněny v jednotlivých článcích, nebylo uvedeno konkrétní nastavení, simulací. Což zamezilo snadnému porovnávání a vyhodnocování algoritmů.

# Literatura


[1] CAVERS, James K.; LIAO, Maria W. Adaptive Compensation for Imbalance and Offset Losses in Direct Conversion Transceivers. *IEEE Transactions on Vehicular Technology, Vol. 42, November 1993,* p. 581 – 588.

[2] HELD, Ingolf; KLEIN, Oliver; CHEN, Albert; MA, Vincent. Low Complexity Digital IQ Imbalance Correction in OFDM WLAN Receivers. *IEEE Integrated System Solution Corporation. Hsinchu, Taiwan, 2004,* p. 1172 – 1176.

[3] ŠEBESTA, Vladimír. *Teorie sdělování*. Brno: VUT, 1998. Fakulta elektrotechniky a komunikačních technologií. ISNB 80-214-1247-X.

[4] ZHU, Zhiwen; HUANG, Xinping. Adaptive Compensation of Gain/Phase Imbalances and DC-Offsets Using Constant Modulus Algorithm. *Communications Research Centre, Ottawa, Ontario, K2H 8S2, Canada, IEEE, 2004.*

[5] CAVERS, James K. New Methods for Adaptation of Quadrature Modulators and Demodulators in Amplifier Linearization Circuits. *IEEE Transactions on Vehicular Technology, Vol. 46, NO. 3, August 1997,* p. 707 – 716.

[6] CAVERS, James K. The Effect of Quadrature Modulator and Demodulator Errors on Adaptive Digital Predistorters for Amplifier Linearization. *IEEE Transactions on Vehicular Technology, Vol. 46, NO. 2, May 1997,* p. 456 – 466.

[7] PERRY, Douglas L. *VHDL. Programming by Example*. 4th ed. New York: McGraw – Hill, 2002. 476 s. ISBN 0-07-140070-2

[8] KOLOUCH, Jaromír. *Programovatelné logické obvody a modelování číslicových systémů v jazycích ABEL a VHDL*. Brno: VUT, 2000. 83 s. Fakulta elektrotechniky a komunikačních technologií. ISBN 80-214-1733-1.

[9] MEMEC DESIGN, *Virtex 2 VM2MB1000 Development Board User's Guide*. [online]. Listopad 2002. 44 s. Dostupné na WWW:
http://www.cs.lth.se/EDA385/doc/restricted/V2MB_User_Guide_3_0.pdf

[10] MEMEC DESIGN, *Memec P160 Analog Module User Guide*. [online]. Červenec 2003. 21 s. Dostupné na WWW:
http://www.ux.uis.no/~karlsk/MIK200/dok/P160Analog_UserGuide_1_2.pdf

[11] XILINX, *Virtex 2 1.5V Field-Programmable Gate Arrays*. [online]. Říjen 2001. 8 s. Dostupné na WWW:
http://www.datasheetcatalog.org/datasheets/134/287682_DS.pdf