

**RESTRICTED**

FP7-ICT Future Networks  
SPECIFIC TARGETTED RESEARCH PROJECT  
Project Deliverable

<b>PHYDYAS Doc. Number</b>	PHYDYAS_021
<b>Project Number</b>	ICT - 211887
<b>Project Acronym+Title</b>	PHYDYAS – PHYsical layer for DYnamic AccesS and cognitive radio
<b>Deliverable Nature</b>	Report
<b>Deliverable Number</b>	D9.3
<b>Contractual Delivery Date</b>	January 1 <sup>st</sup> , 2010
<b>Actual Delivery Date</b>	February 8, 2010
<b>Title of Deliverable</b>	WiMAX simulation results – Lab setup and measurements
<b>Contributing Workpackage</b>	WP9
<b>Project starting date; Duration</b>	01/01/2008; 30 months
<b>Dissemination Level</b>	RE
<b>Author(s)</b>	Frank Schaich (ALUD); Jurgen Vandermot (AGI); Vidar Ringset, Helge Rustad (SINTEF); Montse Najar, Francisco Rubio, Marius Caus (CTTC); Zhao Zhipeng (COMSIS);

**Abstract:**

The simulator that is developed in work package 9, based on the existing WiMAX standard (Worldwide interoperability for Microwave Access) and enhanced with algorithms developed by other work packages to include the FBMC principle has already extensively been used. In the first section of this deliverable, new results of those simulation tests are analyzed and discussed.

Section 2 is dedicated to the demonstrator setup. According to the milestone at month 24 included in the DoW, some measurement results should be provided., but because we're still integrating the demonstrator, this section is about the steps and status of the development and the tests that were performed during this process.

## Contents

<b>1</b>	<b>THE SIMULATOR .....</b>	<b>3</b>
<b>1.1</b>	<b>Additions to the simulator .....</b>	<b>3</b>
1.1.1	Preamble based synchronization (downlink) .....	3
1.1.2	RF Impairments.....	9
1.1.3	MIMO .....	14
1.1.4	OFDM-specific additions.....	14
1.1.5	Planned Additions .....	15
<b>1.2</b>	<b>Complexity assessment .....</b>	<b>16</b>
<b>1.3</b>	<b>Simulation results.....</b>	<b>19</b>
1.3.1	Spectral efficiency.....	19
1.3.2	Synchronization .....	34
1.3.2.1	Pilot based synchronization (downlink PUSC).....	34
1.3.2.2	Preamble based synchronization.....	57
1.3.3	Adaptive channel aware subcarrier spacing.....	65
<b>2</b>	<b>OFDM-FBMC DEMONSTRATOR SETUP .....</b>	<b>80</b>
<b>2.1</b>	<b>Transmitter.....</b>	<b>80</b>
2.1.1	Physical Architecture .....	81
2.1.2	Transmitter set-up .....	82
2.1.2.1	Deriving setup parameters for the hardware .....	82
2.1.2.2	Downlink PUSC.....	83
2.1.2.3	Uplink AMC .....	87
2.1.2.4	Transmitter Development Status.....	88
<b>2.2</b>	<b>Channel Emulator.....</b>	<b>90</b>
2.2.1	N5106A PXB Setup .....	90
2.2.2	N5102A DSIM.....	91
2.2.3	TX-DSIM Digital Interface.....	93
2.2.4	N5183A MXG .....	95
<b>2.3</b>	<b>Receiver.....</b>	<b>96</b>
2.3.1	Module Description.....	96
2.3.2	Data Description .....	97
2.3.3	Interface Description.....	99
<b>2.4</b>	<b>2x2 MIMO Decoder .....</b>	<b>104</b>
2.4.1	General Architecture .....	104
2.4.1.1	Norm Inversion Calculation.....	106
2.4.1.2	Y Projection an H Scalar Production .....	108
2.4.1.3	BLAST Decoder .....	111
<b>3</b>	<b>REFERENCES .....</b>	<b>115</b>

---

# 1 The Simulator

## 1.1 Additions to the simulator

### 1.1.1 Preamble based synchronization (downlink)

Beside the exploitation of the linear phase ramps (in time (CFO) and frequency (FTD), respectively) after the analysis filter bank and the FFT, respectively, with the help of pilots, another possibility to determine time and frequency offsets is the use of a preamble. The least squares approach (LS) exploits a preamble consisting of two identical parts. In the following a single multicarrier symbol is used. A detailed description of the LS algorithm is elaborated in [1].

The cost function to be minimized is:

$$(\hat{\varepsilon}, \hat{\tau}) = \arg \min_{(\tilde{\varepsilon}, \tilde{\tau})} \left\{ \sum_{m=m_0}^{m_0+M/2} \left| r[m + \tilde{\tau}/T_s] - r[m + M/2 + \tilde{\tau}/T_s] e^{j2\pi\tilde{\varepsilon}} \right|^2 \right\}$$

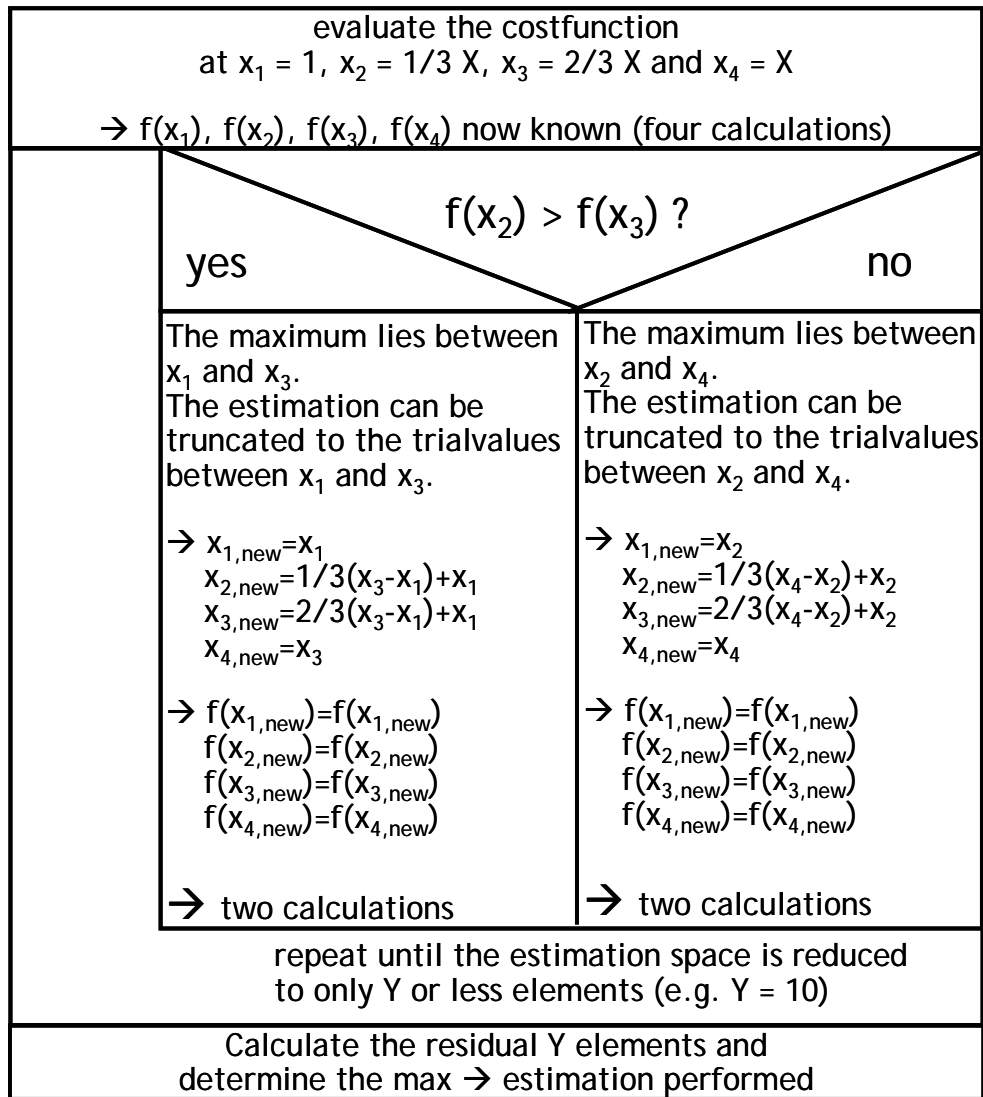
$\hat{\varepsilon}$  and  $\hat{\tau}$  are the estimates,  $\tilde{\varepsilon}$  and  $\tilde{\tau}$  the trial values for CFO and FTD, respectively.  $m_0$  defines the beginning of the preamble,  $M$  is the number of subcarriers.  $r[\dots]$  is the received signal, including the channel gain, the frequency and time offsets and noise. Finally  $T_s$  is the sampling interval.

After some math the two-dimensional minimization problem can be simplified to a one-dimensional maximization with respect to the timing offset plus an analytical expression for the frequency offset:

$$\begin{aligned} \hat{\tau} &= \arg \max_{\tilde{\tau}} \{2|R(\tilde{\tau})| - Q(\tilde{\tau})\} \\ \hat{\varepsilon}(\hat{\tau}) &= \frac{1}{2\pi} \angle \{R(\hat{\tau})\} \\ \text{with} \\ R(\tilde{\tau}) &= \sum_{m=(K-1)M}^{(K-1/2)M-1} r^* \left[ m + \frac{\tilde{\tau}}{T_s} \right] r \left[ m + \frac{M}{2} + \frac{\tilde{\tau}}{T_s} \right] \\ Q(\tilde{\tau}) &= \sum_{m=(K-1)M}^{(K-1/2)M-1} \left| r \left[ m + \frac{\tilde{\tau}}{T_s} \right] \right|^2 + \sum_{m=(K-1)M}^{(K-1/2)M-1} \left| r \left[ m + \frac{M}{2} + \frac{\tilde{\tau}}{T_s} \right] \right|^2 \end{aligned}$$

Obviously the complexity significantly could be reduced. However, compared to pilot based timing and frequency estimation, complexity is still an issue. The numerical maximization with respect to  $\tilde{\tau}$  calls for  $M$  metric calculations, if the sampling interval is used as step size.

Fortunately this number can be reduced further by applying the following algorithm (The algorithm is applicable for cost functions with a single maximum/minimum. This is the case here):



$x_1, x_2, x_3$  and  $x_4$  are equidistant arguments of the cost function,  $f(x_1), f(x_2), f(x_3)$  and  $f(x_4)$  the respective function value. Then, depending on  $f(x_2)$  and  $f(x_3)$ , the range of the cost function can be reduced by a fourth per iteration. Finally, once there are only few trial values left, these residual elements can be evaluated.

Two examples (low FTD (0.1), and high FTD (0.9)) are given to further clarify the concept:



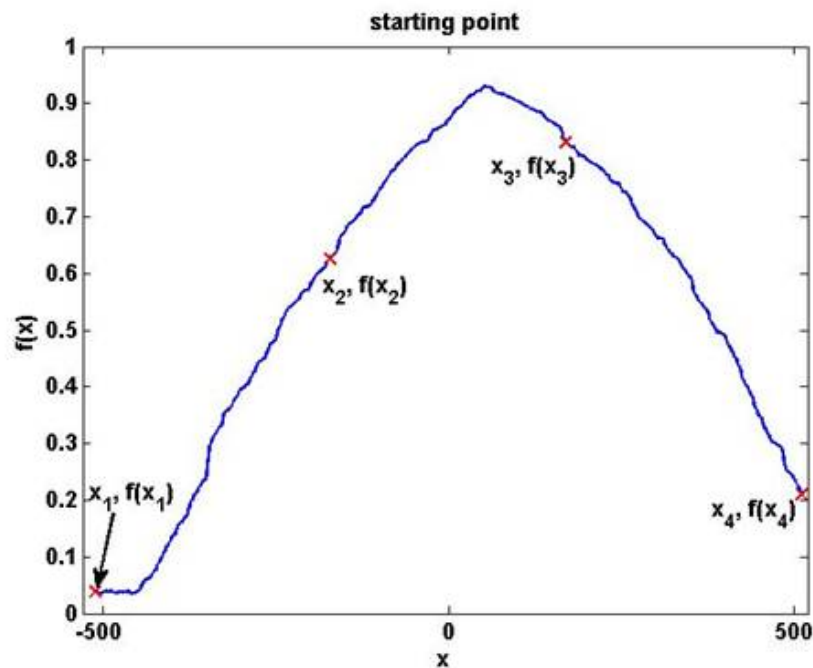


Figure 1: function maximisation, starting point (low FTD)

The blue line depicts the cost function. Without the proposed algorithm it would have to be calculated for all  $x$ . The first step of the algorithm is to calculate  $f(x_1)$ ,  $f(x_2)$ ,  $f(x_3)$  and  $f(x_4)$  (four calculations). Obviously  $f(x_2) < f(x_3)$  holds. Thus, the looked for maximum cannot be located between  $x_1$  and  $x_2$ . Hence, all trial values smaller than  $x_2$  can be excluded:

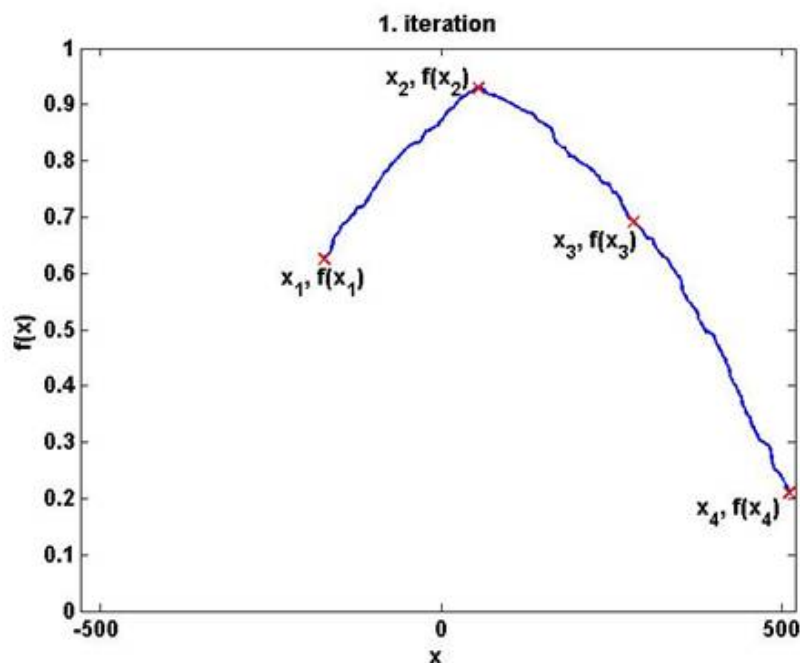


Figure 2: function maximisation, 1. iteration (low FTD)

The new trial values to be used are the two border values and two values in between that way, that all four values divide the complete value space into 3 equal parts. Again  $f(x_1)$ ,  $f(x_2)$ ,  $f(x_3)$  and

$f(x_4)$  are to be calculated (two calculations, as  $f(x_1)$  and  $f(x_4)$  are already known from the previous iteration). Now  $f(x_2) > f(x_3)$  holds. Thus, all trial values bigger than  $x_3$  can be excluded:

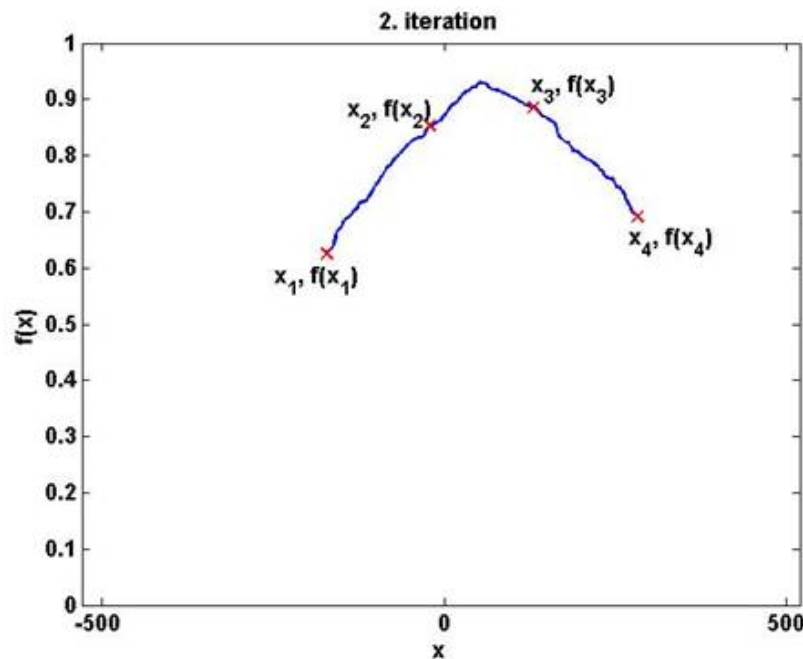


Figure 3: function maximisation, 2. iteration (low FTD)

This goes on until at some point the number of trial values is smaller than a given threshold:

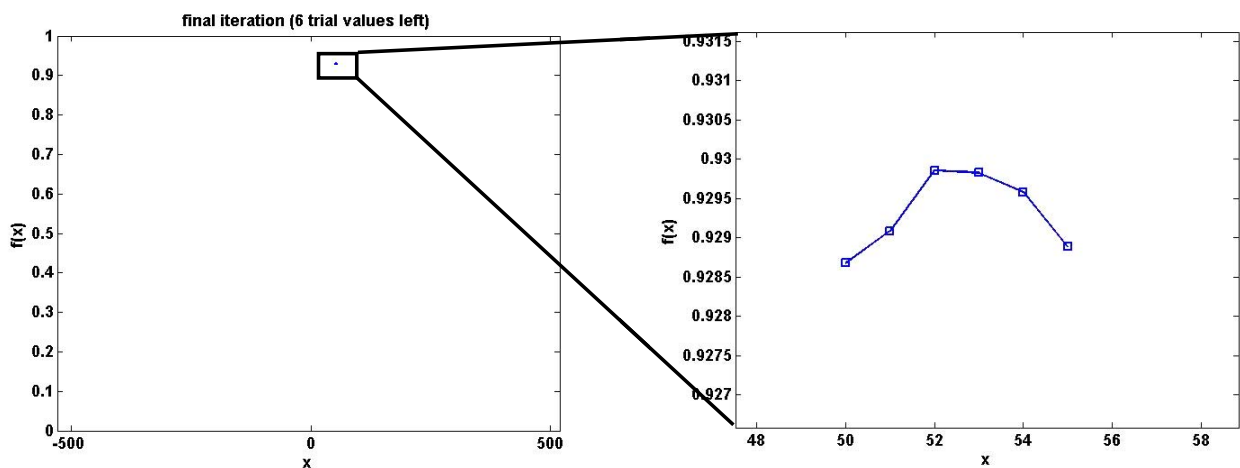


Figure 4: function maximisation, final calculation of the maximum (low FTD)

At this point it is sensible to evaluate the remaining trial values and chose the one maximizing the cost function. The number of metric calculations to be performed with this example is 62 compared to the 1024 when not applying the algorithm.

The second example with high FTD:

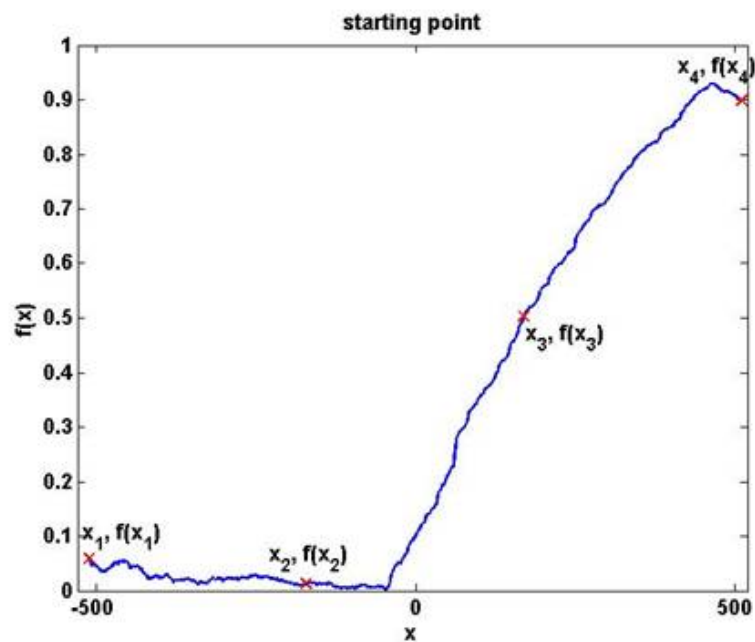


Figure 5: function maximisation, starting point (high FTD)

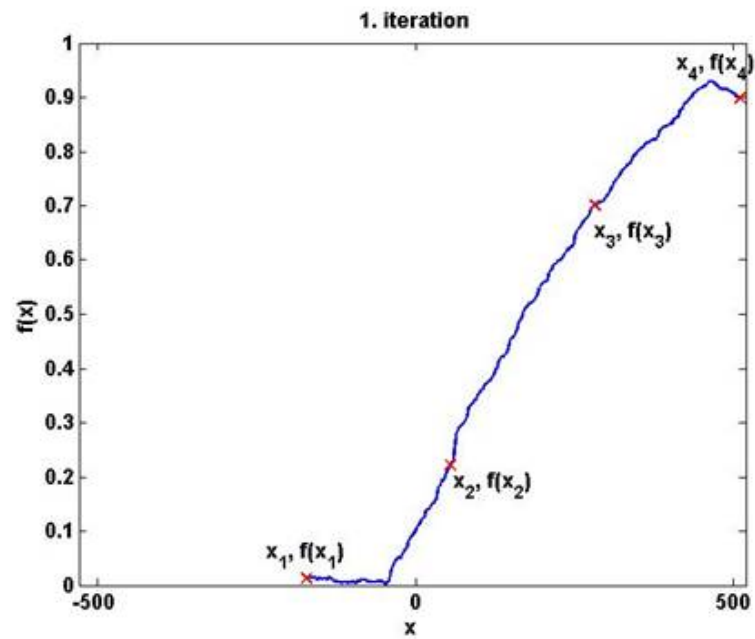


Figure 6: function maximisation, 1. iteration (high FTD)

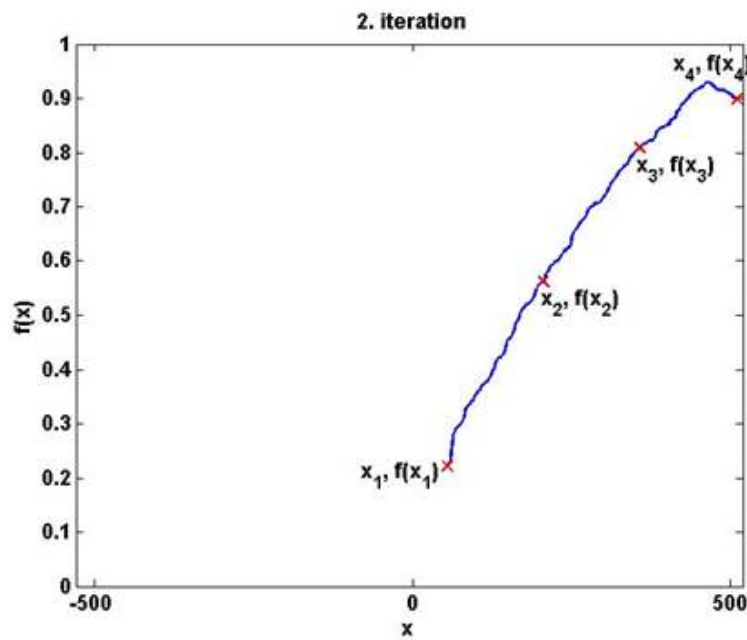


Figure 7: function maximisation, 2. iteration (high FTD)

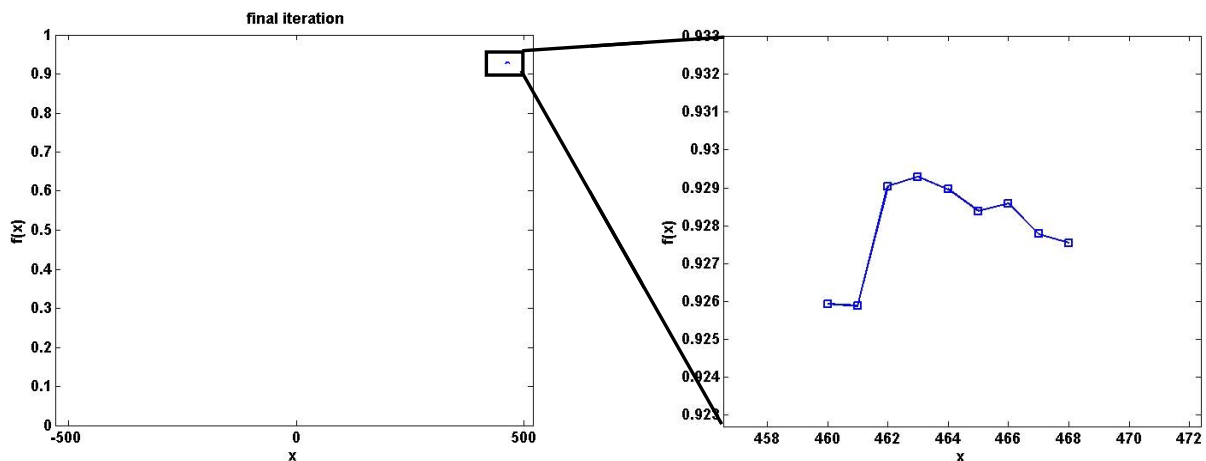


Figure 8: function maximisation, final calculation of the maximum (high FTD)

To get reliable estimates both the envelope and the signal must exhibit the above mentioned symmetry. To achieve a periodic signal someone simply has to null the symbol of every second carrier. Recalling the response of the filter bank we obviously have to take appropriate actions to produce the periodicity of the envelope. One possibility is to apply the memory preloading concept followed by a proper windowing. Another way is to simply bypass the filter bank during the construction of the preamble, directly transmitting the FFT output.

Once the offsets are estimated the compensation is done in time domain simply by shifting the signal in time and applying the frequency correction. I.e. not the symptoms are combated as it is done with pilot based synchronization (i.e. phase distortion, ICI, ISI), but the actual disease (i.e. time and frequency shifts) is cured.

The parameters to activate and define preamble based synchronization are (within PAR1\_CELL\_... and PAR6\_MS\_DIGI\_...):

**Table 1: Parameter space for preamble based synchronization**

Parameter name	Possible values	Remarks
C1_CELL.Frm.PreambleType	'LS'	<b>LS:</b> Preamble with two identical parts for preamble based synchronisation with the least squares approach.
C1_MS_DIGI.FBMC.CFO_correction	'LS_Preamble'	<b>LS_Preamble:</b> apply preamble based CFO estimation using the LS approach
C1_MS_DIGI.FBMC.FTD_estimation	'LS_Preamble'	<b>LS_Preamble:</b> apply preamble based FTD estimation using the LS approach
C1_MS_DIGI.FBMC.FasterMax	'true', 'false'	boolean to activate faster cost function maximisation

### 1.1.2 RF Impairments

The RF impairments implemented are:

- static DC offset
- phase noise
- IQ-imbalance
- Power amplifier distortions
- non-linearities (LNA, Mixer)
- ADC

For a detailed description of the models the interested reader is referred to [2] and [3].

The additions to the parameter space are (within PAR1\_CELL\_...and PAR2\_MS\_... (TX) and PAR5\_MS\_ANA\_... and PAR5\_BS\_ANA\_..., respectively (RX)):

**Table 2: Parameter space for RF impairments**

Parameter name	Possible values	Remarks
C1_MS_ANA.NF (DL) C1_BS_ANA.NF (UL)	real value	Noise factor and implementation loss (dB) (RX).
C1_MS_ANA.Oversampling (DL, RX) C1_CELL.Frm.Oversampling (DL, TX) C1_BS_ANA.Oversampling (UL,	0, 1	<b>0:</b> No oversampling. <b>1:</b> Apply oversampling.

RX) <a href="#">C1_MS.Oversampling</a> (UL, TX)		
<a href="#">C1_MS_ANA.OversamplingFactor</a> (DL, RX) <a href="#">C1_CELL.FrmOversamplingFactor</a> (DL, TX) <a href="#">C1_BS_ANA.OversamplingFactor</a> (UL, RX) <a href="#">C1_MS.OversamplingFactor</a> (UL, TX)	positive integer	Oversampling factor. The baseband signal is oversampled at input of TX and RX RF imperfection blocks and then filtered and downsampled at their respective outputs.
<a href="#">C1_CELL.Frm.TX_IQ_mismatch</a> (DL) <a href="#">C1_MS.TX_IQ_mismatch</a> (UL)	0,1	<b>0</b> : Deactivate IQ imbalance model (TX). <b>1</b> : Activate IQ imbalance model (TX).
<a href="#">C1_CELL.Frm.TX_IQ_delta_gain_dB</a> (DL) <a href="#">C1_MS.TX_IQ_delta_gain_dB</a> (UL)	real value	Amplitude imbalance between I and Q branches (according to the definition chosen by selector <a href="#">TX_IQ_delta_gain_def</a> ) (dB) (TX).
<a href="#">C1_CELL.Frm.TX_IQ_delta_gain_def</a> (DL) <a href="#">C1_MS.TX_IQ_delta_gain_def</a> (UL)	1, 2	<b>1</b> : ... <a href="#">TX_IQ_delta_gain_dB</a> is defined as the ratio of the highest power branch (among I and Q) to the mean power of the complex IQ signal, i.e. $\epsilon = 10^{(\dots TX\_IQ\_delta\_gain\_dB/20)-1}$ . <b>2</b> : ... <a href="#">TX_IQ_delta_gain_dB</a> is defined as the I to Q power ratio, i.e. $\epsilon = (10^{(\dots TX\_IQ\_delta\_gain\_dB)/20}-1)/(10^{(\dots TX\_IQ\_delta\_gain\_dB)/20}+1)$ .
<a href="#">C1_CELL.Frm.TX_IQ_delta_phase_deg</a> (DL) <a href="#">C1_MS.TX_IQ_delta_phase_deg</a> (UL)	real number	Phase orthogonality imbalance (TX). The branch I is advanced by <a href="#">TX_IQ_delta_phase</a> and branch Q is delayed by <a href="#">TX_IQ_delta_phase</a> ., relatively to the non impaired signal.
<a href="#">C1_CELL.Frm.TX_PhaseNoise</a> (DL) <a href="#">C1_MS.TX_PhaseNoise</a> (UL)	0, 1	<b>0</b> : Phase noise present (TX). <b>1</b> : No phase noise (TX).
<a href="#">C1_CELL.Frm.TX_PhaseNoise_PSD</a> (DL) <a href="#">C1_MS.TX_PhaseNoise_PSD</a> (UL)	integer	$1/f^2$ phase noise level @ frequency reference from the carrier (dBc/Hz) (TX).
<a href="#">C1_CELL.Frm.TX_PhaseNoise_FRef</a> (DL) <a href="#">C1_MS.TX_PhaseNoise_FRef</a> (UL)	integer	Frequency reference offset (Hz) (TX).
<a href="#">C1_CELL.Frm.TX_PhaseNoise_wpn</a> (DL) <a href="#">C1_MS.TX_PhaseNoise_wpn</a> (UL)	integer	White phase noise (dBc/Hz) (TX).

C1_CELL.Frm.TX_PhaseNoise_pll_cutoff (DL) C1_MS.TX_PhaseNoise_pll_cutoff (UL)	integer	PLL cutoff frequency (Hz) (TX).
C1_CELL.Frm.PA (DL) C1_MS.PA (UL)	0, 1	<b>0</b> : No power amplifier distortion (TX). <b>1</b> : Power amplifier distortion present (TX).
C1_CELL.Frm.PA_AMAM_model (DL) C1_MS.PA_AMAM_model (UL)	'sspa', 'twi', 'spline', 'limiter'	Selection for the power amplifier AMAM characteristic model. See Phydias D8.1 or D9.2 for more details.
C1_CELL.Frm.PA_AMPM_model (DL) C1_MS.PA_AMPM_model (UL)	'sspa', 'twi', 'spline'	Selection for the power amplifier AMPM characteristic model. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_renormalize (DL) C1_MS.PA_renormalize (UL)	0, 1	<b>0</b> : Do not renormalize the signal passed to the PA model block (debug mode only). <b>1</b> : Renormalize the signal passed to the PA model block (default).
C1_CELL.Frm.PA_IBO (DL) C1_MS.PA_IBO (UL)	real number	Input power back-off relatively to the PA saturation power (dB).
C1_CELL.Frm.PA_A0 (DL) C1_MS.PA_A0 (UL)	real number	SSPA model parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_p (DL) C1_MS.PA_p (UL)	real number	SSPA model parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_alpha_A (DL) C1_MS.PA_alpha_A (UL)	real number	TWT model Parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_beta_A (DL) C1_MS.PA_beta_A (UL)	real number	TWT model Parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_alpha_phi (DL) C1_MS.PA_alpha_phi (UL)	real number	TWT model parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_beta_phi (DL) C1_MS.PA_beta_phi (UL)	real number	TWT model parameter. See Phydias [12] or [2] for more details.
C1_CELL.Frm.PA_TWTAM_Pinsat_dBm (DL) C1_MS.PA_TWTAM_Pinsat_dBm (UL)	real number	Input saturation power (dB) for TWT AMAM model. Above this value, the AMAM output is saturated. (This allows to deal with the non realistic behaviour of the TWT model for high input power values.)
C1_CELL.Frm.PA_TWTPM_Pinsat_dBm (DL)	real number	Input saturation power (dB) for the PA TWT AMPM model. Above this value, the

C1_MS.PA_TWTPM_Pinsat_dBm (UL)		AMPM output is saturated. (This is to deal with the non realistic behaviour of the TWT model for high input power values.)
C1_CELL.Frm.PA_splineAM_Pinmin_dBm (DL) C1_MS.PA_splineAM_Pinmin_dBm (UL)	real number	Minimum input power (dB) for the spline AMAM model. Below this value, the AMAM is modelled as linear.
C1_CELL.Frm.PA_splineAM_Pinsat_dBm (DL) C1_MS.PA_splineAM_Pinsat_dBm (UL)	real number	Input saturation power (dB) for the spline AMAM model. Above this value, the AMAM output is saturated.
C1_CELL.Frm.PA_splinePM_Pinmin_dBm (DL) C1_MS.PA_splinePM_Pinmin_dBm (UL)	real number	Minimum input power (dB) for the spline AMPM model. Below this value, the AMPM is modelled as linear.
C1_CELL.Frm.PA_splinePM_Pinsat_dBm (DL) C1_MS.PA_splinePM_Pinsat_dBm (UL)	real number	Input saturation power (dB) for the spline AMPM model. Above this value, the AMPM output is saturated.
C1_CELL.Frm.PA_pp_phase (DL) C1_MS.PA_pp_phase (UL)	real vector	Parameters for the <i>ppval</i> matlab function (see matlab documentation) describing the AMAM normalized characteristic as a piecewise polynomial.
C1_CELL.Frm.PA_pp_amp (DL) C1_MS.PA_pp_amp (UL)	real vector	Parameters for the <i>ppval</i> matlab function (see matlab documentation) describing the AMAM normalized characteristic as a piecewise polynomial.
C1_CELL.Frm.PA_limiter_Pinsat_dBm (DL) C1_MS.PA_limiter_Pinsat_dBm (UL)	real vector	Input saturation power (dB) for the simple limiter model. Above this value, the AMAM output is saturated.
C1_MS_ANA.LNA (DL) C1_BS_ANA.LNA (UL)	0, 1	<b>0</b> : Deactivate LNA non linearity (RX). <b>1</b> : Activate LNA non linearity (RX).
C1_MS_ANA.LNA_IIP3 (DL) C1_BS_ANA.LNA_IIP3 (UL)	real number	LNA 3 <sup>rd</sup> Order Input Interception Point (dBm) (RX).
C1_MS_ANA.LNA_IIP5 (DL) C1_BS_ANA.LNA_IIP5 (UL)	real number	LNA 5 <sup>rd</sup> Order Input Interception Point (dBm) (RX).
C1_MS_ANA.LNA_Gain (DL) C1_BS_ANA.LNA_Gain (UL)	real number	LNA Gain (dB) (RX).
C1_MS_ANA.Split_Loss (DL)	real number	Split losses between LNA and mixers in the



C1_BS_ANA.Split_Loss (UL)		receiver (dB) (RX).
C1_MS_ANA.Mixer (DL) C1_BS_ANA.Mixer (UL)	0, 1	<b>0:</b> Deactivate mixer non linearity (RX). <b>1:</b> Activate mixer non linearity (RX).
C1_MS_ANA.Mixer_IIP3 (DL) C1_BS_ANA.Mixer_IIP3 (UL)	real number	Mixer 3 <sup>rd</sup> Order Input Interception Point (dBm) (RX).
C1_MS_ANA.Mixer_Gain (DL) C1_BS_ANA.Mixer_Gain (UL)	real number	Mixer Gain (dB) (RX).
C1_MS_ANA.LO_Power (DL) C1_BS_ANA.LO_Power (UL)	real number	LO Power (dBm) (RX).
C1_MS_ANA.LO_IF_Isol (DL) C1_BS_ANA.LO_IF_Isol (UL)	real number	LO-IF Isolation (RX).
C1_MS_ANA.PhaseNoise (DL) C1_BS_ANA.PhaseNoise (UL)	0, 1	<b>0:</b> No phase noise at the receiver. <b>1:</b> Phase noise at the receiver present.
C1_MS_ANA.PhaseNoise_PSD (DL) C1_BS_ANA.PhaseNoise_PSD (UL)	real number	1/f <sup>2</sup> phase noise level @ frequency reference from the carrier (dBc/Hz) (RX).
C1_MS_ANA.PhaseNoise_FRef(DL) C1_BS_ANA.PhaseNoise_FRef (UL)	real number	Frequency reference offset (Hz) (RX).
C1_MS_ANA.PhaseNoise_wpn (DL) C1_BS_ANA.PhaseNoise_wpn (UL)	real number	White phase noise (dBc/Hz) (RX).
C1_MS_ANA.PhaseNoise_pll_cutoff (DL) C1_BS_ANA.PhaseNoise_pll_cutoff (UL)	real number	PLL cutoff frequency (Hz) (RX).
C1_MS_ANA.DCO (DL) C1_BS_ANA.DCO (UL)	0, 1	<b>0:</b> No DC offset. <b>1:</b> DC offset present.
C1_MS_ANA.DC_offset (DL) C1_BS_ANA.DC_offset (UL)	real number	DC offset (% of mean signal amplitude).
C1_MS_ANA.IQ_mismatch (DL) C1_BS_ANA.IQ_mismatch (UL)	0, 1	<b>0:</b> deactivate IQ imbalance model (RX). <b>1:</b> activate IQ imbalance model (RX).
C1_MS_ANA.IQ_delta_gain_def (DL) C1_BS_ANA.IQ_delta_gain_def (UL)	1, 2	<b>1:</b> ...IQ_delta_gain_dB is defined as the ratio of the highest power branch (among I and Q) to the mean power of the complex IQ signal, i.e. $\epsilon = 10^{(\dots \text{IQ\_delta\_gain\_dB}/20)-1}$ <b>2:</b> ... IQ_delta_gain_dB is defined as the I to Q power ratio, i.e. $\epsilon = (10^{(\dots \text{IQ\_deltagain\_dB})/20}-1)/(10^{(\dots \text{IQ\_deltagain\_dB})/20}+1)$
C1_MS_ANA.IQ_delta_gain_dB	real value	Amplitude imbalance between I and Q

(DL) <a href="#">C1_BS_ANA.IQ_delta_gain_dB</a> (UL)		branches according to the definition chosen by selector <a href="#">IQ_delta_gain_def</a> (dB) (RX).
<a href="#">C1_MS_ANA.IQ_delta_phase_deg</a> (DL) <a href="#">C1_BS_ANA.IQ_delta_phase_deg</a> (UL)	real value	Phase orthogonality imbalance. The branch I is advanced by <a href="#">IQ_delta_phase</a> and branch Q is delayed by <a href="#">IQ_delta_phase</a> , relatively to the non impaired signal (RX).
<a href="#">C1_MS_ANA.DCFilter</a> (DL) <a href="#">C1_BS_ANA.DCFilter</a> (UL)	0, 1	<b>0</b> : deactivate DC filter (RX). <b>1</b> : activate DC filter (RX).
<a href="#">C1_MS_ANA.ADConv</a> (DL) <a href="#">C1_BS_ANA.ADConv</a> (UL)	0, 1	<b>0</b> : deactivate AD converter quantization noise (RX). <b>1</b> : activate AD converter converter quantization noise (RX).
<a href="#">C1_MS_ANA.ADC_Input_Power</a> (DL) <a href="#">C1_BS_ANA.ADC_Input_Power</a> (UL)	real number	<a href="#">ADC Input Power</a> (dB) (RX).
<a href="#">C1_MS_ANA.ADC_Nb_bits</a> (DL) <a href="#">C1_BS_ANA.ADC_Nb_bits</a> (UL)	positive integer	Resolution of the ADC (Number of bits) (RX).

### 1.1.3 MIMO

Due to a delay within the respective workpackage the inclusion of MIMO is shifted to the next deliverable of workpackage 9.

### 1.1.4 OFDM-specific additions

To fully align the OFDM base to the FBMC approach the channel estimator used by the OFDM part of the simulator was expanded to apply two-dimensional linear interpolation. Furthermore the CFO/FTD estimation algorithms based on linear regression provided by the project partners were adapted for the use in OFDM mode.

The new parameters are (within PAR6\_BS\_DIGI... (UL) and PAR6\_MS\_DIGI...(DL)):

**Table 3: Parameter space for the OFDM specific additions**

Parameter name	Possible values	Remarks
C1_MS_DIGI.WiMAX. CFO_correction (DL)  C1_BS_DIGI.WiMAX. CFO_correction (UL)	‘PCI’, ‘None’, ‘RegressionEstimation’	algorithm to be used for CFO estimation: <b>PCI:</b> use perfect channel knowledge <b>None:</b> no CFO correction <b>RegressionEstimation:</b> linear regression (in time direction) using the estimates at the pilot positions as supporting points
C1_MS_DIGI.WiMAX. FTD_estimation (DL)  C1_BS_DIGI.WiMAX. FTD_estimation (UL)	‘PCI’, ‘Estimation’	algorithm to be used for FTD estimation: <b>PCI:</b> use perfect channel knowledge <b>Estimation:</b> linear regression (in frequency direction) using the estimates at the pilot positions as supporting points

### 1.1.5 Planned Additions

Depending on the advances of the respective work packages the planned additions within the next semester are:

- MIMO
  - transmit/receive diversity
  - spatial multiplexing
  - channel estimation for multiple antennas
  - synchronization for multiple antennas
- Preamble based synchronization (uplink)

## 1.2 Complexity assessment

The increased complexity due to the insertion of the filter bank is one of the concerns against FBMC. First studies were performed to compare the complexity of FBMC to OFDM [4]. The outcome was that FBMC is around 3 to 4 times more complex than OFDM for typical systems (i.e. using an overlap factor of 4 and 1024 subcarriers).

This study shall position the complexity of the filter bank into the overall complexity of the physical layer investigated (in terms of complex multiplications per complex symbol) to assess the impact of this increase in complexity to the overall system. The by far most complex algorithm within the complete processing chain is the turbo decoding. Therefore, the complexity of the complete system is approximated by the complexity of the turbo decoder in the following.

The following figure depicts the general structure of a turbo decoder for parallel concatenated codes [5]:

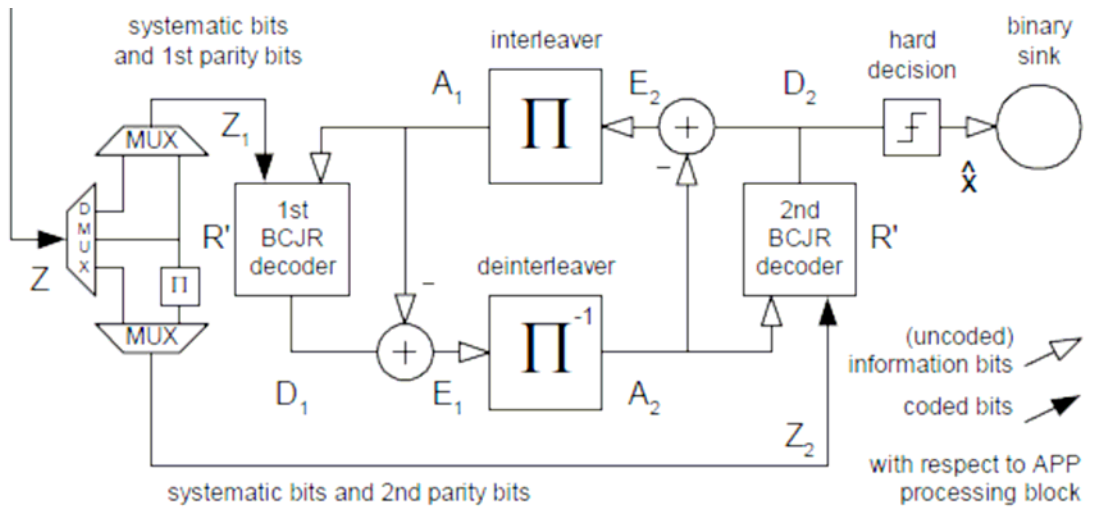


Figure 9: turbo decoder for parallel concatenated codes

Neglecting the sum operations the complexity of the scheme resides within the two BCJR decoders and the interleaver/deinterleaver.

The deInterleaver is not adding to the overall complexity with respect to the number of complex multiplications per complex symbol. The number of complex multiplications of a single stage of the BCJR algorithm per time step is according to [6]:

$$N_{BCJR} = 2 \cdot M \cdot 2^{k_0} + M$$

with

$$M = 2^{\nu-1}$$

The classic turbo encoder at the transmitter consists of two component encoders (convolutional encoders).  $k_0$  is the number of shift registers per component code,  $\nu$  is their constraint length.

Thus the number of complex multiplications per time step depending on the number of iterations  $I$  performed is:

$$N_{mult} = I \cdot 2 \cdot N_{BCJR}$$

The physical layer investigated in PHYDYAS is based on WiMAX [7][8]. Here the parameters settings are:  $\nu = 3$ ,  $k_0 = 2$ .

Thus, the number of complex multiplications per time step is:

$$N_{mult} = I \cdot 72$$

Typically  $I = 3 \dots 5$  iterations are sufficient:

$$N_{mult} = 216 \dots 360$$

Within a single time step two information bits are processed. To calculate the number of complex multiplications per QAM symbol the number of information bits per QAM symbol is needed:

**Table 4: Number of information bits per complex symbol**

	<b>number of information bits per complex symbol</b>
QPSK, code rate 1/2	1
QPSK, code rate 3/4	1.5
16 QAM, code rate 1/2	2
16 QAM, code rate 3/4	3
64 QAM, code rate 1/2	3
64 QAM, code rate 2/3	4
64 QAM, code rate 3/4	4.5
64 QAM, code rate 5/6	5

Finally the number of complex multiplications per complex symbol needed for CTC decoding is:

**Table 5: Number of complex multiplications per complex symbol**

	<b>number of complex multiplications per complex symbol</b>
QPSK, code rate 1/2	108...180
QPSK, code rate 3/4	162...270
16 QAM, code rate 1/2	216...360
16 QAM, code rate 3/4	324...540
64 QAM, code rate 1/2	324...540
64 QAM, code rate 2/3	432...720
64 QAM, code rate 3/4	486...810
64 QAM, code rate 5/6	540...900

Recalling [4] the number of real multiplications per QAM symbol (overlap factor is 4 and 1024 subcarriers are used) is 23/30 (synthesis filterbank/analysis filterbank). Thus the complexity of the filterbank and thus the increase in complexity is negligible compared to the overall complexity of the physical layer used.

## 1.3 Simulation results

### 1.3.1 Spectral efficiency

FBMC is a way to address the inherent inefficiencies of OFDM:

- Cyclic prefix (e.g. 1/8)  $\rightarrow \sim 11.1$  % spectral efficiency wasted
- Frequency guards (e.g. WiMAX)  $\rightarrow \sim 8$  % spectral efficiency wasted

FBMC does not apply a cyclic prefix and due to the low out-of-band leakage much smaller guards can be applied. Hence, FBMC does not introduce these inefficiencies.

We have calculated the spectral efficiency as follows:

$$c = c_{\max} r R_T R_G R_P$$

with

$$\begin{aligned} c_{\max} &= R_{CTC} \log_2(B) \\ r &= 1 - PER \end{aligned}$$

$c_{\max}$  is the maximal achievable spectral efficiency depending on the modulation order  $B$  and the coderate  $R_{CTC}$ . We have based the normalized throughput  $r$  on the packet error rate (PER), i.e. a Bit is treated as erroneous if any of the Bits within its packet (64 Bytes) is received erroneous.  $R_T$  reflects the efficiency in time direction, i.e. it is the ratio of time duration used for data transmission and the time duration of the complete subframe:

$$R_T = \frac{N_S T_S}{T_{SF}}$$

$N_S$  is the number of multicarrier symbols within the subframe used for data transmission,  $T_S$  the useful symbol length.  $T_{SF}$  is the frame length.

$R_G R_P$  reflects the efficiency in frequency direction, i.e. the overhead due to the guard bands and the pilot symbols is accounted for here.

$$\begin{aligned} R_G &= \frac{N_{SC} - N_G - 1}{N_{SC}} \\ R_P &= \frac{N_{SC} - N_G - 1 - N_P}{N_{SC} - N_G - 1} \end{aligned}$$

$N_{SC}$  is the number of subcarriers,  $N_G$  the number of subcarriers used as guard and  $N_P$  the number of pilots per multi carrier symbol.

As already mentioned, the base version of the physical layer under investigation is based on WiMAX. In downlink we have adopted PUSC in uplink AMC permutation schemes. Although it is just an optional solution in the WiMAX standard [8], we use convolutional turbo coding (CTC, code rate 1/2) as channel coding scheme. We have chosen a transform size of 1024, out of which the active subcarriers cover a bandwidth of approximately 10 MHz.

The channel is modelled as Pedestrian B (Ped-B) and Vehicular A (Veh-A), respectively, according to [9]. The receiver first estimates the frequency response of the channel at the pilot locations. Then, 2-D linear interpolation over time and frequency yield the channel estimate at each subcarrier and at each time instant. In the OFDM-based WiMAX system, the length of the cyclic prefix is sensibly chosen to cover most channel delay spreads, so that single tap equalization can be performed. This is the case for scenarios with delay spreads similar to Ped-B/Veh-A and a CP with a normalized length of 1/8.

At the CTC decoding stage four iterations are performed before the bits are decided and fed into the bit sink.

The general parameter settings are summarized in **Erreur ! Source du renvoi introuvable..**

**Table 6: General simulation parameters**

Parameter	chosen value
number of subcarriers (M)	1024
carrier frequency	2.5 GHz
bandwidth	10 MHz
sampling rate	11.2 MHz
subcarrier spacing	10.94 kHz
overlapping factor	4
permutation scheme	PUSC (DL), AMC (UL)
Number of symbols per frame	47 (WiMAX), 53 (FBMC)
packet size	64 Bytes
pilot boost	2.5 dB
coding scheme	CTC
code rate	1/2
number of turbo iterations	4
channel model	Ped-B, 3 km/h Veh-A, 60 km/h

The distribution of the usable multicarrier symbols to the subframes is (one symbol is used as preamble):

**Table 7: TDD uplink/downlink partitioning**

downlink subframe	WiMAX: 31 FBMC: 35
uplink subframe	WiMAX: 15 FBMC: 17



Due to the absence of the CP FBMC can modulate more symbols per frame (WiMAX: 47, FBMC: 53).

#### a) DL PUSC:

Due to the broadcast nature of the downlink and in the case that the coherence bandwidth of the channel exceeds two subcarriers, there is no need for guards between the users. Otherwise the PUSC permutation would not be applicable efficiently anyway.

The following figures compare the spectral efficiency of the WiMAX reference (applying OFDM) and the FBMC system. First perfect channel knowledge (PCI) is assumed. Thus the results are upper bounds. The increased efficiency due to the usage of the filter bank is apparent.

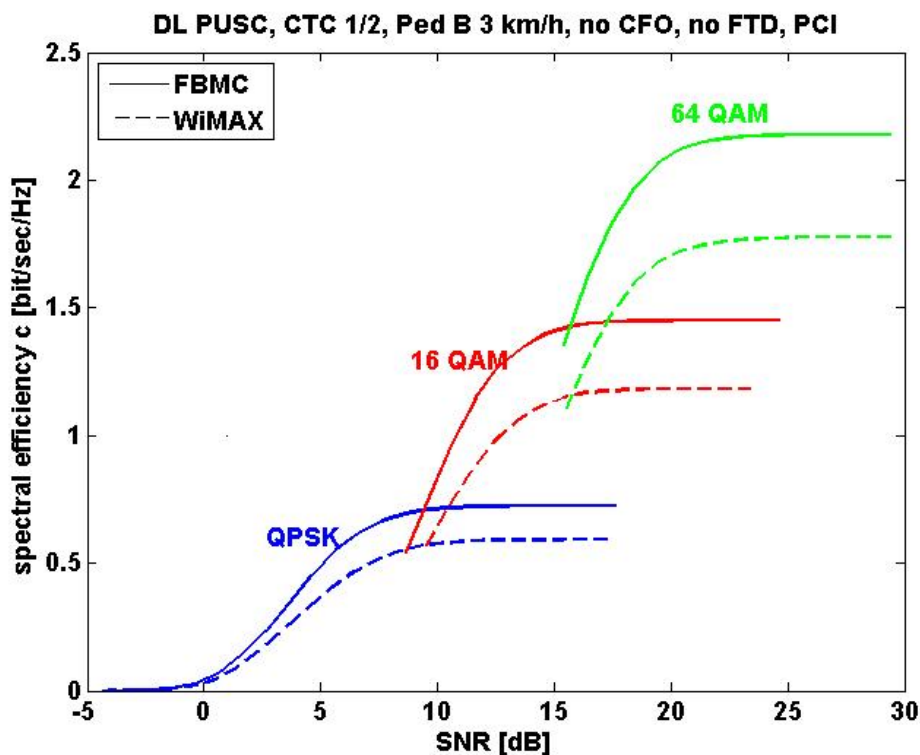


Figure 10: Spectral efficiency for downlink PUSC, Ped B 3 km/h, perfect channel knowledge.

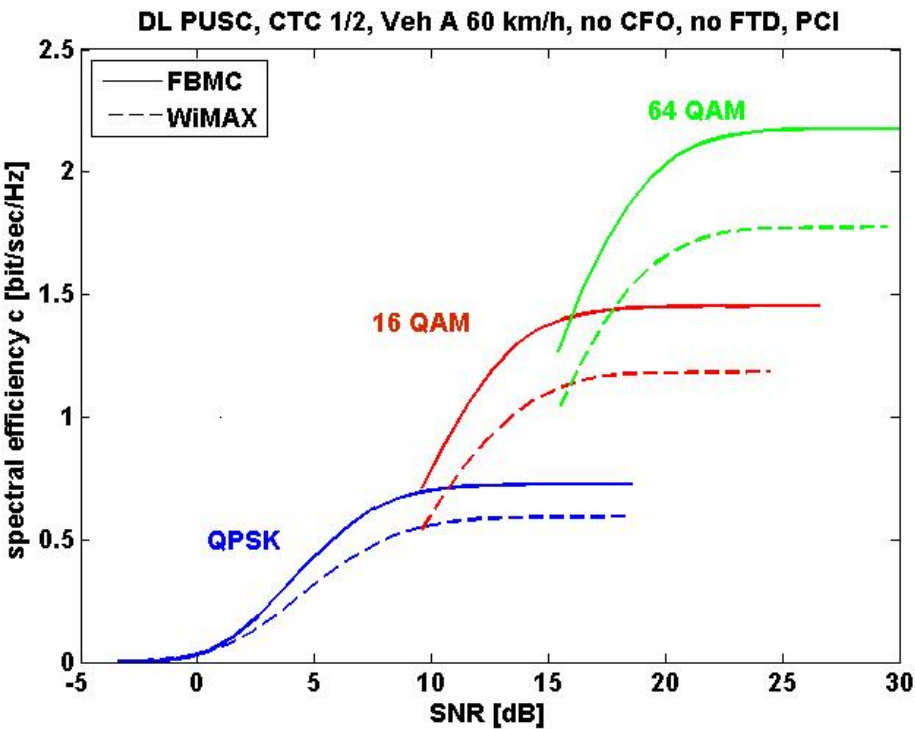


Figure 11: Spectral efficiency for downlink PUSC, Veh A 60 km/h, perfect channel knowledge.

With pilot based channel estimation and linear interpolation:

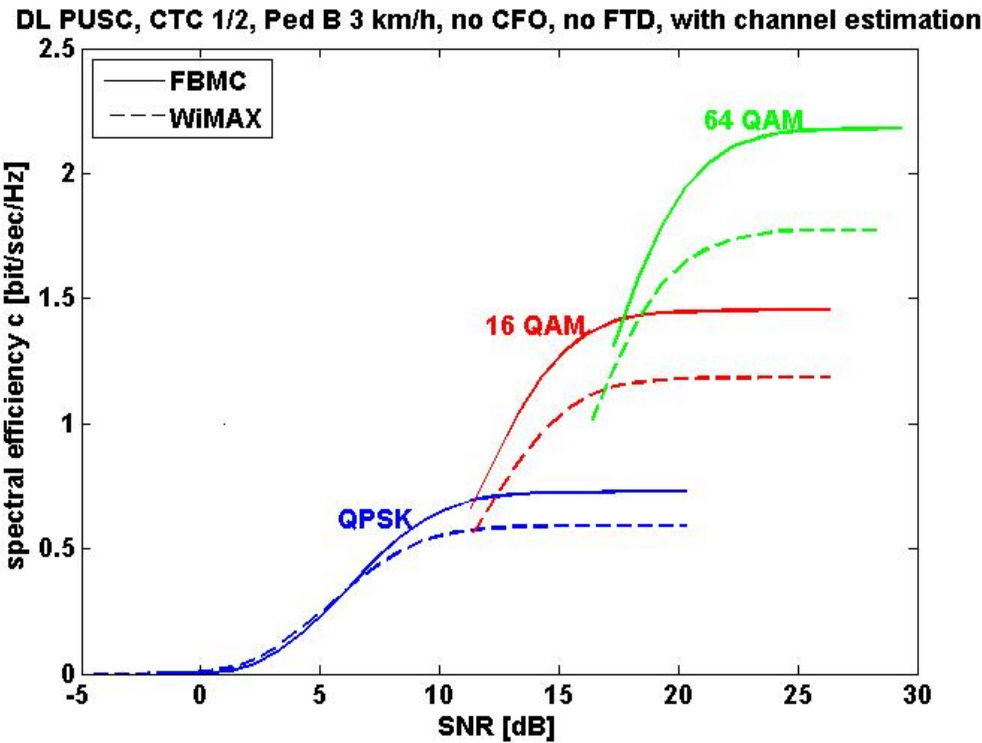


Figure 12: Spectral efficiency for downlink PUSC, Ped B 3 km/h, real channel estimation

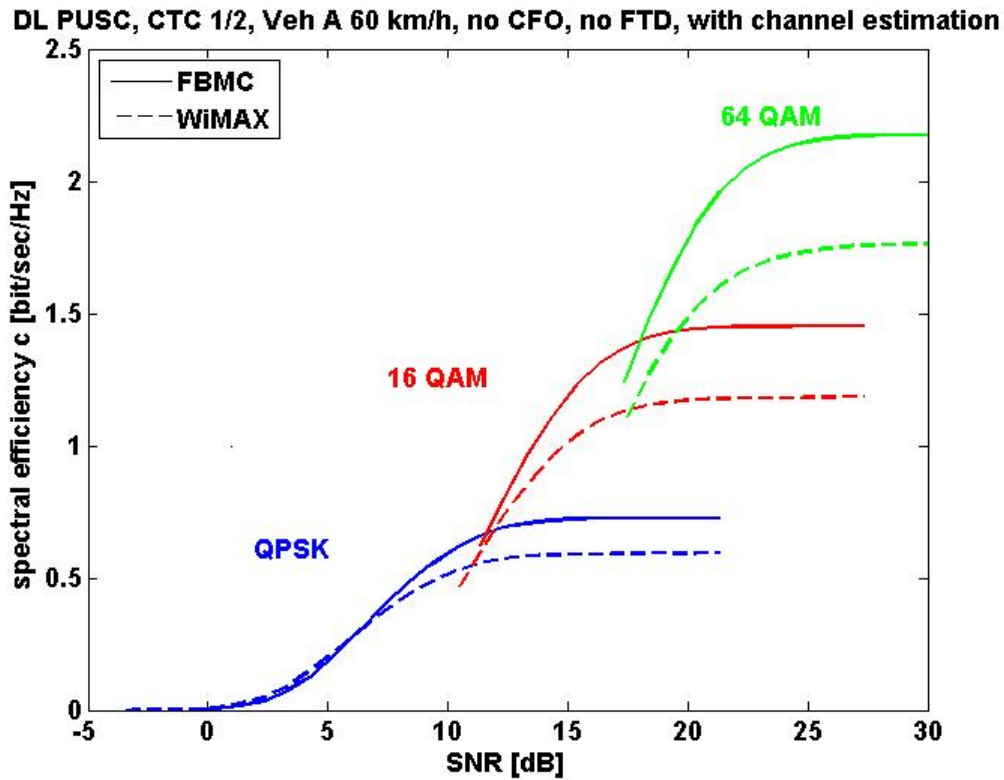


Figure 13: Spectral efficiency for downlink PUSC, Veh A 60 km/h, real channel estimation

#### b) UL AMC:

Contrary to the downlink the uplink is a multiple access channel. Different users access the channel separately. Thus, adjacent subcarriers may encounter different complex channel gains (if they are transmitted by different users), orthogonality is destroyed. Therefore time and frequency guards to separate the users are necessary, reducing the spectral efficiency. The following figures depict the single user case (upper bound), thus no subcarriers are left empty as guards between users, the multi user case follows afterwards. Again first perfect channel knowledge is assumed.

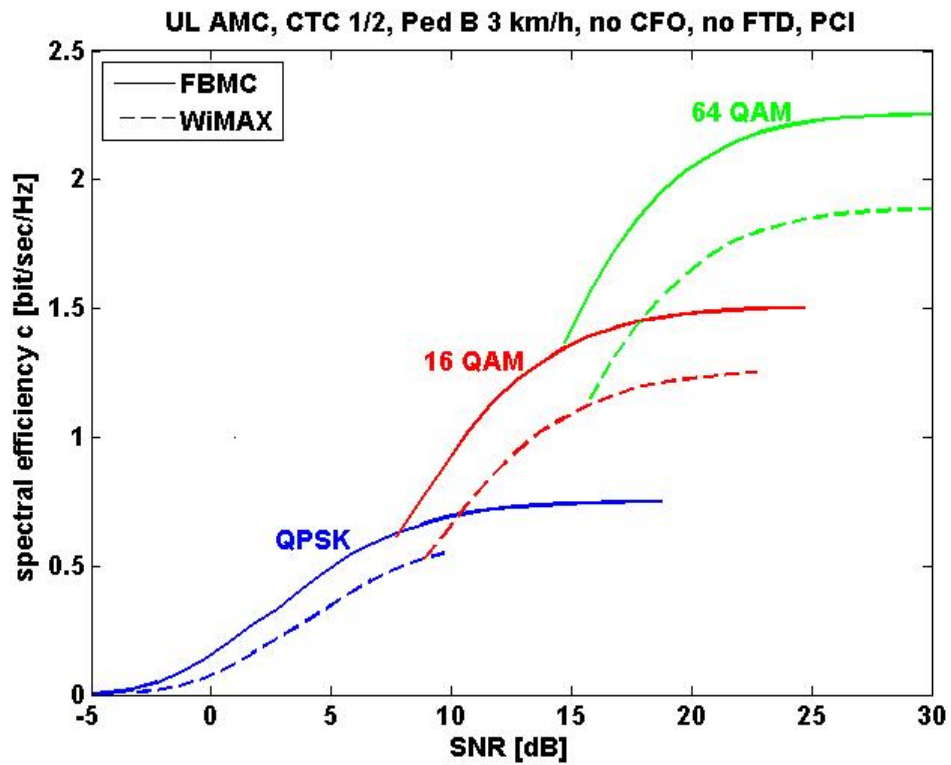


Figure 14: Spectral efficiency for uplink AMC, Ped B 3 km/h, perfect channel knowledge, single user case (upper bound)

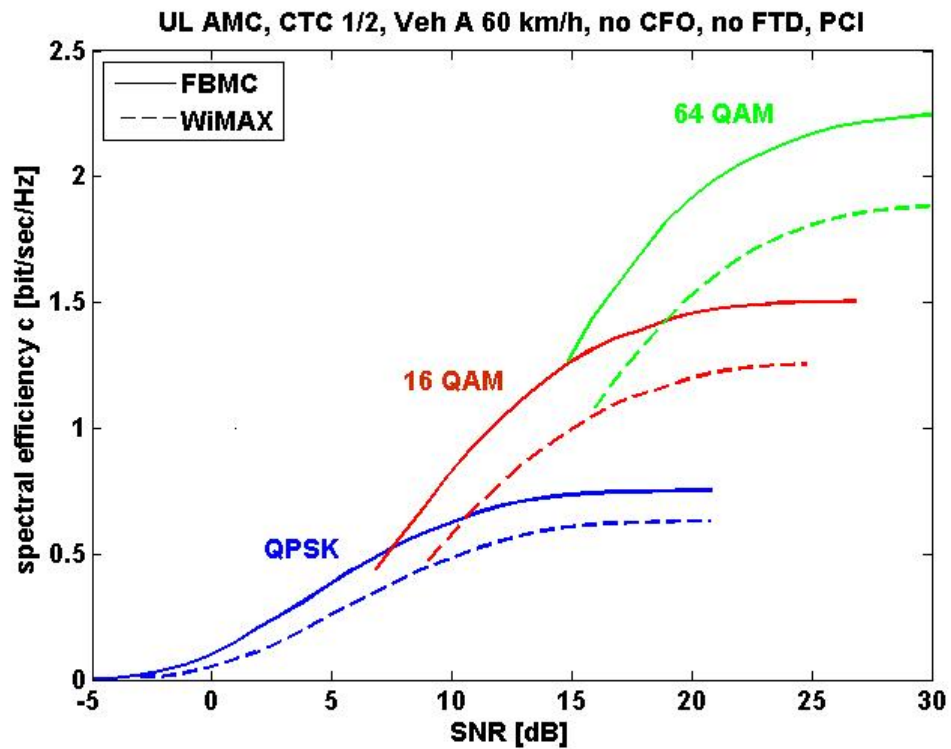


Figure 15: Spectral efficiency for uplink AMC, Veh A 60 km/h, perfect channel knowledge, single user case (upper bound)

Once more the system applying FBMC outperforms the WiMAX reference significantly.

With real channel estimation, whereas the same power is spent for the main pilots in the case of FBMC as it is done in the OFDM case:

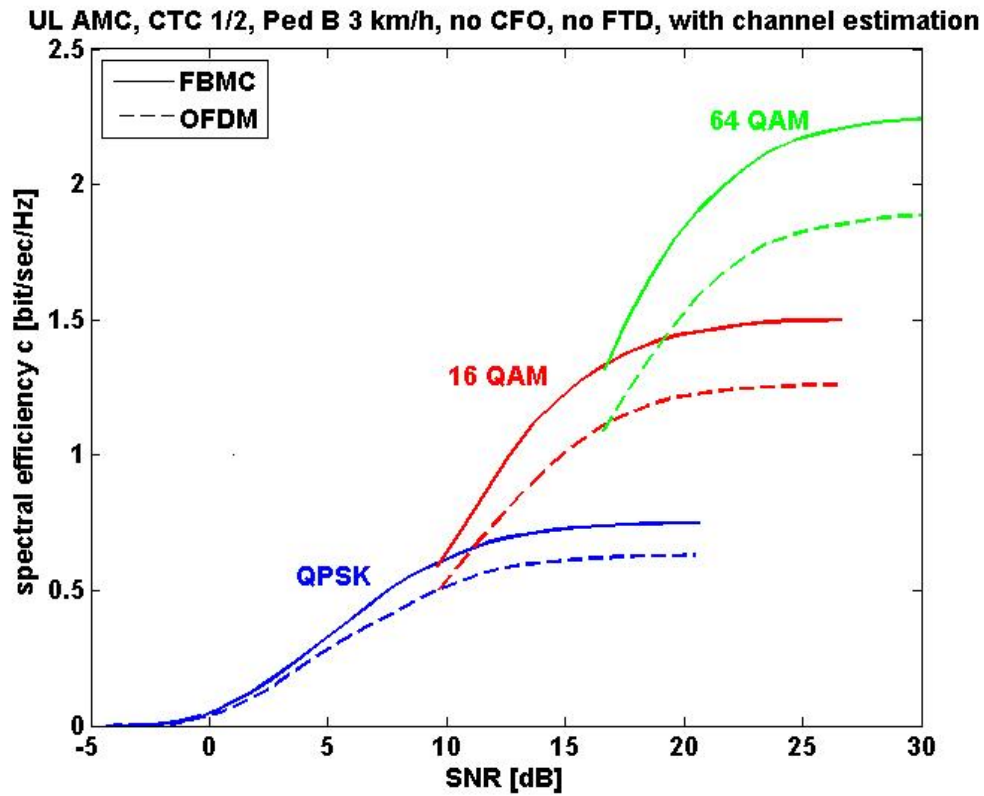


Figure 16: Spectral efficiency for uplink AMC, Ped B 3 km/h, real channel estimation, single user case (upper bound)

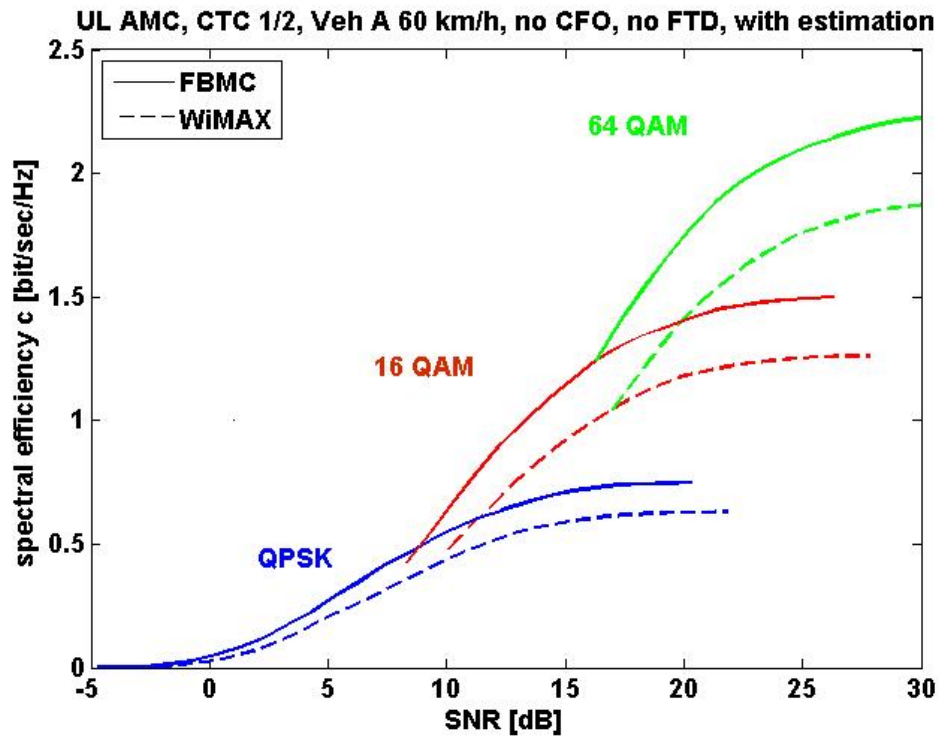
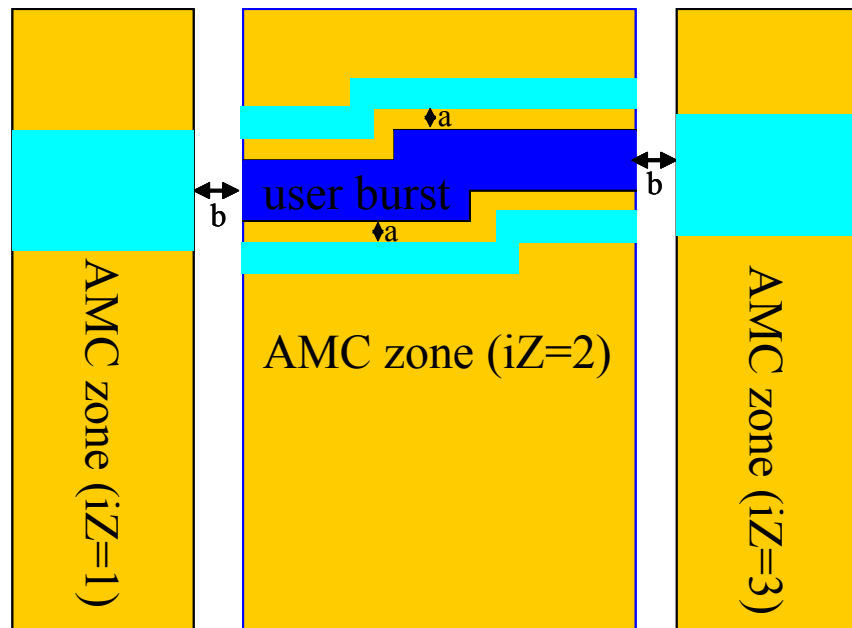


Figure 17: Spectral efficiency for uplink AMC, Veh A 60 km/h, real channel estimation, single user case (upper bound)

As already mentioned the uplink is a multiple access channel. Different users transmit data from different positions within the cell. Thus the channel gains two adjacent subcarriers encounter, are independent, if they trace back to different users. Then, the orthogonality due to the offset QAM is destroyed. Therefore guards are necessary to separate them:



**Interferer:** 16QAM, ideal synch, not decoded

a: frequency guards (step size: subcarriers; 0,1,...)

b: time guards (step size: multicarrier symbols; 0,1,...)

Contiguous data allocation

→ Interferer in frequency must not span complete frame

**Figure 18: Physical user data and interferer placement within the frame.**

For the following assessments perfect ranging is assumed. Thus, no carrier frequency offset, perfect symbol timing and identical receive powers (user under investigation and interferers) are supposed. In the case of QPSK and 16 QAM a single subcarrier and symbol, respectively, is enough to separate the users:

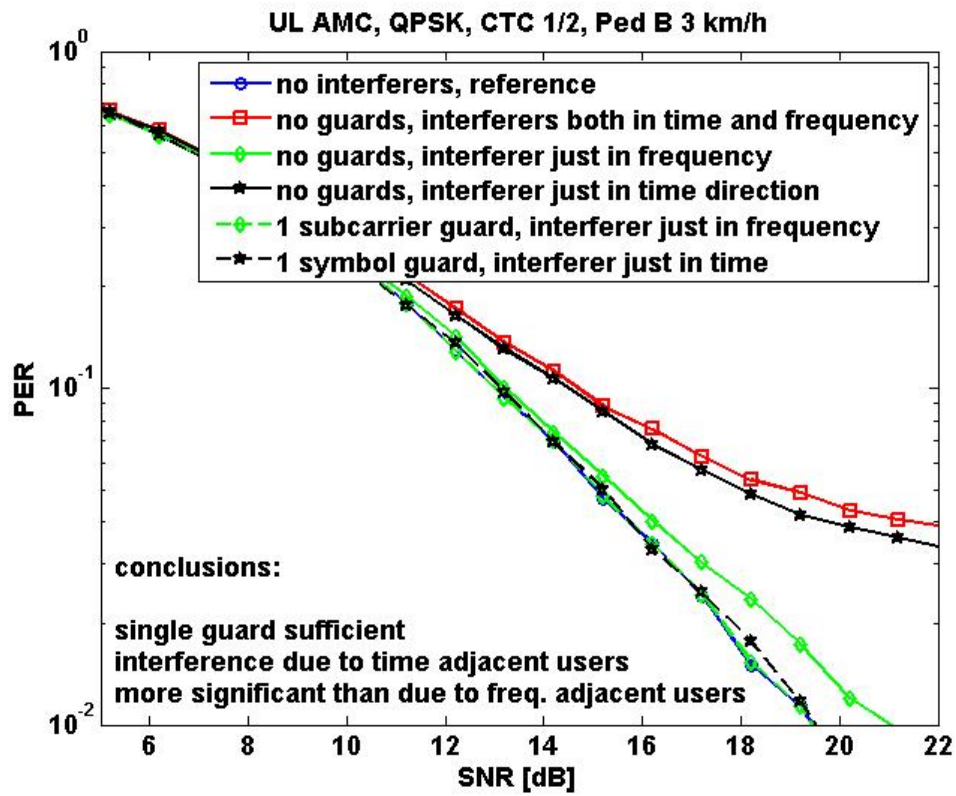


Figure 19: PER depending on the guard dimensions (QPSK).

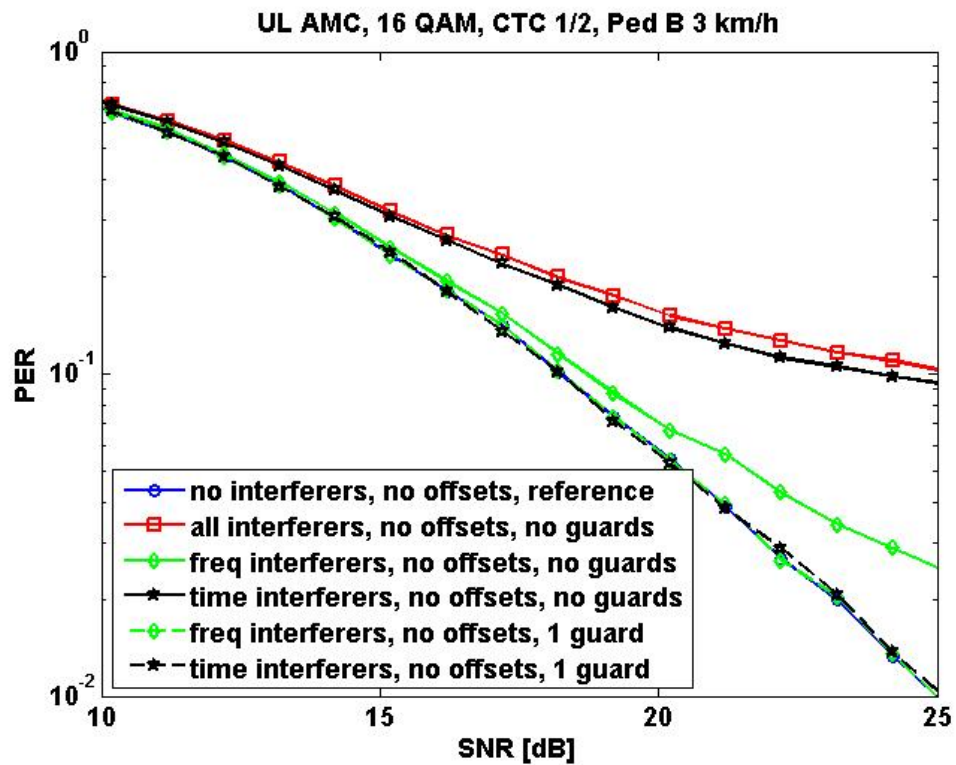


Figure 20: PER depending on the guard dimensions (16 QAM).



64 QAM is more sensitive to multi user interference. If the latter is to be eliminated completely, two symbols are necessary to separate time adjacent users:

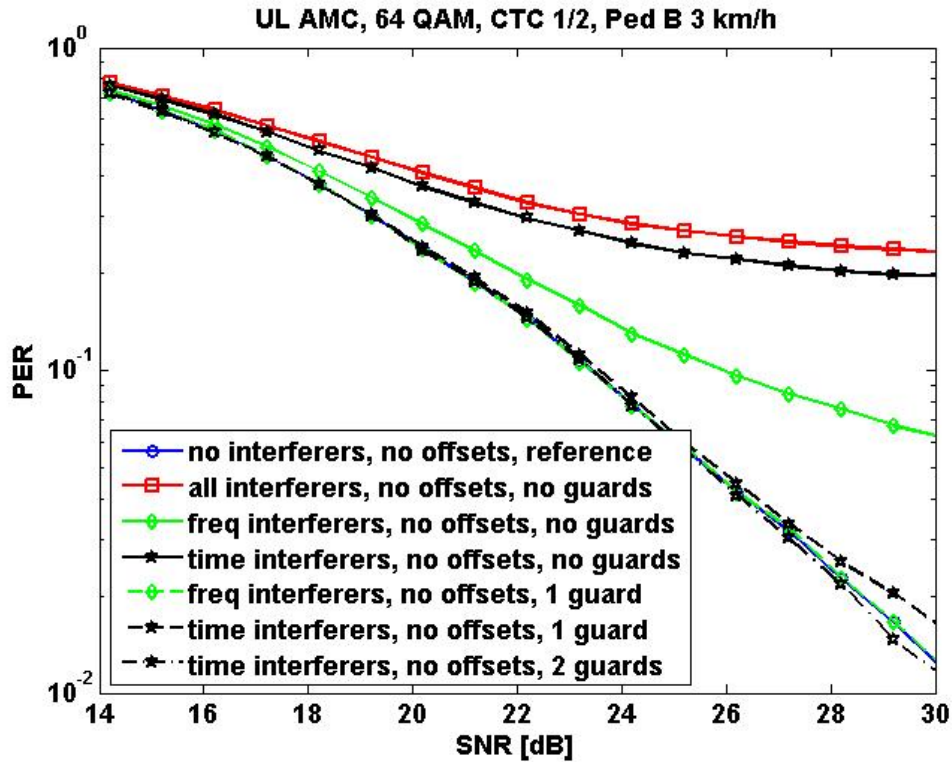


Figure 21: PER depending on the guard dimensions (64 QAM).

For QPSK the loss if no frequency guards are used is comparatively low. Thus they may be omitted increasing spectral efficiency while losing some robustness. With higher order modulation it may be possible to use the frequency guards to transmit QPSK symbols without losing too much performance while gaining efficiency. However, this is not further treated within this study. In the following sufficient guards to completely separate users are assumed.

The following figures depict the spectral efficiency depending on the number of users. The placement of the users (not of the data) is adopted from WiMAX. It is done frequency first.

Assuming that every user got at least as much slots scheduled as the length of the subframe is, the frequency guards obviously span the complete subframe. With the used subframe length (5 slots) and with typical user burst (e.g. 11 slots) sizes in mind, this assumption is justifiable. The number of needed frequency guards ( $N_{\text{freqguards}}$ ) is:

$$N_{\text{freqguards}} = N_{\text{user}} - 1$$

$N_{\text{user}}$  is the number of scheduled users.  $N_{\text{freqguards}}$  can be incorporated into the calculation of  $R_G$  as follows:

$$R_G = \frac{N_{SC} - N_G - N_{\text{freqguards}} - 1}{N_{SC}}$$

Similarly the number of time guards is:

$$N_{timeguards} = (N_{user} - 1) \cdot b \cdot \frac{N_{timeslots} - 1}{N_{timeslots}}$$

The fraction accounts for the cases, when a user burst ends at the end of the subframe (position of the last slot of the user burst in time direction assumed to be equally distributed with the possible values  $[1, 2, \dots, N_{timeslots}]$ ,  $N_{timeslots}$  is the length of the subframe in time slots). In this case there is no need for a time guard between the actual user and the next one. For QPSK and 16 QAM  $b = 1$  holds, in the case of 64 QAM we have  $b = 2$  (remember: 64 QAM needs two symbols as guards). The time guards fortunately do not span a complete multicarrier symbol, but just a single subchannel (18 subcarriers with AMC). Thus the number of subcarriers reserved for user separation in time is (normalized to the number of usable subcarriers per multi carrier symbol):

$$N_{SC\_timeguards} = \frac{(N_{user} - 1) \cdot 18 \cdot b \cdot \frac{N_{timeslots} - 1}{N_{timeslots}}}{N_{SC} - N_G - N_{freqguards} - 1}$$

The number of useable multi carrier symbols for data transmission is then reduced:

$$N'_S = N_S - N_{SC\_timeguards}$$

This value is not necessarily an integer anymore.

Depending on the number of users the spectral efficiency for specific SNRs is, if perfect channel knowledge is present:

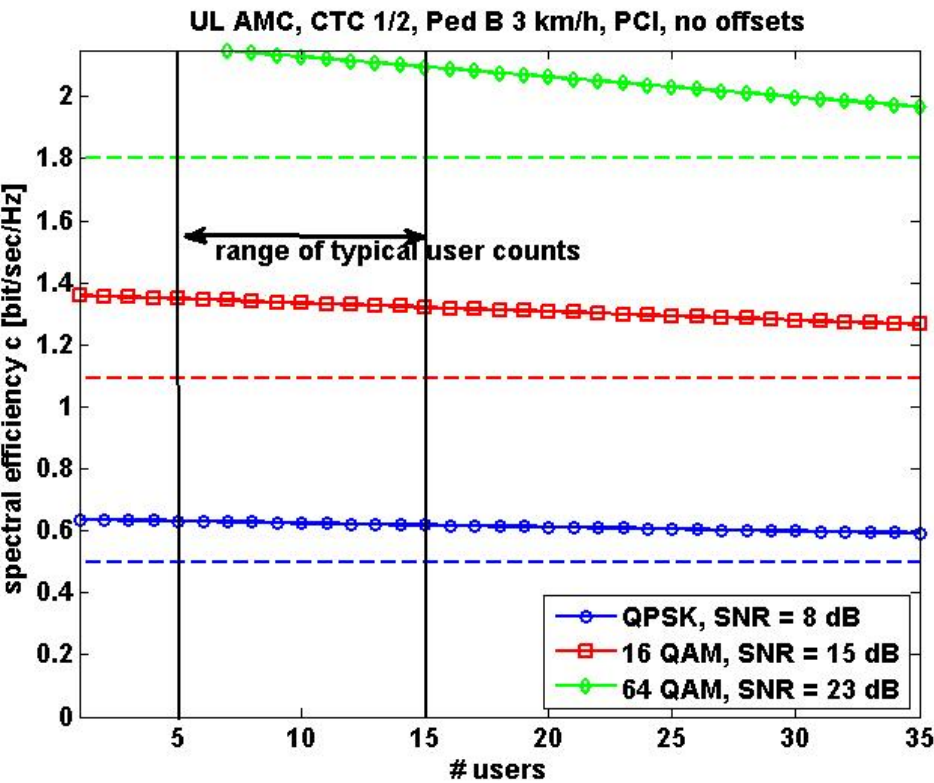


Figure 22: Spectral efficiency for uplink AMC, Ped B 3 km/h, perfect channel knowledge, multi user case.

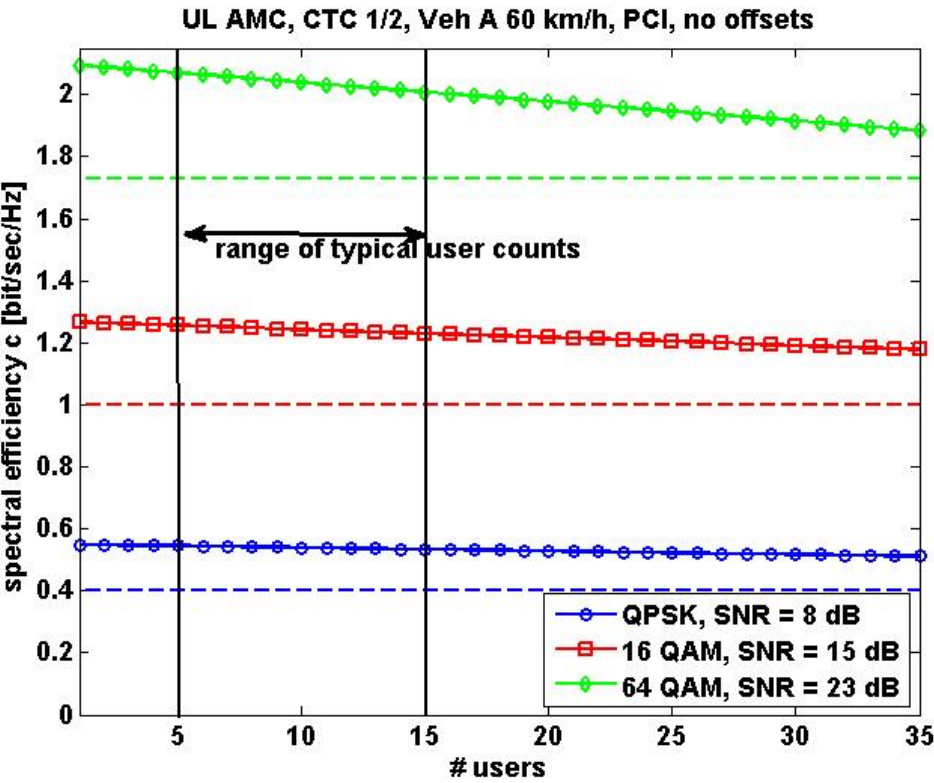


Figure 23: Spectral efficiency for uplink AMC, Veh A 60 km/h, perfect channel knowledge, multi user case.

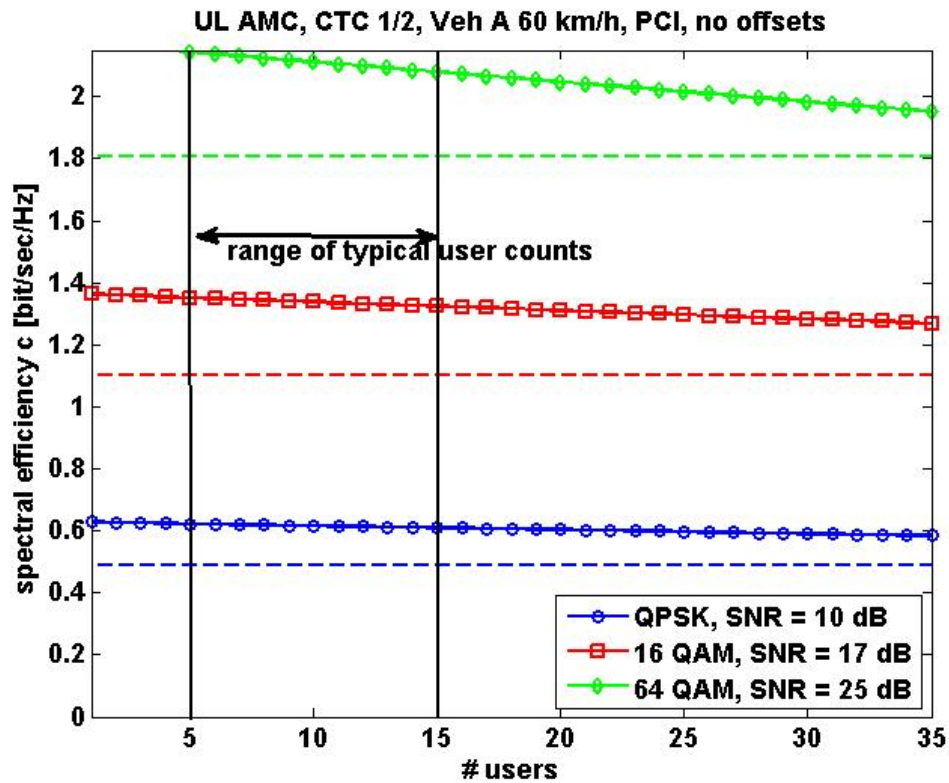


Figure 24: : Spectral efficiency for uplink AMC, Veh A 60 km/h, perfect channel knowledge, multi user case.

The dashed lines are the spectral efficiencies of the respective WiMAX system. As indicated above the more users are present, the more the spectral efficiency is reduced if FBMC is applied. However, even with a high number of scheduled users in the range of 35 (rather unlikely) the FBMC system outperforms the WiMAX system (range of typical user counts: 5...15).

If pilot based channel estimation with linear interpolation is applied:

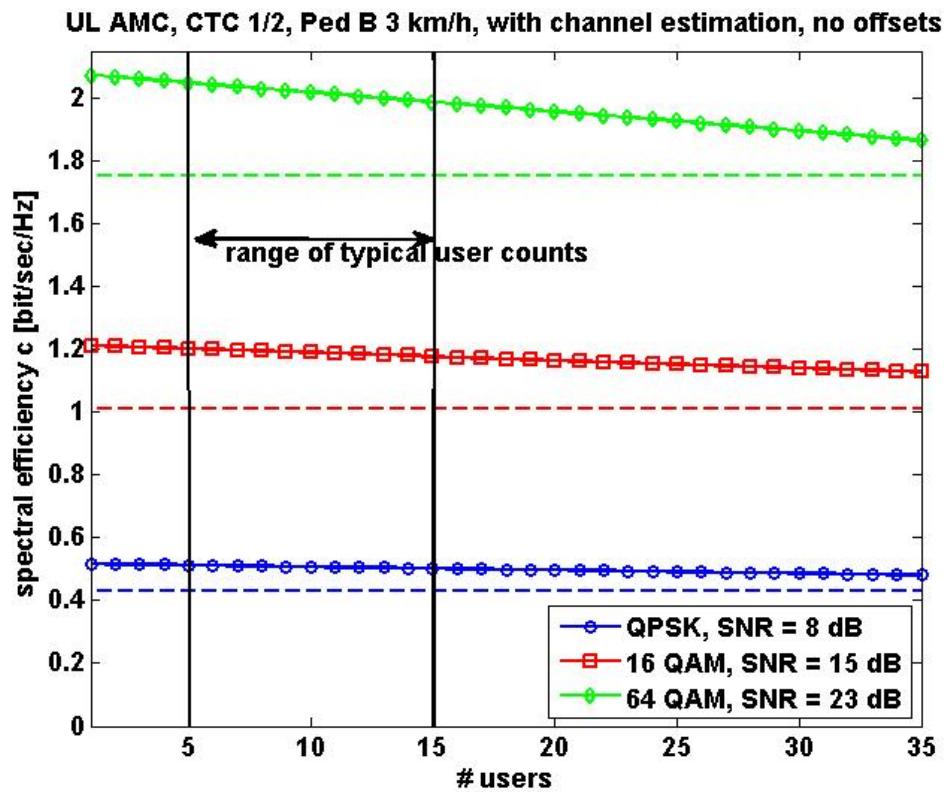


Figure 25: Spectral efficiency for uplink AMC, Ped B 3 km/h, real channel estimation, multi user case.

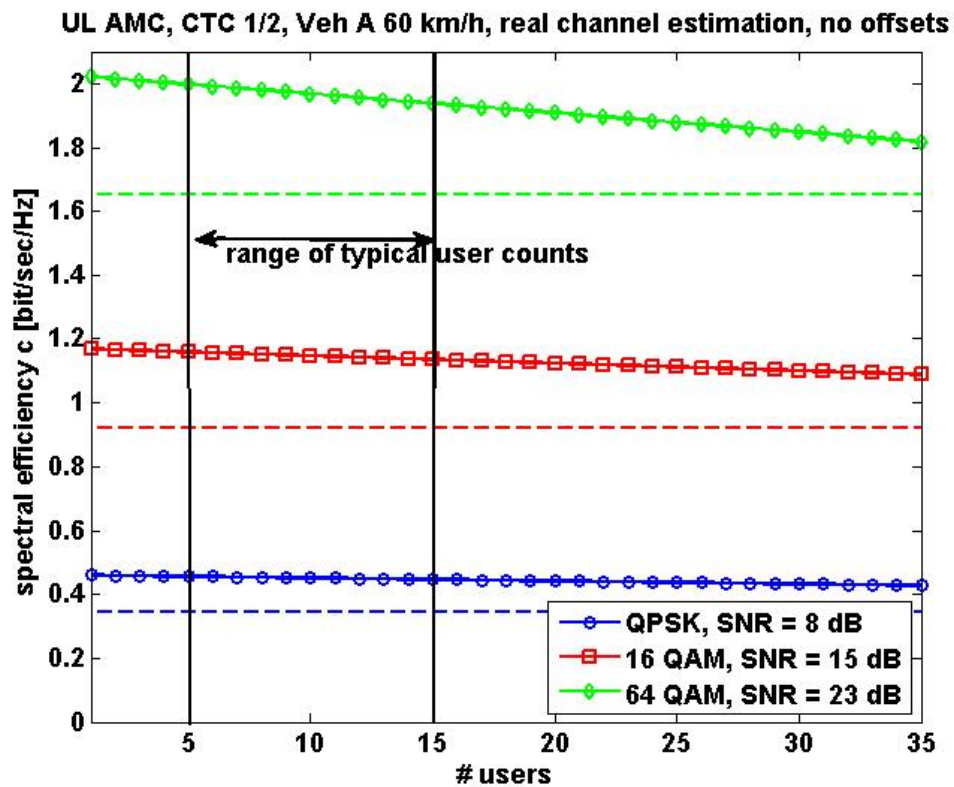


Figure 26: Spectral efficiency for uplink AMC, Veh A 60 km/h, real channel estimation, multi user case.

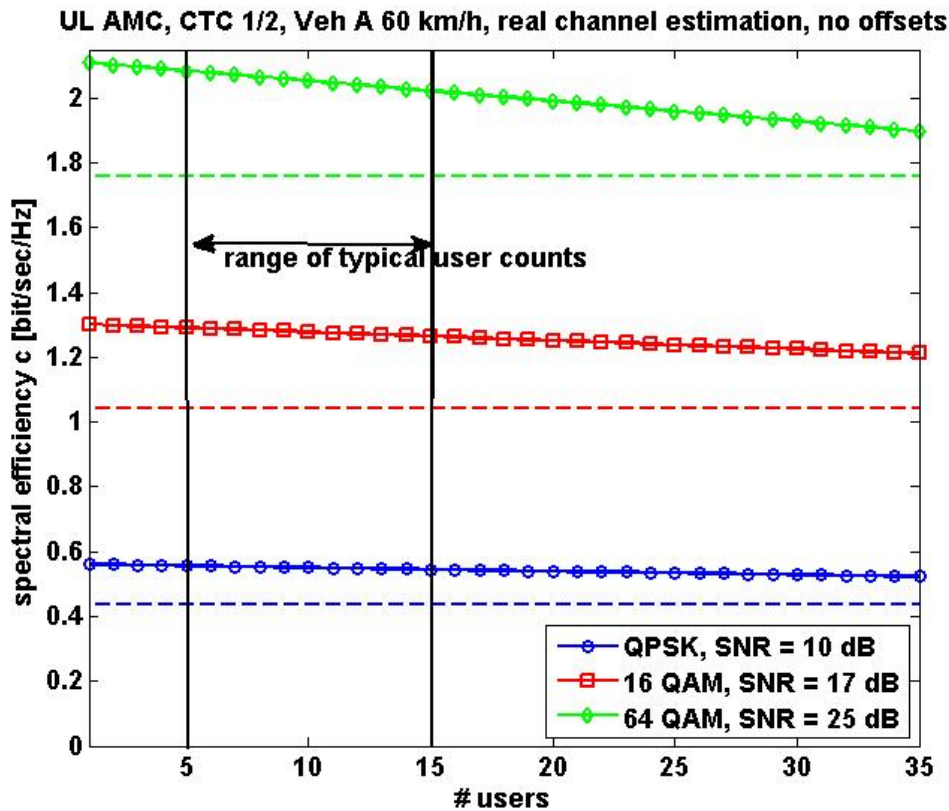


Figure 27: Spectral efficiency for uplink AMC, Veh A 60 km/h, real channel estimation, multi user case.

FBMC obviously outperforms OFDM with respect to spectral efficiency significantly.

### 1.3.2 Synchronization

#### 1.3.2.1 Pilot based synchronization (downlink PUSC)

In [2] the sensitivity to frequency (CFO, carrier frequency offset normalized to the subcarrier spacing) and timing offset (FTD, fractional time delay normalized to the FBMC symbol length, i.e.  $M/2$ ) when using pilot based synchronization in the case of uplink transmission was investigated. The downlink case (PUSC) is presented in the following. Again the linear phase slopes are determined via linear regression using the estimated phases at the pilot positions as supporting points.

Due to the fact that in downlink typically all pilots are modulated much more supporting points can be used (in uplink the basestation naturally only can use the pilots transmitted from a single user to estimate its offsets). This way the estimation gets much more reliable. The estimation error due to noise and the channel is significantly reduced. The following histograms indicate this:



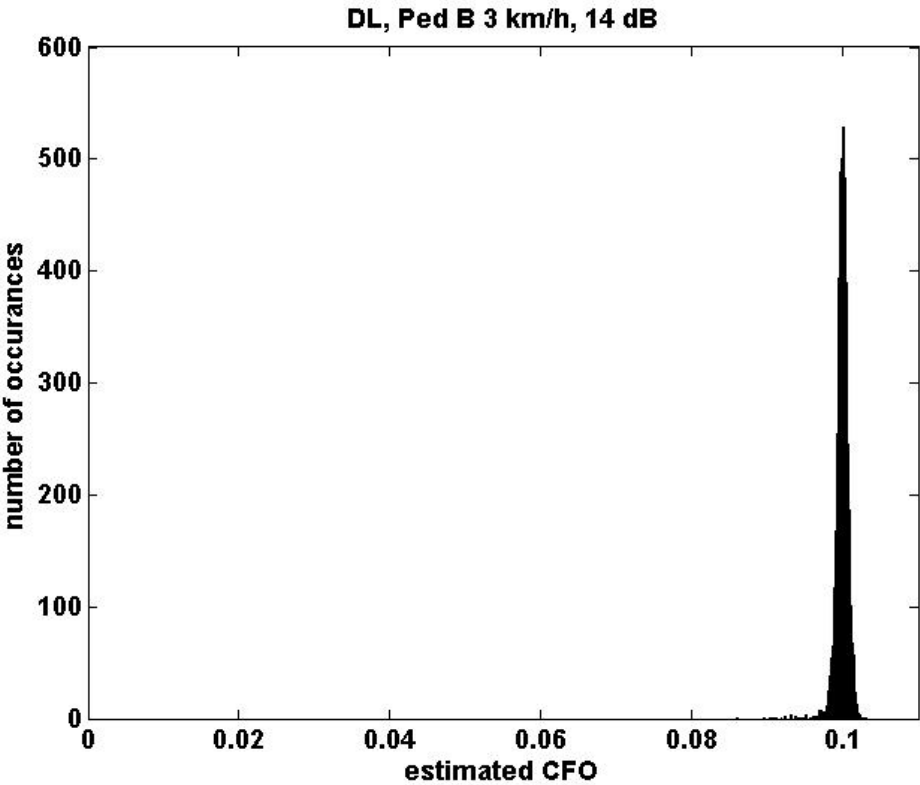


Figure 28: Histogram of CFO estimates (DL PUSC).

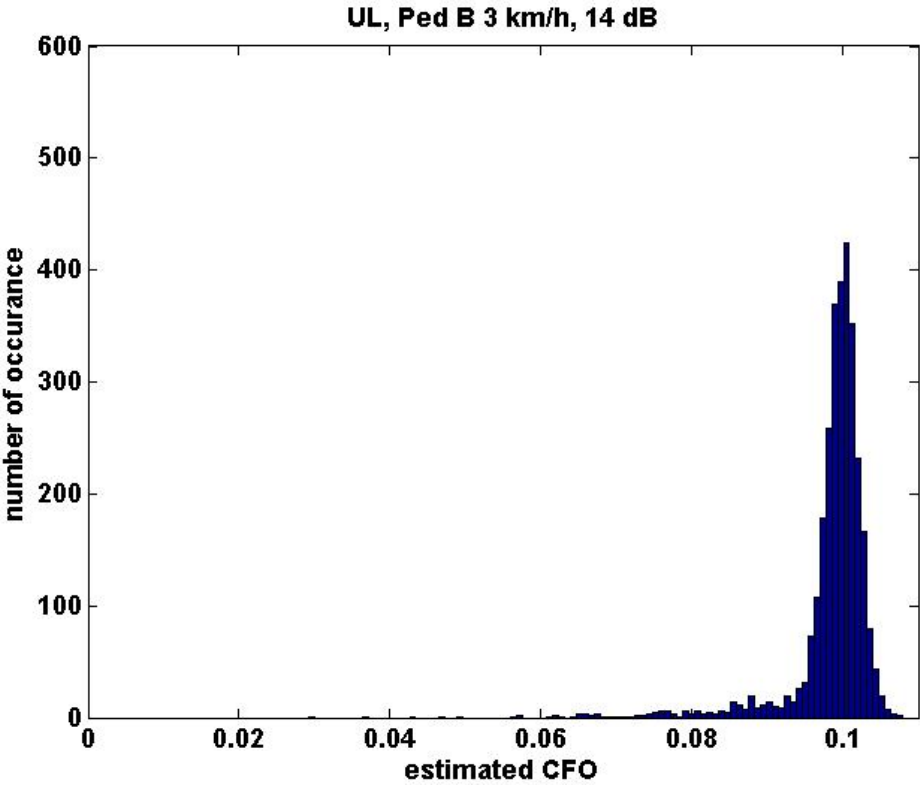


Figure 29: Histogram of CFO estimates (UL AMC).

As a measure for the sensitivity to CFO and FTD again the 3 dB tolerance with respect to a PER of 1% is evaluated. The following figures depict the required SNR to achieve this PER. The blue horizontal line within the figures displays a 3 dB offset relative to the perfectly synchronized system. This way the amount of tolerable CFO and FTD can be determined when spending 3 dB link budget to this matter. The last diagrams within the respective sections depict this tolerance. Channel estimation is done via linear interpolation using the pilot estimates as supporting points. Channel compensation in FBMC mode again is performed with the help of 3 taps using the MSE principle [10]. The common simulation parameters are as defined in [2].

## ● QPSK

First FBMC without dedicated offset compensation to determine the inherent sensitivity:

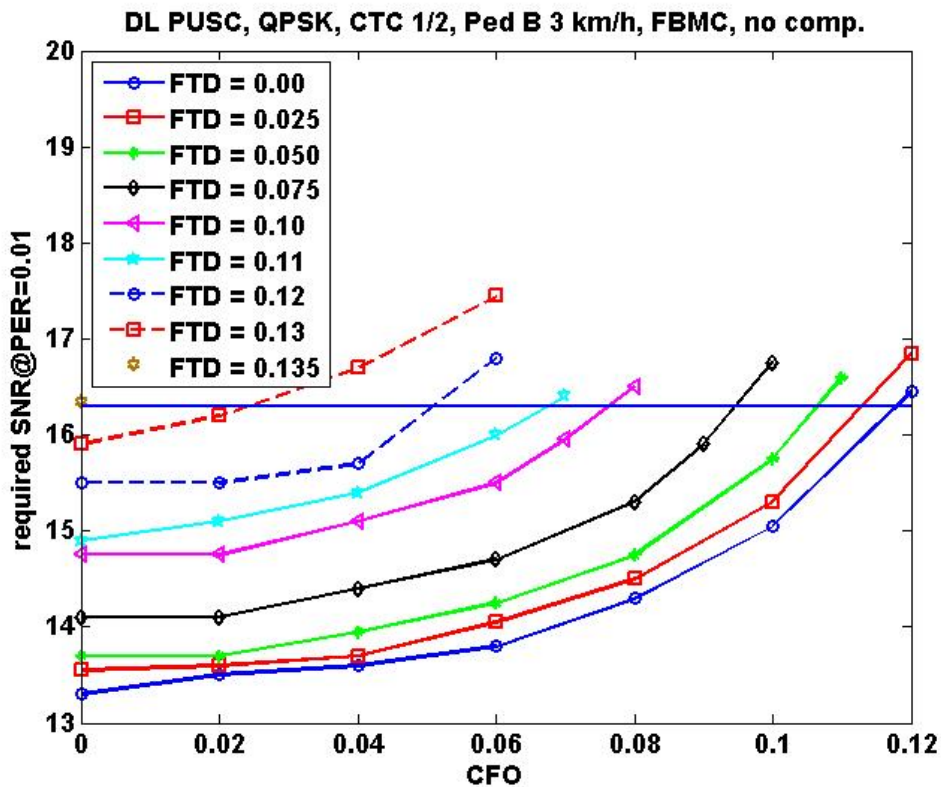


Figure 30: required SNR to achieve PER=1%, no dedicated offset compensation



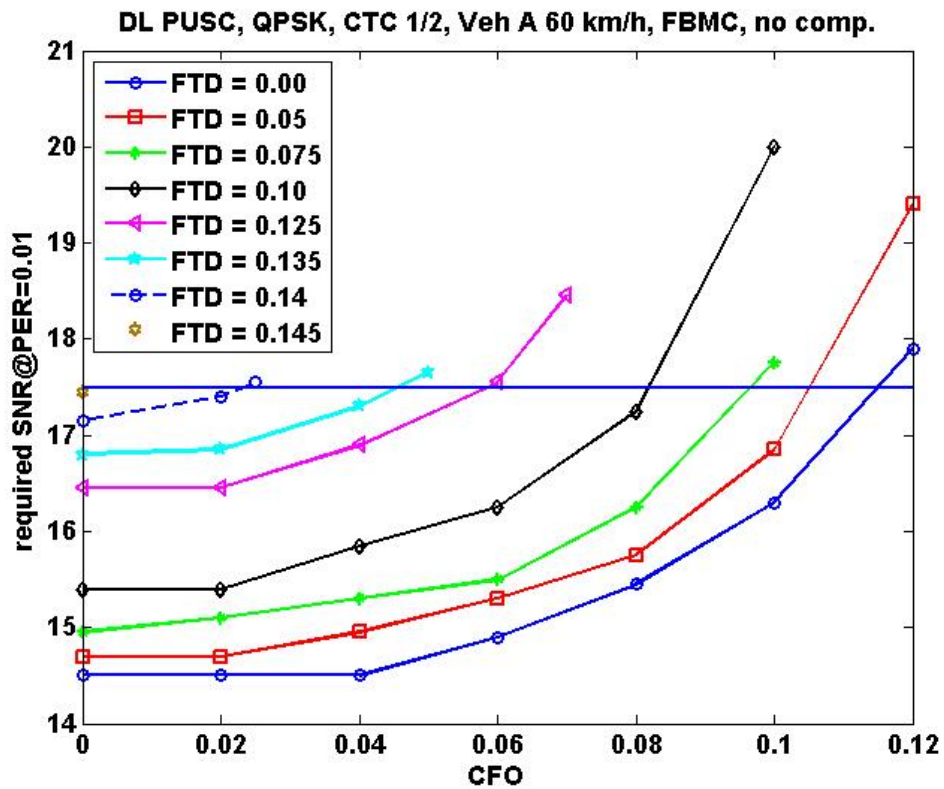


Figure 31: required SNR to achieve PER=1%, no dedicated offset compensation

When using OFDM, again omitting compensation:

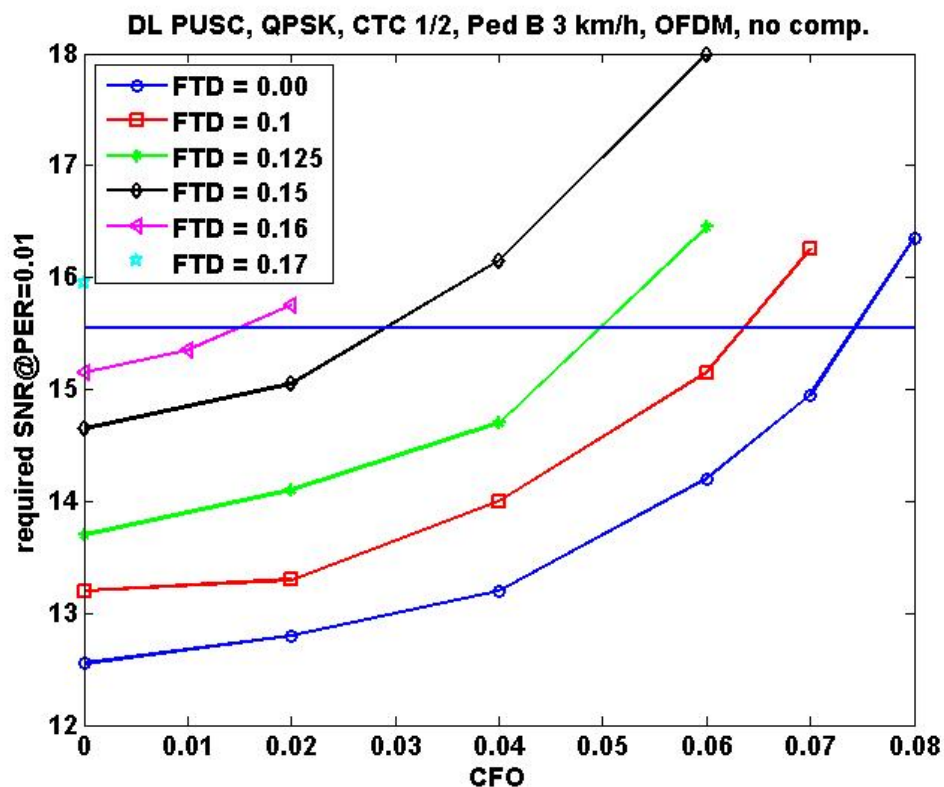


Figure 32: required SNR to achieve PER=1%, no dedicated offset compensation

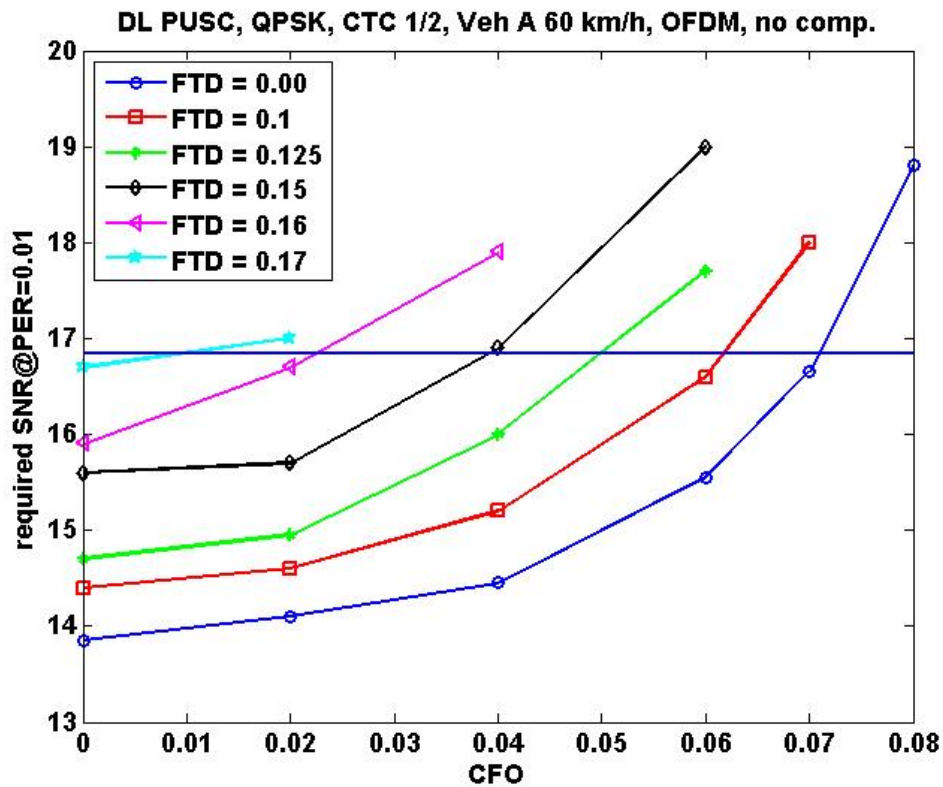


Figure 33: required SNR to achieve PER=1%, no dedicated offset compensation

Now FBMC mode with offset compensation:

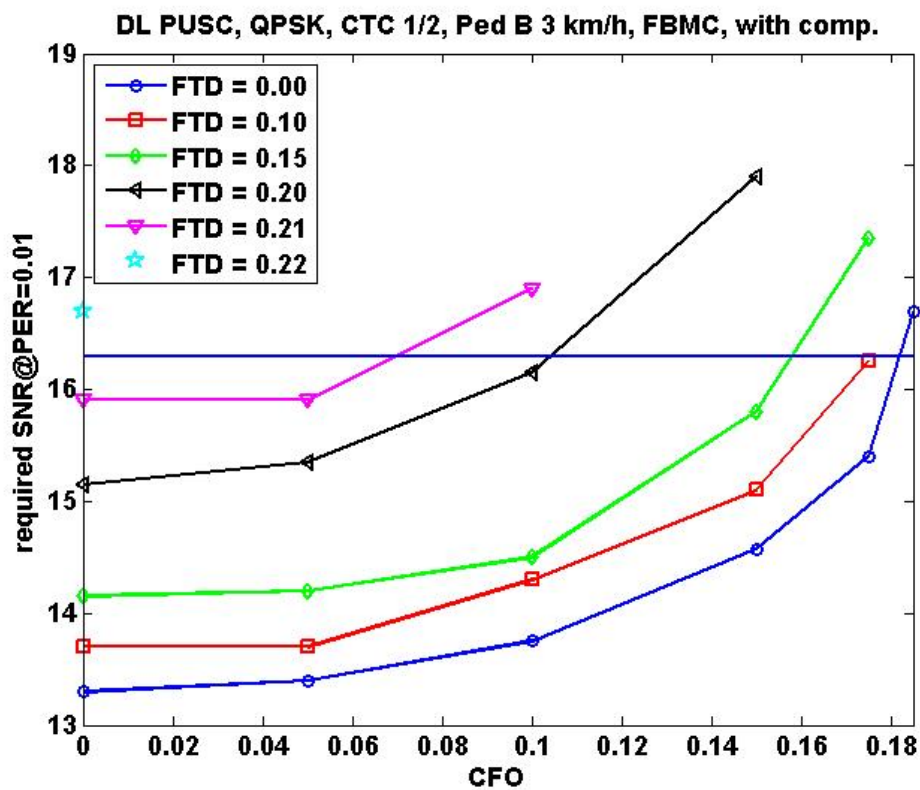


Figure 34: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

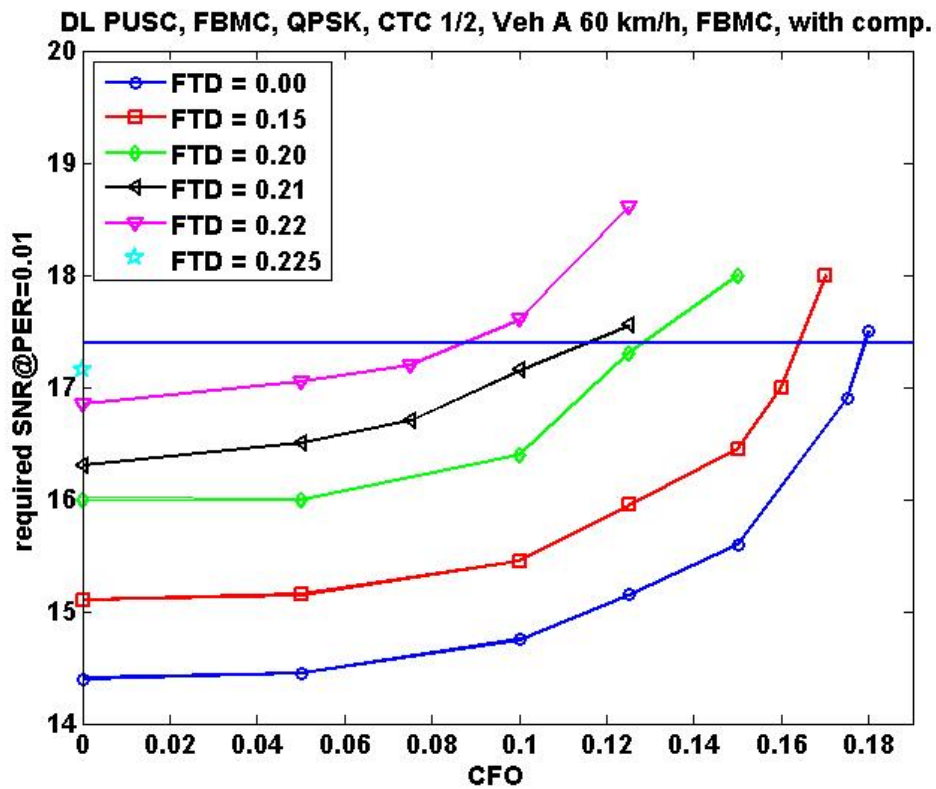


Figure 35: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

OFDM mode with compensation:

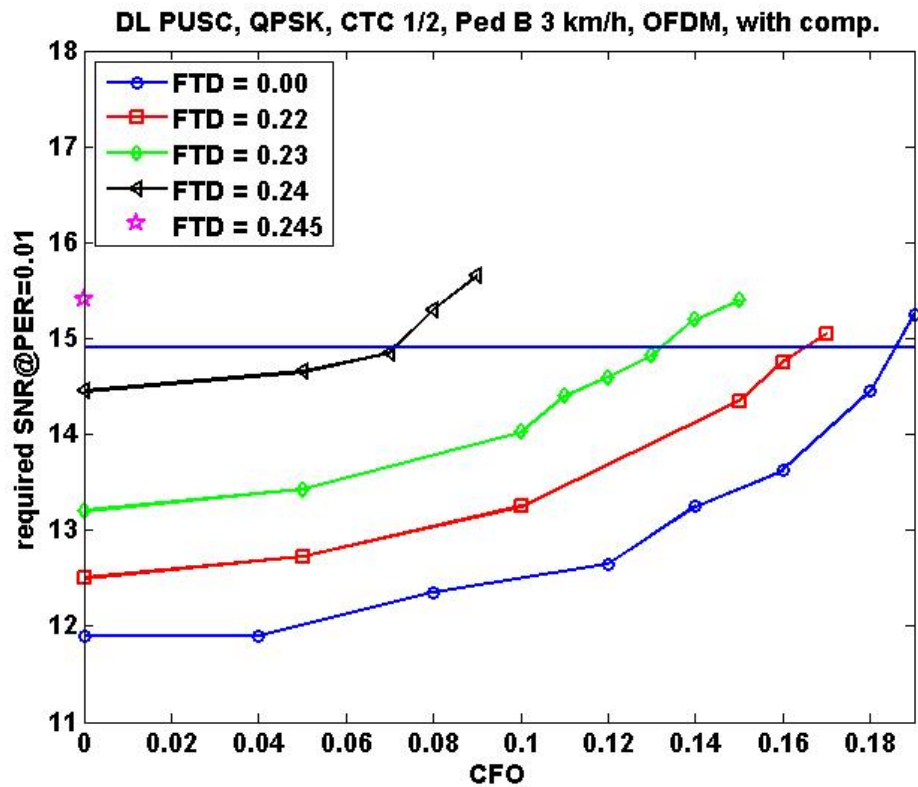


Figure 36: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

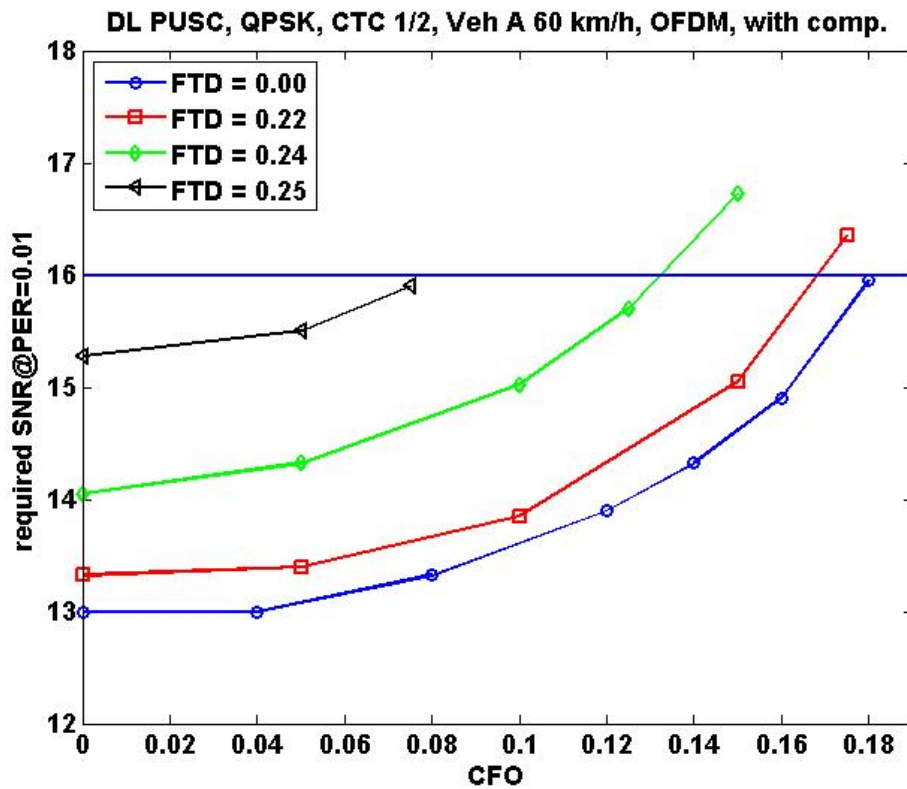


Figure 37: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

Up to this point the pilot power in FBMC mode is shared between the main pilot and the auxiliary pilot, leading to a lower main pilot power compared to the OFDM mode. Naturally estimation accuracy suffers due to the lower pilot SNR. To capture this influence, simulations were done when spending the same power to the main pilot in FBMC mode as it is done in OFDM mode:

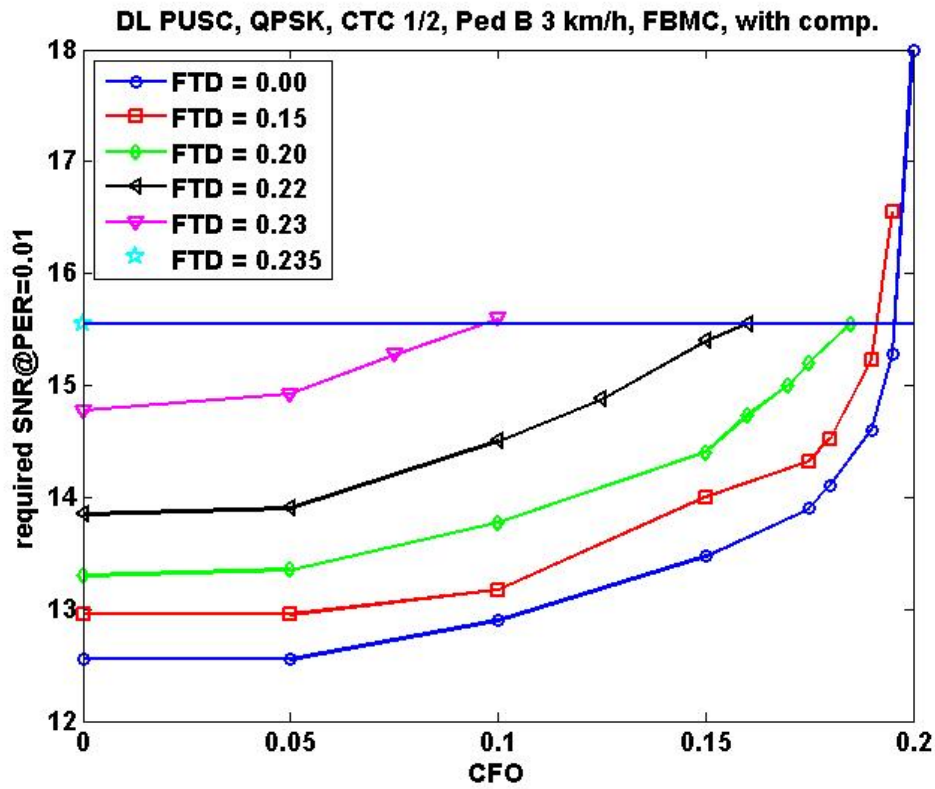


Figure 38: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

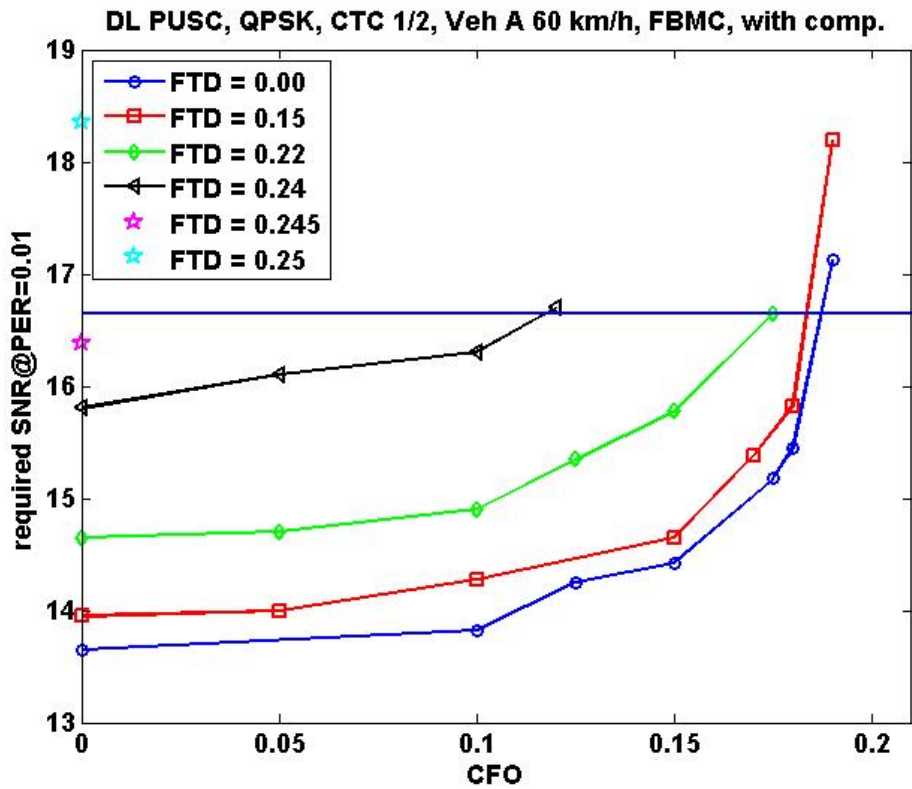


Figure 39: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

Thus the 3 dB tolerances with respect to CFO and FTD are:

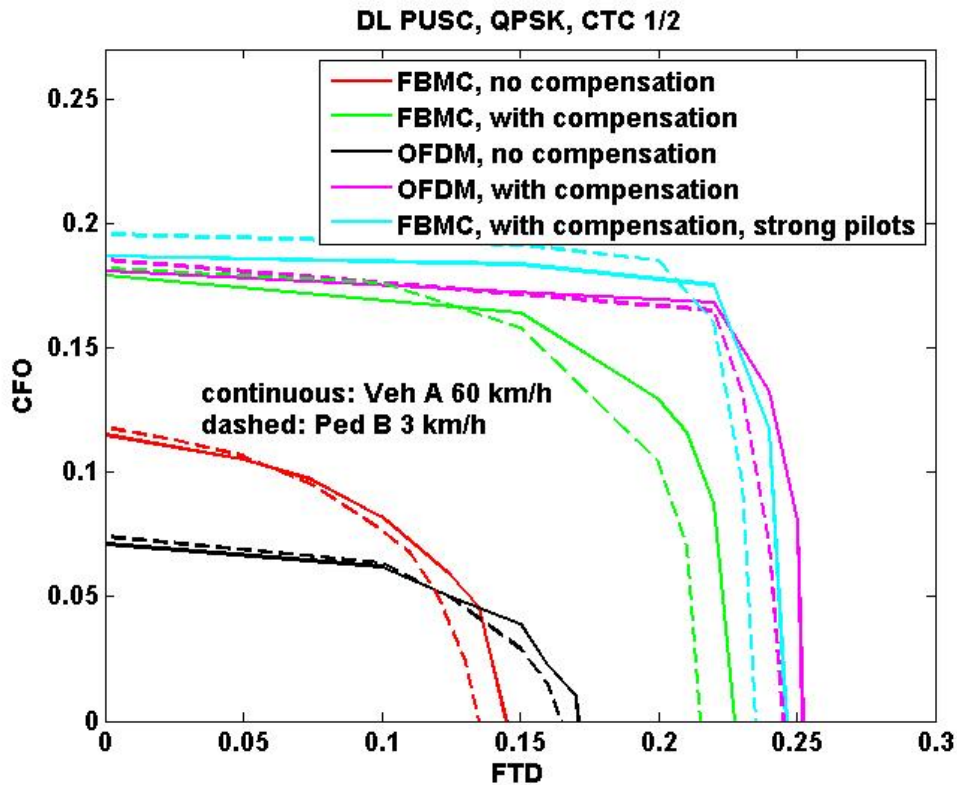


Figure 40: CFO/FTD tolerances with 3 dB link budget dedicated to desynchronization.

As to be expected when applying FBMC CFO sensitivity is reduced, due to the lower out-of-band leakage. Regarding FTD sensitivity OFDM slightly outperforms FBMC mode, however, with the help of a cyclic prefix. The figure shows, that FBMC can approximately achieve the same performance without the need of a cyclic prefix, i.e. without wasting precious resources. Again Ped B at 3 km/h is slightly more tolerant to CFO (lower Doppler) but less tolerant to FTD (higher delay spread) compared to Veh A at 60 km/h.



• 16 QAM

FBMC, no dedicated compensation:

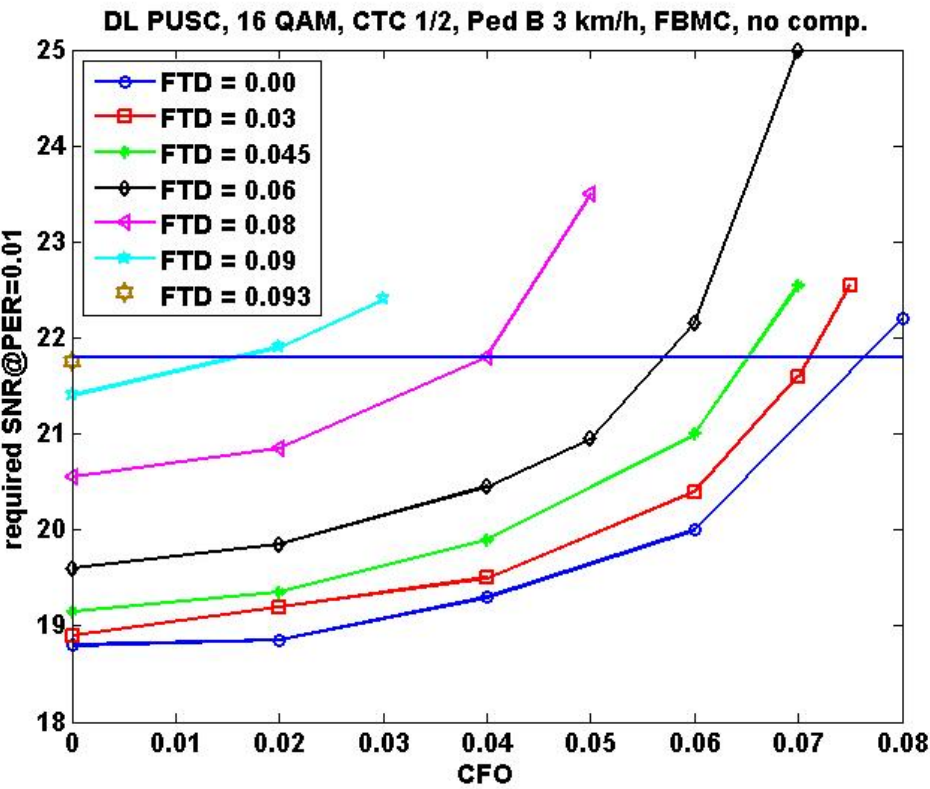


Figure 41: required SNR to achieve PER=1%, no dedicated offset compensation

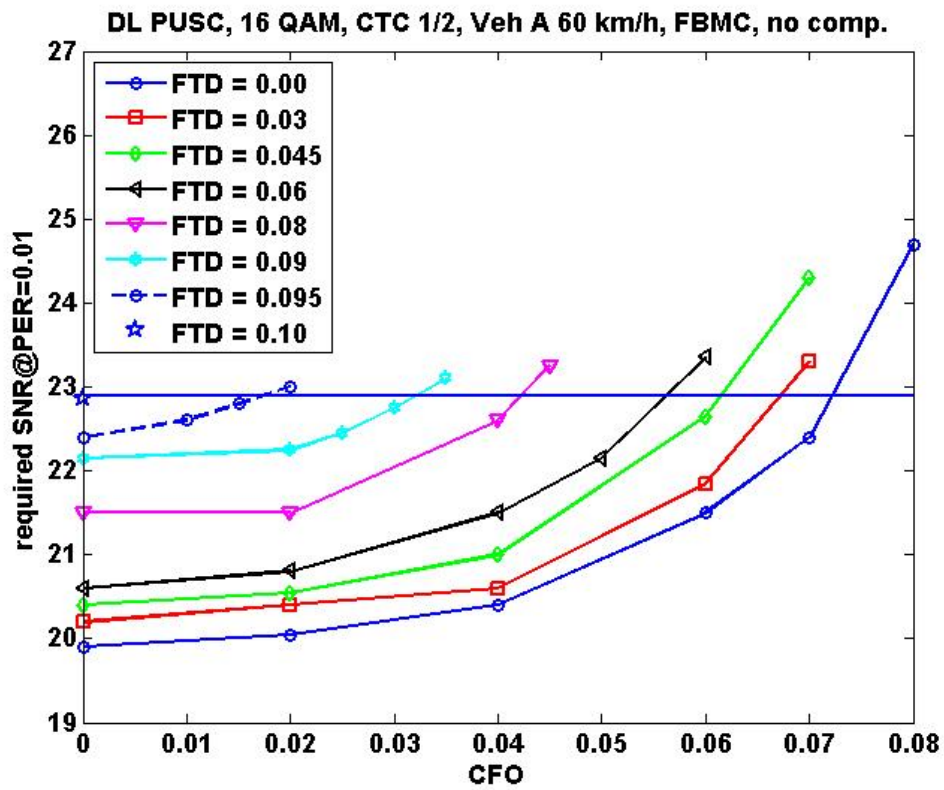


Figure 42: required SNR to achieve PER=1%, no dedicated offset compensation

OFDM, no dedicated compensation:

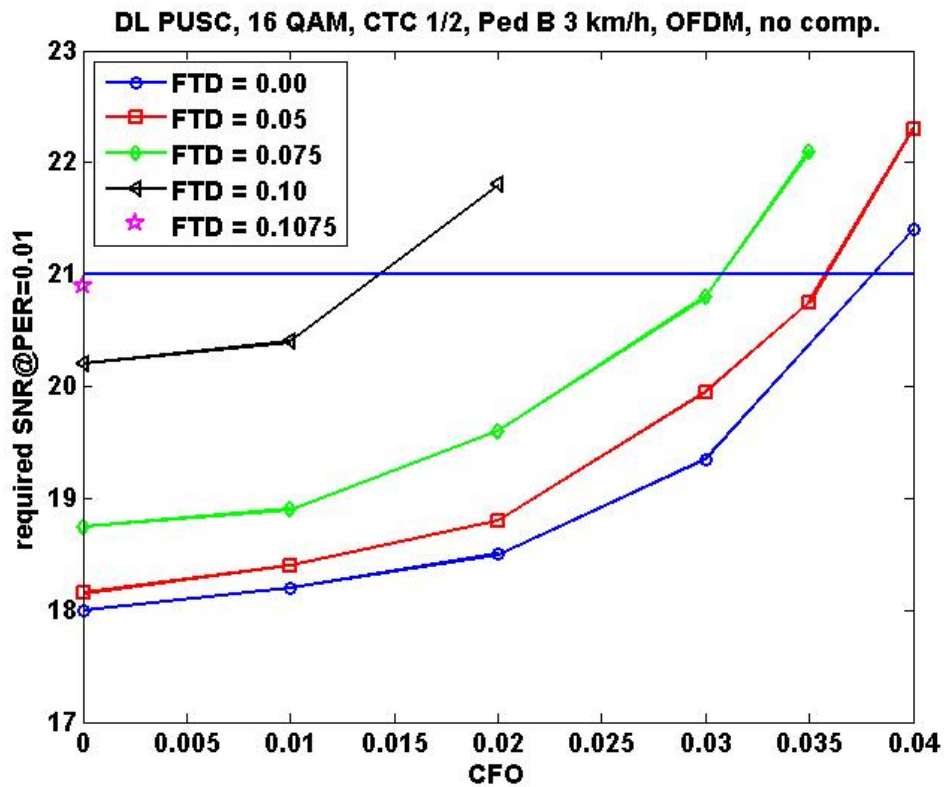


Figure 43: required SNR to achieve PER=1%, no dedicated offset compensation



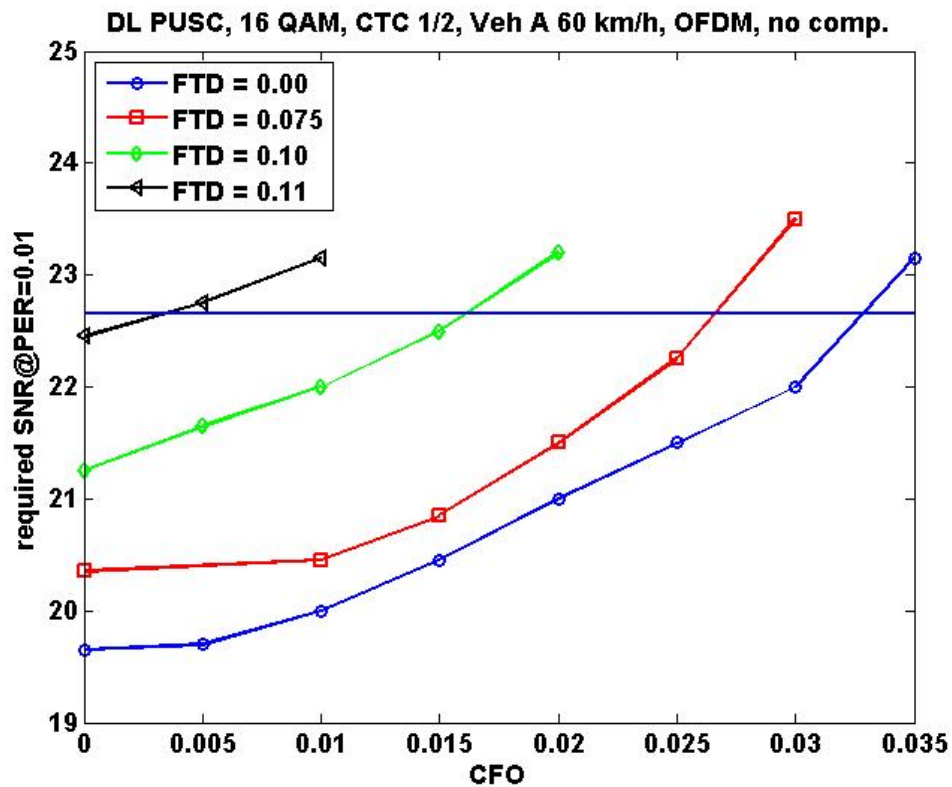


Figure 44: required SNR to achieve PER=1%, no dedicated offset compensation

FBMC, with dedicated compensation:

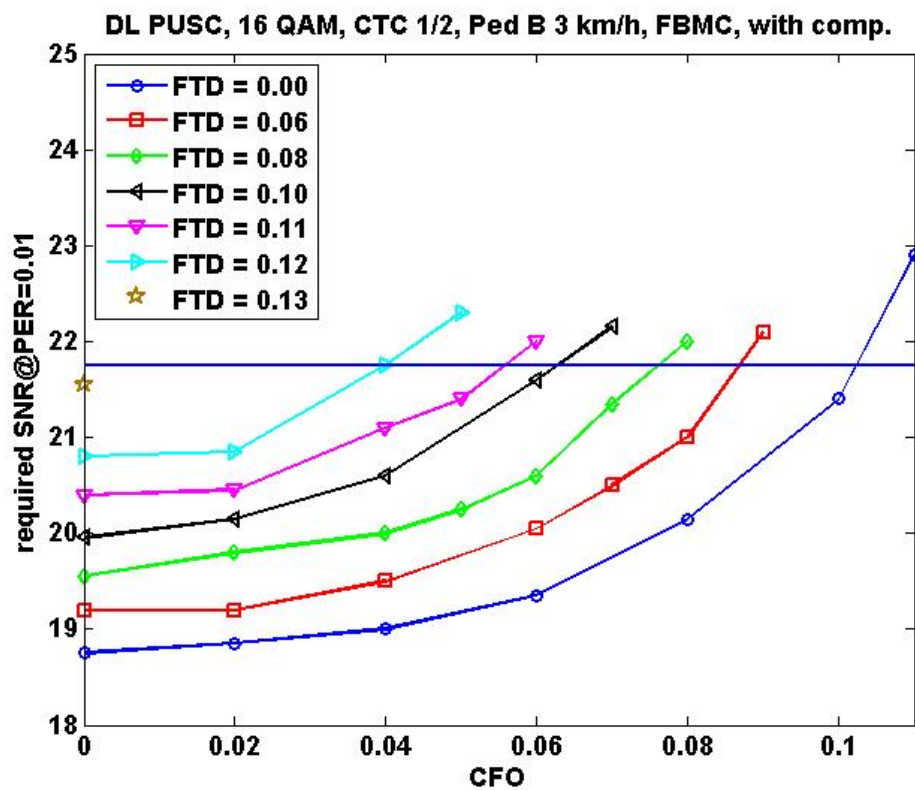


Figure 45: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

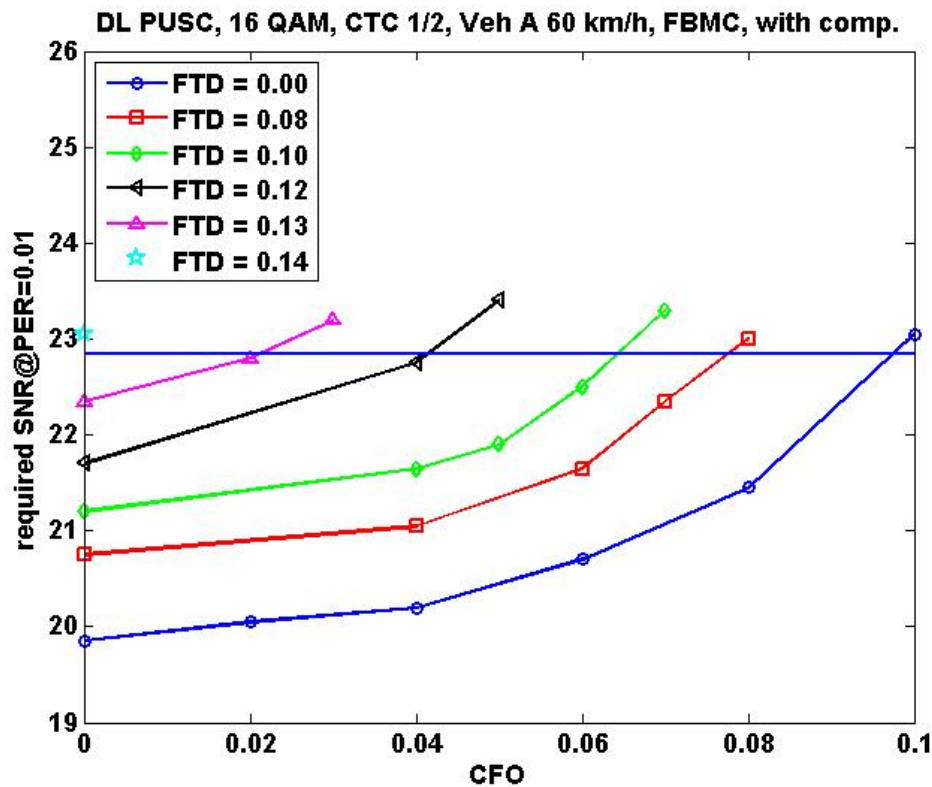


Figure 46: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

OFDM, with dedicated compensation:

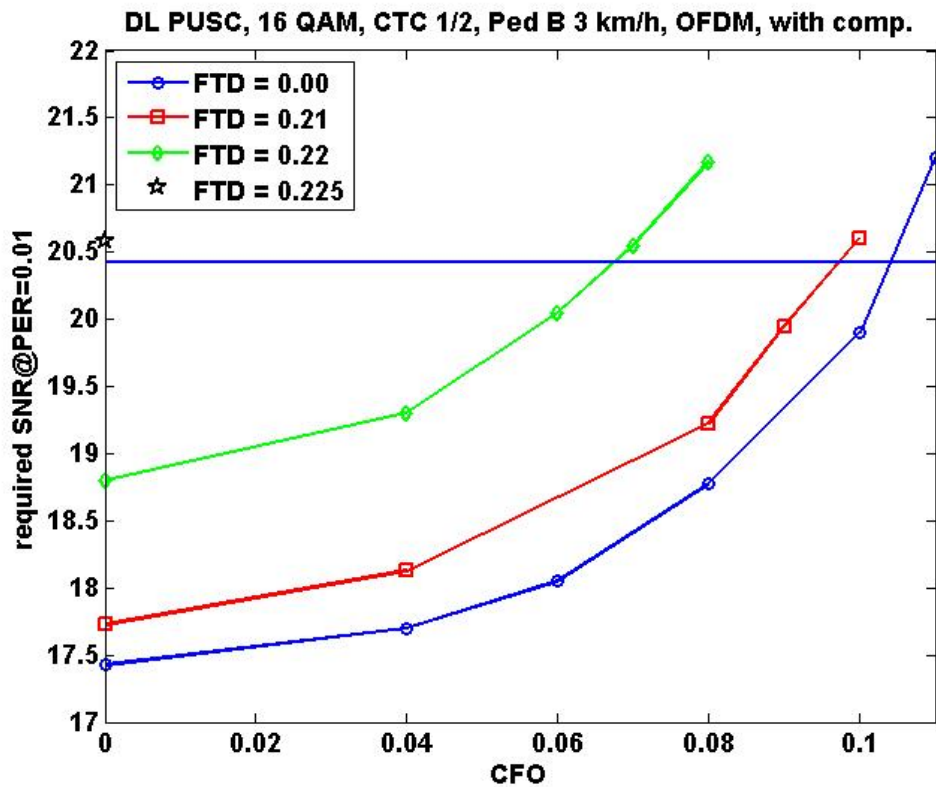


Figure 47: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

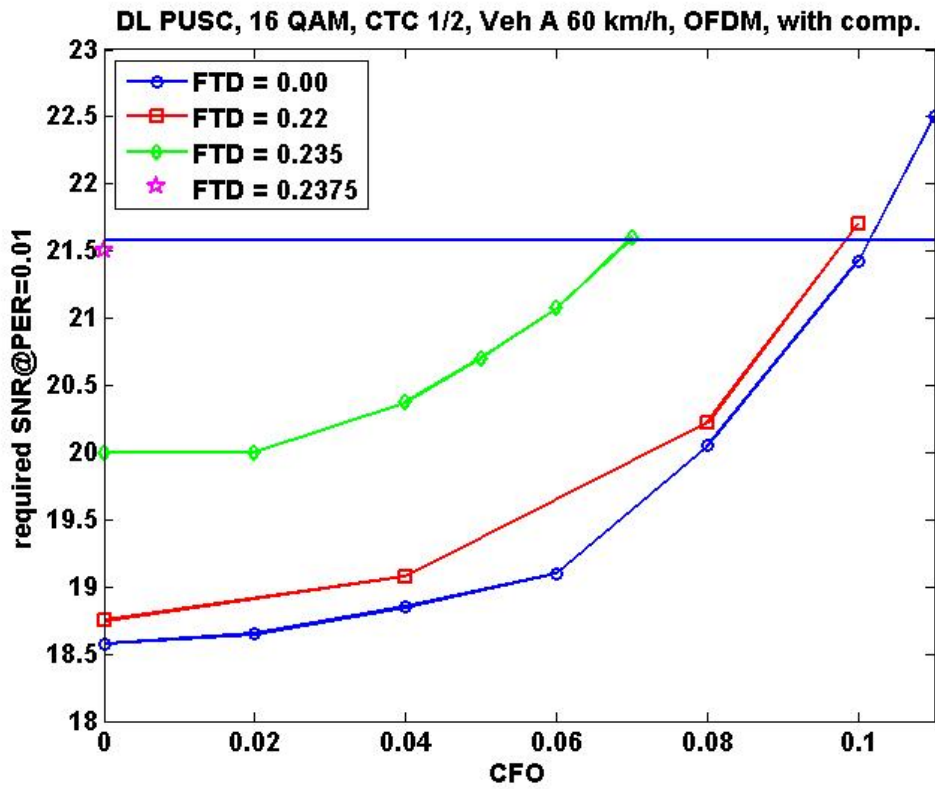


Figure 48: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

FBMC mode with compensation and the same main pilot power as in OFDM mode:

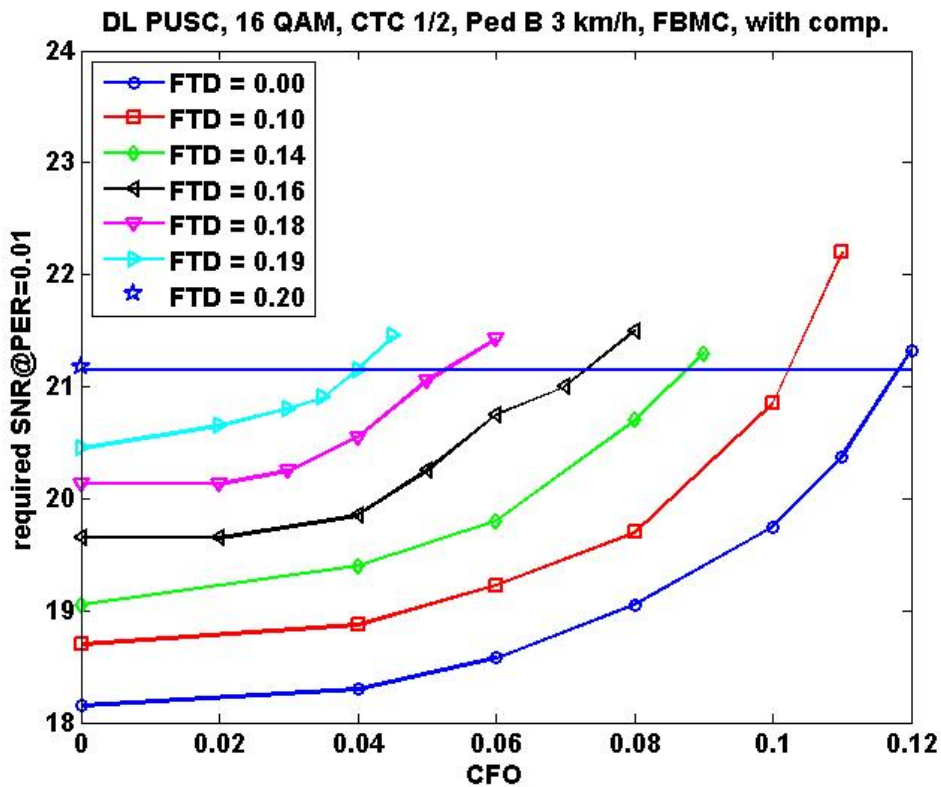


Figure 49: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

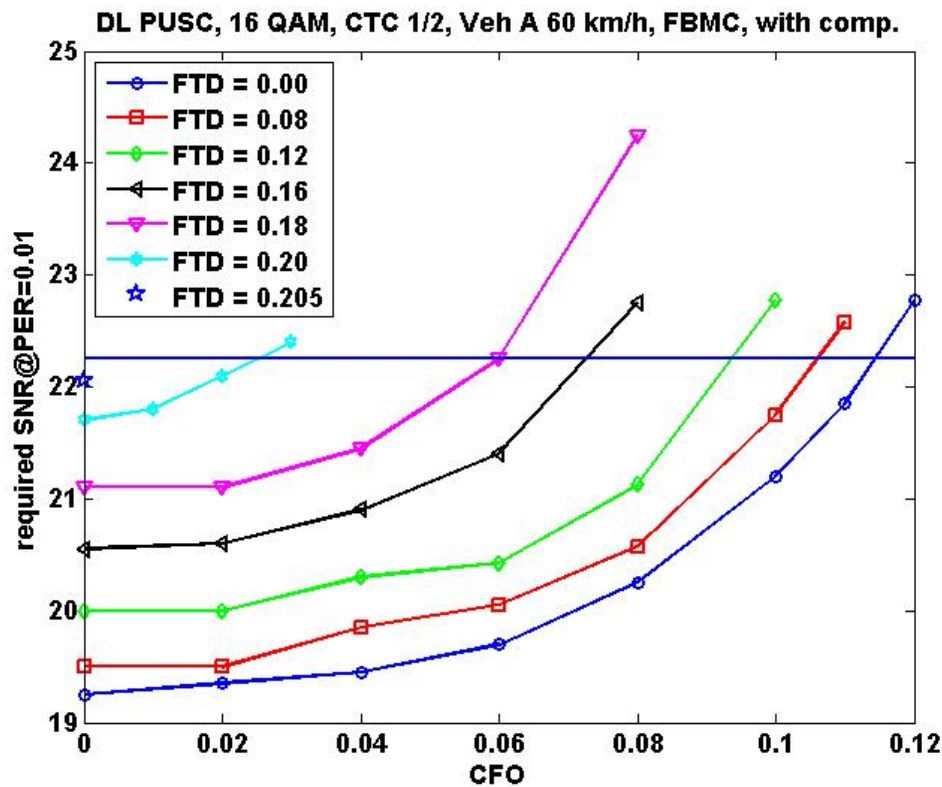


Figure 50: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

Thus the 3 dB tolerances with respect to CFO and FTD are:

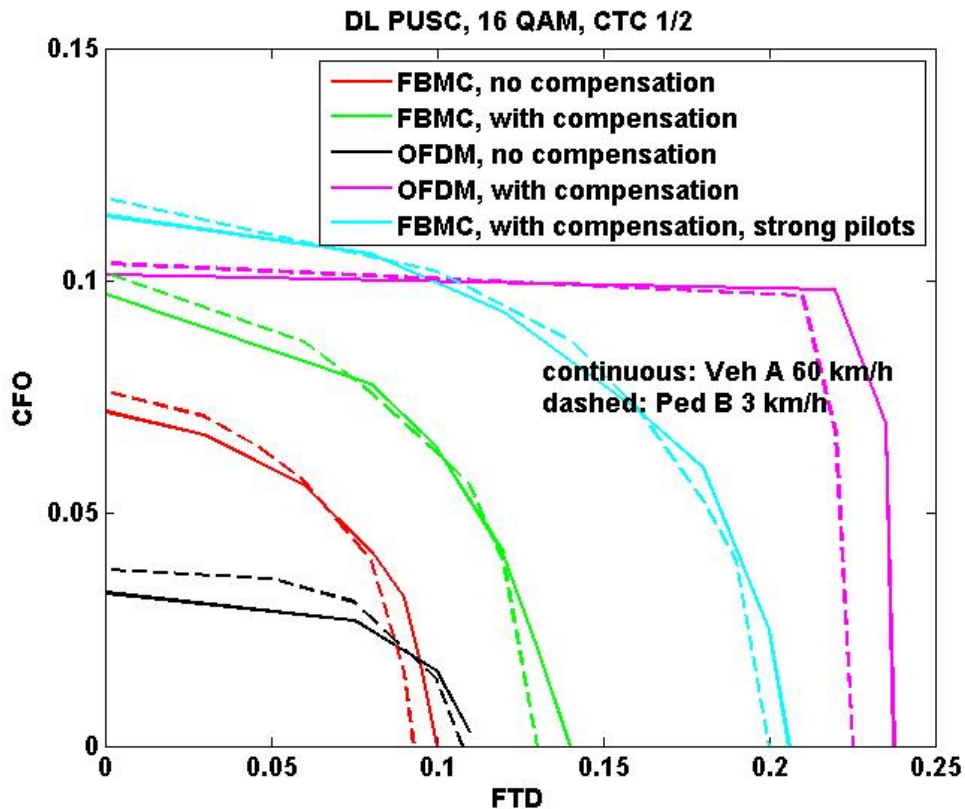


Figure 51: CFO/FTD tolerances with 3 dB link budget dedicated to desynchronization.

Again by using the filterbank the sensitivity to CFO can be reduced. Due to the fact, that the cyclic prefix avoids ISI independently of the chosen modulation order (up to a certain amount depending on the length of the CP), sensitivity to FTD for 16 QAM when applying OFDM is similar to the case of QPSK. With FBMC sensitivity degrades more due to the fact, that the higher modulation order is more sensitive to ISI than QPSK.

• 64 QAM

FBMC, no dedicated compensation:

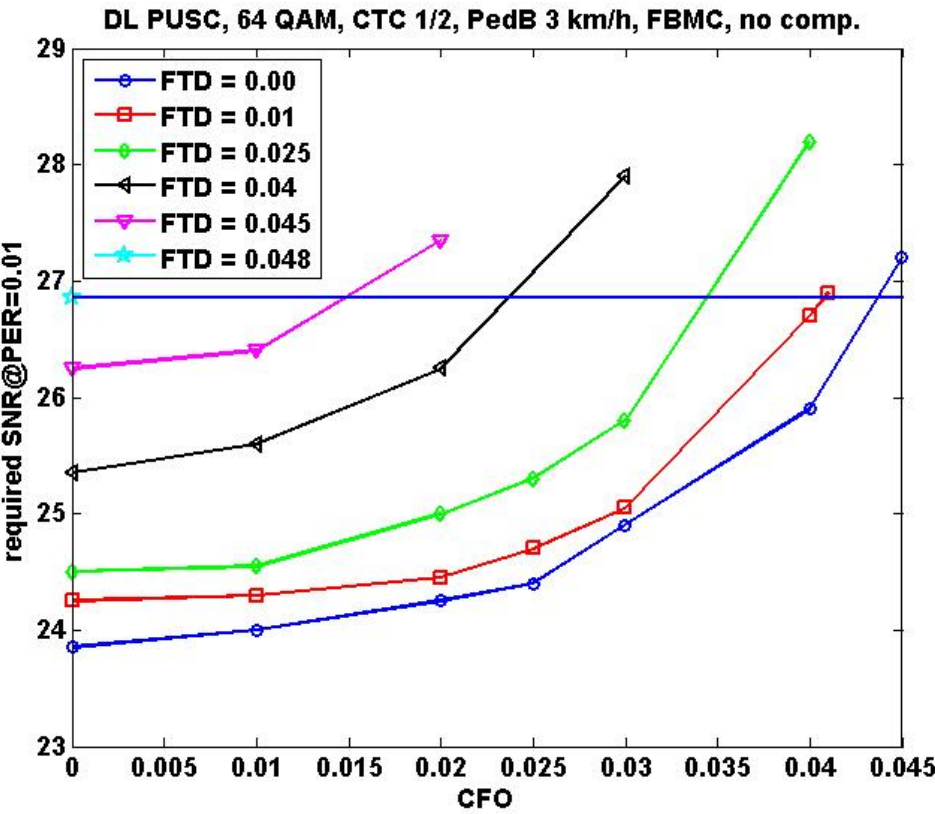


Figure 52: required SNR to achieve PER=1%, no dedicated offset compensation



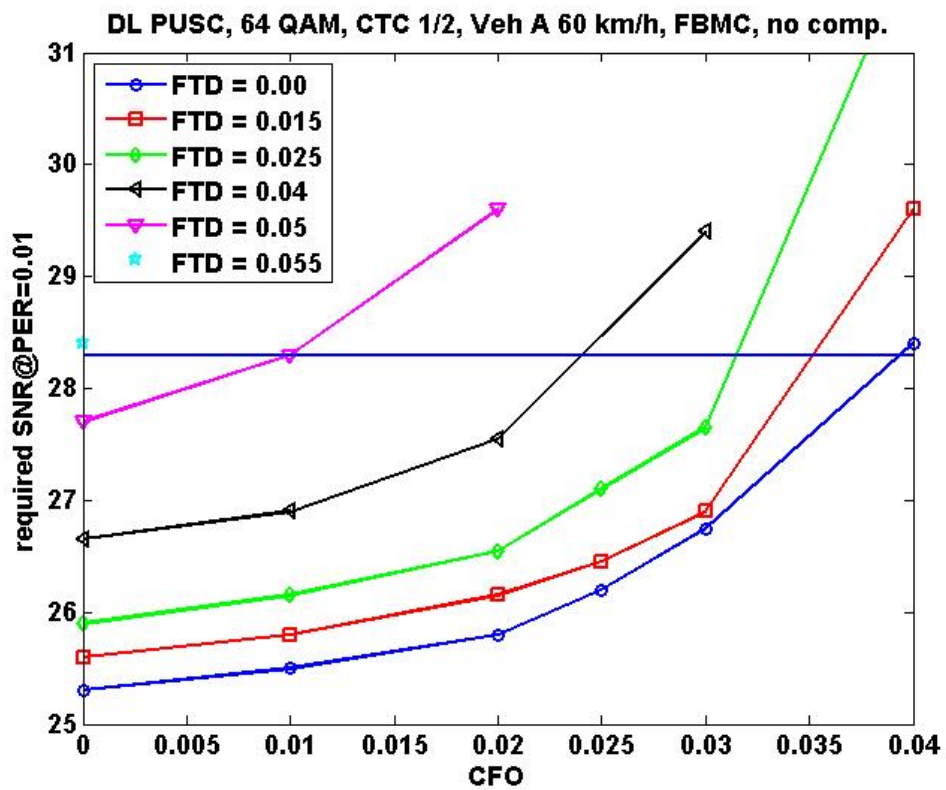


Figure 53: required SNR to achieve PER=1%, no dedicated offset compensation

OFDM, no dedicated compensation:

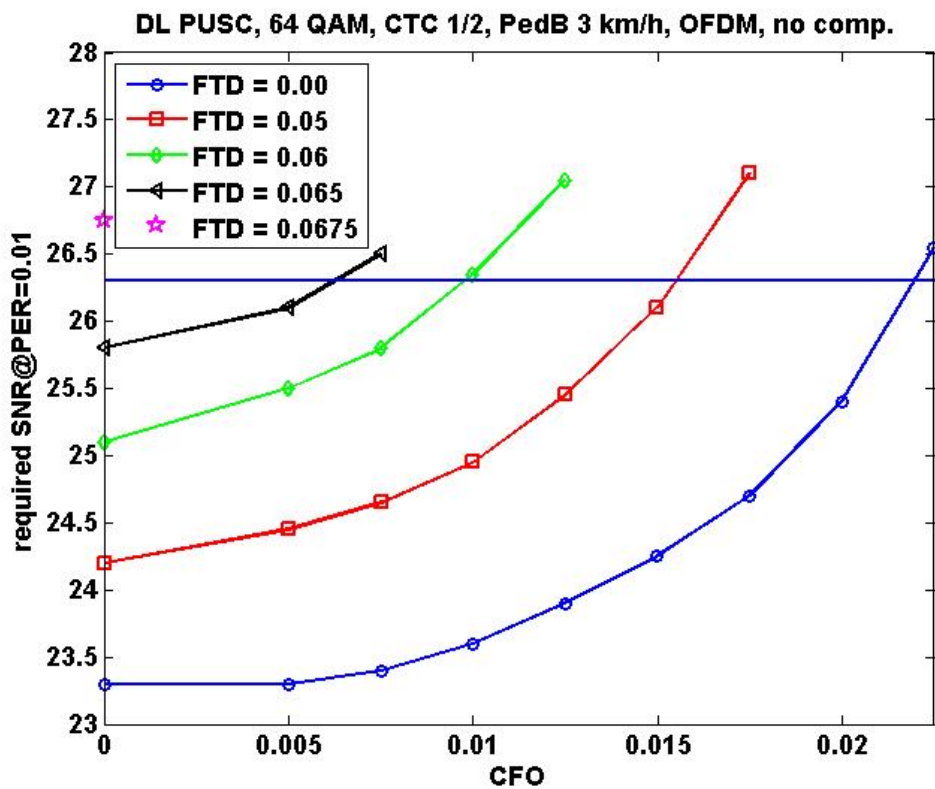


Figure 54: required SNR to achieve PER=1%, no dedicated offset compensation

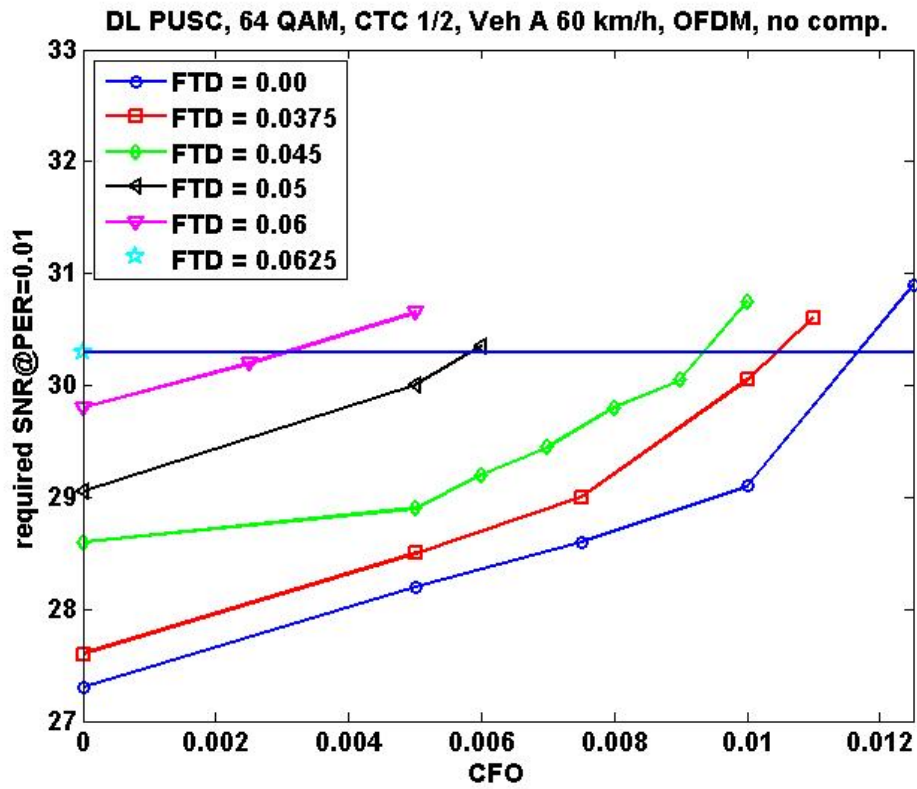


Figure 55: required SNR to achieve PER=1%, no dedicated offset compensation

FBMC, with dedicated compensation:

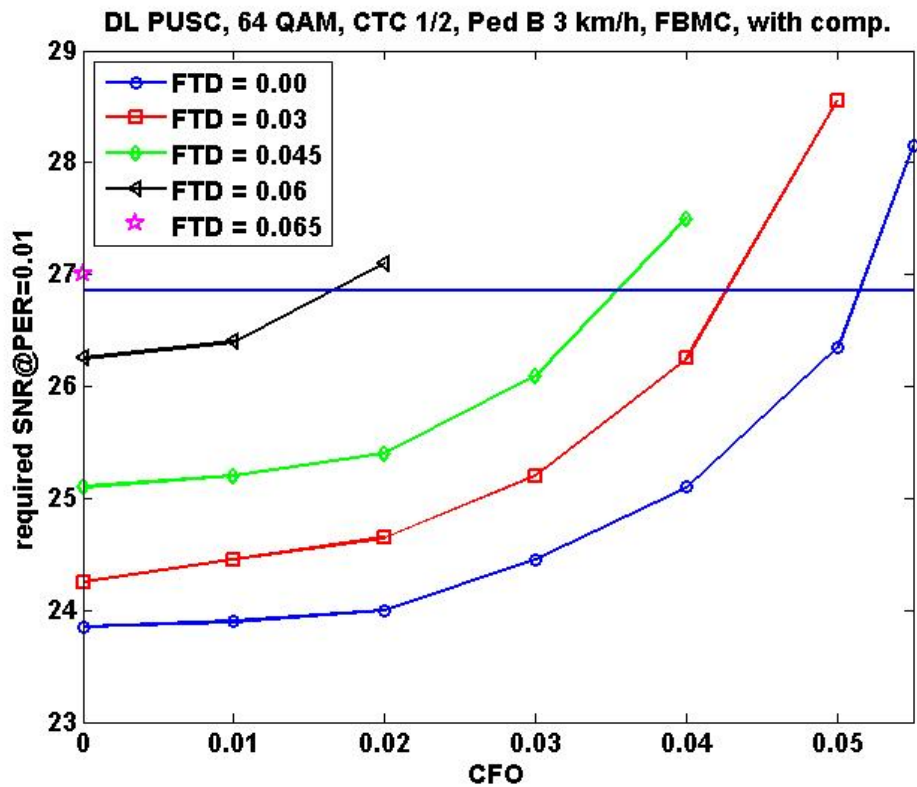


Figure 56: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)



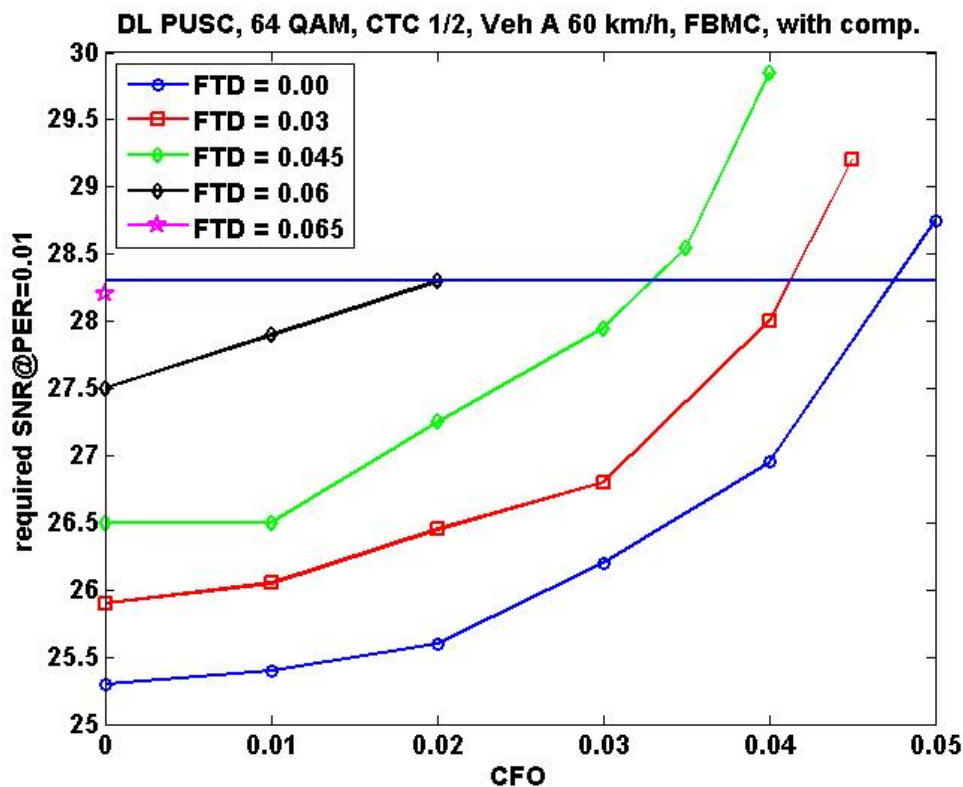


Figure 57: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

OFDM, with dedicated compensation:

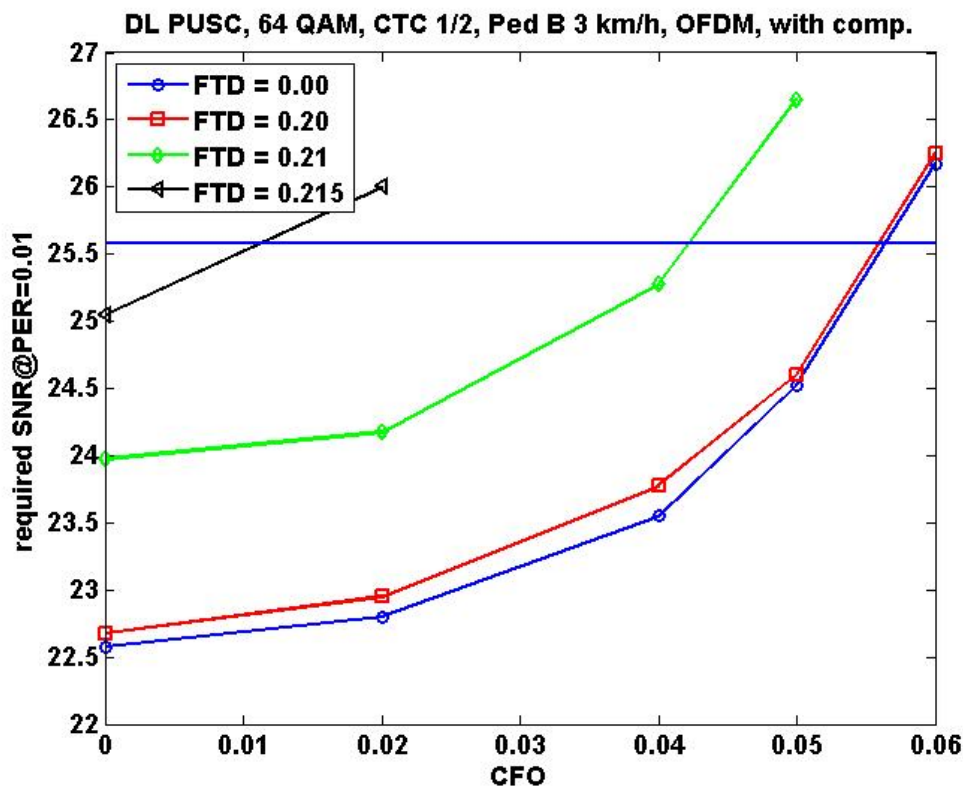


Figure 58: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

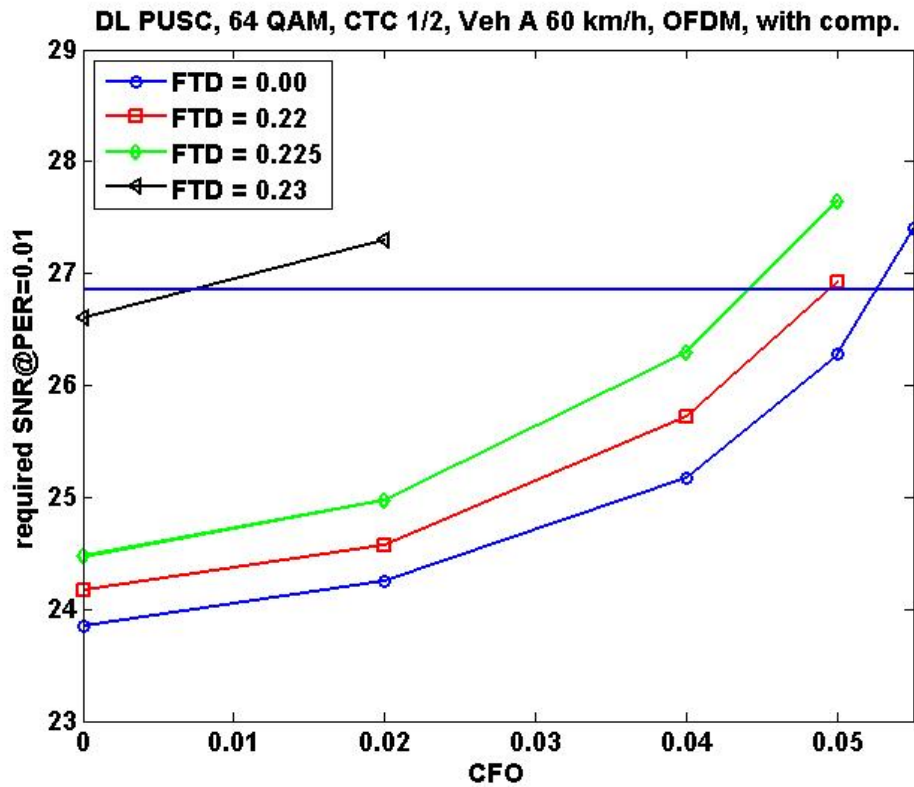


Figure 59: required SNR to achieve PER=1%, with dedicated offset compensation (regression based)

FBMC mode with compensation and the same main pilot power as in OFDM mode:

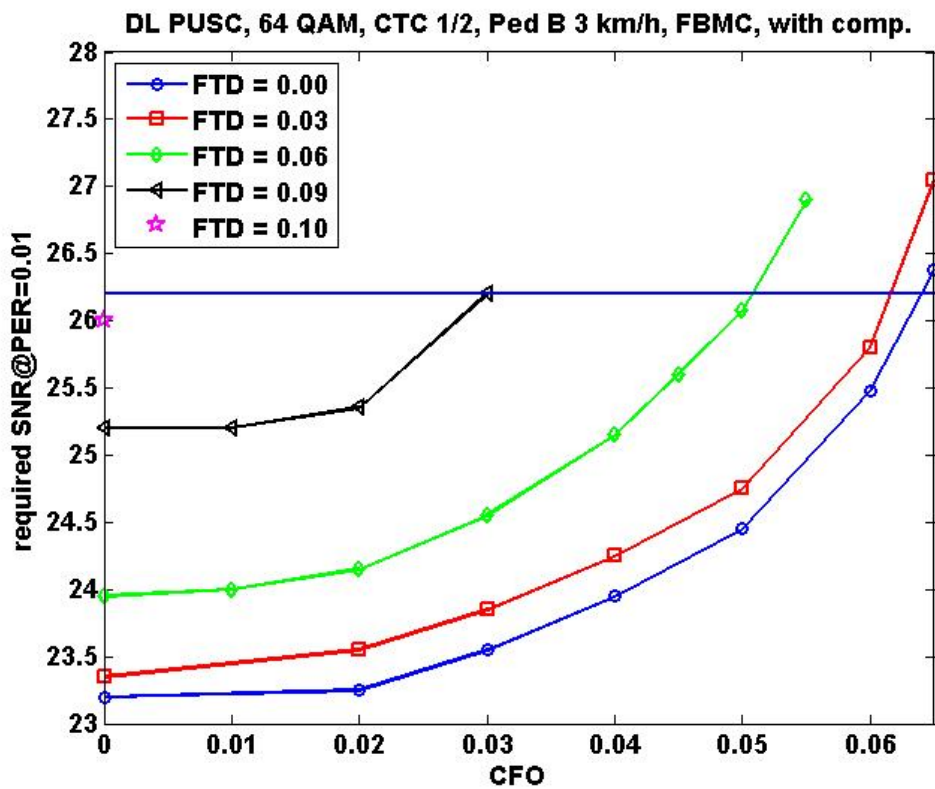


Figure 60: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

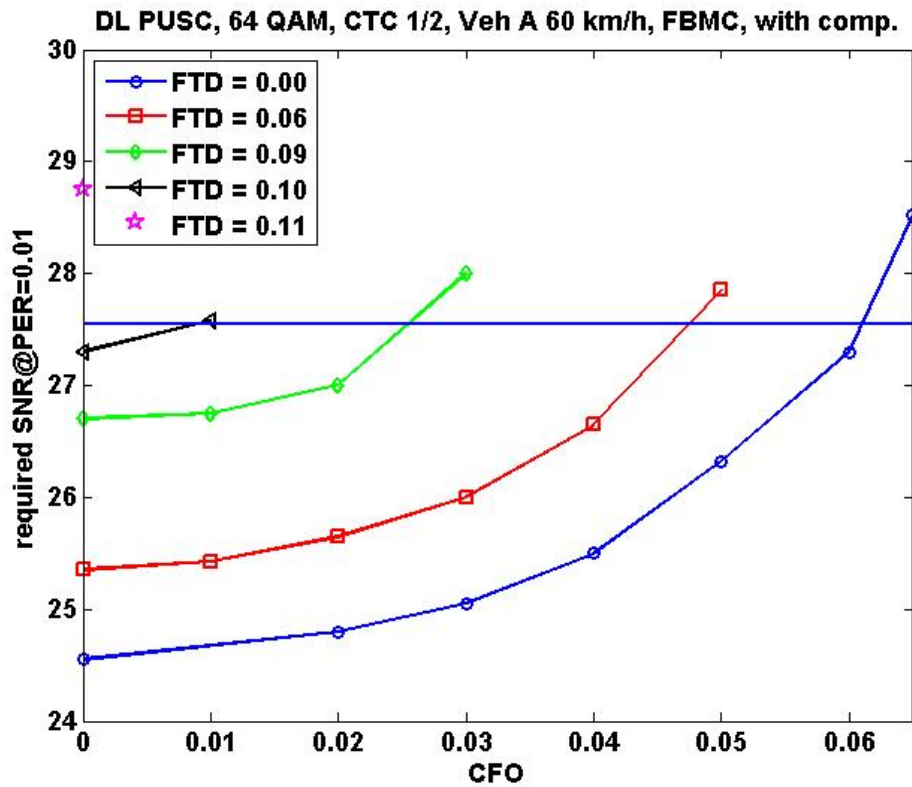


Figure 61: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots)

The compensation schemes used here estimate the carrier offset and the time delay separately. This is not optimal as the estimation of the phase slope introduced by the CFO suffers due to the phase distortion due to the FTD (FTD estimation is done after CFO estimation). In [1] an iterative approach is suggested. Here the chain of channel, timing and frequency estimation/compensation is passed through several times. Hence, the estimations are improved iteratively.

When passing through the chain two times the required SNRs are:

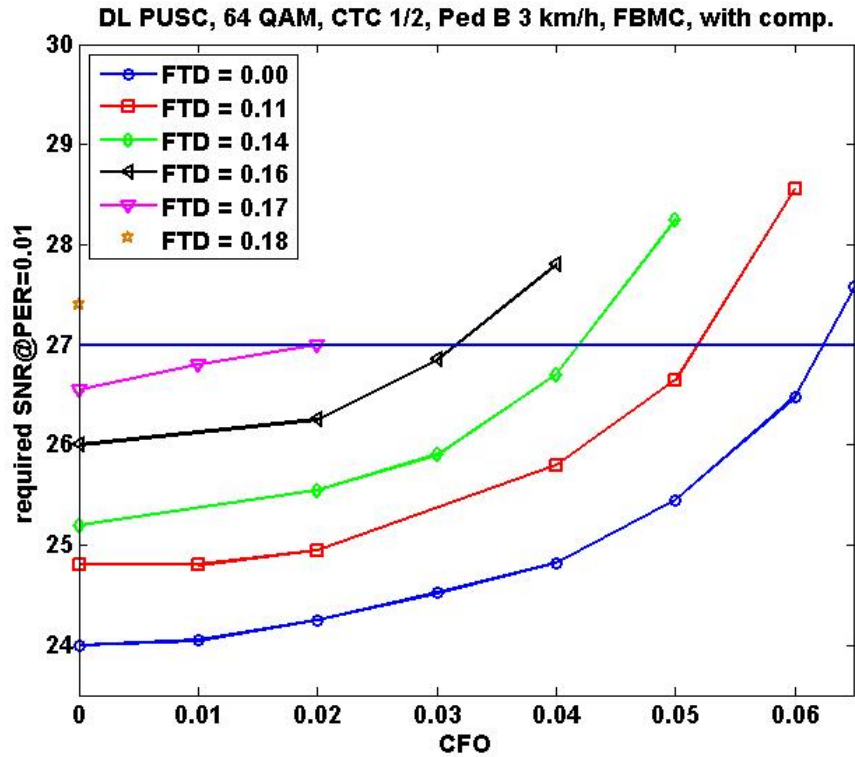


Figure 62: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots, passing the chain twice)

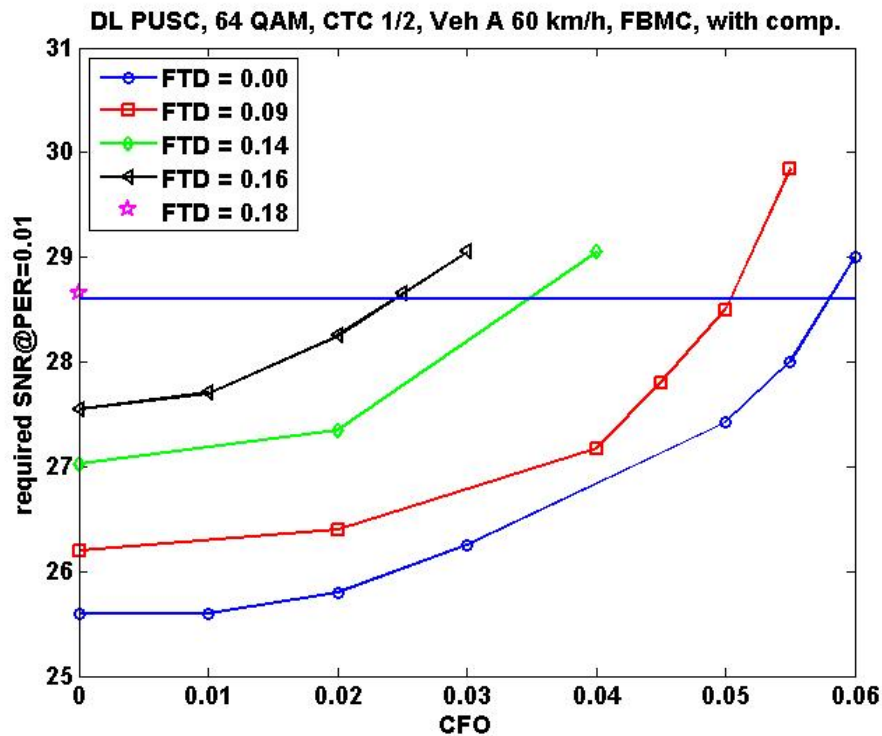


Figure 63: required SNR to achieve PER=1%, with dedicated offset compensation (regression based, strong pilots, passing the chain twice)

Passing through the chain more than twice delivers no further gains.

Thus the 3 dB tolerances with respect to CFO and FTD are:

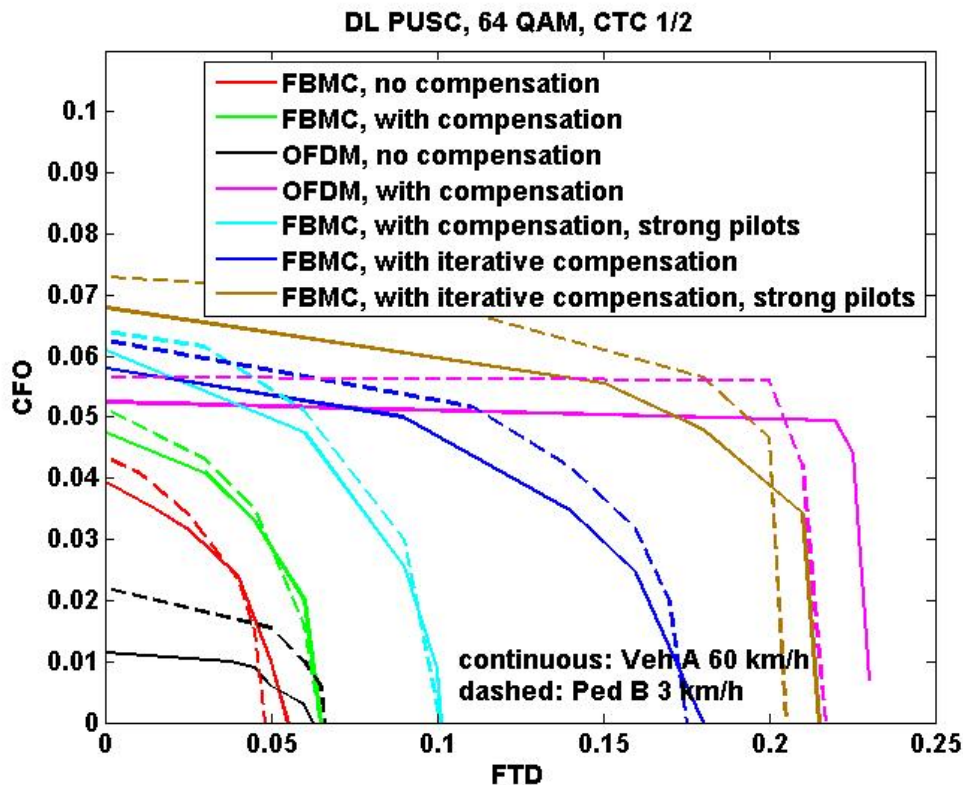


Figure 64: CFO/FTD tolerances with 3 dB link budget dedicated to desynchronization.

As with QPSK and 16 QAM OFDM is rather insensitive to time delay due to the cyclic prefix. FBMC suffers more. However, by applying iterative compensation the robustness of FBMC to FTD significantly can be increased. Once the main pilots are as strong as with OFDM and an additional iteration is performed the sensitivity of FBMC to FTD is nearly as low as with OFDM without the need of a CP. Again the filterbank solution supports higher frequency offsets.

At this point it has to be emphasized, that the results presented here are based on pure phase compensation schemes. By applying more complex schemes combating ISI and ICI in addition shall lead to even more promising results.

### 1.3.2.2 Preamble based synchronization

As already mentioned within a previous chapter preamble based synchronization in downlink (LS approach) is a new addition to the simulator. A simulation study was performed to assess the capability of using a preamble to synchronize the receiver. The basic simulator settings are unchanged. The diagrams show the achievable packet error rate (PER) depending on the SNR for different FTDs and CFOs, respectively.

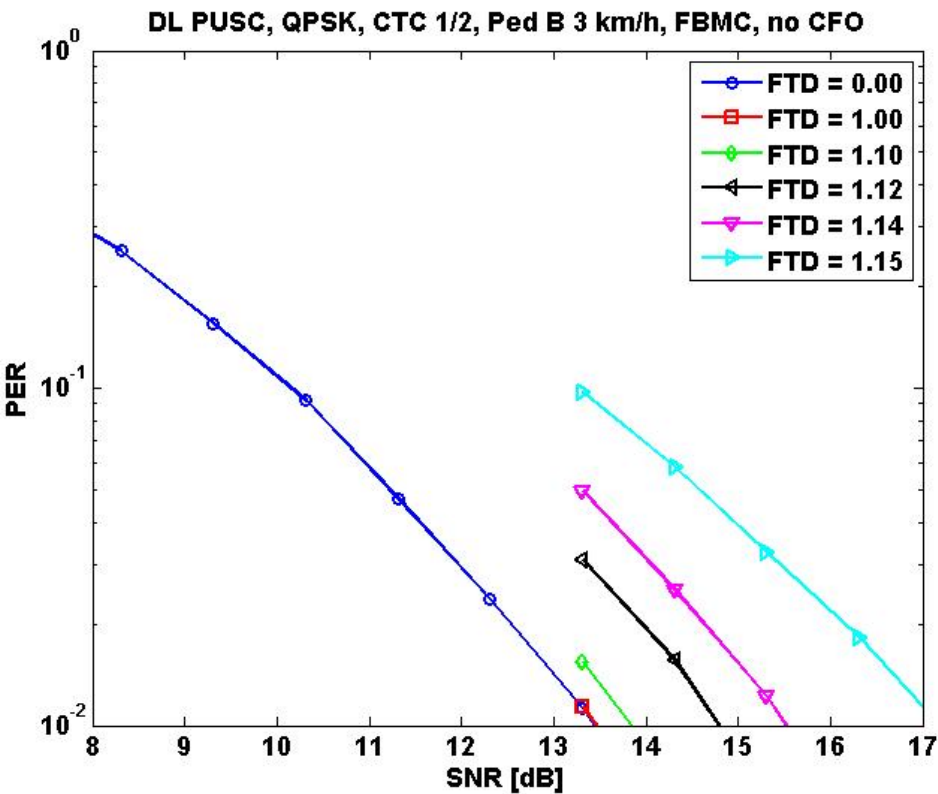


Figure 65: PER depending on the SNR for different FTDs.

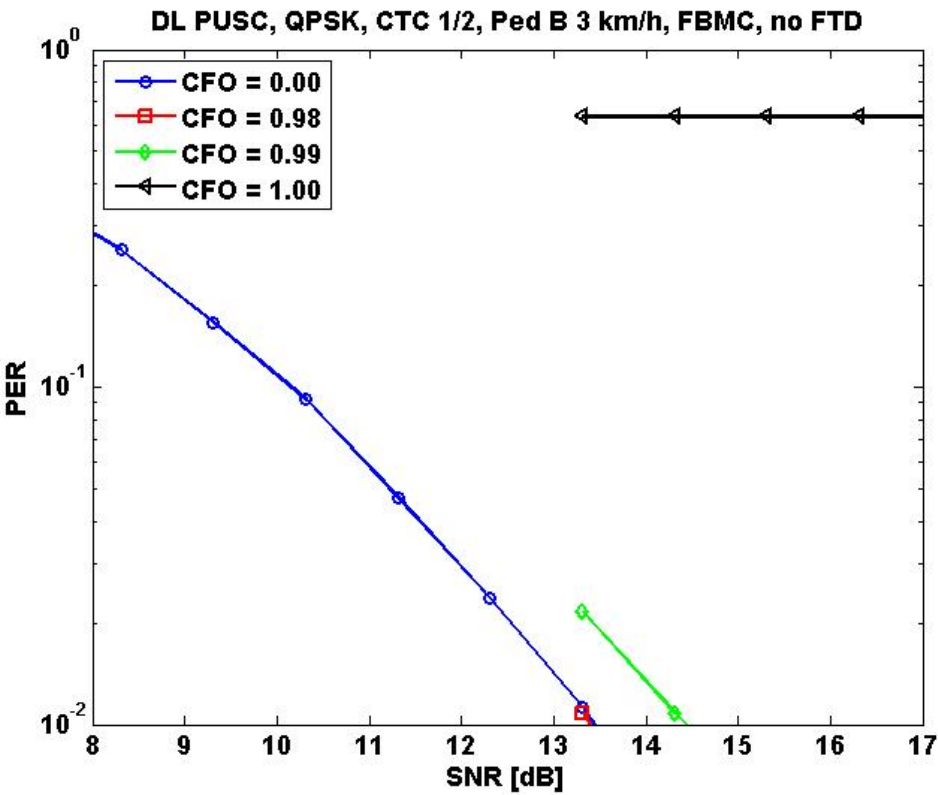


Figure 66: PER depending on the SNR for different CFOs.



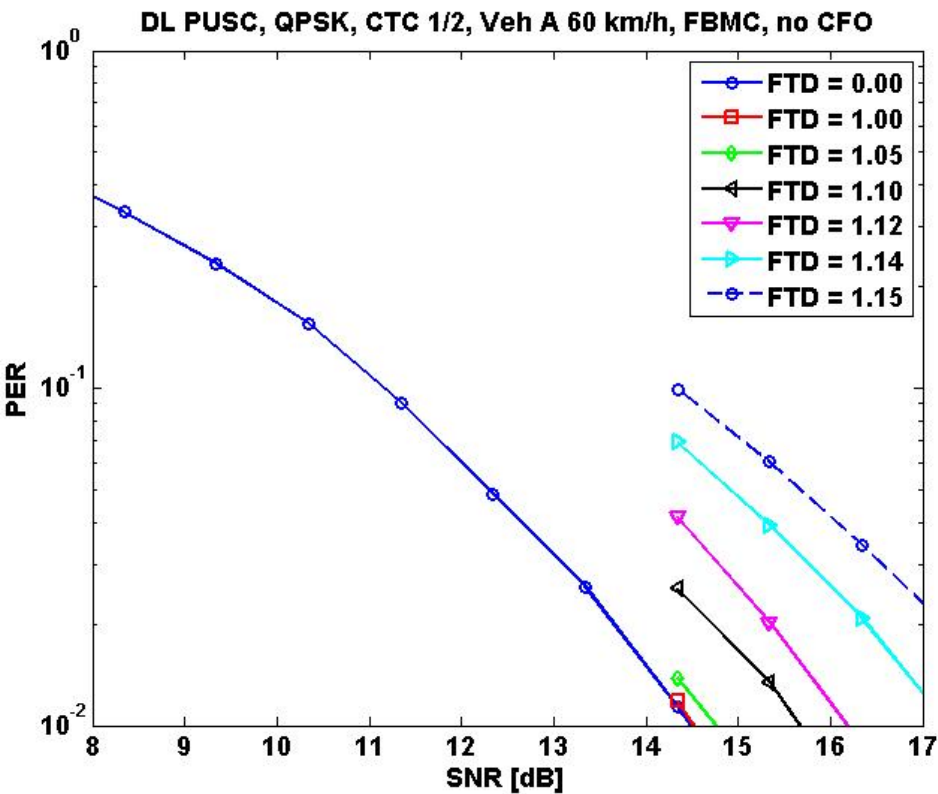


Figure 67: PER depending on the SNR for different FTDs.

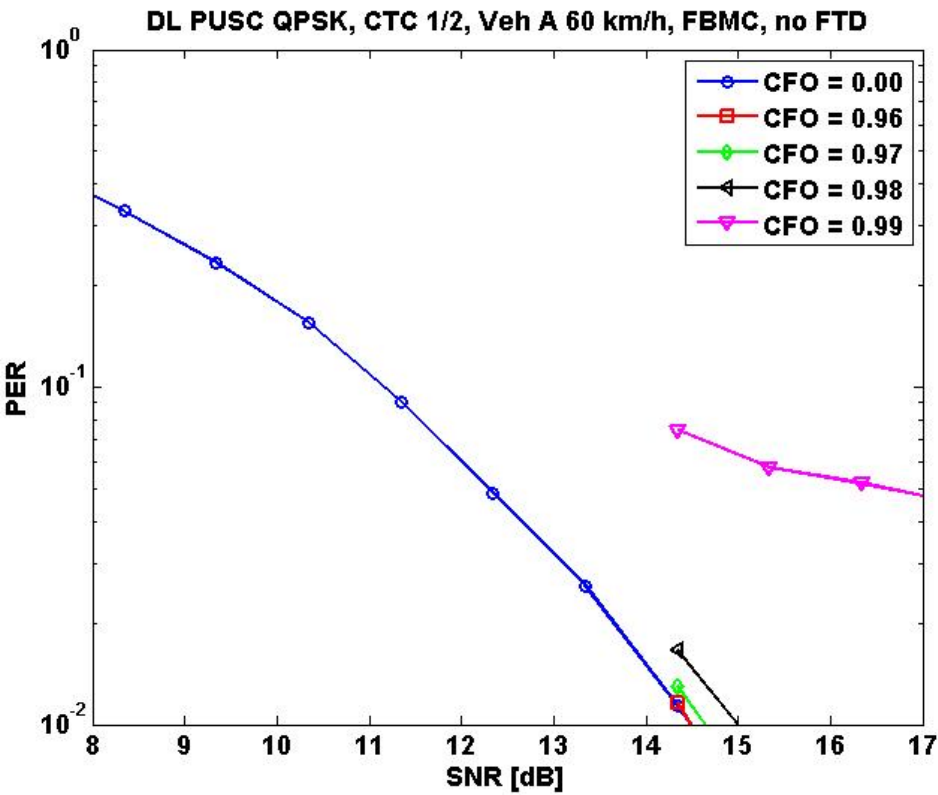


Figure 68: PER depending on the SNR for different CFOs.

As long as the actual offsets are within the estimation range there is no performance degradation at all as they are virtually perfect estimated. Once the estimation range is exceeded a residual timing offset remains which afterwards can be compensated via a pilot based scheme.

CFO compensation on the other side breaks down completely once the estimation range exceeds half the length of the preamble. This happens due to the transformation of the two-dimensional minimization to a one-dimensional maximization followed by an analytical calculation of the CFO. To make this more clear again this analytical expression:

$$\hat{\varepsilon}(\hat{\tau}) = \frac{1}{2\pi} \angle \{R(\hat{\tau})\}$$

with

$$R(\tilde{\tau}) = \sum_{m=(K-1)M}^{(K-1/2)M-1} r^* \left[ m + \frac{\tilde{\tau}}{T_s} \right] r \left[ m + \frac{M}{2} + \frac{\tilde{\tau}}{T_s} \right]$$

If the estimated FTD ( $\hat{\tau}$ ) equals or is greater than  $\frac{M}{2}T_s$  (half the length of the preamble here, remember: a single symbol was used as preamble) the right multiplier in  $R(\tilde{\tau})$  (i.e.  $r[\dots]$ ) does not contain an useful part of the preamble anymore and thus the calculation of  $\hat{\varepsilon}$ , the CFO, is not possible.

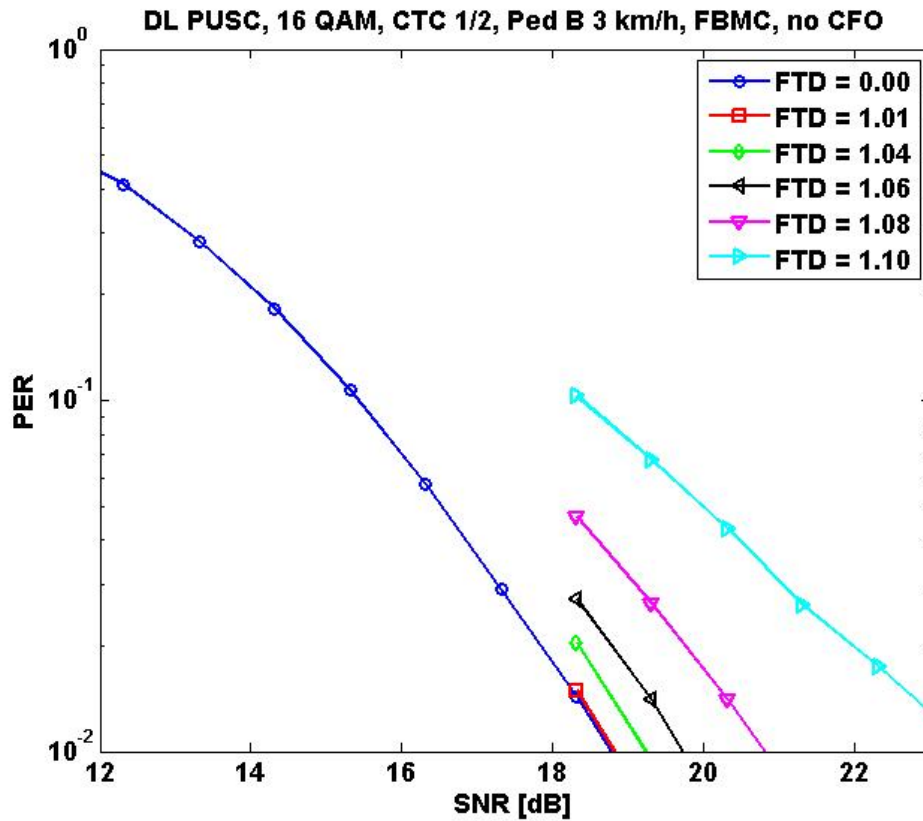


Figure 69: PER depending on the SNR for different FTDs.



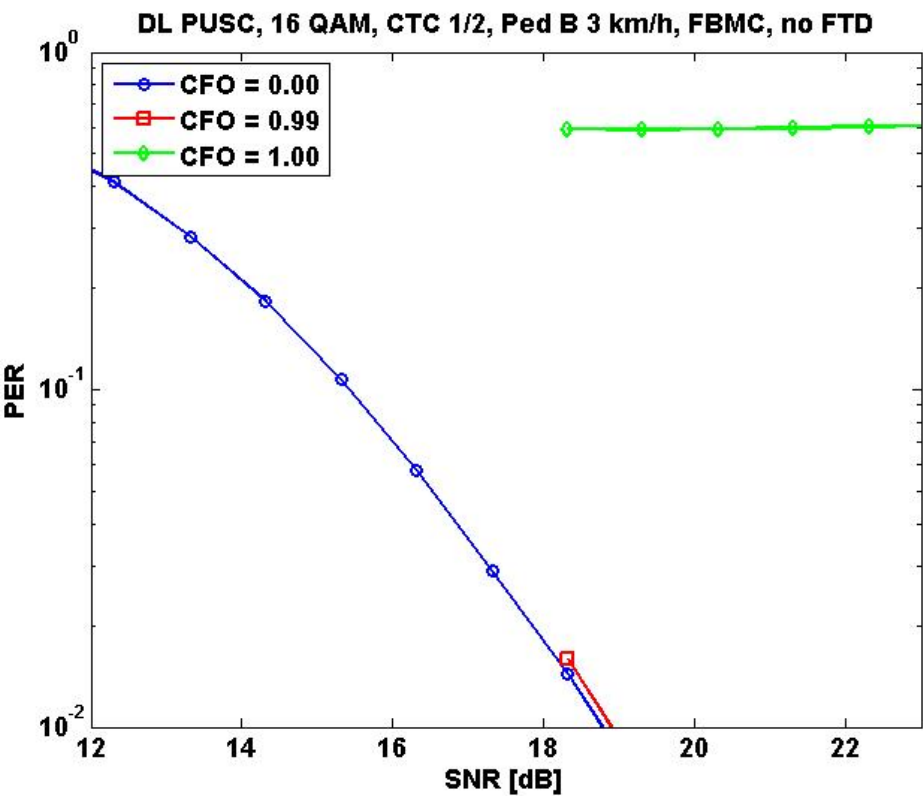


Figure 70: PER depending on the SNR for different CFOs.

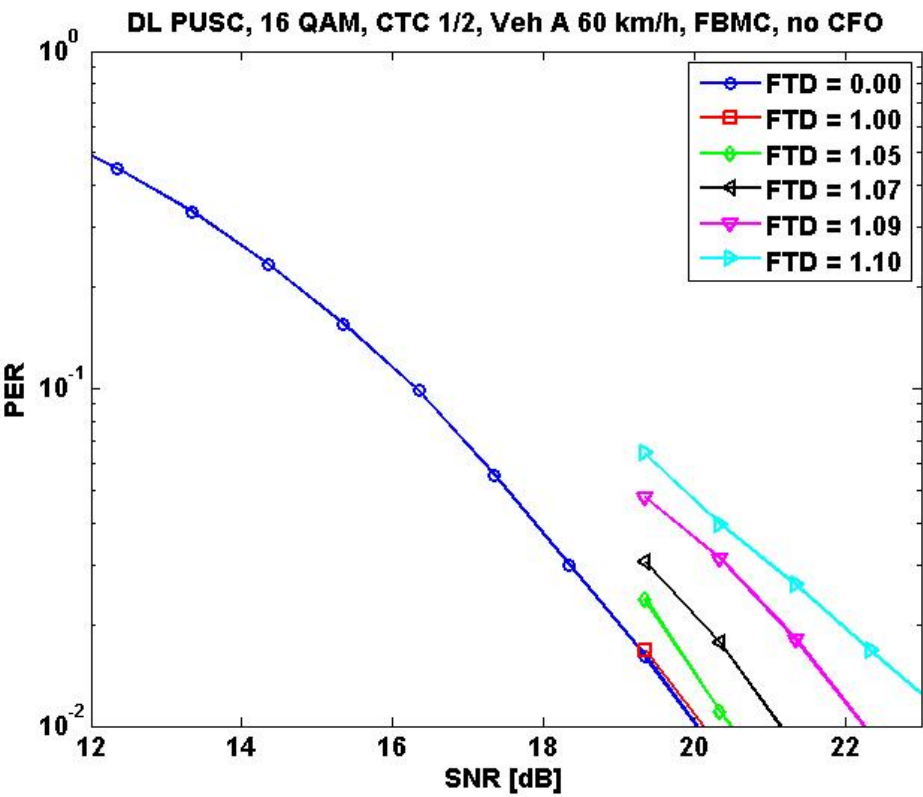


Figure 71: PER depending on the SNR for different FTDs.

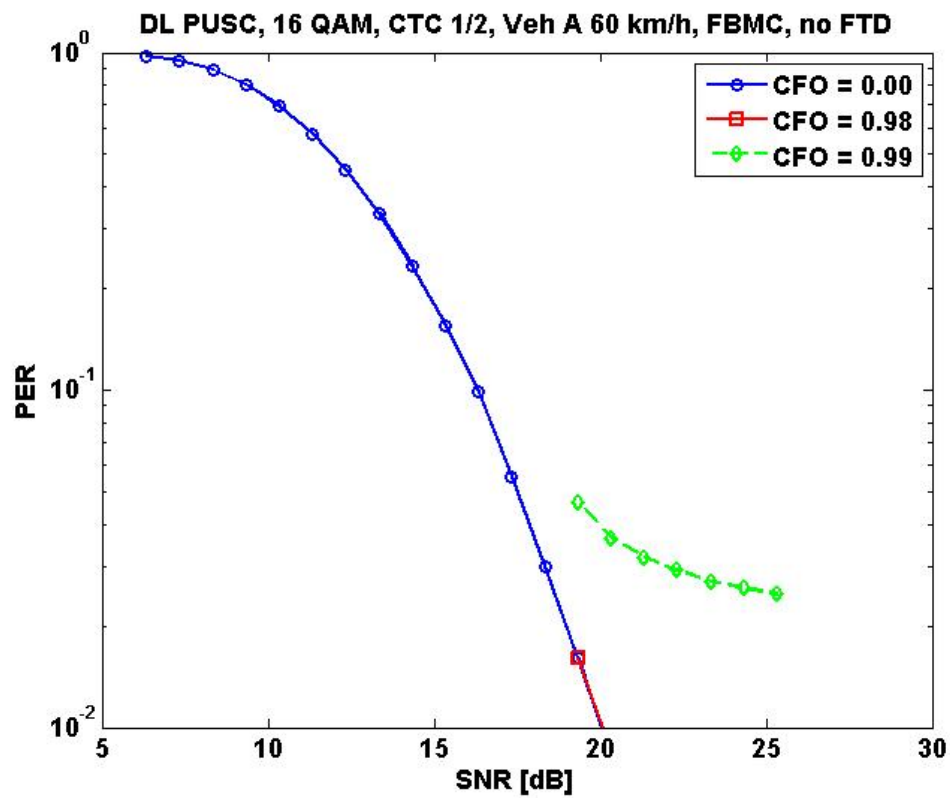


Figure 72: PER depending on the SNR for different CFOs.

As with QPSK again the estimation of the offset works perfect as long as the estimation range is not exceeded. Once this is the case degradation is faster, as 16 QAM is more sensitive to the residual offsets as QPSK.

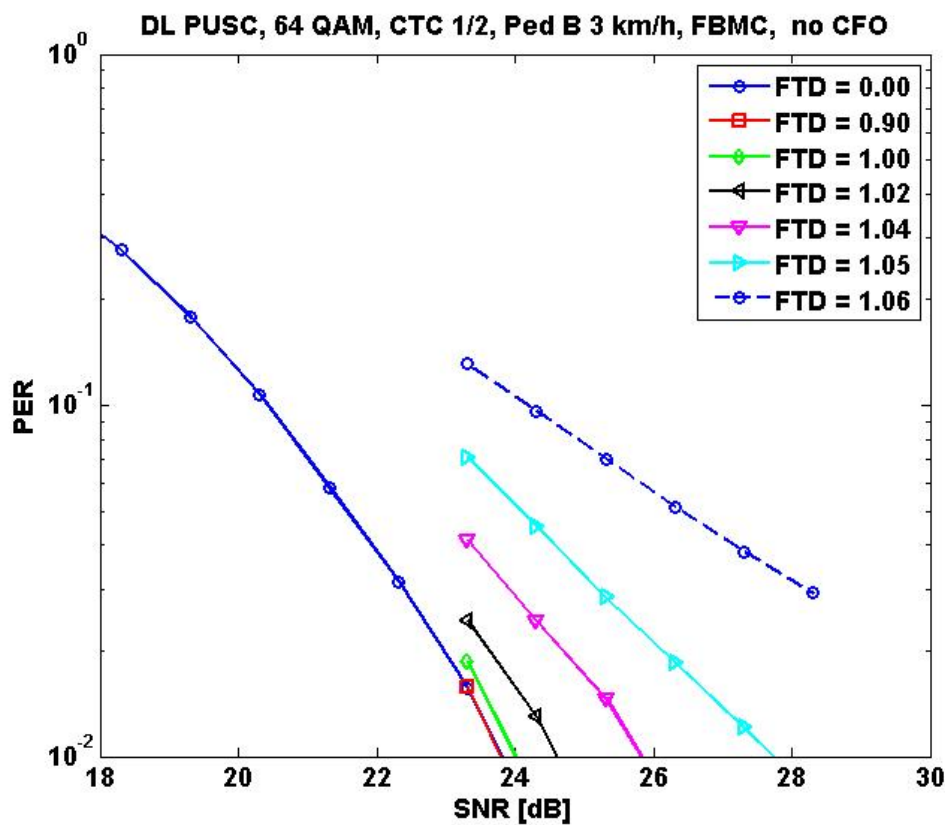


Figure 73: PER depending on the SNR for different FTDs.

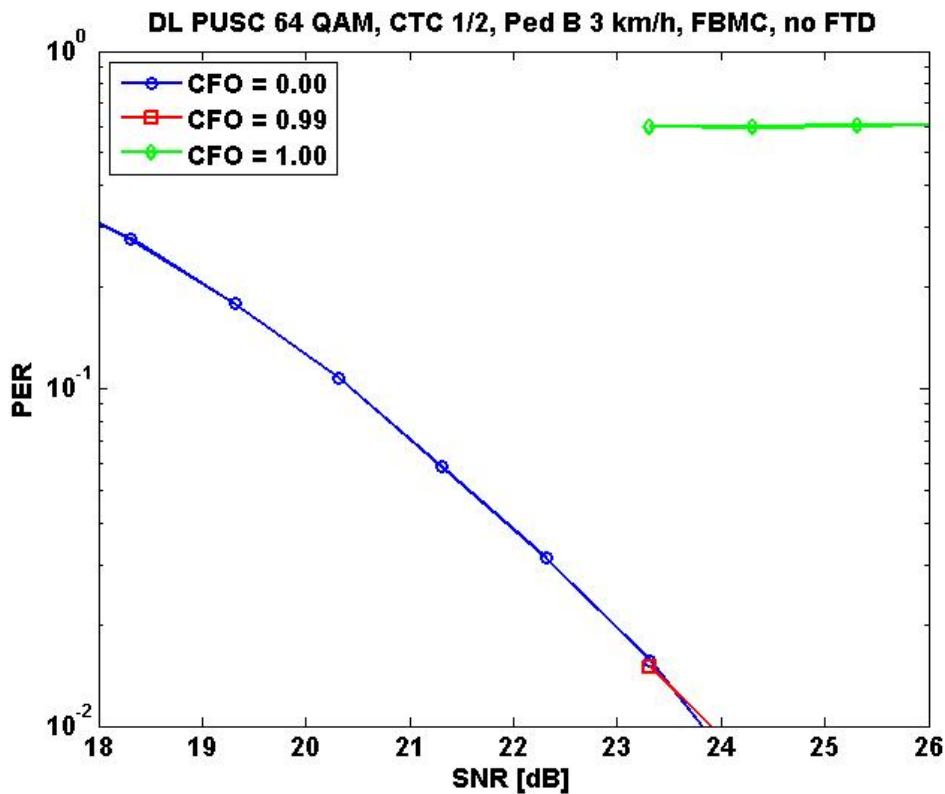


Figure 74: PER depending on the SNR for different CFOs.

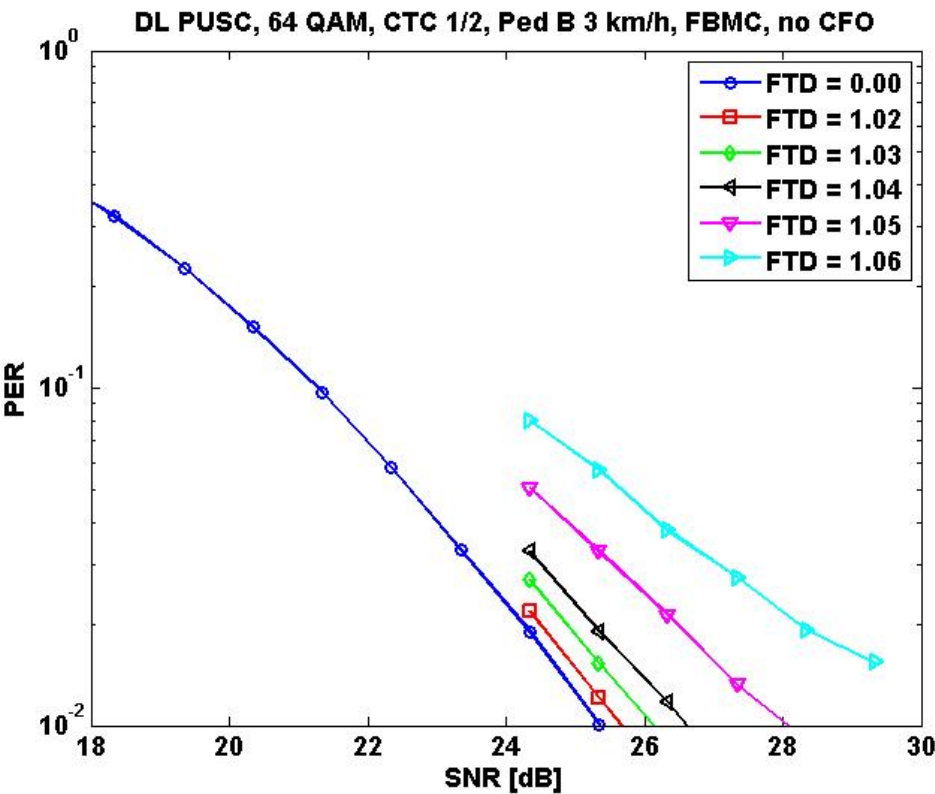


Figure 75: PER depending on the SNR for different FTDs.

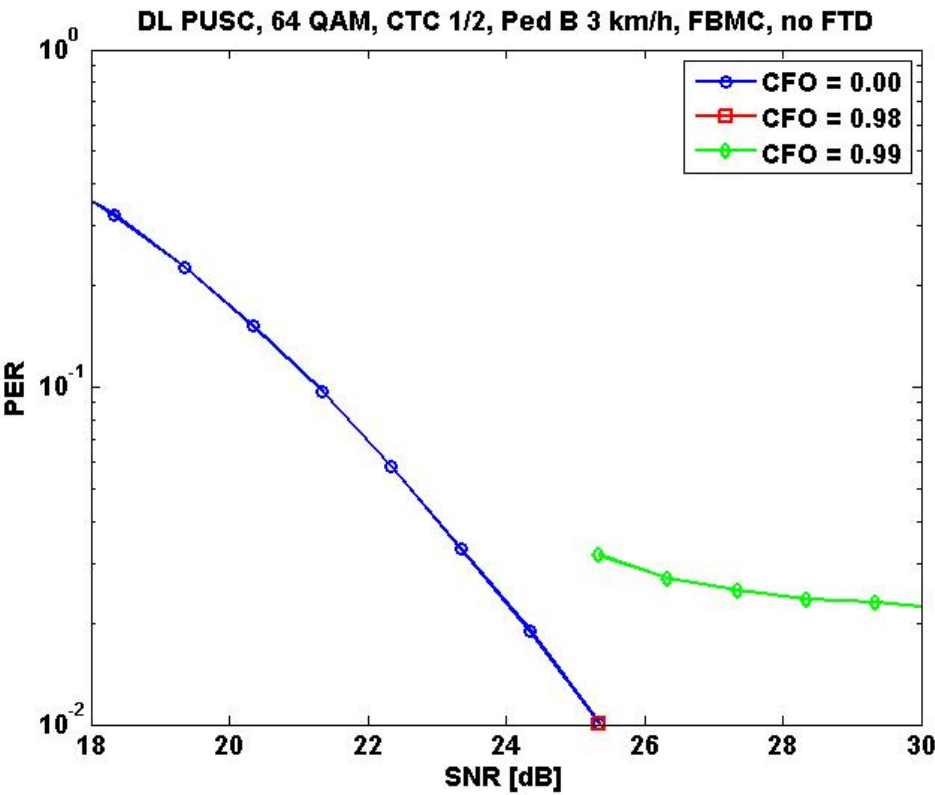


Figure 76: PER depending on the SNR for different CFOs.

Finally with 64 QAM preamble based synchronization again perfectly aligns timing and frequency within the estimation range. Beyond performance degrades.

In summary the range of acceptable offsets if 3 dB of the link budget are spent for this matter:

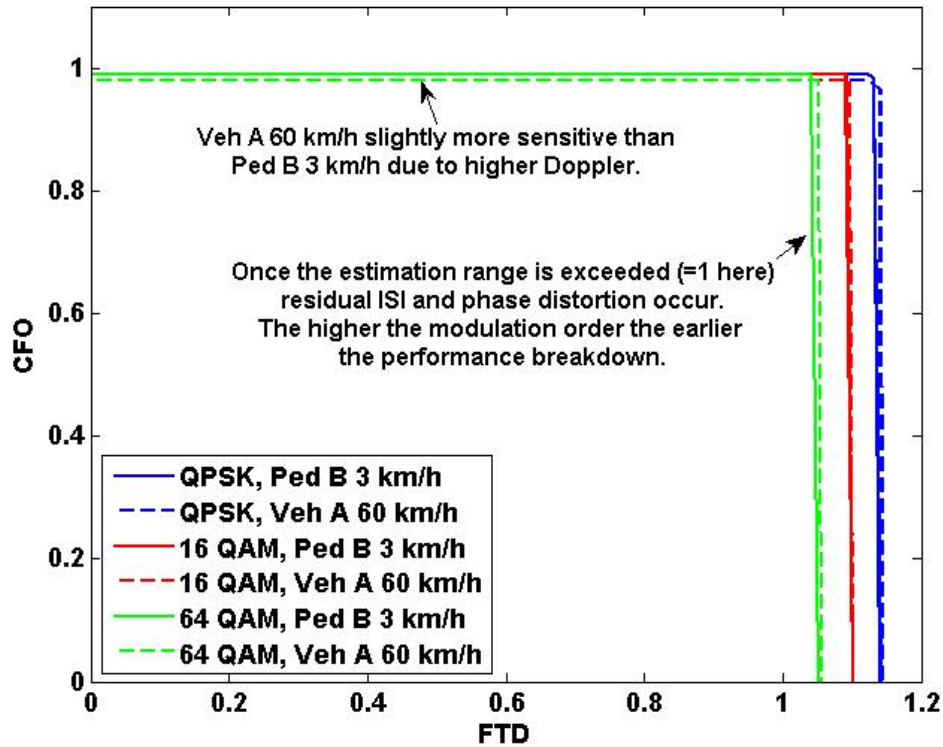


Figure 77: CFO/FTD tolerances with 3 dB link budget dedicated to desynchronization.

In a nutshell preamble based synchronization obviously is much better suited for high offsets than pilot based, however at the cost of a higher complexity (actually the range only is limited by the selection of the space of trial values with respect to the FTD, if the analytical expression is used to calculate the CFO the limit is half the length of the preamble). This is self-evident as preamble based synchronization cures the disease (i.e. the offsets) while pilot based synchronization just combats the symptoms (i.e. phase deviation, ISI, ICI).

### 1.3.3 Adaptive channel aware subcarrier spacing

Multicarrier schemes for wireless communications elaborated currently (LTE, WiMAX) are typically rather tight with respect to the available design space. There are some degrees of freedom for defining a transmission system, however, some aspects are carved in stone wasting opportunities for performance improvement. For example the subcarrier spacing in LTE is fixed to 15 kHz, in WiMAX to 10.94 kHz (and connected to this the symbol length for a given FFT size). These values are chosen taking into account typical channel characteristics (delay spread, Doppler spread, CFO, FTD) leading to rather robust data transmission. However, these channel characteristics somehow are antithetic with respect to the degradation they cause depending on the subcarrier spacing/symbol length. CFO and Doppler spread call for broader spacings/shorter symbols, while FTD and delay

spread demand smaller spacings/longer symbols. Thus, the predefined values are a compromise leading to a robust transmission of data in average. Different users encounter different channel characteristics. E.g. a user riding within a train encounters a high Doppler spread while the delay spread may be rather small, depending on the environment. To serve this user efficiently a broad spacing of the subcarriers while having a short symbol length should be applied. On the other side a pedestrian walking at the edge of the cell encounters typically a high delay spread while the Doppler spread is negligible. To serve him efficiently the length of the symbols should be long while the spacing between the subcarriers would be rather small.

This chapter continuous and wraps up the simulation study started in [11] regarding flexible subcarrier spacings.

Data transmission in multi carrier communication systems (e.g. WiMAX, LTE) is segmented into frames. This study proposes to segment these frames into subframes with different subcarrier spacings each:

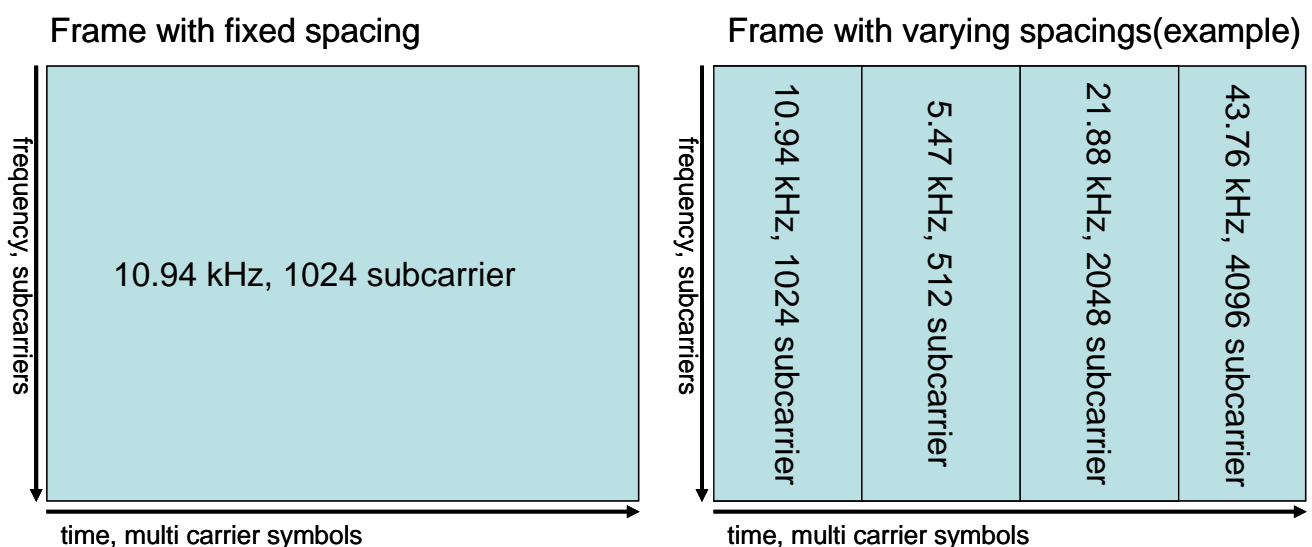


Figure 78 Frames with and without varying subcarrier spacings

Alternatively, spacing could be changed from frame to frame, either.

The characteristics of the channel a user is confronted with are measured. Based on these measurements the user is scheduled into the subframe/frame granting the best performance.

The channel characteristics to be measured are:

### 1. Delay spread, coherence bandwidth:

The smaller the coherence bandwidth (the bigger the delay spread), the smaller the spacing should be (leading to longer symbols), to assure flat fading within a single subcarrier and to aid pilot based channel estimation. On the other side, spacings that are chosen too small, are reducing the achievable frequency diversity gain exploited by the error correction in contiguous allocation schemes (e.g. AMC in WiMAX, LTE).

### 2. Doppler spread, coherence time (speed):

The higher the time variance (speed) is the smaller the symbol length should be (equivalent to broader spacings), to keep the channel within a single symbol constant and to reduce the impact of the frequency jitter due to Doppler.

### 3. Carrier frequency offset (CFO):

The higher the frequency offset is, the broader the spacings should be, to reduce ICI.

### 4. Time delay (FTD):

The higher the time delay, the longer the symbol length should be (equivalent to smaller spacings), to reduce ISI.

Metrics to measure the channel characteristics (examples):

1. autocorrelation of the estimated channel in frequency direction → The width of the correlation peak is a measure of the coherence bandwidth.
2. autocorrelation of the estimated channel in time direction → The width of the correlation peak is a measure of the coherence time.  
alternatively:  
cross correlation between the channel estimations in frequency direction for different times (e.g. in WiMAX scenario: cross correlation of the channel estimate at frame  $x$  with the channel estimate at frame  $x+j$  ( $j=1,2,3,4,\dots$ ))
3. linear regression of the phase with the pilots as supporting points in time direction → phase slope is depending on the frequency offset
4. linear regression of the phase with the pilots as supporting points in frequency direction → phase slope is depending on the time delay

Generally, the parameterization of the divers subframes may be chosen freely (spacing, number of subcarriers, bandwidth,...). However, to minimize complexity the following is recommended:

- To keep the sampling rate constant over the complete frame/frames, the overall bandwidth should be kept constant.  
→ the number of subcarriers is varying between the zones/frames
- To be able to apply FFT/IFFT, the number of subcarriers should be a power of two.  
→ the subcarrier spacings of the divers subframes/frames should be power of two multiples of a given base ( $*1, *2, *4,\dots$ )

If these recommendations are met, the only increase in complexity is the provision of several FFTs within the transmitter/receiver and the additional constraints for scheduling.

To demonstrate the impact of the chosen subcarrier spacings to the performance, link level simulations have been executed. The following figures depict the packet error rate (a single packet carries 64 Bytes) depending on the signal-to-noise ratio for different spacings. The achievable gain when applying the proposed scheme gets apparent.

Both, a system using OFDM with cyclic prefix (WiMAX, relative CP length: 1/8) and a multi-carrier system applying a filterbank (FBMC, filterbank based multi carrier) – a candidate for replacing the OFDM in future communication systems – are used. The system parameters for the latter are the same as in WiMAX (e.g. bandwidth, permutation schemes, ...)

**Table 8: System Parameters**

sampling rate	11.2 MHz	11.2 MHz	11.2 MHz	11.2 MHz	11.2 MHz	11.2 MHz
subcarrier spacing	2.74 kHz	5.47 kHz	10.94 kHz	21.88 kHz	43.76 kHz	87.52 kHz
FFT width	256	512	1024	2048	4096	8192
permutation scheme	AMC	AMC	AMC	AMC	AMC	AMC

### Impact of delay spread and speed:

First the **FBMC** system is investigated:

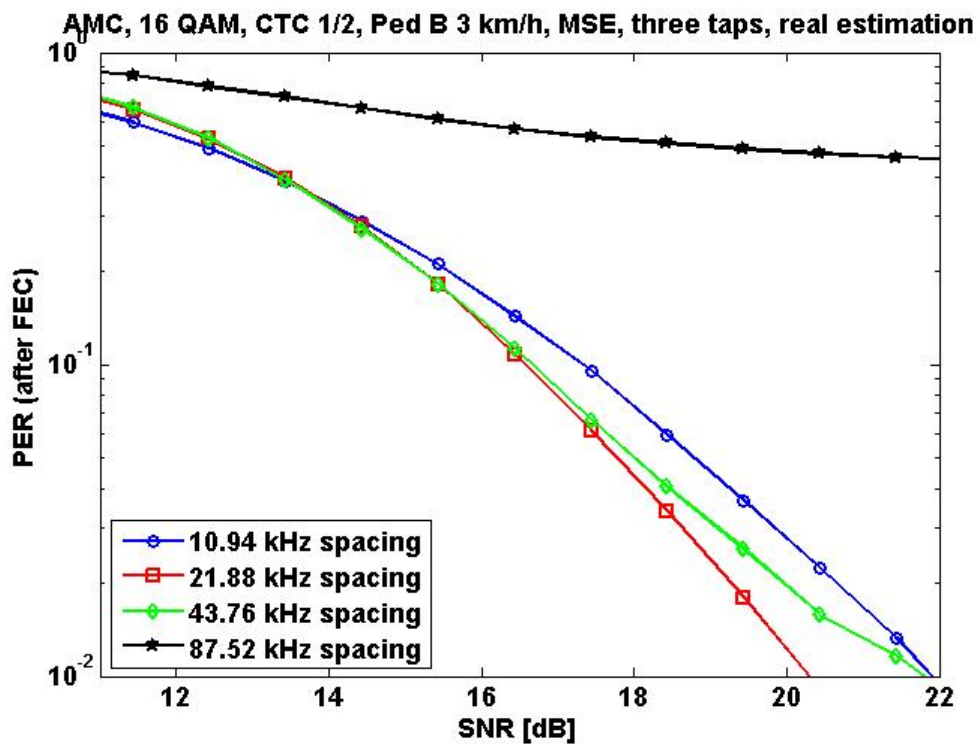




Figure 79: Ped B 3 km/h, best choice 21.88 kHz

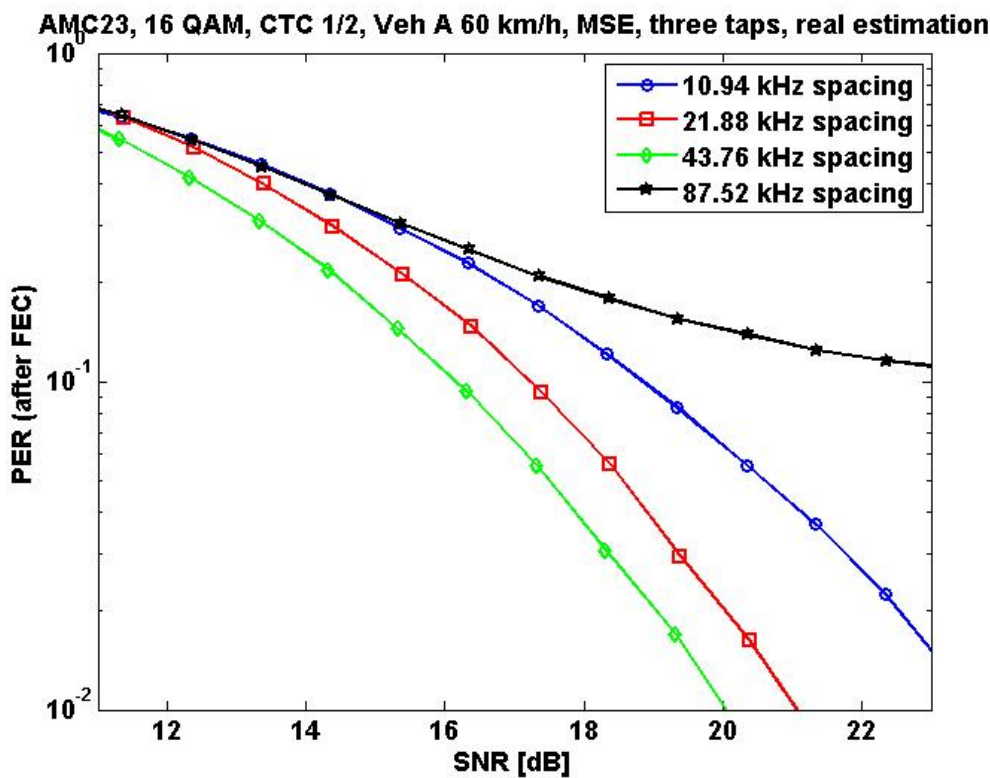


Figure 80: Veh A 60 km/h, best choice 43.76 kHz

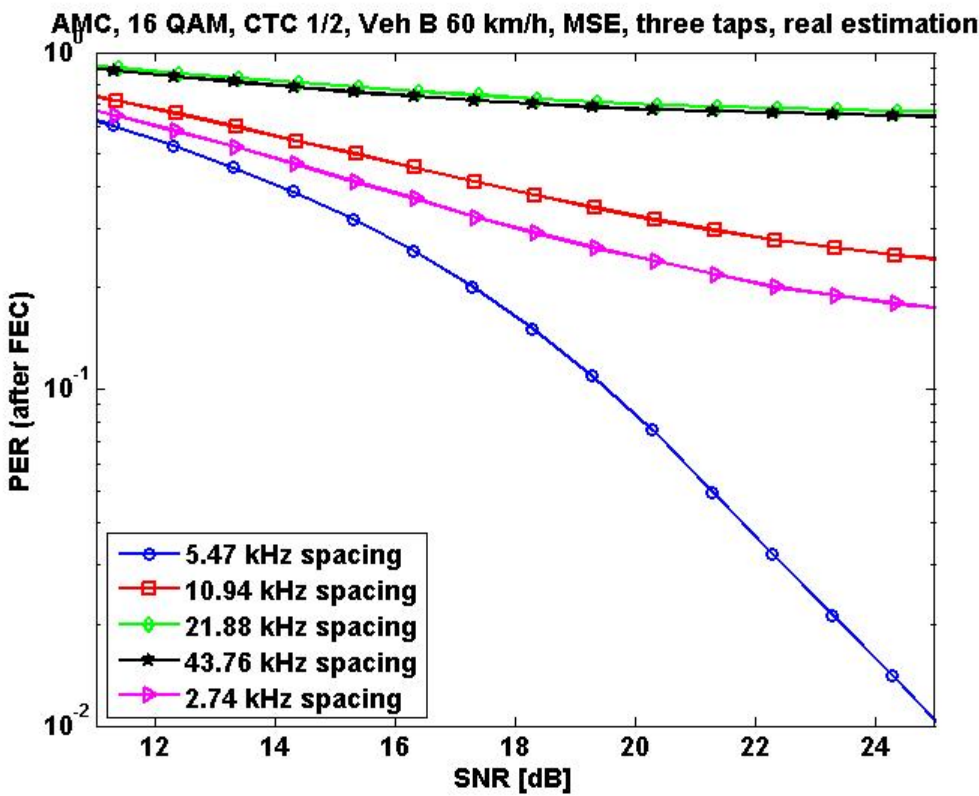


Figure 81: Veh B 60 km/h, best choice 5.47 kHz

Thus, taking the simulation results into account the optimal spacings are:

Table 9: Optimal Spacings (delay spread, FBMC)	
channel model	optimal spacing
Ped B (medium delay spread)	21.88 kHz
Veh A (low delay spread)	43.76 kHz
Veh B (high delay spread)	5.47 kHz

If **OFDM** is used:

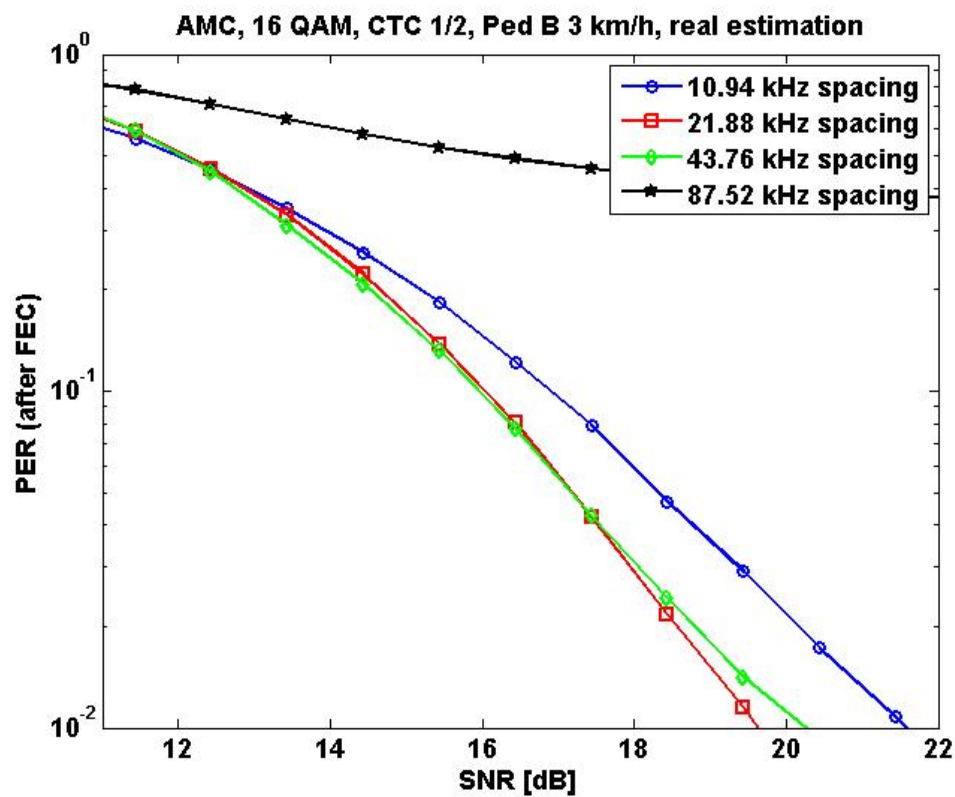


Figure 82: Ped B 3 km/h, best choice 21.88 kHz

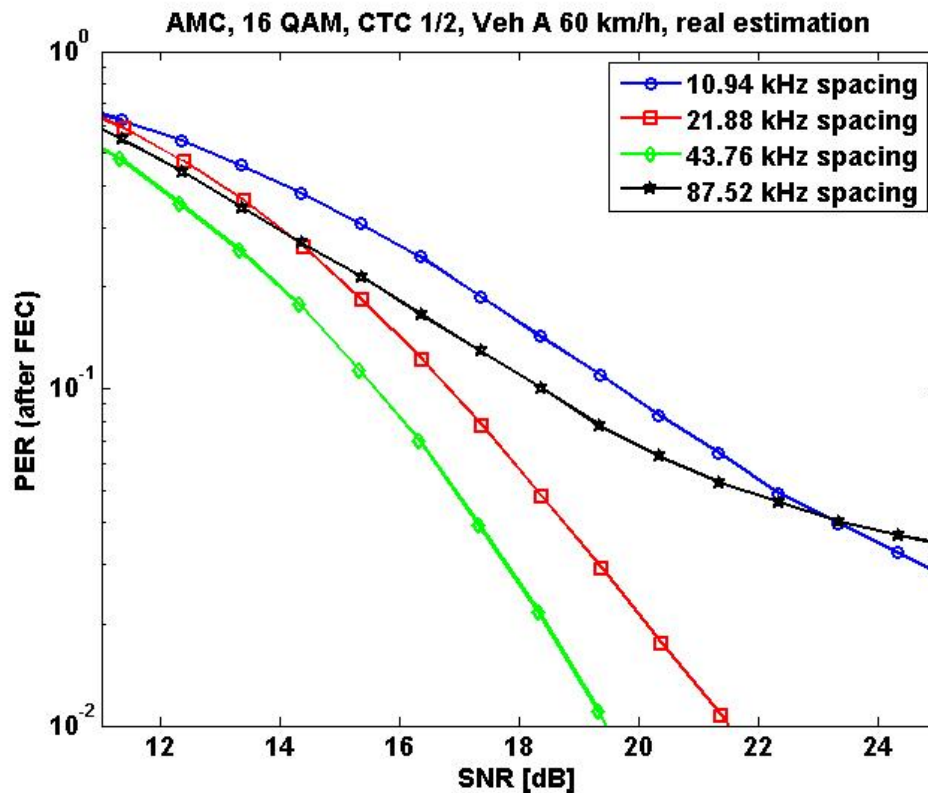


Figure 83: Veh A 60 km/h, best choice 43.76 kHz

Thus, taking the simulation results into account the optimal spacings are:

Table 10: Optimal Spacings (delay spread, OFDM)

channel model	optimal spacing
Ped B (medium delay spread)	21.88 kHz
Veh A (low delay spread)	43.76 kHz
Veh B (high delay spread)	not applicable

The cyclic prefix of WiMAX is too short to support channels with delay spreads modeled with Veh B.

### Impact of frequency offsets:

Again we start with the **FBMC** system:

To highlight the impact of the frequency offset ideal channel estimation is assumed (known frequency response). The carrier offset is estimated via linear regression in time direction using the pilots as supporting points.

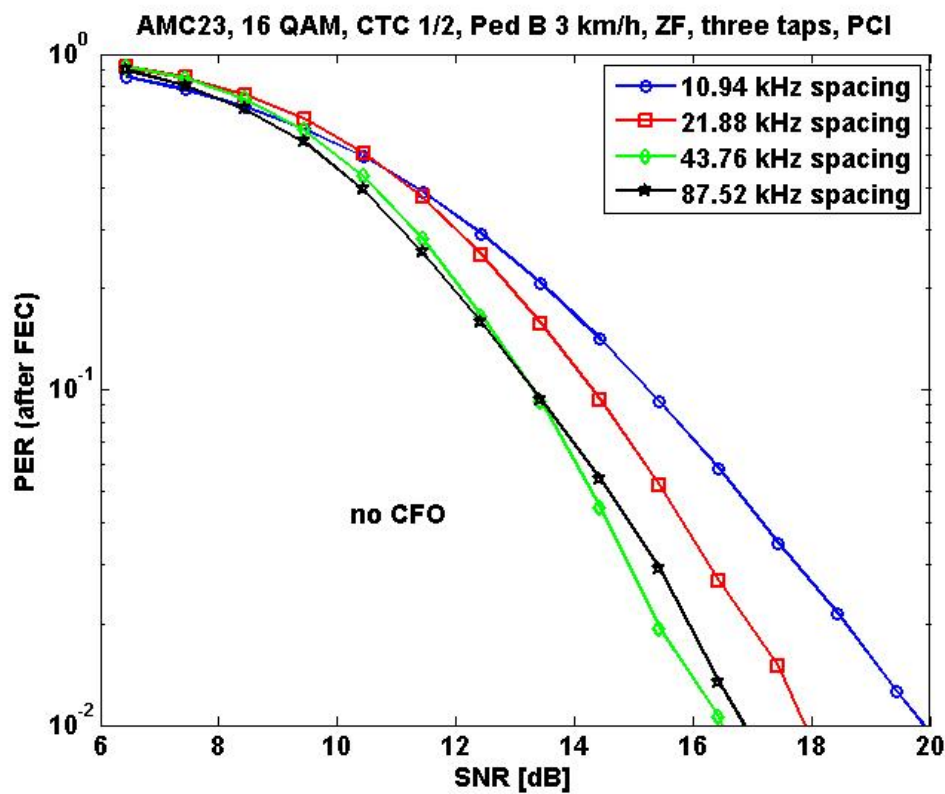


Figure 84: no CFO, best choice 43.76 kHz

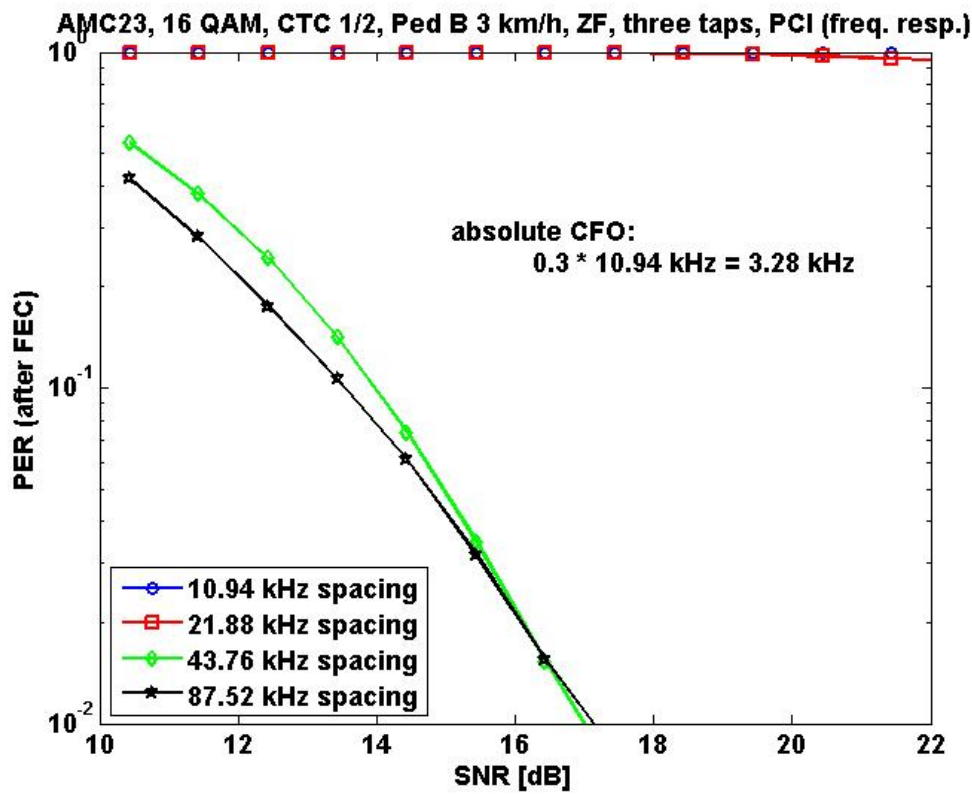


Figure 85: 3.28 kHz CFO, best choice: 43.76 kHz.

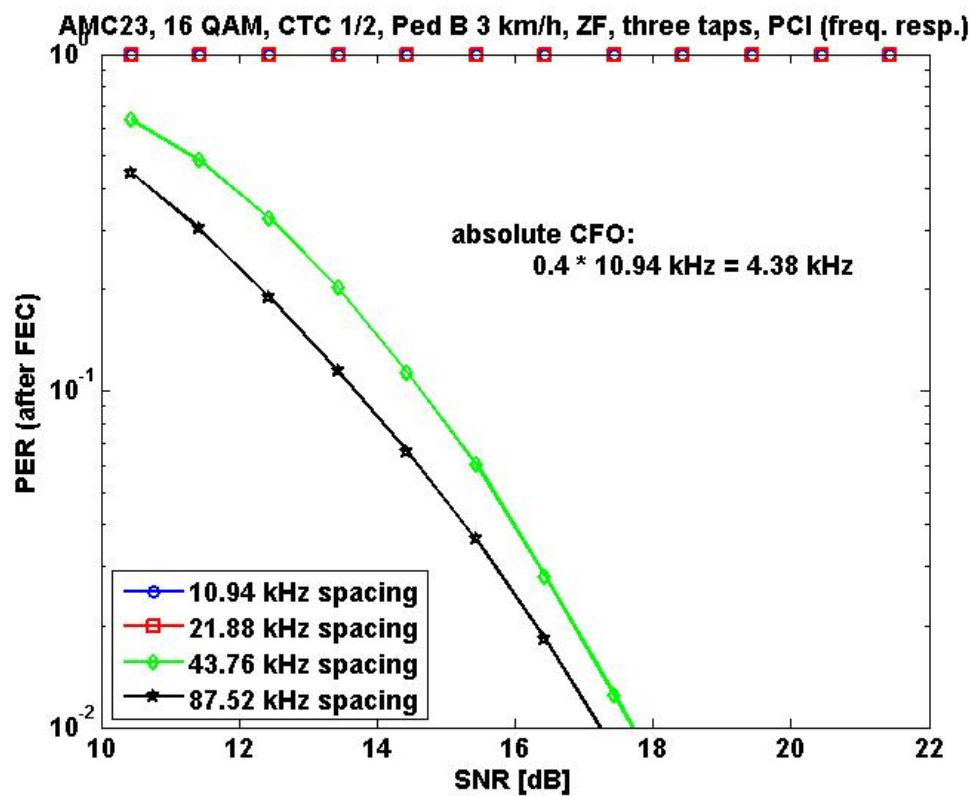


Figure 86: 4.38 kHz CFO, best choice: 87.52 kHz.

Thus, taking the simulation results into account the optimal spacings are:

Table 11: Optimal Spacings (CFO, FBMC)

frequency offset	optimal spacing
0 kHz	43.76 kHz
3.28 kHz	43.76 kHz
4.38 kHz	87.52 kHz

**OFDM** system:

Now real channel estimation is applied.

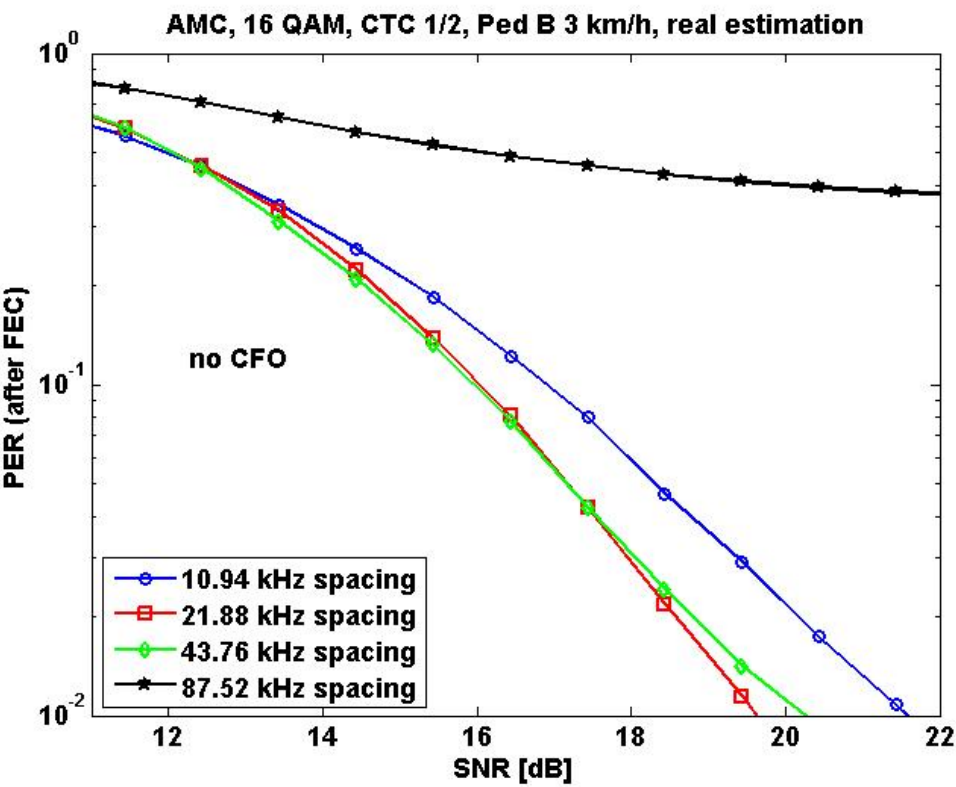


Figure 87: no CFO, best choice: 21.88 kHz.

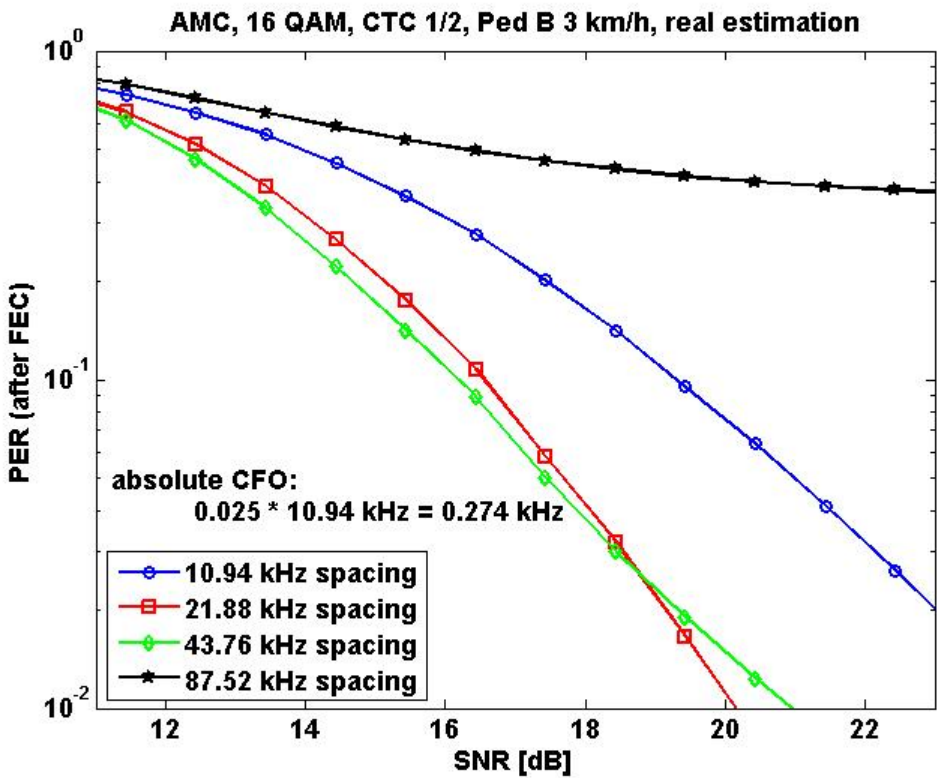


Figure 88: 0.274 kHz CFO, best choice: 21.88 kHz.

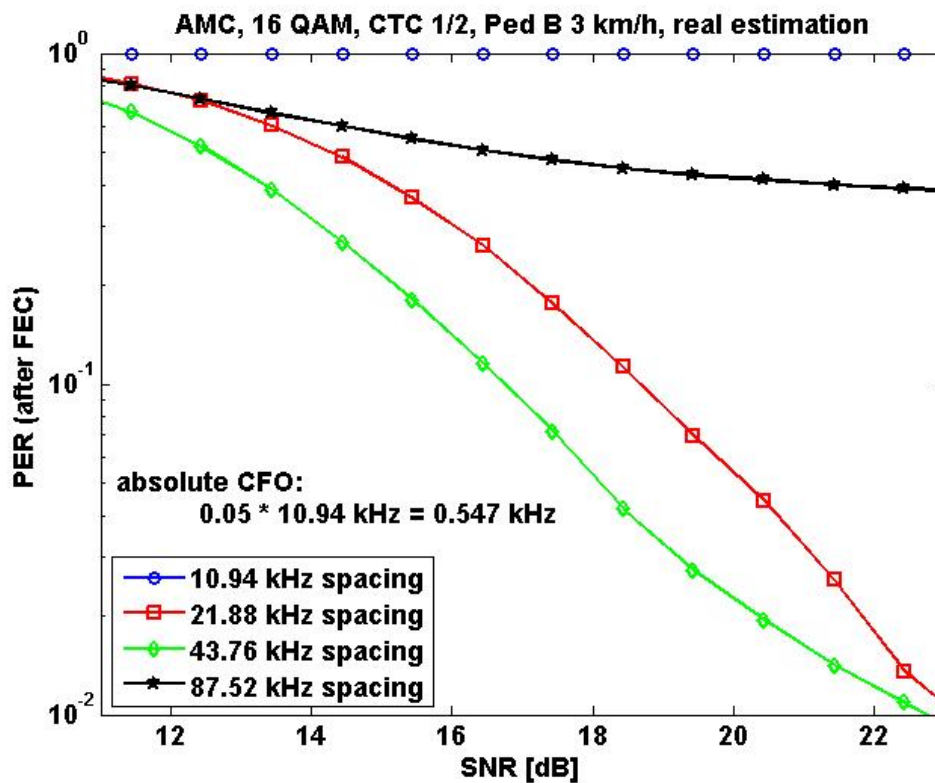


Figure 89: 0.547 kHz CFO, best choice: 43.76 kHz.

Thus, taking the simulation results into account the optimal spacings are:

Table 12: Optimal Spacings (CFO, OFDM)

frequency offset	optimal spacing
0 kHz	21.88 kHz
0.274 kHz	21.88 kHz
0.547 kHz	43.76 kHz

### Impact of time delay:

#### FBMC system:

To highlight the impact of the time delay ideal channel estimation is assumed (known frequency response). The time delay is estimated via linear regression in frequency direction using the pilots as supporting points.



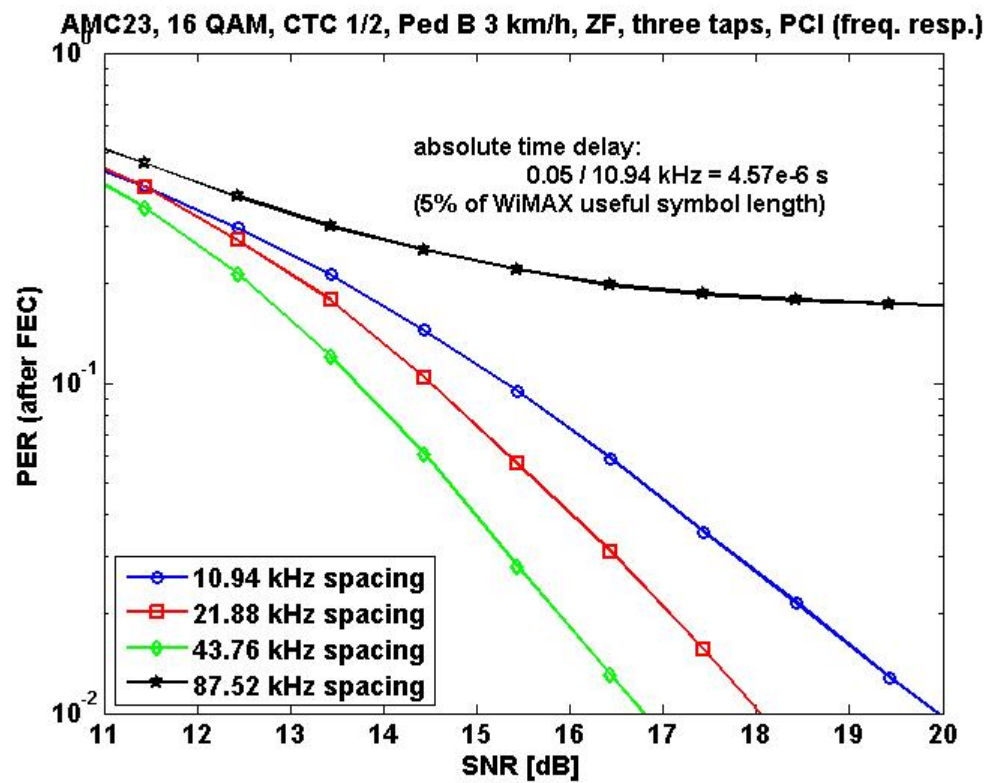


Figure 90: 4.57  $\mu\text{s}$  FTD, best choice: 43.76 kHz.

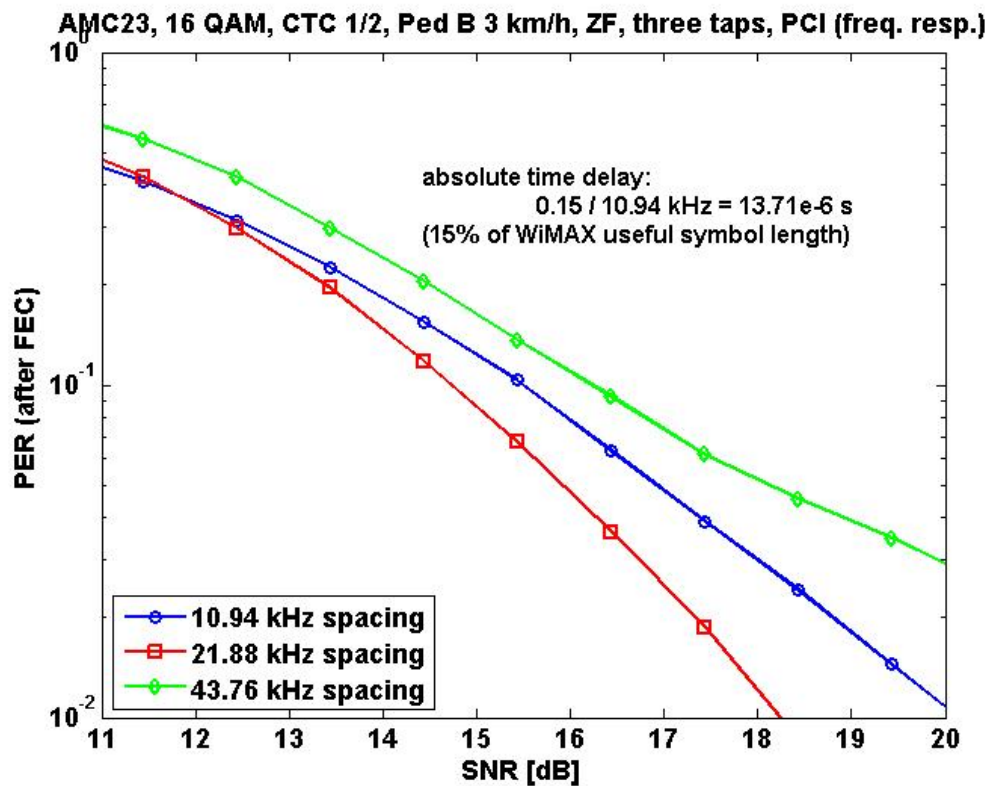


Figure 91: 13.71  $\mu\text{s}$  FTD, best choice: 21.88 kHz.

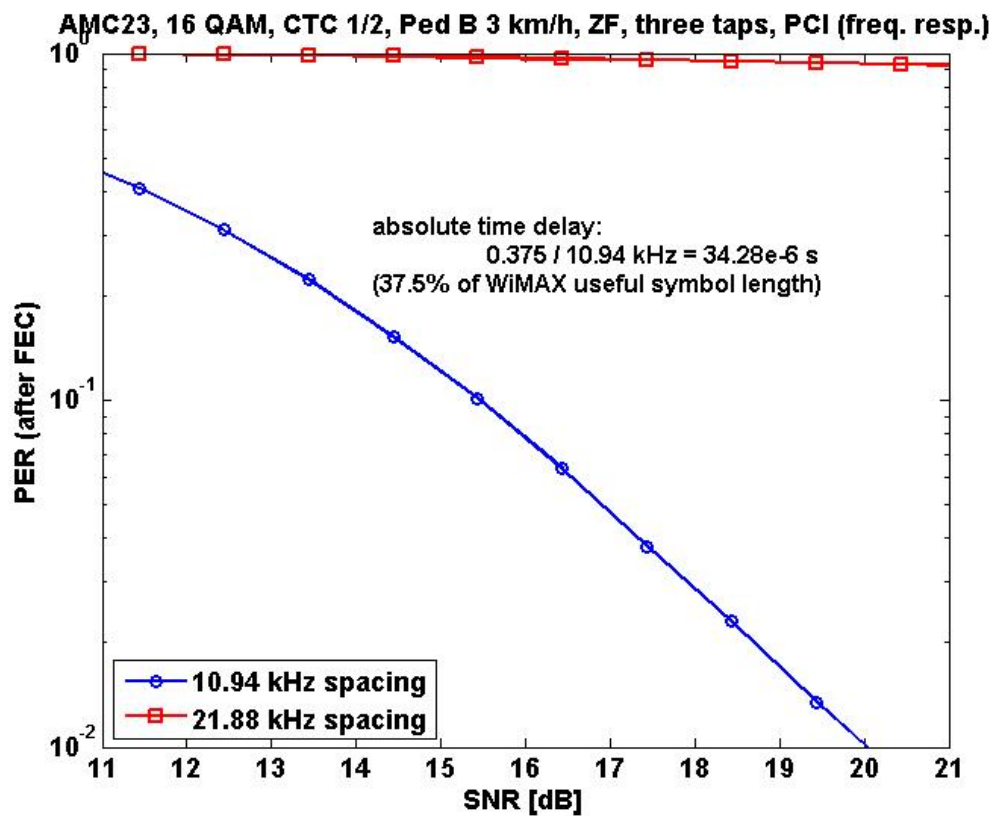


Figure 92: 34.28  $\mu\text{s}$  FTD, best choice: 10.94 kHz.

Thus, taking the simulation results into account the optimal spacings are:

Table 13: Optimal Spacings (FTD, FBMC)

time delay	optimal spacing
4.57 $\mu\text{s}$	43.76 kHz
13.71 $\mu\text{s}$	21.88 kHz
34.28 $\mu\text{s}$	10.94 kHz

**OFDM** system:

Real channel estimation is applied.

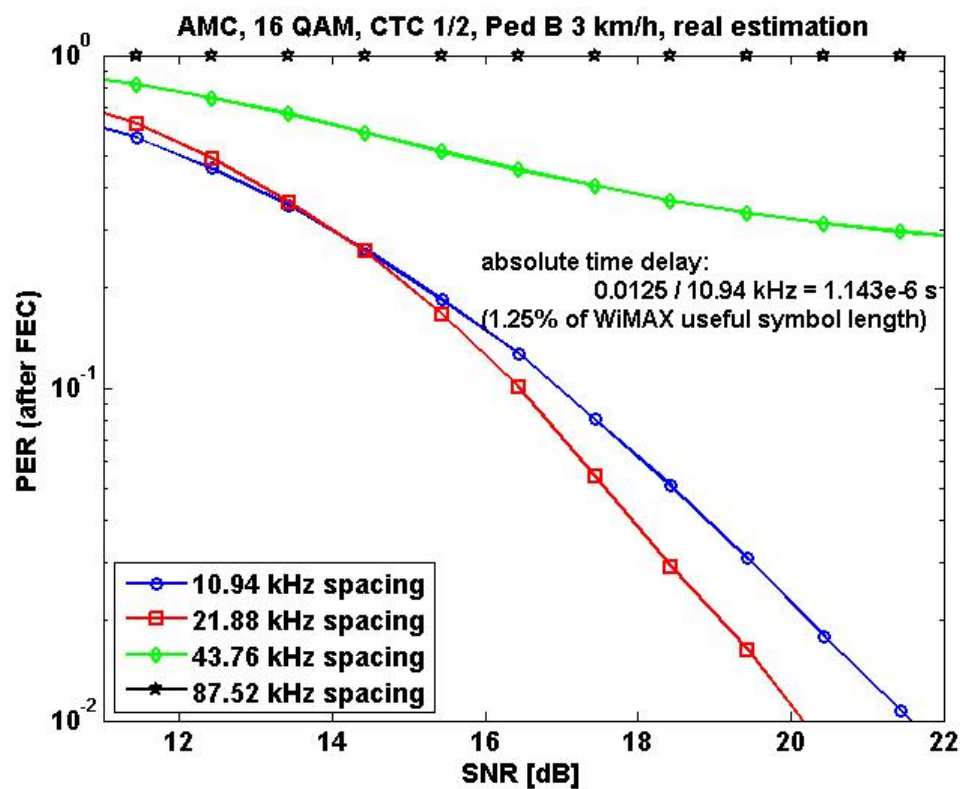


Figure 93: 1.143  $\mu$ s FTD, best choice: 21.88 kHz.

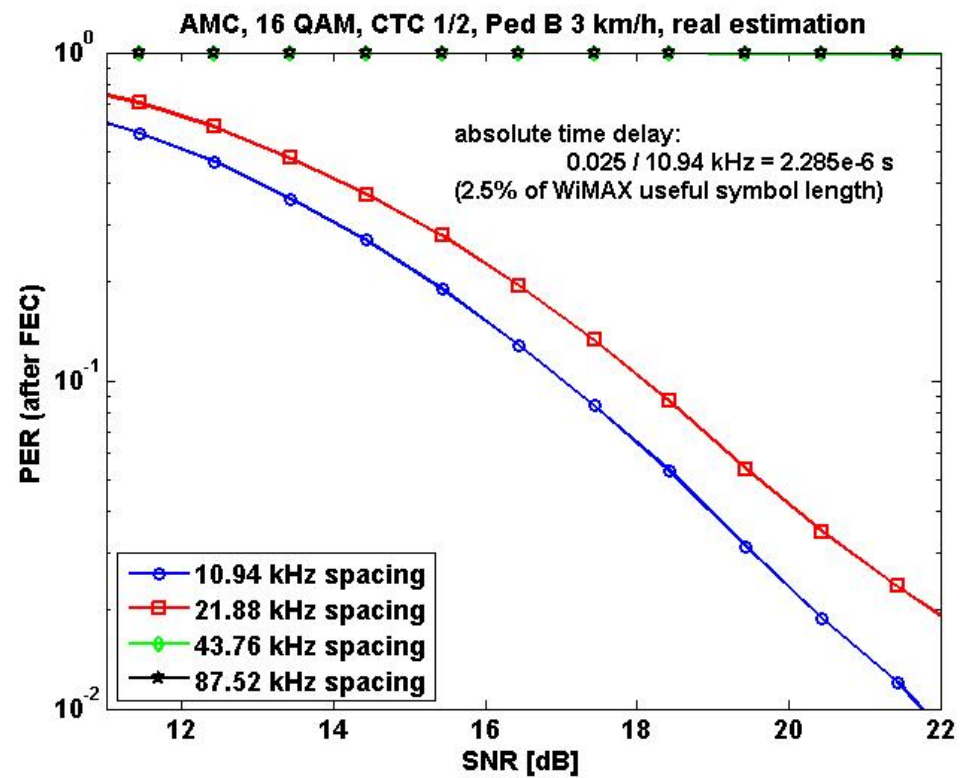


Figure 94: 2.285  $\mu$ s FTD, best choice: 10.94 kHz.

Thus, taking the simulation results into account the optimal spacings are:

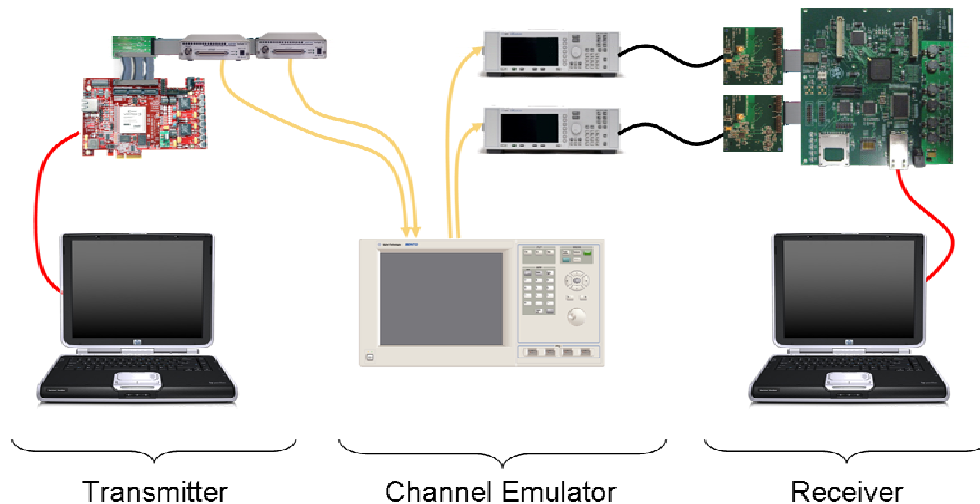
**Table 14: Optimal Spacings (FTD, OFDM)**

time delay	optimal spacing
1.143 $\mu$ s	21.88 kHz
2.285 $\mu$ s	10.94 kHz

## 2 OFDM-FBMC Demonstrator Setup

One of the objectives of the laboratory setup is to test, measure and demonstrate the advantages of an FBMC-based approach as compared to an OFDM-based approach. It will try to bring additional insight on top of the simulation results. Realistic performance expectations can only be checked if we can evaluate the impact of large time scale channel effects in a repeatable way.

The figure below shows the demonstrator setup that will be build for the Phydys project.



**Figure 95: Demonstrator Setup**

### 2.1 Transmitter

The test transmitter will together with the fading simulator and test receiver make near real-time measurements of FBMC performance possible. It will also allow comparisons between WiMAX and FBMC performance.

SINTEF is responsible for the test transmitter design and implementation. This chapter describes the specifications and architecture of the test transmitter.

The test transmitter is normally connected to the receiver side through the fading simulator. The transmitter and fading simulator run in real time, The receiver captures data in real time and demodulates offline. But it is also possible to demodulate data from the test transmitter in the simulator's receiver. This cross connection is mainly intended for test and verification.

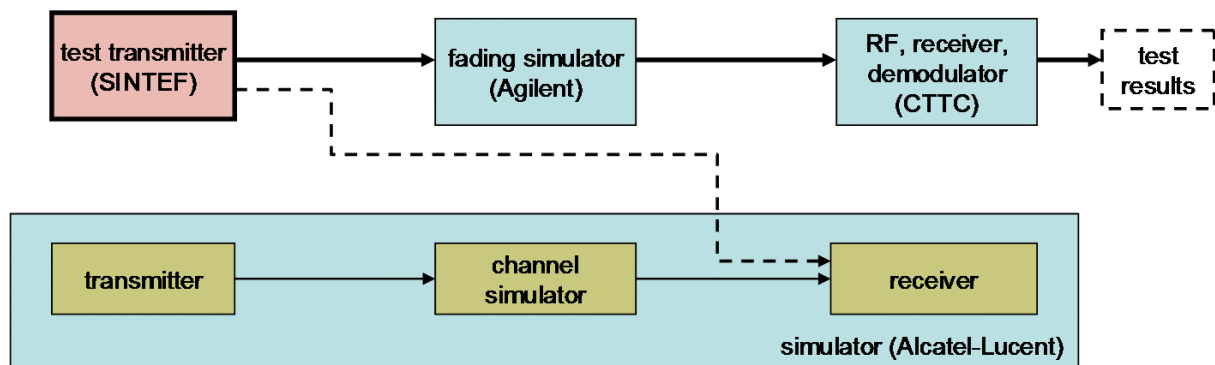


Figure 96: Demonstrator-Simulator Cross Connection

### 2.1.1 Physical Architecture

The transmitter is implemented on an FPGA board (HTG-V5-DDR3-PCIE) from [www.hitechglobal.com](http://www.hitechglobal.com). It will connect to a PC through an USB port for configuration and control. It will connect to the fading simulator through flat cable interface. Two instances of the transmitter can run simultaneously on the FPGA board using two different interfaces to the fading simulator. The two transmitter instances can be coordinated for MIMO simulation or one can be used for data while the second transmit a disturbing neighbour station signal.

The main component on the board is a Virtex-5 SX 95 FPGA from Xilinx. This is a large FPGA well suited to our design. The board was chosen for its large FPGA and because it has connectors with many digital IO lines. It has many other resources like SRAM, DRAM, PCI Express interface, Gigabit serial interfaces and Ethernet interface but these will not be used in our design.

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices <sup>(2)</sup>	Block RAM Blocks			CMTs <sup>(4)</sup>	PCI Express Endpoint Blocks	Ethernet MAC Blocks	Maximum RocketIO GTP Transceivers <sup>(5)</sup>	Total I/O Banks <sup>(7)</sup>	Max User I/O <sup>(6)</sup>
	Array (Row x Col)	Virtex-5 Slices <sup>(1)</sup>	Max Distributed RAM (Kb)		18 Kb <sup>(3)</sup>	36 Kb	Max (Kb)						
XC5VSX95T	160 x 46	14,720	1,520	640	488	244	8,784	6	1	4	16	19	640

Figure 97: FPGA Specification

The FPGA board is a PCI Express board but we will use it as a standalone board with an external power supply. The control PC will communicate with the board using a serial port over USB. The serial port runs at 115 kbaud which is sufficient for configuration and control.

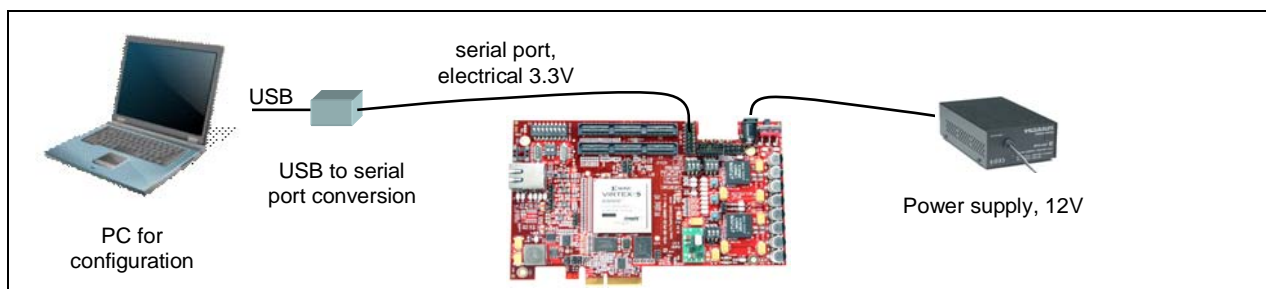


Figure 98: USB and Power Connection

To support MIMO with two channels two separate interfaces to the fading simulator are needed. Each interface uses a flat cable and has one clock line and 16 single ended data lines plus ground lines. The clock comes from the fading simulator at 8 times the sample rate. The data lines alternate between 16 bit I and Q values.

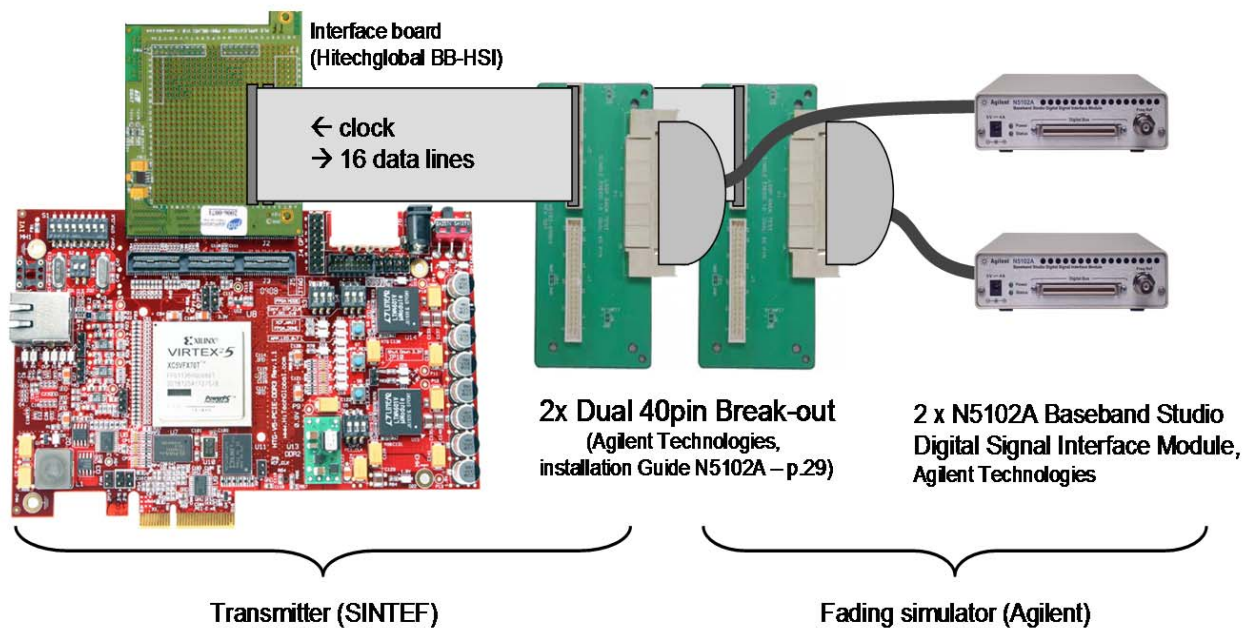


Figure 99: Interface between Transmitter and Fading Emulator

The FPGA board with the interconnection boards will be mounted in a suitable cabinet to avoid damage to the boards and cables.

## 2.1.2 Transmitter set-up

The setups are used for the first tests and verification of the transmitter. The purpose of this setup is to have a common configuration such that the demonstrator transmitter can be verified both against the Matlab simulator and the demonstrator receiver. The interface testing between the transmitter and channel emulator has started, but the tests involving the receiver has not yet commenced. The last section in this chapter describes the current status of the transmitter.

### 2.1.2.1 Deriving setup parameters for the hardware

The demonstrator transmitter is configured by loading a large set of parameters from configuration files describing each step in the processing from data generation through encoding, frame assembly, modulation and up-sampling. All these parameters are loaded from the configuration PC at the start of a demonstration session.



The basic configuration files are generated manually while two large tables are extracted from a specially prepared version of the Matlab simulator. We configure the modified Matlab simulator with the scenario we want to implement and when the simulator runs special extraction code extracts configuration data for the demonstrator transmitter.

- The *instruction table* is a large table describing each carrier in the full frame transmitted.
  - The zone type in this part of the frame (Wimax, FBMC)
  - The data source for the carrier (burst data, random data, constant data)
  - The type of carrier (preamble, data, pilot, auxiliary pilot)
  - The modulation type (BPSK, QPSK, 16QAM)
  - Scaling (power boost)
  - Physical location (randomization)

This table is extracted by (automatically) inserting special values at specific points in the processing chain in the Matlab simulator. Our configuration file is then generated by automated reverse engineering of the generated frame (before the FFT step). By using this method we avoid the need to implement all the complicated frame-mapping algorithms in the standard once more.

- The *filterbank coefficients* are generated by the Matlab simulator and extracted and scaled to our use.
- Smaller data tables for the interleaver are generated by Python scripts.

```
#WW 70 v0111010_1_011_0000001001 ;PREAMBLE even symbol=0 line=3 subcarrier=9
#WW 70 v0111010_0_000_0000001100 ;PREAMBLE odd symbol=0 line=4 subcarrier=12
. . .
#WW 70 v11000_0_01_010_1111111000 ;RANDOM 16QAM symbol=22 line=570 subcarrier=1016
#WW 70 v11000_0_01_011_1111111001 ;RANDOM 16QAM symbol=22 line=571 subcarrier=1017
#WW 70 v0111100_0_100_1111111010 ;PILOT + AUX_PILOT symbol=22 line=120 subcarrier=1018
#WW 70 v11000_0_01_001_1111111011 ;RANDOM 16QAM symbol=22 line=572 subcarrier=1019
#WW 70 v11000_0_01_010_1111111100 ;RANDOM 16QAM symbol=22 line=573 subcarrier=1020
#WW 70 v10110_0_00_011_1111010011 ;BURST symbol=22 line=288 subcarrier=979 sign=1
#WW 70 v10110_0_00_011_0011110001 ;BURST symbol=22 line=289 subcarrier=241 sign=0
```

Each line in a configuration file represents a write of a single value (16 or 32 bits) to a single register, RAM or FIFO in the FPGA. Thus the loading of a configuration file is similar to running a low level driver in a computer program. There are no read operations during the configuration however.

### 2.1.2.2 Downlink PUSC

In downlink direction the PUSC permutation is chosen as the test case. The signal of interest is surrounded by signals from interfering users as shown in Figure 100. **Erreur ! Source du renvoi introuvable.** The content of the OFDM symbols are listed below:

- OFDM symbol number 1: Preamble
- OFDM symbol number 2 – 7: Only pilots
- OFDM symbol number 8 – 9: Interferer 2
- OFDM symbol number 10 – 13: Interferer 5, Desired signal, Interferer 3

- OFDM symbol 14 – 15: Interferer 4
- OFDM symbol: 16 – 31: Empty

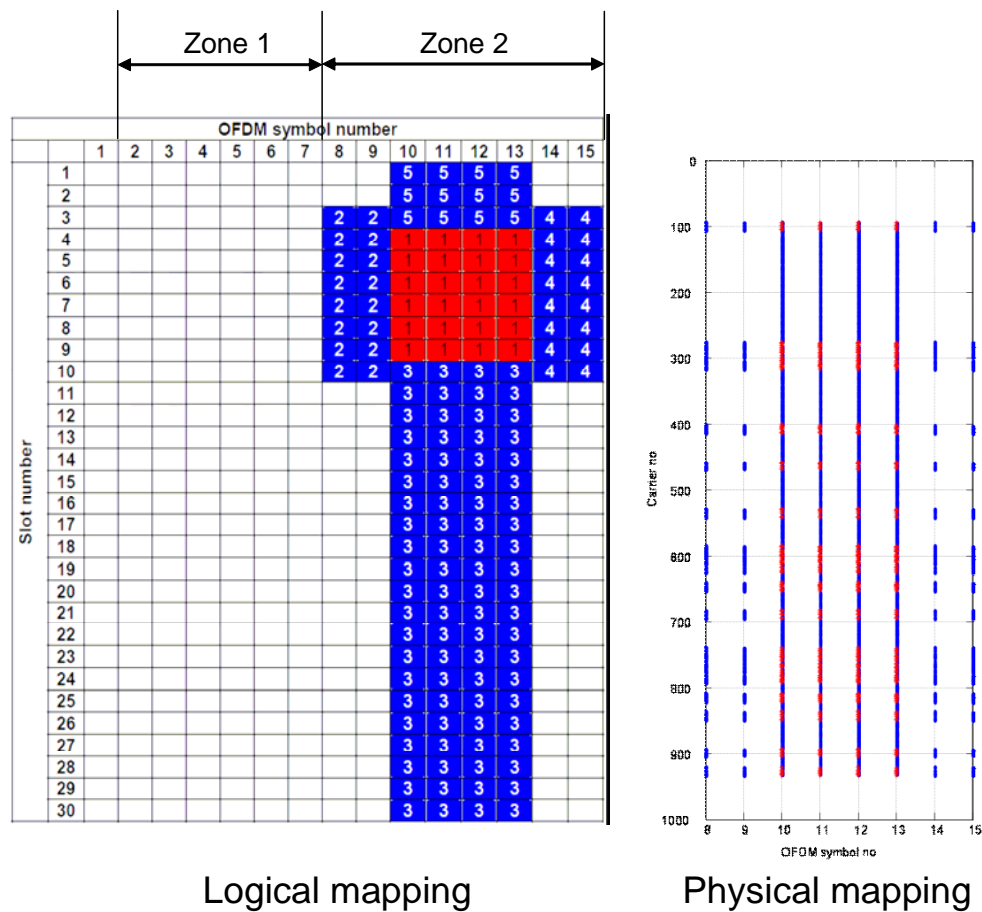
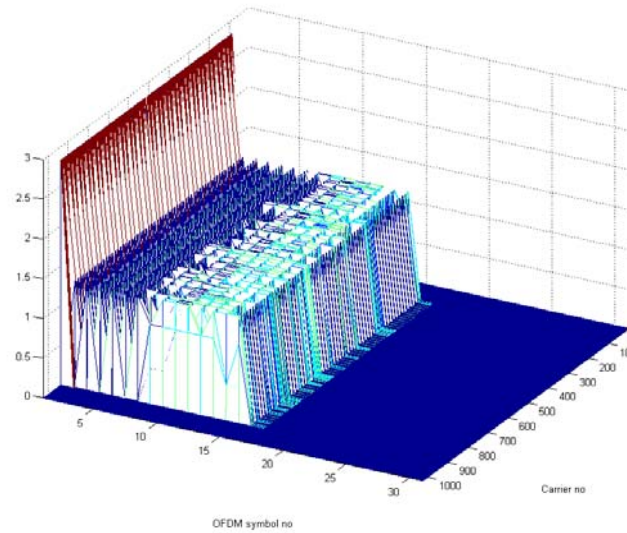


Figure 100: PUSC Signal Mapping

After physical mapping these signals are scattered over the entire bandwidth as can be seen in the right part of Figure 101.



**Figure 101: PUSC Bandwidth Occupation**

The parameters that control the logical allocation of the data is shown in Table 15, overruling parameters in the Matlab simulator are given in Table 16 to Table 18.

**Table 15: Parameters controlling data allocation in downlink PUSC**

	Desired signal	Interferer 2	Interferer 3	Interferer 4	Interferer 5
C1.MS{iS}.DATA.Diuc	1	3	3	3	3
C1.MS{iS}.DATA.OfsTslots	1	0	1	3	1
C1.MS{iS}.DATA.OfsSchn	3	2	9	2	0
C1.MS{iS}.DATA.NbTslots	2	1	2	1	2
C1.MS{iS}.DATA.NbSchn	6	8	21	8	3
C1.MS{iS}.DATA.NbSlots	12	8	42	8	6
C1.MS{iS}.DATA.PacketSize	64	64	64	64	64

**Table 16: Common overruling parameters for Down Link PUSC**

SNRoffset	7
C1.TermCond.NbFrms	100
C1.Cell{1}.Frm.CycPrefix	0
C1.Cell{1}.Frm.FBMC	false
C1.Cell{1}.Zone{2}.PermuMode	'PUSC'
C1.Cell{1}.CHN.MS{1}.FBMC.f_off	0.1
C1.Cell{1}.CHN.MS{1}.FBMC.ph_off	0
C1.Cell{1}.CHN.MS{1}.FBMC.fractional_time_delay	0.1
C1.Cell{1}.CHN.MS{1}.PreDefModel	'PEDB'
C1.Cell{1}.CHN.MS{1}.Fad.Speed_kmh	3

C1.MS{1}.DATA.Diuc	10
C1.MS{1}.DIGI.ChnEstType	40
C1.MS{1}.DIGI.FBMC.CFO_correction	'WeightedEstimation'
C1.MS{1}.DIGI.FBMC.cfo_recompensation	false
C1.MS{1}.DIGI.FBMC.FTD_estimation	'EstIIC'
C1.MS{1}.DIGI.FBMC.estimation_method	'Estimation'
C1.MS{1}.DIGI.FBMC.NbSynchIterat	0

**Table 17: Overruling parameters for Down Link PUSC FBMC**

FBMC case{100}%QPSK 1/2	
C1.Cell{1}.Frm.CycPrefix	0
C1.Cell{1}.Frm.FBMC	true
C1.Snr_dB	[-7:12]+SNROffset
C1.Cell{1}.Frm.PreambleType	'WiMAX'
C1.Cell{1}.Frm.NbPreambleSyms	1
C1.Cell{1}.Frm.pilot_method	'AuxPilots'
C1.MS{1}.DATA.Diuc	1
C1.MS{1}.DATA.OfsTslots	1
C1.MS{1}.DATA.OfsSchn	3
C1.MS{1}.DATA.NbTslots	2
C1.MS{1}.DATA.NbSchn	6
C1.MS{1}.DATA.NbSlots	12
C1.MS{1}.DATA.PacketSize	64
C1.Cell{1}.Zone{1}.OfsOsym	1

**Table 18: Overruling parameters for Down Link PUSC WiMAX**

WiMAX case{200}%QPSK 1/2	
C1.Cell{1}.Frm.FBMC	false
C1.Snr_dB	[-7:12]+SNRoffset
C1.Cell{1}.Frm.PreambleType	'WiMAX'
C1.Cell{1}.Frm.NbPreambleSyms	1
C1.MS{1}.DATA.Diuc	1
C1.MS{1}.DATA.OfsTslots	1
C1.MS{1}.DATA.OfsSchn	3
C1.MS{1}.DATA.NbTslots	2
C1.MS{1}.DATA.NbSchn	6
C1.MS{1}.DATA.NbSlots	12
C1.MS{1}.DATA.PacketSize	64
C1.Cell{1}.Zone{1}.OfsOsym	1

### 2.1.2.3 Uplink AMC

In uplink direction the AMC permutation is chosen as the test case. The uplink zone covers 15 OFDM symbols. The allocation of the signal in the frame is shown in Figure 102. The overruling parameters for both WiMAX and FBMC are given in **TABAAA** to **TABCCC**.

#### Uplink AMC

- The UL signal covers
  - 15 OFDM symbols in time
  - 54 subcarriers in frequency
  - 11 slots
- 18 x 3 carriers per slot incl 6 pilots

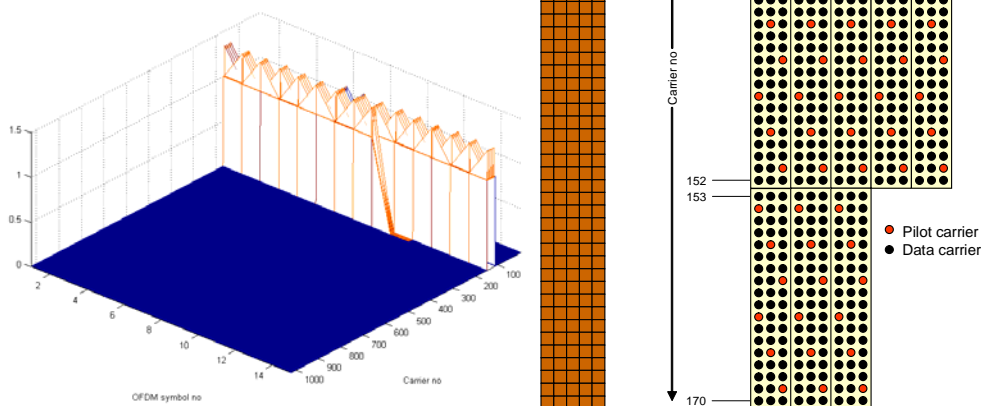


Figure 102: Uplink AMC Permutation

**Table 19: Common overruling parameters for AMC Up Link**

C1.SNR_Modoffset	0
C1.Cell{1}.Frm.CycPrefix	0
C1.Cell{1}.Frm.FBMC	true
C1.Cell{1}.Frm.proto_opt	1
C1.Cell{1}.Frm.overlapping_factor	4
C1.Cell{1}.Frm.excess_ratio	1
C1.Cell{1}.Frm.rcos_ratio	0.1
C1.Cell{1}.Zone{1}.PermuMode	'AMC23'
C1.MS{1}.CHN.FBMC.f_off	0.0
C1.MS{1}.CHN.FBMC.ph_off	0
C1.MS{1}.CHN.FBMC.fractional_time_delay	0.0
C1.MS{1}.DATA.Uiuc	8
C1.BS{1}.DIGI.FBMC.CFO_correction	'PCI'
C1.BS{1}.DIGI.FBMC.FTD_estimation	'PCI'

**Table 20: Overruling parameters for Up Link FBMC AMC**

C1.MS{1}.DATA.Uiuc	1
C1.Cell{1}.Frm.pilot_method	'AuxPilots'
C1.Snr_dB	$[-5:10] + \text{C1.SNR\_Modoffset}$
C1.Cell{1}.Frm.excess_ratio	-0.1
C1.Cell{1}.Frm.rcos_ratio	-0.1

**Table 21: Overruling parameters for Up Link WiMAX AMC**

C1.Cell{1}.Frm.CycPrefix	1/8;
C1.Cell{1}.Frm.FBMC	false;
C1.MS{1}.DATA.Uiuc	1;
C1.Cell{1}.Frm.pilot_method	'AuxPilots';
C1.Snr_dB	$[-5:10] + \text{C1.SNR\_Modoffset};$
C1.Cell{1}.Frm.excess_ratio	-0.1;
C1.Cell{1}.Frm.rcos_ratio	-0.1;

## 2.1.2.4 Transmitter Development Status

The physical hardware is ready. It consists of a commercial FPGA board with a custom made interface. Two copies have been made and assembled in laboratory cases for easy transportation.



**Figure 103: Transportable Transmitter**

Currently the cable connection to the fading simulator is being tested and verified. The transmitter generates test patterns that are received and checked on the fading emulator side.

All FPGA firmware code (VHDL code) has been written except for the CTC data coder. (The CC encoder is currently used instead) The Matlab code that extracts configuration data from the simulator has been written.

All the code is up and running. At the moment the code is being debugged and verified to obtain correct output data. Output from simulation of the VHDL code is being verified against the Matlab simulator step by step in the processing chain.

We have had some examples of AMC FBMC correctly through the whole chain and are now working on the helper pilot calculation step in the PUSC downlink scenario. The remaining issues are details and no known serious obstacles remain.

---



## 2.2 Channel Emulator

The Agilent N5106A PXB MIMO Receiver Tester is a commercially available instrument that can emulate complex MIMO channels to test wireless systems that operate in a high multipath environment. The PXB can use Agilent's N5183A MXG signal generator or an internally generated baseband signal as transmitter source. After the signal generation, the channel characteristics are applied onto this signal to provide an accurate presentation of realistic multipath channels that can be used to test MIMO components in a receiver.

This chapter describes the integration of the Agilent N5106A PXB MIMO Receiver Tester with the real-time WiMAX/FBMC transmitter from Sintef and the adjustment that were needed on the PXB to use an external non-Agilent transmitter as source for signal generation.

### 2.2.1 N5106A PXB Setup

A channel emulator, such as the PXB, that replicates real-world MIMO conditions using powerful digital signal processing technology will provide a quick path for troubleshooting advanced radio components and systems. The channel emulator also has the advantages that it can generate realistic fading scenarios including path and channel correlations.

The PXB provides up to 8 faders useful for testing and troubleshooting up to 4x2 MIMO systems.

**Erreur ! Source du renvoi introuvable.** shows a simplified configuration diagram for testing a 2x2 MIMO transmitter and receiver using the PXB. Its internal faders can be independently configured with a standards-compliant fading model, such as a WiMAX ITU Pedestrian B, or custom configured model using a variety of path and fading conditions.

It will be possible that all fading scenarios that are handled by the simulator can also be applied by the channel emulator.

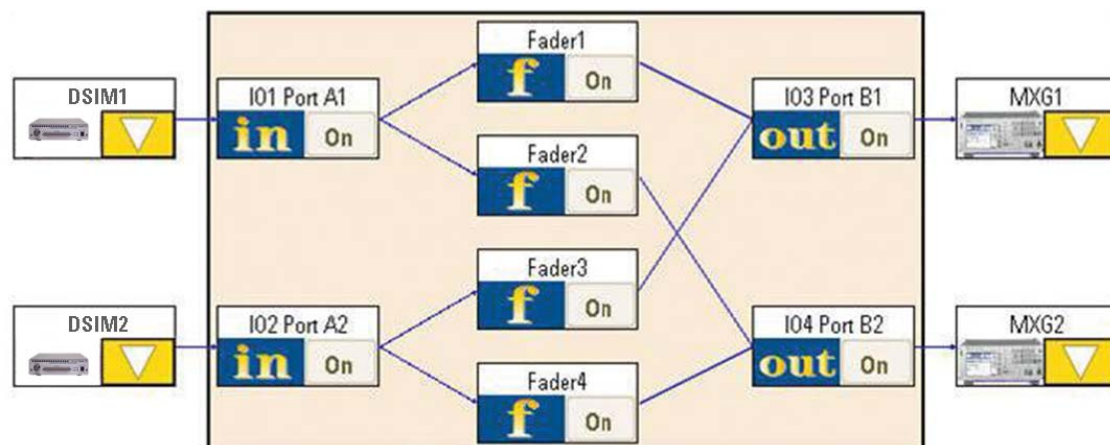


Figure 104: PXB 2x2 MIMO with external input

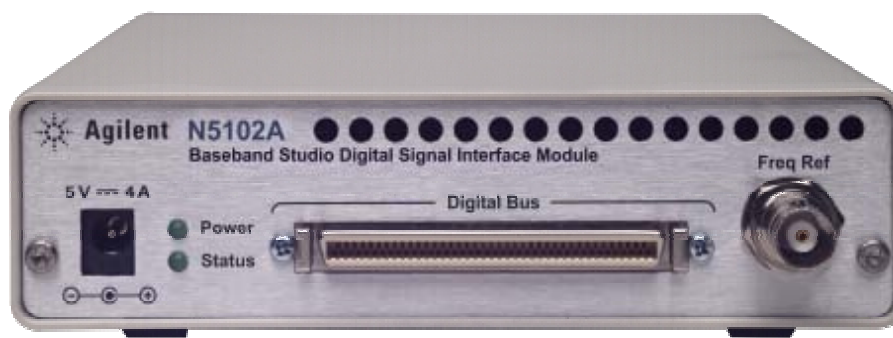
As stated before, the PXB channel emulator has to take externally generated signals as inputs for its channel faders. These signals are composed of 16-bit digital baseband I and Q samples at 44.81024MHz and are 2.5Vpp and single ended.

But because the commercially available PXB can only handle LVDS signal with an Agilent proprietary physical interface and protocol, the N5102A DSIM is used to convert these signals and interface them to the emulator.

At the output of the emulator, Agilent's N5183A MXG signal generator is connected to convert the LVDS signal to analog I/Q or an up converted version of the same signal to an arbitrary frequency.

### 2.2.2 N5102A DSIM

The Digital Signal Interface module or DSIM is an instrument that was design to convert digital signals, coming from a custom source, to an LVDS signal that could be used as input on Agilent equipment such as the N5183A MXG, the E4438C ESG or the E8267PSG, all signal generators. The module can also be used the other way round to convert Agilent's LVDS signals to digital signals that can be captured by a custom receiver or to be used to stress a Device Under Test (DUT).



**Figure 105: N5102A DSIM**

The N5102 module delivers the digital IQ or digital IF singals to a custom device with the data requirements, clock features and signaling the user needs. With its selection of several logic types and different break-out board connectors, the interface module can connect to wide variety of test systems, in most cases eliminating the need for custom fixtures.

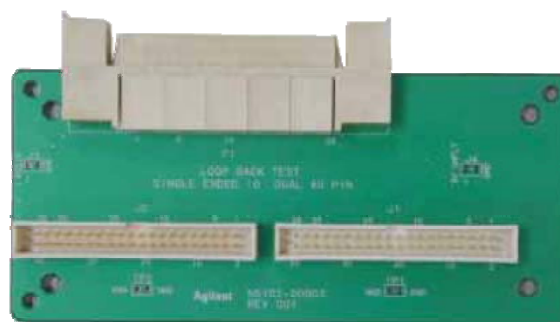
The N5102A module provides many features:

- Output mode
- Input mode
- Bit level access to data from a wide range of signal creation applications
- Simple user interface
- Flexible data formats
  - Variable 4- to 16-bit word size
  - Serial, parallel and parallel interleaved data transmission
  - 2's complement or offset binary word presentation
  - MSB or LSB bit order
  - Digital IQ or digital IF signals
- Flexible clocking
  - Automatic resampling
  - 1kHz to 100MHz sample rates
  - Multiple clock inputs and outputs
  - Adjustable clock phase, skew and polarity
  - Multiple clocks per sample
- Flexible signal interface
  - Multiple logic types, single ended or differential
  - Interchangeable connector break-out boards

To set these parameters, the user interface (UI) on the connected MXG/ESG/PSG signal generators is normally used. But for the demonstrator in the Phydyas project we used a PXB channel emulator and therefore we had to develop and extend the UI software and firmware of the PXB so that it could talk to the interface module.

To connect to the WiMAX/FBMC Transmitter that is developed by Sintef, we use the following list of parameters:

- Input mode
- Parallel interleaved
- Word size: 16-bit
- 2's complement
- Digital IQ
- Sample rate: 44.81024MHz
- Single ended 2.5V
- Dual 40-pin break-out board



**Figure 106: Dual 40-pin break-out board**

Only one connector of the break-out board is used because the DSIM is configured in the parallel interleaved mode. This means that I- and Q-samples are interleaved on the I-connector, meaning that if the I-sample is sent on the rising edge of the clock, the Q-sample will be sent on the falling edge (see Figure 107).

Note that the figure shows a 4-bit word, but in the demonstrator we will use a 16-bit interface.

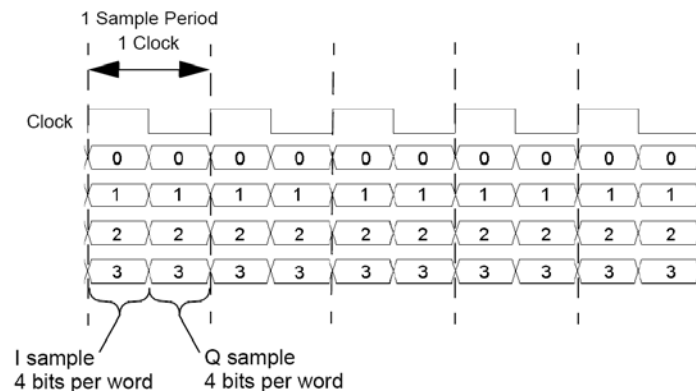


Figure 107: IQ Interleaving

At the start of the Phydias project, there was no digital interface available on the PXB. The integration of the DSIM and the PXB was done in the context of the project. As a result, the research and the developments for this interface were handed over to product developers and became publicly available.

### 2.2.3 TX-DSIM Digital Interface

To connect the high speed Samtec header on the transmitter board to the digital input of the DSIM, we used Hitech Global's break-out board and soldered 2 40-pin header onto it, 1 connector for each IQ Channel (see Figure 108).

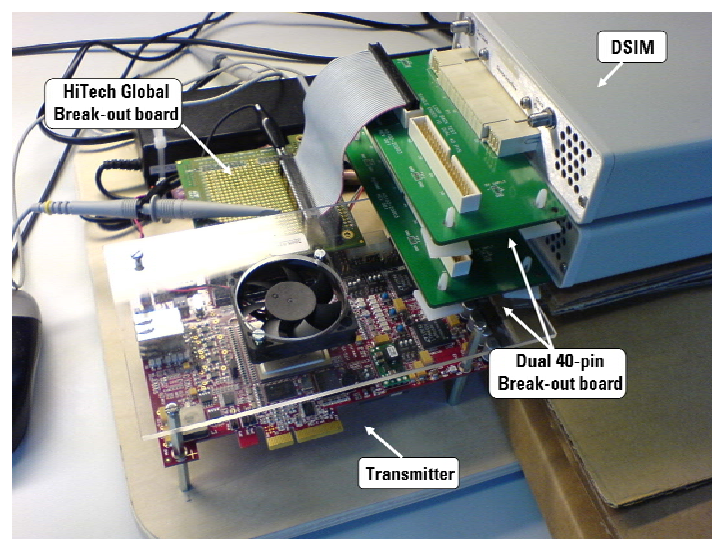
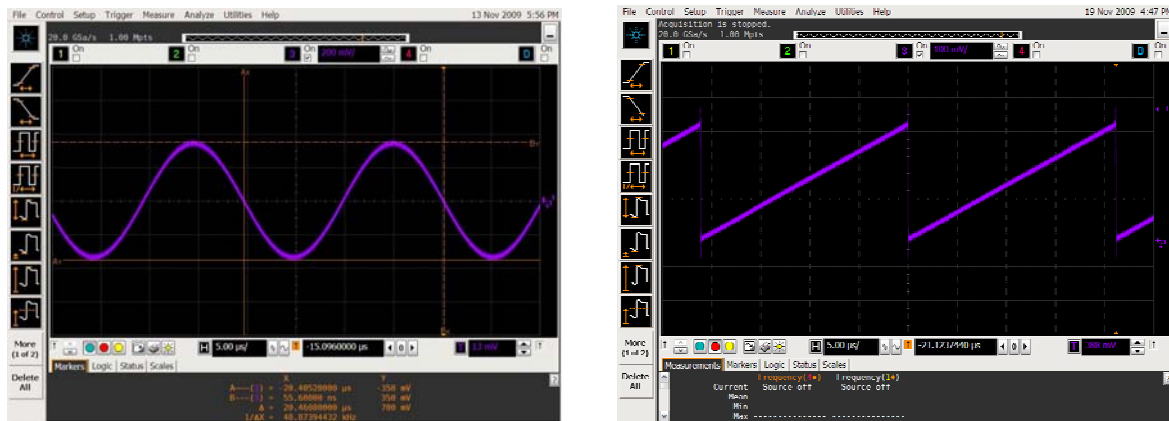


Figure 108: Transmitter - DSIM Interface

To test the interface, the transmitter generated known patterns like a sinesoid or a sawtooth (see figures below). The analog representation of those signals looked fine at first.



But closer inspection and zooming into more detail revealed that much more precaution on grounding, cable lengths and clock domain crossing has to be taken to guarantee signal integrity. Figure 109 shows how badly data was jittering with respect to clock signal. This was due to bad grounding and a longer flatcable.

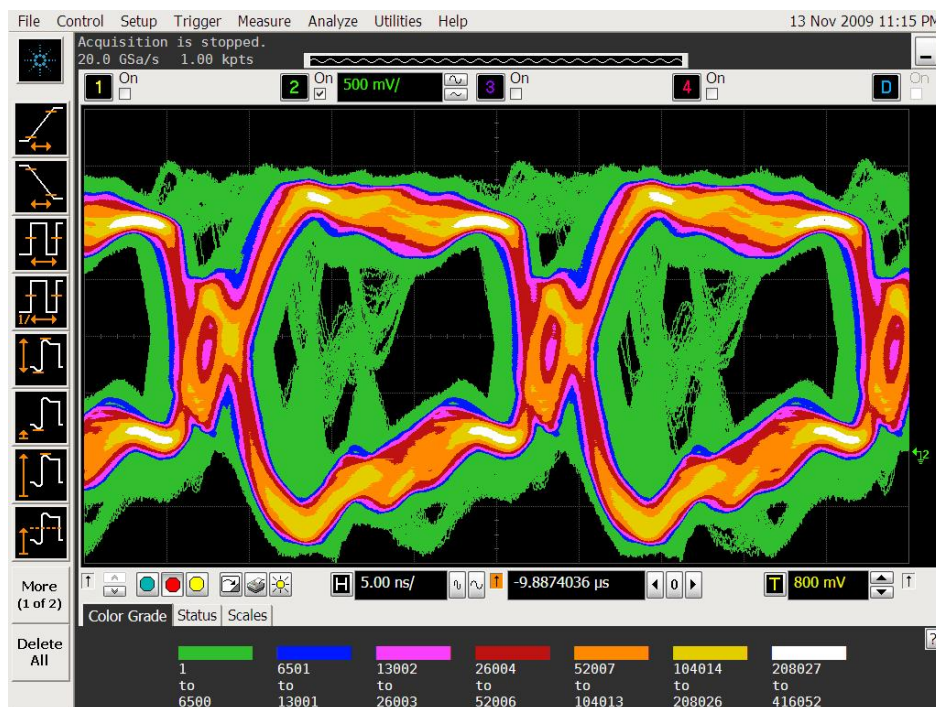


Figure 109: Jittering data



Another problem was detected by triggering the oscilloscope on one peak of the saw-tooth and looking at the next peak. This test showed that also the data was jittering, but with a period equal to the clock period (see Figure 110). It indicated that there were sometimes some samples missing. Further investigation learned that the transmitter sometimes saw several clock edges instead of one due to bad signal integrity, therefore it sends out more samples than needed. The DSIM only captured one of those samples and the rest was missed. This problem was solved by improving grounding even more.

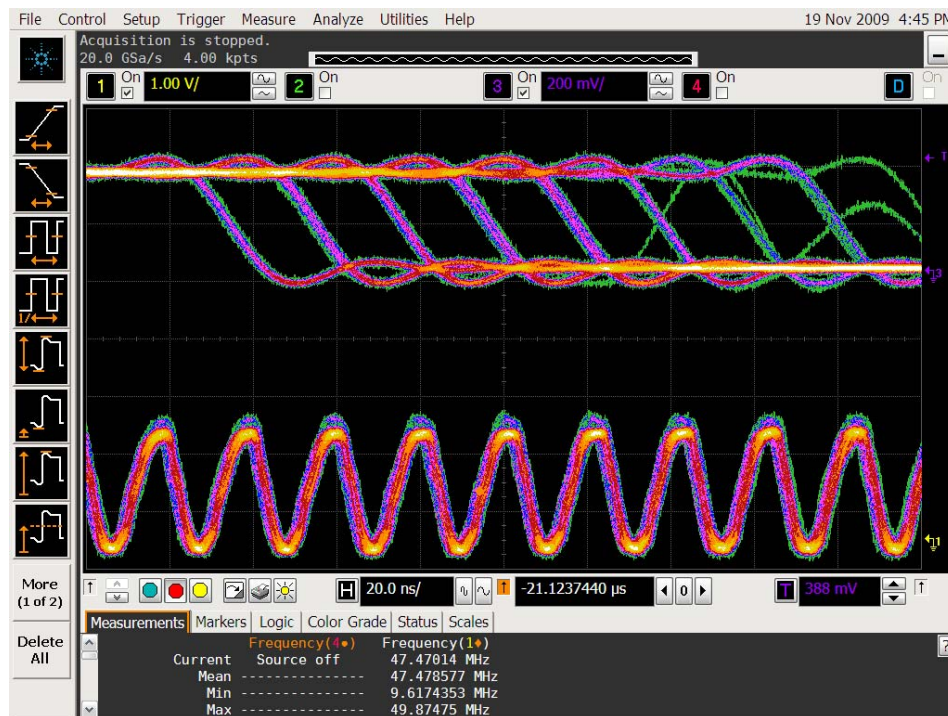


Figure 110: Missing samples

The interface between transmitter and channel emulator is working fine now, all problems are solved and further integration can continue.

## 2.2.4 N5183A MXG

The N5183A MXG Signal Generator is an analog and vector signal generator that meets the challenges facing manufacturers and designers of general purpose, cellular communications and wireless networking systems. Standard options include frequency ranges up to 6GHz, AM/FM/PM, and pulse modulation. To be up converted signals can come from the internal baseband generator, internal memory with software generated waveforms or externally generated baseband signals.

For the Phydias project, the MXG is configured in a mode where it can accept external IQ baseband signals from the PXB. For each channel we need an MXG to up convert the output of the channel emulator. In a 2x2 MIMO setup, we therefore need 2 MXGs.

The connection between the PXB Channel Emulator and the MXG Signal Generator was already available from Agilent, no extra development or tests were needed to interface those two instruments.

## 2.3 Receiver

The aim of this Section is to describe exhaustively all modules which constitute the software subsystem of the PHYDYAS concept demonstrator presented in D9.2 using C++ as the implementation language.

### 2.3.1 Module Description

#### Synchronization

This module starts the software part of the FBMC receiver. This module performs the initial FTD and CFO correction to provide the following modules with the samples to be processed.

#### Analysis Filter Bank

This module performs the analysis filter bank of the samples in the frame. This module works with the output of the previous block. The following operations are involved in this module:

Filter prototype creation.

Block by block processing. This processing consists of the polyphase filtering, the FFT and the multiplication by some weighting factors.

System delay application due to the filter bank structure.

The output of this module is organized into a 2D array of complex elements with the sent symbols organized by frequency (subcarriers) and time (symbols).

#### Equalization

This module compensates the channel effects from the output of the previous block. Several iterations can be performed in order to refine the output of this module. This module is composed by the following sub-modules:

- **Pilot Channel estimation:** The purpose of this sub-module is to generate the channel estimation at the pilot location.
- **Channel Estimation:** The purpose of this sub-module is to expand the channel estimation at the pilot location to extract whole channel estimation.
- **Channel Equalization:** The purpose of this sub-module is to equalize the input of the module with the channel estimation obtained from the previous sub-module.

The output of this module is a 2D array of complex elements organized according to frequency, time.

#### Demapping

The demapping operation is performed in order to extract the symbols and channel information from the output of the previous block. This module is composed by the following sub-modules:

- **Demodulation:** remove the OQAM modulation performed in transmission due to FBMC. Channel information from the previous module is important to obtain the weighting factors for the following entity. The channel information used for this purpose is the one obtained at the first channel estimation.
  - **FEC\_Block decomposition:** symbols and channel information are divided in blocks and grouped by slots in order to feed the decoder.
-



The outputs of this module are 2D arrays of complex elements grouped by FEC\_Blocks. These FEC\_Blocks are divided into a slot organization. The symbol information and the channel information are placed in this structure.

### Decoding

The purpose of this module is to obtain the byte stream from the symbols recovered in the previous block. This module is composed by the following sub-modules:

- **Data preparation:** previous to the decoding operations, data must be prepared. This preparations consist of two sub-modules:
  - Extraction of Symbol Data: the different elements in the slots which compose a FEC\_Block are organized into a 1D array. This operation is performed not only for symbol information but also for channel information.
  - Softbits generation: softbits are generated for each FEC\_Block according to modulation type, channel information from the previous sub-module and coding type.
- **FEC\_Block Decoding:** according to the coding type, the decoding operations are performed in different order.
  - The operations involved in convolutional decoding are *deinterleaving, depuncturing and decoding*.
  - The operations involved in convolutional turbo decoding are *depuncturing, deinterleaving and decoding*.

The output of the decoders is an array of binary values. Previous to recover the byte stream, this binary arrays has to be converted to decimal values.
- **Byte stream:** groups the byte stream of each FEC\_Block into one byte-stream.

The output of this module is an array containing the byte-stream from the input data corresponding to one frame. The length of this array depends on the parameters chosen in transmission.

## 2.3.2 Data Description

The data handled by the system is separated into three planes corresponding to the role they develop in the system.

### Signal information

These are the variables which handles the inputs and outputs for the different modules.

### Control parameters

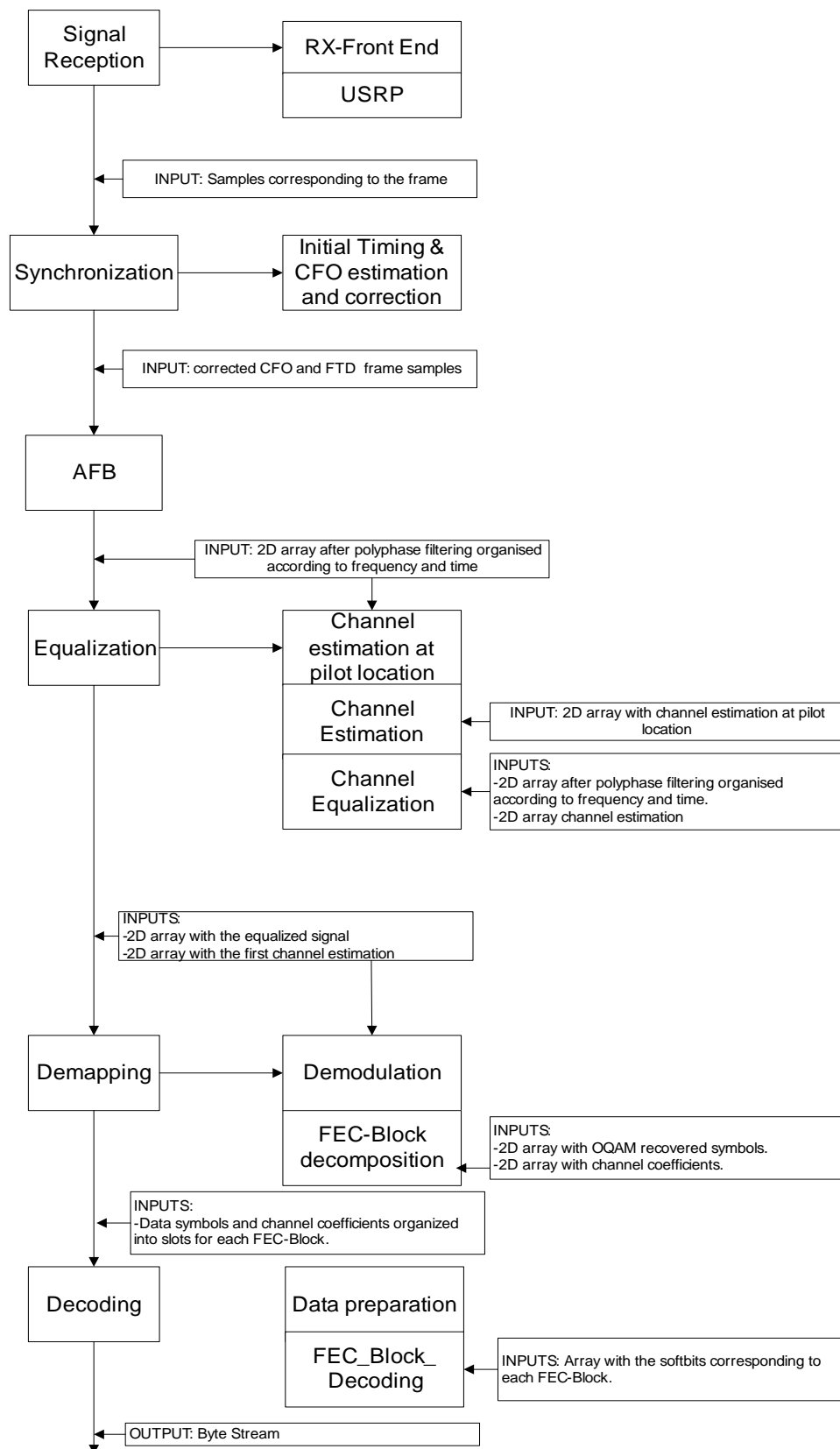
They control the way the different calculations in the modules are done. They correspond to parameters which select the type of prototype filters to be used, the frame organization and the methods to be implemented at the equalization stage between others. A comprehensive list of these parameters can be found in D9.1.

### Auxiliary parameters

These parameters are pre-calculated from the control parameters and such elements are needed for the different modules to develop its functionalities. As these values are constant in the whole

---

transmission of data, they can be calculated and stored at the beginning of the program to be available for the different modules.



**Figure 111: Module System Decomposition**

### 2.3.3 Interface Description

#### Synchronization

Input:

- Array of complex samples (I & Q components) at the output of the hardware subsystem.

Output:

- *Array of complex samples (I & Q components) with the CFO and FTD corrected.* The length of this array is equal to  $L = (M * (2 * (N + K) - 1)) / 2$ , where  $M$  is the number of points of the FFT,  $N$  the number of symbols in the frame and  $K$  the overlapping factor between filters.
- Control parameters:
  - *FFT Transform Size (M).* It is an integer value
  - *Overlapping factor (K).* It is an integer value

#### Filter Bank

Input:

- Array of complex samples (I & Q components) at the output of the synchronization module. The length of this array is equal to  $L = (M * (2 * (N + K) - 1)) / 2$ .

Output:

- 2D array of components with the symbols organized in frequency and time. The dimension is  $M \times 2N$ .
- Control parameters:
  - FFT transform size (M). It is an integer value.
  - Prototype filter type (K). It is an integer value.
  - Overlapping factor between filters (opt). It is an integer value.
- Auxiliar parameters:
  - *Filter prototype:* 2D array of double values. The dimension of this array is  $M \times K$ .
  - *RS\_multipliers:* 2 vector of size M.
- Intermediate data storing:
  - No intermediate data storing for this module is needed because in this module there is no sub-modules.

#### Equalization

Input:

- 2D array of components with the symbols in a frame organized in frequency and time. This is the output from the previous module and the dimension is  $M \times 2N$ .

Outputs:

- *2D array with the equalized signal.* 2D array of complex values of dimension  $M \times 2N$ .
- *2D array with the whole first channel estimation.* 2D array of complex values whose dimension depends on the control parameter Equalization case.
  - Equalization case = 1, the dimension is  $M \times 2N$
  - Equalization case  $\neq 1$ , the dimension is  $2M \times 2N$

Control parameters:

---

- *Iteration number*. It is an integer value.
- *Pilot boosting factor dB*. It is an integer value.
- *Pilot method estimation*. Value which selects between Auxiliary Pilots and Pair of Pilots.
- *Overlapping factor*. It is an integer value.
- *Prototype filters type*. It is an integer value.
- *FFT Transform size*. It is an integer value.
- *Estimation average window*. It is an integer value.
- *Equalization case (number of taps)*. It is an integer value whose value can be 1, 2 or 3.
- *Estimation method*
- *CFO recompensation*. It is a logical value.
- *Equalization criterion*. Value which selects between ZF or MSE to define the subcarrier equalizer response.

Auxiliary parameters:

- *Matrix containing the pilot location*. It is logical matrix of  $M \times 2N$  with ones in the position of the pilots. The position of the pilots depends on the type the transmission (AMC23 or PUSC) is performed and the disposition of the FEC\_Blocks in the frame. The complete algorithm is found in the simulator function F1430\_MsPhyMapVec.
- *Vector with the active carriers*. It is an array with the indices of the active carriers in the frame. The number of active carriers depends on the permutation used and the number of points of the FFT, which define the right and left guard band. The complete algorithm is found in the simulator function F1300sub\_locate\_pilots\_FBMC.

Intermediate data storing:

- *2D array of complex values of dimension  $M \times 2N$* . This array will contain the different channel estimation at the pilot location and the whole channel estimation through the successive iterations. Important to remember that the values in this array will coincide with the values of one output if the number of iterations is 1.

## Demapping

Input:

- *2D array with the equalized signal organized in frequency and time*. This is an output from the previous module whose dimension is  $M \times 2N$ .
- *2D array with the whole first channel estimation*. 2D array of complex values whose dimension depends on the control parameter Equalization case.
- *Equalization case = 1*, the dimension is  $M \times 2N$
- *Equalization case  $\neq 1$* , the dimension is  $2M \times 2N$

Output:

- *2D arrays of data symbols organized into slots for each FEC\_Block*.
- *The distribution between FEC\_Block and slots in the user data burst is performed in the simulator function F1400sub\_FecBlockLenCalc*. This output is an array of complex elements with the dimensions  $F \times S \times O \times S_c$ , where  $F$  is the number of FEC\_Blocks,  $S$  is the number of slots in each block,  $O$  is the length of the slot in time symbols and  $S_c$  is the length of the slot in subcarriers.

- 2D arrays of channel coefficients organized into slots for each FEC\_Block. This output presents the same structure as the previous one.

Control parameters:

- Number of OFDM symbols in the frame. It is an integer value.
- Number of Active Multicarrier Symbols in the zone. It is an integer value.
- Number of Offset symbols in the zone. It is an integer value.
- *Equalization case*. It is an integer value.

Auxiliary parameters:

- *Number of FEC\_Block*. It is an integer value.
- Number of Slots per FEC\_Block. It is an integer value.
- The number of FEC\_Blocks and the number of slots in the FEC\_Blocks are calculated in the simulator function *F1400sub\_FecBlockLenCalc*.
- *Length of slot in symbols*. It is an integer value. Its value depends on the pilot permutation mode.
- *Temporal position for each slot compounding each FEC\_Block*. This is an integer value which indicates where the data for this slot start, temporally speaking. This value is needed for each slot in the FEC\_Block. The procedure to find these values is presented in the simulator function *F1410\_MsSlotPosVec*.
- *Subcarriers indexes in each slot of each FEC\_Block*. Each slot in each FEC\_Block has a vector of integer values which contains the subcarriers that are used by this slot. The length of this array is an integer value which depends on the pilot permutation mode. The procedure to find these values is presented in the simulator function *F1430\_MsPhyMapVec*.
- In case the transmission mode is AMC23, previous to obtain the subcarriers indexes, the frequential position of the subchannel has to be assigned to the slot, similarly to the temporal position. The procedure to find the subchannel position for a slot is determined in the simulator function *F1410\_MsSlotPosVec*.
- In case the transmission mode is PUSC, the tile mapping parameter is necessary. The procedure to find these values is included in the simulator function *F1320\_IdxLog2Phy*. It exists an integer value for ea
- ch used subchannel.

Intermediate data storing:

- *2D array of complex values of dimension  $M \times N$* . This array will contain the data information after the demodulation sub-module.
- *2D array of complex values of dimension  $M \times N$* . This array will contain the channel information after the demodulation sub-module.

## Decoding

Input:

- 2D arrays of data symbols organized into slots for each FEC\_Block
  - 2D arrays of channel coefficients organized into slots for each FEC\_Block
  - Both inputs present the type of data and the dimensions specified for the output of the “Demapping” module.
-

## Output:

- *ID array which contains the byte stream of the user data in the frame.* This is a 1D array of integer elements comprised between 0-255 (byte format). The dimension of this array is proportional to the number of data symbols per slot divided by 8, the total number of slots in the burst (sum of the slots which compose each FEC\_Block), the modulation type and the coding rate.

## Control parameters:

- Number of data symbols per slot. It is an integer value.
- *Transmission mode.* Value which selects between AMC23 or PUSC.
- Scaling bits to control CTC decoding resolution. It is an integer value
- Number of iterations CTC decoding. It is an integer value
- *CTC Scaling factor.* It is an integer value
- *Packet Size.* It is an integer value

## Auxiliary parameters:

- *Number of FEC\_Blocks.* It is an integer value
  - Number of Slots per FEC\_Block. It is an integer value
  - *Data permutation indexes.* The data elements present in each slot have been permuted in transmission. This array of integer values contains the permutation order. Its dimension is equal to the number of data symbol per slot. Each slot has its own data permutation indexes. The procedure to find this array is determined in the simulator function *F1420\_MsInSlotMapVec*.
  - In case the transmission mode is PUSC, the data permutation indexes are found from the value of the subcarrier rotation for PUSC. There is an integer value for each slot which defines the subcarrier rotation for PUSC.
  - *Modulation Type.* It is an integer value which identifies between QPSK (2), 16QAM (4) and 64QAM (6).
  - *Coding Type.* It is a value which selects between CC (convolutional coding) and CTC (convolutional turbo coding).
  - *Coding Rate.* It is a double value.
    - The modulation type, the coding type and the coding rate are found from the value of one parameter through the simulator function *Fmulti\_FecTypeBreakDown*. The way these parameters are calculated in the simulator is quite convenient because by changing one value, the previous three parameters are changed.
  - *Array with interleaving indexes for CTC encoding.* An array of this type exists for each FEC\_Block. It is an array of integer values whose dimensions depends on the number of slots in the FEC\_Block, the number of data symbols per slot, the modulation type, the coding rate and the number of elements the FEC\_Block is divided to feed the decoder. The complete algorithm to find this array of values is found in the Matlab simulator function *F1400sub\_IdxSubInterleaver*.
-

- *P parameters for CTC encoding.* An array of this type is necessary for each FEC\_Block. It is an array of four integer elements necessary for the turbo decoding. The algorithm to find this array of values is found in the simulator function *F1400sub\_IdxCtcInterleaver*.
- *Number Bytes per Slot.* It is an integer value whose value is proportional to the number of data symbols per Slot divided by 8, the modulation type and the coding rate.

Intermediate data storing:

- *1D Array of complex values to store the data elements in a FEC\_Block.* The length of this array is proportional to the number of slots in a FEC\_Block and the number of symbols in the slot. One array of this kind is needed for each FEC\_Block.
  - *1D Array of complex values to store the channel coefficients in a FEC\_Block.* The length of this array is proportional to the number of slots in a FEC\_Block and the number of symbols in the slot. One array of this kind is needed for each FEC\_Block.
  - *1D array of complex values to store the softbits.* The length of this array  $M_t \times N_s \times N_{Sb}$ , where  $M_t$  is a value proportional to the modulation type (2 for QPSK, 4 for 16-QAM and 6 for 64-QAM),  $N_s$  is the number of slots in a FEC\_Block and  $N_{Sb}$  is the number of data symbols per slot. One array of this kind is needed for each FEC\_Block.
  - In case of convolutional turbo decoding:
    - *1D array to store the result of the depuncturing and the deinterleaving operation.* This array is constituted by non-integer real values and its dimension is three time the product of the number of slots in the FEC\_Block, the number of data symbols per slot, the modulation type and the coding rate. One array of this kind is needed for each FEC\_Block.
  - In case of convolutional decoding:
    - 1D array to store the result of the deinterleaving operation. Non-integer real values. Its dimension is proportional to the number of slots in the FEC\_Block and the number of data symbols per slot.
    - 1D to store the result of the depuncturing operation. The length of this element depends on the coding rate. One array of this kind is needed for each FEC\_Block.
  - *1D array to store the result of the convolutional/turbo decoding.* It is constituted by binary information. The length of this element is proportional to the number of slots in the FEC\_Block and the number of data symbols per slot. One array of this kind is needed for each FEC\_Block.
-



## 2.4 2x2 MIMO Decoder

This section describes the VHDL implementation of COMSIS RxV1.2h FPGA-based 2x2 Decoder. Details of the algorithm are provided in PHYDYAS deliverable D9.2 MIMO decoder section. This generic architecture is based on the MIMO-OFDM prototype and it can be integrated in wireless systems using MIMO-OFDM technique, for example 802.11n and WiMAX. Several interface parameters need to be defined to meet the multi-carriers specification and latency constraints. In addition, FMBC operation mode is reserved for future updates.

In this report, we present the principal components of the MIMO decoder. Both the interface and general architecture are given at each functional block. For the sake of readers' convenience, we will focalize on the channel processing and word processing by neglecting the trivial control procedures.

### 2.4.1 General Architecture

This MIMO decoder implementation operates at clock 80MHz and it supports the constellations of BPSK, QPSK, 16-QAM and 64-QAM. The throughput is up to 130Mbps with COMSIS' 802.11n test bench. This MIMO decoder operates for both  $N_{SS}=1$  and  $N_{SS}=2$ , specified by the control signal BLAST. This implementation is compatible with all OFDM/QAM system so it can be integrated in either WiMAX system or Wi-Fi system with trivial configuration modification. The interface of this module is illustrated in Figure 112: the inputs signals contain the channel coefficients, the received signals and decoder mode control signals.

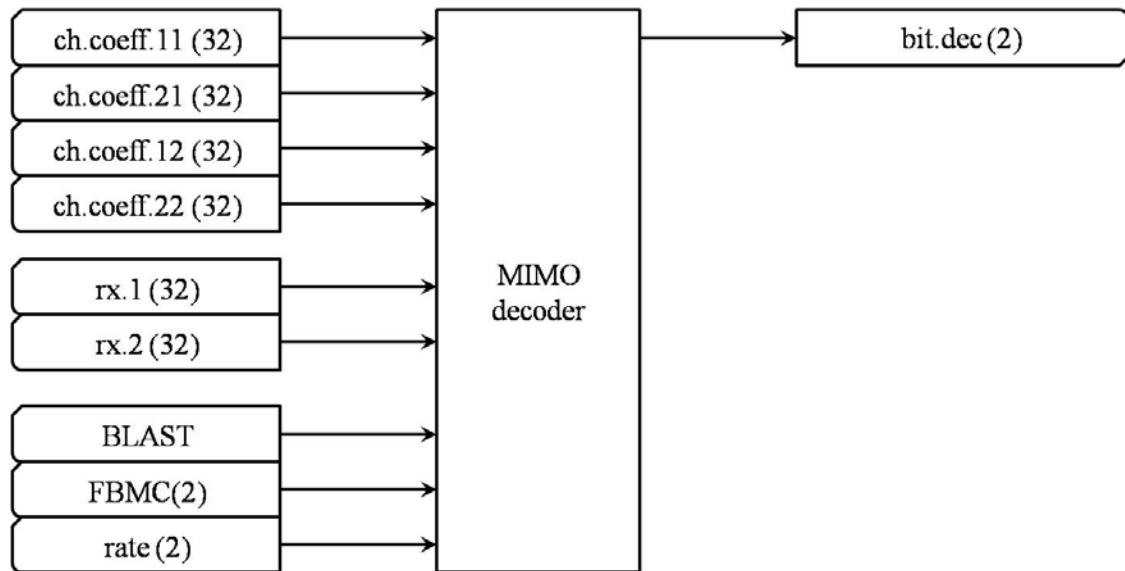


Figure 112: MIMO Decoder Interface

The channel coefficients are fed into MIMO decoder in the order of frequency index. The received signal of is processed in the pipeline manner. The mode control signals, including BLAST, rate and FMBC, need to be specified before the decoding. The signal mapping table is given in Table 22.

**Table 22: Signal Mapping of MIMO Decoder Interface**

Signal	Math. expression	Description of signal
ch.coeff.i1 (32)	$h_{1,1}$ and $h_{2,1}$	channel coefficients at receive antenna $i$
ch.coeff.i2 (32)	$h_{1,2}$ and $h_{2,2}$	channel coefficients for the second spatial stream at receive antenna $i$ : only valid in case of $N_{SS}=2$
rx.1 (32)	$y_1$	received signal at receive antenna 1
rx.2 (32)	$y_2$	received signal at receive antenna 2
BLAST		BLAST decoder control: '0' for $N_{SS}=1$ and '1' for $N_{SS}=2$
FBMC (2)		FBMC control: "0x" for mode OFDM/QAM; "10" for mode FBMC/OQAM real part, "11" for FBMC/OQAM imaginary part
rate (2)		QAM/OQAM constellation : "00" for BPSK, "01" for QPSK, "10" for 16-QAM and "11" for 64-QAM
bit.dec (2)		decoded bits: the outputs are in parallel for 2 spatial streams; in case of $N_{SS}=1$ , only bit.dec[0] is valid

For readers' convenience, signals or ports are denoted with their logical name and its number of bits in the following parentheses<sup>1</sup>. The complex signals are denoted by the real part and the imaginary part with the same precision<sup>2</sup>.

This MIMO decoder consists of *FSM controller*, *ram of channel coefficients*, *norm inversion calculation*, *Y projection and H scalar production*, *BLAST decoder* and *output stage*, as shown in Figure 113. FSM (finite state machine) controller manages the interface and coordinates the internal components. It provides a data bus for all the components inside of the MIMO decoder. The ram, whose size depends on the system configuration, memorizes the channel coefficients for the data sub-carriers which are supposed valid for several data symbols. The channel coefficients will be first processed in the norm inversion module then in the Y projection and scalar production module<sup>3</sup> in order to perform the channel processing. The receive signal is pre-processed in the Y projection and scalar production module. The intermediary results, including the projection of the received signal and the normalized channel coefficients, are fed into BLAST decoder. The output stage module controls the decoding results and the interface with external module.

<sup>1</sup> For the signals of one bit, only their logical name is given.

<sup>2</sup> For example, rx.1(32) means the receive signal  $y_1$  is denoted by 32 bits with 16 bits for the real part and 16 bits for the imaginary part.

<sup>3</sup> The function of scalar production is activated during channel processing, while the function of projection is active during word processing.

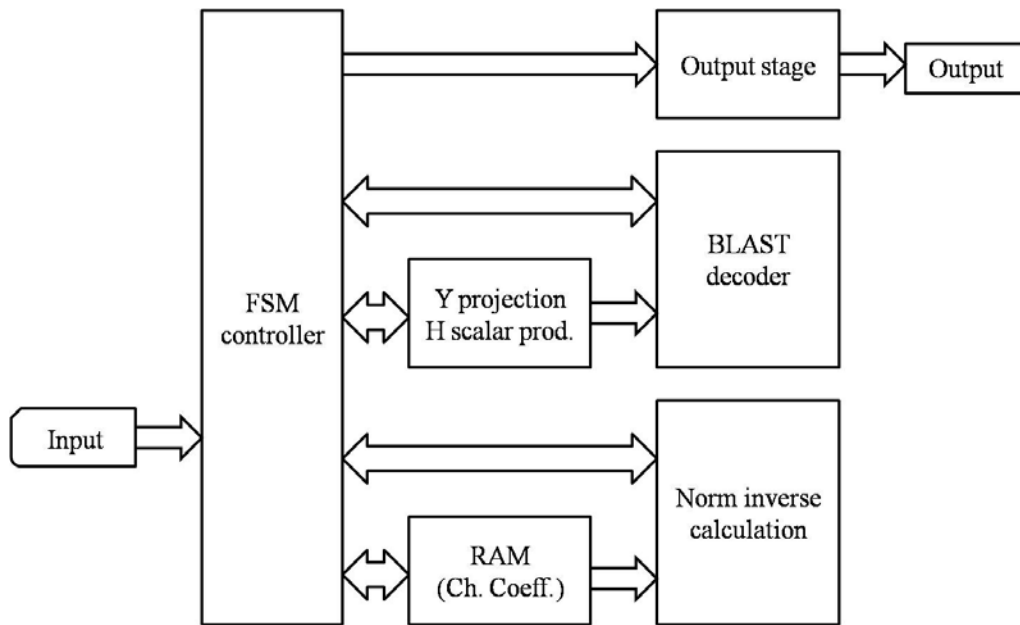


Figure 113: Global MIMO Decoder Architecture

### 2.4.1.1 Norm Inversion Calculation

In the norm inversion calculation module calculates the normalization coefficient as well as the SWAP signal for the channel processing procedure. This module receives the channel coefficients from the ram module and reports to the FSM the final results which will be used in the Y projection and H scalar production module, as shown in Figure 114.

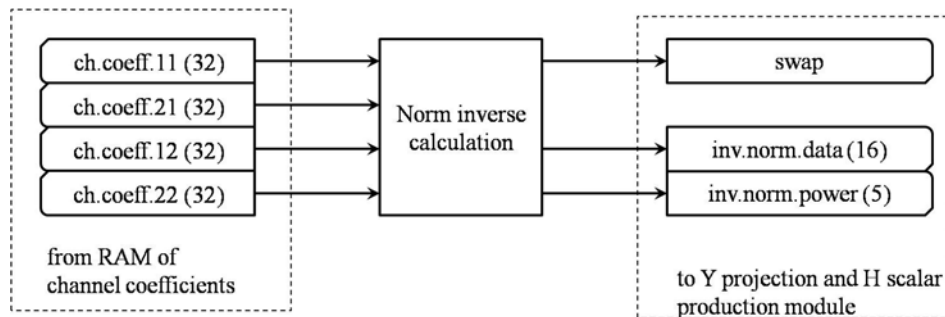


Figure 114: Norm Inversion Calculation Interface

The norm inverse calculation module contains 2 units of calculation: the norm calculation unit and the inversion unit, as shown in Figure 115.

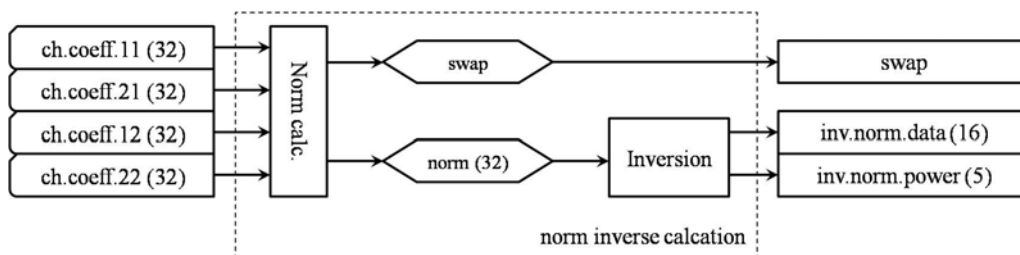


Figure 115: Norm Inversion Calculation Architecture

An illustration of the norm calculation unit is given in Figure 116. The channel coefficients are grouped according to the column index by  $(h_{1,1}, h_{2,1})$  and  $(h_{1,2}, h_{2,2})$ . The nom of each column is calculated in such way:

$$\begin{cases} N_1 = |\text{Re}(h_{1,1})|^2 + |\text{Im}(h_{1,1})|^2 + |\text{Re}(h_{2,1})|^2 + |\text{Im}(h_{2,1})|^2 \\ N_2 = |\text{Re}(h_{1,2})|^2 + |\text{Im}(h_{1,2})|^2 + |\text{Re}(h_{2,2})|^2 + |\text{Im}(h_{2,2})|^2 \end{cases}$$

In the implementation, the SEL function selects the real part or the imaginary part of the channel coefficient and transfer it to the square calculation unit. The square results will be summed in the accumulator which gives  $N_1$  and  $N_2$ .

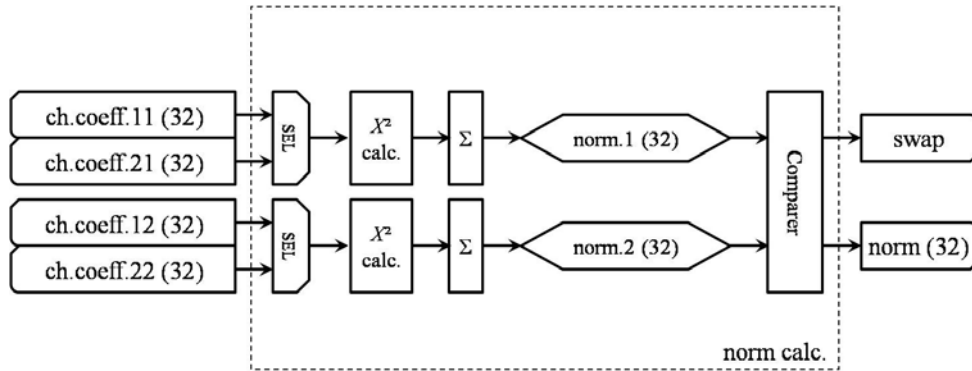


Figure 116: Norm Calculation Unit Architecture

The norm results  $N_1$  and  $N_2$  are compared and the SWAP signal is given as well as the  $N_{\max}$ . The  $N_{\max}$  is fed into the inversion unit where the inversion is performed with a looking table. The inverse result is denoted in a floating manner in order to provide larger dynamic range of notation: 16 bits denote the significant denoted by  $a$  and 5 bits for the exponent denoted by  $b$ . The relation of  $N_{\max}^{-1}$  and  $(a, b)$  is given by:

$$N_{\max}^{-1} = a \cdot 2^b$$

The signal mapping is described in Table 23: Signal Mapping of Norm Inversion Calculation

Table 23: Signal Mapping of Norm Inversion Calculation

Signal	Math. expression	Description of signal
ch.coeff. $ij$ (32)	$h_{i,j}$	channel coefficients at receive antenna $i$ for spatial stream $j$
norm.1 (32)	$N_1 =  h_{1,1} ^2 +  h_{2,1} ^2$	the norm of channel matrix first column
norm.2 (32)	$N_2 =  h_{1,2} ^2 +  h_{2,2} ^2$	the norm of channel matrix second column
swap	$SWAP$	SWAP indicator: '0' if $N_2 > N_1$ and '1' otherwise
norm (32)	$N_{\max}$	the maximal column norm of channel matrix
inv.norm.dat (16)	$N_{\max}^{-1}$	inverse of the maximal norm, denoted in floating manner
inv.norm.power (5)		

### 2.4.1.2Y Projection an H Scalar Production

The Y projection and scalar production module participates in both channel processing and word processing in order to reuse the calculation unit. The interface to the other modules is given in Figure 117: Y Projection and H Scalar Production. During channel processing phase, it receives the channel coefficients from the ram and  $N_{\max}^{-1}$  with swap indicator from the norm inverse calculation module. During word processing phase, it receives  $y_1$  and  $y_2$  from the FSM controller.

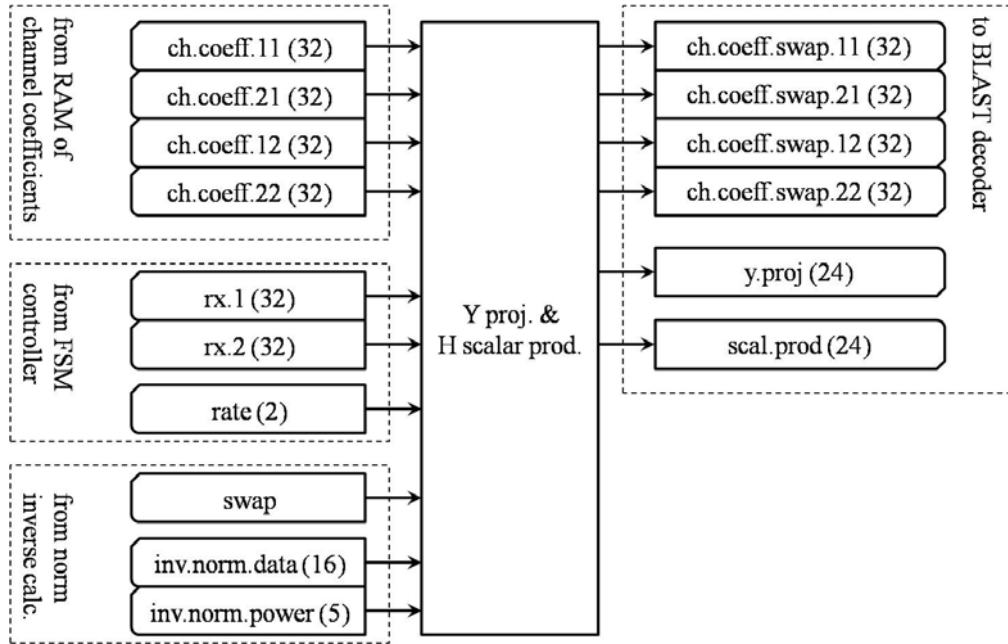


Figure 117: Y Projection and H Scalar Production

It generates both the normalized channel coefficients and Y projection in function of the rate signal. The general calculation structure is shown in Figure 118 and the signal mapping description is shown in Table 24.

During the channel processing phase, the channel scalar production is performed in the H scalar production unit which gives the result of  $h_{1,1+SWAP}h_{1,2-SWAP}^* + h_{2,1+SWAP}h_{2,2-SWAP}^*$ . This result is normalized by multiplying  $N_{\max}^{-1}$  at the alignment unit where a bit shift is performed according to the exponent of  $N_{\max}^{-1}$ . In the meanwhile, the channel coefficients are processed by the SWAP unit where we permute the columns of channel matrix according to the swap signal. The constellation normalization is performed before this permutation by multiplying  $K$  in function of rate parameter. For the final output, we have:

$$P = N_{\max}^{-1} \left( h_{1,1+SWAP}h_{1,2-SWAP}^* + h_{2,1+SWAP}h_{2,2-SWAP}^* \right)$$

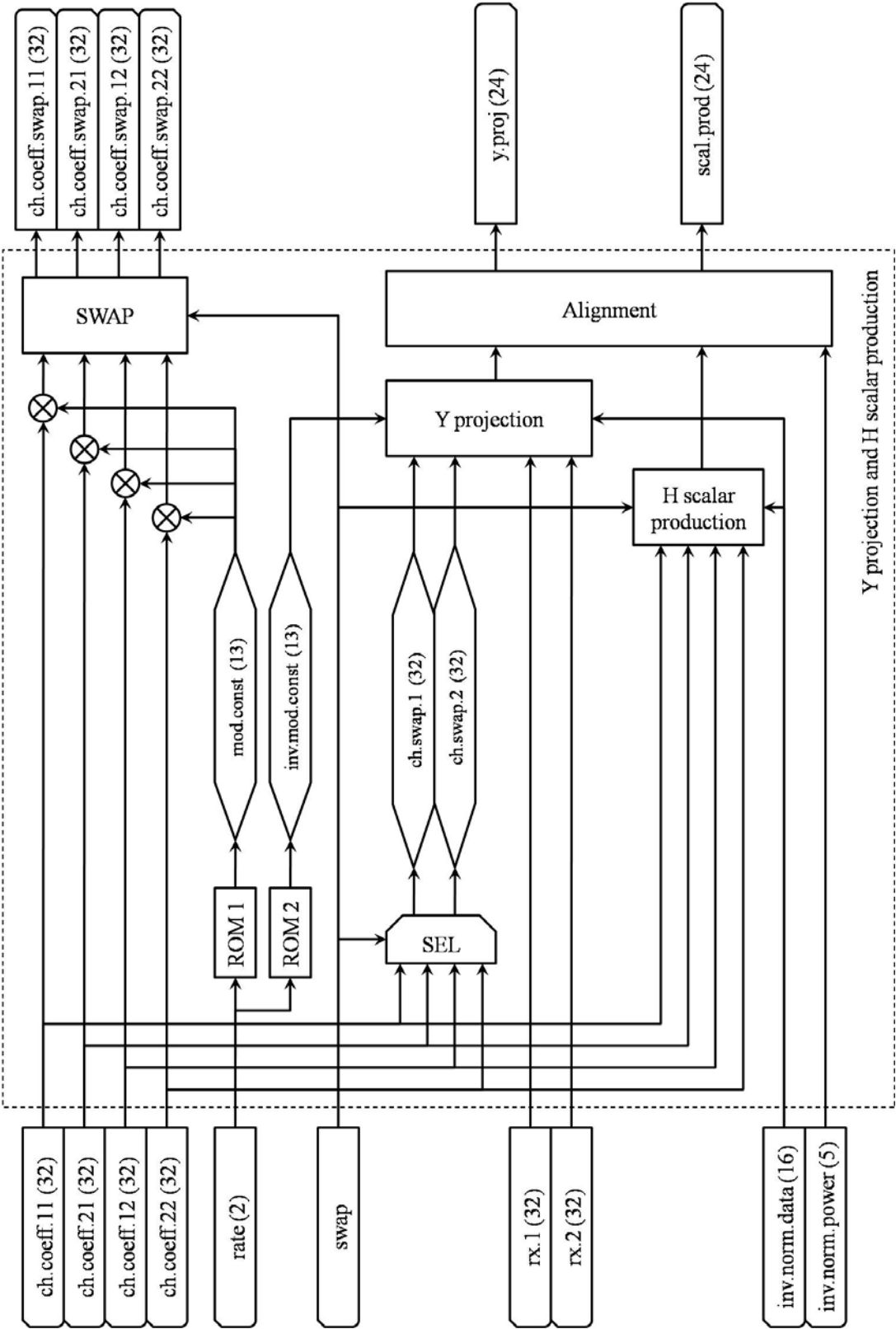


Figure 118: Y Projection and H Scalar Production Architecture

**Table 24: Signal Mapping of Y Projection and H Scalar Production**

Signal	Math. expression	Description of signal
ch.coeff.ij (32)	$h_{ij}$	channel coefficients at receive antenna $I$ for spatial stream $j$
rate (2)		QAM/OQAM constellation : “00” for BPSK, “01” for QPSK, “10” for 16-QAM and “11” for 64-QAM
swap	$SWAP$	SWAP indicator: ‘0’ if $N_2 > N_1$ and ‘1’ otherwise
rx.1 (32)	$y_1$	received signal at receive antenna 1
rx.2 (32)	$y_2$	received signal at receive antenna 2
inv.norm.dat (16)	$N_{\max}^{-1}$	inverse of the maximal norm, denoted in floating manner
inv.norm.power (5)		
mod.const (13)	$K$	constellation normalization factor
inv.mod.const (13)	$K^{-1}$	inverse of constellation normalization factor
ch.swap.i (32)	$h_{i,2-SWAP}$	channel coefficients for the projection of received signals
ch.coeff.swap.i1 (32)	$h_{i,I+SWAP}$	permuted channel coefficients in function of SWAP indicator
ch.coeff.swap.i2 (32)	$h_{i,2-SWAP}$	
y.proj (24)	$z$	projection of received signal on the channel matrix column of small norm
scal.prod (24)	$P$	scalar product of the channel coefficient matrix

During the word processing phase, the received signals  $Y_1$  and  $Y_2$  are projected with the selected channel coefficients  $h_{1,2-SWAP}$  and  $h_{2,2-SWAP}$  by  $h_{1,2-SWAP}^* y_1 + h_{2,2-SWAP}^* y_2$ . The projection result is normalized by multiplying  $K^{-1}$  and  $N_{\max}^{-1}$ : for the normalization with  $N_{\max}^{-1}$ , we reuse the alignment unit for the same operation. The final result is given by:

$$z = N_{\max}^{-1} K^{-1} \begin{bmatrix} h_{1,2-SWAP}^* & h_{2,2-SWAP}^* \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

The normalization coefficients  $K$  and  $K^{-1}$  are pre-calculated and memorized in ROM. Besides, since the complex multiplications of the channel processing and the word processing are realized with the same structure and the same precision, the Y projection and H scalar product function are combined in the same module for different operation phase.



### 2.4.1.3BLAST Decoder

The BLAST decoder received the swapped channel coefficients and processed word from the Y projection and H scalar production module. The unprocessed received signals  $Y_1$  and  $Y_2$  are provided directly from the FSM controller for the calculation of metric. The interface is shown in Figure 119.

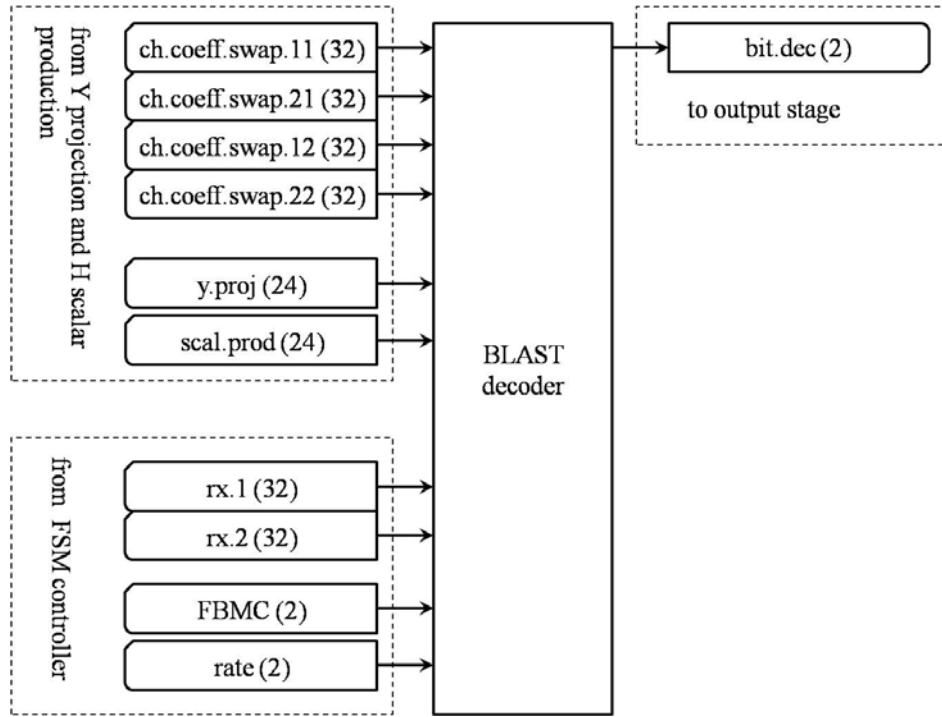


Figure 119: BLAST Decoder Interface

The BLAST decoder is enabled when  $N_{SS}=2$ . In case of  $N_{SS}=1$ , the Y projection term noted by  $z$  is directly fed into the output stage. The architecture is given in Figure 120 and the signal mapping is given in **Erreur ! Source du renvoi introuvable.**

For the implementation, we adopt a structure by placing several metric units for parallel calculation. The possible constellation points are generated by configuring the real part of  $x_1$  which is fixed at each metric unit and by scanning the imaginary part in the [x.im.control] unit. Totally, there are 8 metric units is 8, which is the full-used for 64-QAM. For other constellation modes, only a necessary number of metric units are active.

With this structure, some imaginary part related signals are calculated in order to reduce the metric unit's charge. For  $x_2$  computation, the  $x_2$  projection is given by:

$$t = z - Px_1$$

The intermediary result  $P \cdot \text{Im}(x_1)$  is a common parameter for all the metric units. This result is represented by  $\text{Re}(P) \cdot \text{Im}(x_1)$  and  $\text{Im}(P) \cdot \text{Im}(x_1)$ . Besides, the metric of the pair  $(x_1, x_2)$  is defined by:

$$M(x_1, x_2) = |y_1 - h'_{1,1}x_1 - h'_{1,2}x_2|^2 + |y_2 - h'_{2,1}x_1 - h'_{2,2}x_2|^2$$

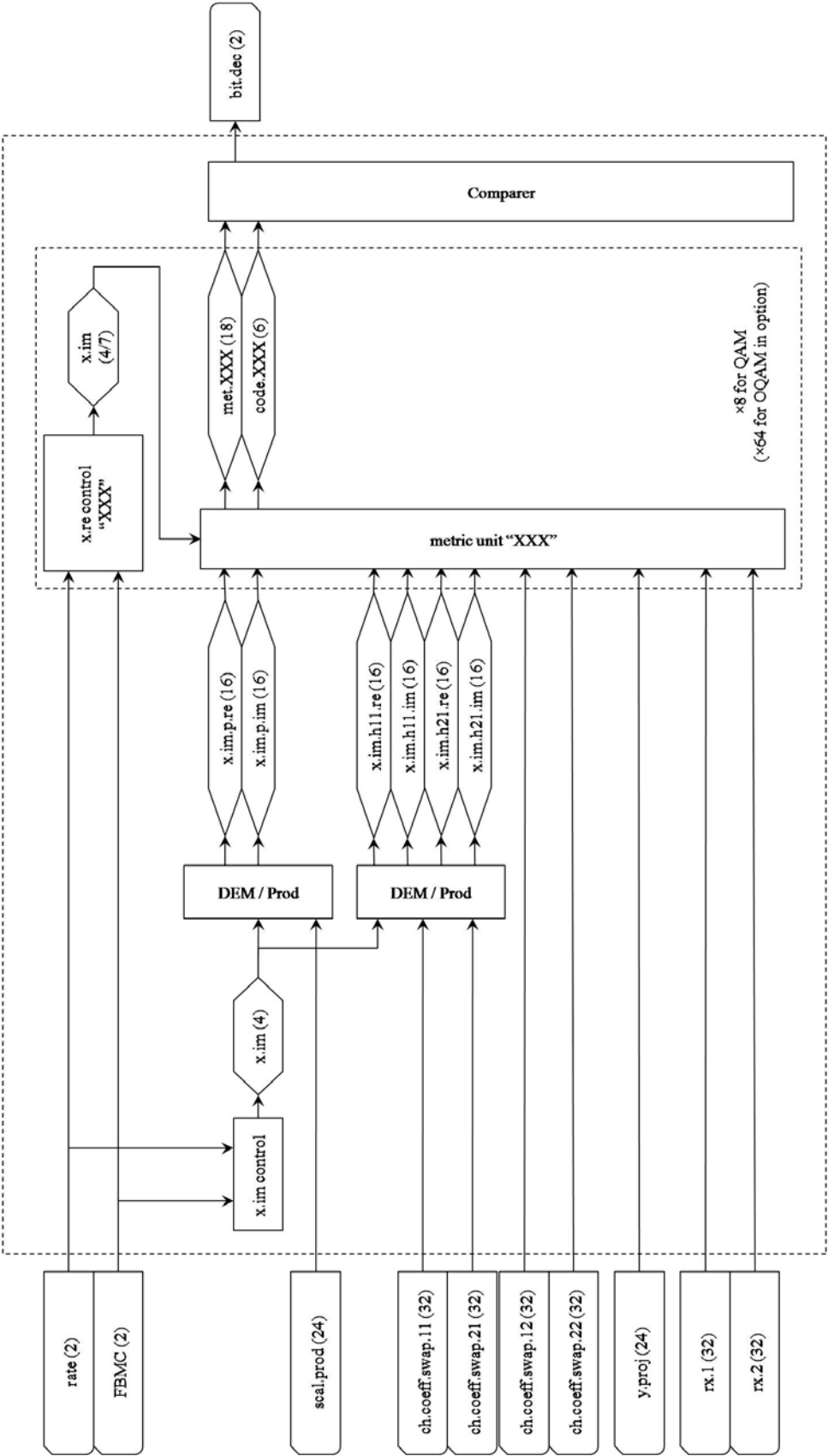


Figure 120: BLAST Decoder Architecture

The  $h'_{1,1} \cdot \text{Im}(x_1)$  and  $h'_{2,1} \cdot \text{Im}(x_1)$  are also computed before the metric calculation. The 2 products are expressed respectively by  $\text{Re}(h'_{1,1}) \cdot \text{Im}(x_1)$ ,  $\text{Im}(h'_{1,1}) \cdot \text{Im}(x_1)$ ,  $\text{Re}(h'_{2,1}) \cdot \text{Im}(x_1)$  and  $\text{Im}(h'_{2,1}) \cdot \text{Im}(x_1)$ . In the same way, once the channel parameters are available, the metric unit pre-calculates the  $P \cdot \text{Re}(x_1)$ ,  $h'_{1,1} \cdot \text{Re}(x_1)$  and  $h'_{2,1} \cdot \text{Re}(x_1)$  according to the fixed  $\text{Re}(x_1)$ .

**Table 25: Signal Mapping of BLAST Decoder Module**

Signal	Math. expression	Description of signal
rate (2)		QAM/OQAM constellation : “00” for BPSK, “01” for QPSK, “10” for 16-QAM and “11” for 64-QAM
FBMC (2)		FBMC control: “0x” for mode OFDM/QAM; “10” for mode FBMC/OQAM real part, “11” for FMBC/OQAM imaginary part
scal.prod (24)	$P$	scalar product of the channel coefficient matrix
ch.coeff.swap.i1 (32)	$h'_{i,1} (=Kh_{i,1+SWAP})$	permuted and normalized channel coefficients in function of SWAP indicator
ch.coeff.swap.i2 (32)	$h'_{i,2} (=Kh_{i,2-SWAP})$	
y.proj (24)	$z$	projection of received signal on the channel matrix column of small norm
rx.1 (32)	$y_1$	received signal at receive antenna 1
rx.2 (32)	$y_2$	received signal at receive antenna 2
x.im (4)	$\text{Im}(x_1)$	imaginary part of $x_1$
x.re (4/7)	$\text{Re}(x_1)$	real part of $x_1$ , 4 bits by default for OFDM/QAM configuration, it can be updated to 7 bits for FMBC/OQAM decoding
x.im.p.re (16)	$\text{Im}(x_1)\text{Re}(P)$	intermediary result of $Px_1$
x.im.p.im (16)	$\text{Im}(x_1)\text{Im}(P)$	
x.im.h11.re (16)	$\text{Im}(x_1)\text{Re}(h'_{1,1})$	intermediary result of $h_{1,1}x_1$
x.im.h11.im (16)	$\text{Im}(x_1)\text{Im}(h'_{1,1})$	
x.im.h21.re (16)	$\text{Im}(x_1)\text{Re}(h'_{2,1})$	intermediary result of $h_{2,2}x_1$
x.im.h21.im (16)	$\text{Im}(x_1)\text{Im}(h'_{2,1})$	
met.XXX (18)	$M(x_1, x_2)$	metric of the candidates when $\text{Re}(x_1) = \text{XXX}$
code.XXX (6)		real part of $x_1$ , 4 bits by default for OFDM/QAM configuration, it can be updated to 7 bits for FMBC/OQAM decoding
bit.dec (2)		decoded bits without swap processing

During the word processing phase, the metric unit uses the Y project results  $z$  to compute the  $x_2$  by projecting the  $t$ . With the detected  $x_2$ , the metric is calculated and communicated to the *comparer* unit. The associated Gray code is also provided at the output of each unit. In the last step, the comparer gathers the metric information and the binary codes from the metric units by examining the real part value. The minimal metric is saved as well as the corresponding codes.

For future FBMC option, more metric units can be implemented with configurable fixed real or imaginary part inputs: we propose to place 64 metric units instead of 8 for the OQAM case.

The decoded bit stream is fed into the output stage where the data bits will be eventually permuted according to the SWAP signal.

### 3 References

- [1] M. Tanda, M. Renfors, J. Louveaux, M. Bellanger, “Synchronization and initialization with single antenna. Blind Techniques”, ICT-211887, PHYDYAS deliverable D2.2, January 2009.
  - [2] F. Schaich, “WiMAX simulation results – Lab setup and measurements”, ICT-211887, PHYDYAS deliverable D9.2, July 2009.
  - [3] A. Viholainen, M. Bellanger, M. Huchard, “WP5: Prototype filter and filter bank structure”, ICT-211887, PHYDYAS deliverable D5.1, January 2009.
  - [4] M. Renfors, “Complexity Comparison”, PHYDYAS internal document.
  - [5] Stephan ten Brink, “Design of Concatenated Coding Schemes based on Iterative Decoding Convergence”, Dissertation, University of Stuttgart, Institute of Telecommunications, 2001.
  - [6] L. R. Bahl, J. Cocke, F. Jelinek, J. Raviv, “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate”, IEEE Transactions on Information Theory, Volume 20, Issue 2, March 1974, pps 284-287.
  - [7] “IEEE standard for local and metropolitan area networks. Part 16: Air interface for fixed and mobile broadband wireless access systems. Amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1,” IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004), pp. 1–822, 2006.
  - [8] WiMAX Forum “Mobile System Profile”, Release 1.0 Approved Specification (Revision 1.5.0), Nov. 2007.
  - [9] ITU-R, “Guidelines for evaluation of radio transmission technologies for IMT-2000,” Recommendation M.1225, 1997
  - [10] J. Louveaux, L. Baltar, D. Waldhauser, M. Renfors, M. Tanda, C. Bader, E. Kofidis, “Equalization and demodulation in the receiver (single antenna)”, ICT-211887, PHYDYAS deliverable D3.1, July 2008.
  - [11] F. Schaich, V. Ringset, M. Bellanger, H. Zhang, D. Le Ruyet, “Compatibility of OFDM and FBMC systems and reconfigurability of terminals”, ICT-211887, PHYDYAS deliverable D7.1, July 2009.
  - [12] H. Zhang, D. Le Ruyet, M Terre, D. Roviras, M. Renfors, T. Ihalainen, C. Bader, M. Shaat, A. Merentitis, D. Triantafyllopoulou, M. Huchard, A. Kuzminskiy, “Application of the FBMC physical layer in a cognitive radio scenario”, ICT-211887, PHYDYAS deliverable D8.1, July 2009.
-