DAFTAR PUSTAKA

[1] Guidolin, F. dan Nekovee, M. 2015, "Investigating Spectrum Sharing between 5G Millimeter Wave Networks and Fixed Satellite Systems," *IEEE Globecom Workshops (GC Workshop)*, San Diego, CA, hal. 1-7.

[2] Evans, B. G., 2014, "The role of satellites in 5G," , *the 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Livorno, hal. 197-202.

[3] Halide, L. (2020) 'Aturan Pemodelan Komputasi Untuk Eliminasi Signal Spektrum-Rf Yang Tidak Diinginkan Computational Modeling Rule For Unwanted Signal Rf-Spectrum Elimination', 7(1), Pp. 797–804.

[4] Ericsson Inc. "*More than 50 billion connected devices"*, http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf. [Online akses 27 September 2018].

[5] Payaró, M., Färber, M., Vlacheas, P., Bartzoudis, N., Tillman, F., Ferling, D., Berg, V., Rautio, T., Serrano,P., dan Sabella, D. 2016, "Flexible and efficient hardware/software platforms for 5G network elements and devices. Journal on Transactions on Emerging Telecommunication Technologies, Vol. 27, Iss. 9, hal. 1242–1249.

[6] Chae, K., dan Yoon, S., 2014, "Cancellation of AltBOC Correlation Side-Peaks for Frequency Sharing in Satellite Communication Spectrum", the journal of Korea Information and Communication Society, edisi 39B, Vol. 11, hal 810-816.

[7] Sirmayanti, S., dan Faulkner, M., 2014, "ΣΔ Modulator for Digital Wireless Architecture: A review*", IEEE MICEEI International Conference*, hal. 83-87

[8] Miao, Yao., Munawwar, M. S., Xiaofu, M., Vuk, M., dan Jeffrey, H., R.)19, "Sustainable green networking: exploiting degrees of freedom wards energy-efficient 5G systems", Journal of Wireless Network, olume 25, Issue 3, hal 951–960.

[9] Helaoui, M., Hatami, S., Negra, R., dan Ghannouchi, F. M., 2008, "A novel architecture of delta-sigma modulator enabling all-digital multiband multistandard RF transmitters design", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 55, hal. 1129- 1133.

[10] Bassoo, V., dan Faulkner, M., 2009, "Sigma-delta digital drive signals for switchmode power amplifiers", Journal Electronics Letters, vol. 44, hal. 1299-1300.

[11] Nielsen, M., dan Larsen, T., 2007, "A transmitter architecture based on delta–sigma modulation and switch-mode power amplification", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 54, hal. 735-739.

[12] Basetas, C., Kanteres, A., dan Sotiriadis, P., 2015, "Hardware implementation aspects of Multi- Step Look-Ahead Σ-Δ modulation-like architectures for all-digital frequency synthesis applications", *Joint Conference of the IEEE International Frequency Control Symp. & the European Freq. and Time Forum*, hal 452-455.

[13] Cordeiro, R., Oliveira Arnaldo S. R. dan Vieira, J., 2015, "FPGA-based all-digital transmitters", *International Conference on Field Programmable Logic and Apps. (FPL)*, hal. 1-2.

[14] Anisah, I., Briantoro, H., Zainudin, A., & Permatasari, D. I. (2018). Implementasi Sistem Komunikasi Nirkabel OFDM Berbasis Software Defined Radio (SDR). *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, *7*(2), 183–189.

[15] Kolsrud, A., Hill, M., & Examiner-chieh, P. (2005). Digital Transmitter System and Method. (12) United States Patent. 2(12).

[16] Yuniarto, T. (2019). Masa Depan Jaringan 5G dan Perilaku Komunikasi Digital. *Warta ISKI*, *2*(01), 1–7.

[17] Gohil, A., Modi, H., & Patel, S. K. (2013). 5G technology of mobile ommunication: A survey. *2013 International Conference on Intelligent ystems and Signal Processing, ISSP 2013*, 288–292.

[18] Braithwaite, R.N. (2007). Digital Transmitter System Employing Self-Generating Predistortion Parameter Lists And Adaptive Controller. (12) United States Patent. 2(12).

[19] Schreier, R., Temes, G. C., dan Wiley, J., 2005, "Understanding delta-sigma data converters", IEEE press Piscataway, NJ, vol. 74.

[20] Schreier, R. 'Understanding Delta-Sigma Data Converters'.Bab 1 "Introduction".hlm 8.

[21] Keyzer, J., Hinrichs, J., Metzger, A., Iwamoto, M., Galton, I., dan Asbeck, P., 2001, "Digital generation of RF signals for wireless communications with band-pass DS modulation", *Microwave Symposium Digest IEEE MTT-S International*, hal. 2127-2130.

[22] Keyzer, J., Uang, R., Sugiyama, Y., Iwamoto, M., Galton, I., dan Asbeck, P., 2002, "Generation of RF pulsewidth modulated microwave signals using delta-sigma modulation", *Microwave Symposium Digest IEEE MTT-S International*, hal. 397-400.

[23] Sirmayanti, S., 2018, "Analysis predicted location of harmonic distortion in RF-upconverter structure", International Journal of Telecommu- nication, Computing, Electronics and Control (TELKOMNIKA), Vol. 6., No.6, December 2018.

[24] Mensink, E., Klumperink, E. A. M., & Nauta, B. (2005). Distortion cancellation by polyphase multipath circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *52*(9), 1785–1794.

[25] Nugroho, Y. P., Christyono, Y., & Sukiswo. (2008). "*Makalah Tugas Akhir Pengembangan Transmisi Perangkat Keras Sistem Modulasi Digital 8-Quadrature Ampitude Modulation Menggunakan Modulasi Fm"*.Semarang

[26] Basalamah, A,. & Yudihartono, S. (2019). "Pengembangan Media Pem-belajaran Sistem Modulasi Dasar Tel,ekomunikasi Berbasis Multimedia". Jurnal Instek Informatika dan Sains. Vol. 4,. No.1 April, 2019.

usilawati, I. (2009). 'TEKNIK TELEKOMUNIKASI DASAR Modulasi ulsa'. Universitas Mercubuana.Yogyakarta.

pasulta, R. Y., St, A. S. M. L., & Sompie, S. R. U. A. (2014). *Simulasi*

Sistem Pengacak Sinyal Dengan Metode FFT (Fast Fourier Transform). 1–9. Journal Teknik Elektro dan Komputer 2014.

[29] Kusuma, D. T. (2021). *Fast Fourier Transform (FFT) Dalam Transformasi Sinyal Frekuensi Suara Sebagai Upaya Perolehan Average Energy (AE) Musik*. *14*(1), 28–35.

[30] Konstelasi, D. (2015). *Perancangan Dan Implementasi Sistem Orthogonal Frequency Division Multiplexing (Ofdm) Dengan Menggunakan Dsk-Tms320c6713 Design And Implementation Orthogonal Frequency Division Multiplexing (Ofdm) System By Using DSK-TMS320C6713*. Jurnal Elektro Telekomunikasi Terapan. Desember 2015.126–136.

[31] Armstrong, J., & Member, S. (2009). *OFDM for Optical Communications*. Journal Of Lightwave Technology. Vol. *27*,. No.3,. 1 February 2019, 189–204.

[32] Nightingale, S., Capps, G., Winter, C., & Woloszczuk, G. (n.d.). *RF Interference Cancellation - a Key Technology to support an Integrated Communications Environment*. 1–13.

[33] Bassoo, V. *et al.* (2009) 'A potential transmitter architecture for future generation green wireless base station', *Eurasip Journal on Wireless Communications and Networking*, 2009. doi: 10.1155/2009/821846.

LAMPIRAN

Delta Modulasi (Signal Output)

```
T_sample = 100;%N_sample
Amp = 0.5; %choose: 0.5,1
un=Amp*sin(2*pi*1./T_sample*[0:(T_sample)]);%input signal
en=0;
en_delay=0;
delta=2;

L = length(un);
Delta_step = 0.05 ; %choose: 0.05,0.1
xn = 0;
for i=1:L;
if un(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

for i=1:D;
if D(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

figure ()
plot (un, '--');
%grid on;
hold on;
stairs (xn, 'r');
hold on;
stairs (D, 'k'); %Delta Mod
legend ('signal input', 'Integrated signal','DeltaMod quantized
signal output');
xlabel ('Sample Number')
ylabel ('Amplitude')
title ('Delta Modulation')
xlim([1 100]);
ylim([-1 1]);
```

Delta Modulasi (Output Spectrum)

```
T_sample = 10000;%N_sample
Amp = 0.5; %choose: 0.5,1
sin(2*pi*1./T_sample*[0:(T_sample)]);%input signal

y=0;
```

```
L = length(un);
Delta_step = 0.05 ; %choose: 0.05,0.1
xn = 0;
for i=1:L;
if un(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

for i=1:D;
if D(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

un1_fft_DM = 20*log10(abs(fft(D)));
un1_fft_fftshift_DM =fftshift(un1_fft_DM);
F= [-T_sample/2 : T_sample/2]/T_sample;

figure()
plot(F,un1_fft_fftshift_DM, 'r')
xlabel('Frequency/fs')
ylabel('Ouput to Input Power Spectrum Frequency (dB)')
title ('Output spectrum of Delta Modulation')
grid on
```

Delta-Sigma Modulasi [MOD1 (Signal Output)]

```
T_sample = 100; %N_sample
Amp = 0.5; %choose: 0.5,1
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

for aa=1:length(un)
    yn=un(aa) + en_delay;

            if yn>=0  % imilar with %vn=sgn(yn); as quantizing
            %if yn>0  % imilar with %vn=sgn(yn); as quantizing
            vn=1;
            else
            vn=-1;
            %vn=0;
            end
= yn - vn;
_delay=en;
```

```
     output_vn(aa)=vn;
end

diff = (un-output_vn);
average_power= vn ./ T_sample;

figure ()
t=0:T_sample;
plot(t, un(t+1),'--')
hold on
grid on
stairs(t, output_vn(t+1),'r')
legend ('signal input','DSMod quantized signal output');
xlabel ('Sample Number')
ylabel ('Amplitude')
title ('Delta-Sigma Modulation')
```

Delta-Sigma Modulasi [MOD1 (Output Spectrum)]

```
T_sample = 10000;%N_sample
Amp = 0.5; %choose: 0.5,1
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

for aa=1:length(un)
    yn=un(aa) + en_delay;

            if yn>=0  % imilar with %vn=sgn(yn); as quantizing
            %if yn>0  % imilar with %vn=sgn(yn); as quantizing
            vn=1;
            else
            vn=-1;
            %vn=0;
            end
     en = yn - vn;
     en_delay=en;

     output_vn(aa)=vn;
end

diff = (un-output_vn);
average_power= vn ./ T_sample;

un1_fft = 20*log10(abs(fft(output_vn)));
un1_fft_fftshift =fftshift(un1_fft);
F= [-T_sample/2 : T_sample/2]/T_sample;

          ()
          un1_fft_fftshift)
          'Frequency/fs')
          'Ouput to Input Power Spectrum Frequency (dB)')
          'Output spectrum of Delta-Sigma MOD1')
```

```
grid on
```

Delta Modulasi dan Delta-Sigma Modulasi [MOD1] (Output Spectrum)

```
T_sample = 10000; %N_sample
Amp = 0.5;
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

L = length(un);
Delta_step = 0.05 ;
xn = 0;
for i=1:L;
if un(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

for i=1:D;
if D(i) >= xn(i)
D(i) = 1;
xn(i+1) = xn(i) + Delta_step;
else
D(i) = -1;
xn(i+1) = xn(i) - Delta_step;
end
end

un1_fft_DM = 20*log10(abs(fft(D)));
un1_fft_fftshift_DM =fftshift(un1_fft_DM);
F= [-T_sample/2 : T_sample/2]/T_sample;

for aa=1:length(un)
    yn=un(aa) + en_delay;
            if yn>=0  % imilar with %vn=sgn(yn); as quantizing
            %if yn>0  % imilar with %vn=sgn(yn); as quantizing
            vn=1;
            else
            vn=-1;
            %vn=0;
            end
     en = yn - vn;
     en_delay=en;
     output_vn(aa)=vn;

(un-output_vn);
_power= vn ./ T_sample;

 = 20*log10(abs(fft(output_vn)));
_fftshift =fftshift(un1_fft);
```

```
F= [-T_sample/2 : T_sample/2]/T_sample;

figure()
plot(F,un1_fft_fftshift_DM, 'r')
hold on
plot(F,un1_fft_fftshift)
xlabel('Frequency/fs')
ylabel('Ouput to Input Power Spectrum Frequency (dB)')
title ('Output spectrum')
legend('DeltaMod','DeltaSigmaMod')
grid on
```

Noise Transfer Function (NTF) Delta Sigma Modulasi Orde 1 (MOD1)

```
close all;
clear all;
clc;

%% generate input
ffc= 3.584*10^9; %generate input frekuenesi carrier choose : 2.3
Ghz(2.355) and 3.5 Ghz (3.584)
ffs= 7.168*10^9; %Sampling Frequency choose 4.710 for (2,3 GHz)
BBW= 56*10^6; %Choose 36,8 for (2,3 GHz)

N=50;
OSR = 50;

fc = 0: 1/(OSR*N) : 1;

for i=1:length(fc);

NTFMOD1(i) = 2.*sin(pi.*fc(i))^2;
end

% Figure NTF_DS_out
figure()
plot (20.*log10 (abs(NTFMOD1))), 'k';

figure()
plot ( (abs(NTFMOD1))), 'r--';
```

Deret Fourier Transform – Penentuan *R* (OSR 4, 8, 16, 32, dan 64)

Np=4=OSR

```
even4nol = [0,0,0,0];
even4pos = [1,1,1,1];
even4neg = [-1,-1,-1,-1];
even_g0  = max(abs(fft([     0,0,0,0,       0,0,0,0,       0,0,0,0,
    0,0,0,0,          0,0,0,0,       0,0,0,0,       0,0,0,0,
    0,0,0,0,          0,0,0,0,       0,0,0,0,       0,0,0,0,
    0,0,0,0,          0,0,0,0,       0,0,0,0,       0,0,0,0,
    0,0,0,0,          0,0,0,0,       0,0,0,0,       0,0,0,0,
    0,0,0,0,          0,0,0,0,       0,0,0,0,       0,0,0,0,
```

```
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,]))/256)*2
even_c2 =  max (abs(fft([    1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,]))/256)*2
```

Np=8=OSR

```
even4nol = [0,0,0,0];
even4pos = [1,1,1,1];
even4neg = [-1,-1,-1,-1];
even_c0  = max(abs(fft([     0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,]))/256)*2
even_c2 =  max (abs(fft([    0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,    -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
1,-1,                   -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
1,-1,                   -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
1,-1,                   -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
1,-1,                       0,0,0,0,      0,0,0,0,       0,0,0,0,
,                           0,0,0,0,      0,0,0,0,       0,0,0,0,
,]))/256)*2
```

```
even_c4 =  max (abs(fft([    1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,]))/256)*2
Np=16=OSR
even4nol = [0,0,0,0];
even4pos = [1,1,1,1];
even4neg = [-1,-1,-1,-1];
even_c0  = max(abs(fft([     0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,]))/256)*2
even_c2  = max (abs(fft([    0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       1,1,1,1,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,                -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,                0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,]))/256)*2
even_c4 =  max (abs(fft([    0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
1,1,1,1,    1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
[illegible] 1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
[illegible] 1,1,1,1,        1,1,1,1,      1,1,1,1,       1,1,1,1,
[illegible] 0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
[illegible] 0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
[illegible] 0,0,0,0,        0,0,0,0,      0,0,0,0,       0,0,0,0,
[illegible] 0,0,0,0,    -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
```

```
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,                 0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,                     0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,]))/256)*2
even_c6 =  max (abs(fft([    0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
0,0,0,0,    0,0,0,0,         0,0,0,0,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,         0,0,0,0,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,      0,0,0,0,       0,0,0,0,
0,0,0,0,    0,0,0,0,]))/256)*2
even_c8 =  max (abs(fft([    1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,       1,1,1,1,
1,1,1,1,    1,1,1,1,         1,1,1,1,      1,1,1,1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,   -1,-1,-1,-1,
-1,-1,-1,-1,              -1,-1,-1,-1,  -1,-1,-1,-1,]))/256)*2
```

Function Theta (Even Quantisation Polar Plane Scale, OSR 4, 8, 16)

OSR 4

```
% even
% steps_in_pi=2, NA=2, NP=OSR=4.
function [even_r_pwm, even_R, even_theta]=
         evenquan4(input_qr_yr,input_qtheta_ytheta);
%NA=2;
if (input_qr_yr <0.6367);
    even_R=0;
    even_r_pwm=0/2;

    n_R=1.2733;
    n_r_pwm=2/2;
```

```
%% NP=4;
if (input_qtheta_ytheta >= (-4/32*2*pi)) &
    (input_qtheta_ytheta < (4/32*2*pi));
    even_theta = (0/32*2*pi);
elseif (input_qtheta_ytheta >= (4/32*2*pi)) &
    (input_qtheta_ytheta <= (12/32*2*pi));
    even_theta = (8/32*2*pi);
elseif (input_qtheta_ytheta >= (12/32*2*pi)) &
    (input_qtheta_ytheta < (16/32*2*pi));
    even_theta  = (16/32*2*pi);
elseif (input_qtheta_ytheta >= (-12/32*2*pi)) &
    (input_qtheta_ytheta < (-16/32*2*pi));
    even_theta  = (-16/32*2*pi);
else (input_qtheta_ytheta  >= (-12/32*2*pi)) &
    (input_qtheta_ytheta < (-4/32*2*pi));
    even_theta  = ((-8/32)*2*pi);
end

end
```

OSR 8

```
% even
% steps_in_pi=4, NA=3, NP=OSR=8.
function [even_r_pwm, even_R, even_theta]=
          evenquan8(input_qr_yr,input_qtheta_ytheta);
%NA=3;
if (input_qr_yr < 0.4502);
    even_R=0;
    even_r_pwm=0/4;

elseif (input_qr_yr >= 0.4502)&(input_qr_yr < 1.0869);
    even_R=0.9003;
    even_r_pwm=2/4;

else
    even_R=1.2733;
    even_r_pwm=4/4;

end

%% NP=8;
if(input_qtheta_ytheta >= (-2/32*2*pi)) & (input_qtheta_ytheta <
    (2/32*2*pi));
    even_theta = (0/32*2*pi);
elseif(input_qtheta_ytheta >= (2/32*2*pi)) & (input_qtheta_ytheta
     < (6/32*2*pi));
    even_theta = (4/32*2*pi);
    input_qtheta_ytheta >= (6/32*2*pi)) & (input_qtheta_ytheta
    10/32*2*pi));
    _theta = (8/32*2*pi);
    input_qtheta_ytheta >= (10/32*2*pi)) & (input_qtheta_ytheta
    (14/32*2*pi));
    _theta = (12/32*2*pi);
```

```
elseif(input_qtheta_ytheta >= (14/32*2*pi)) & (input_qtheta_ytheta
     < (16/32*2*pi));
    even_theta = (16/32*2*pi);
elseif(input_qtheta_ytheta  >= (-14/32*2*pi))&(input_qtheta_ytheta
     < (-16/32*2*pi));
    even_theta = (-16/32*2*pi);
elseif(input_qtheta_ytheta >= (-14/32*2*pi))&(input_qtheta_ytheta
     < (-10/32*2*pi));
    even_theta = (-12/32*2*pi);
elseif(input_qtheta_ytheta >= (-10/32*2*pi))&(input_qtheta_ytheta
     < (-6/32*2*pi));
    even_theta = (-8/32*2*pi);
else(input_qtheta_ytheta >= (-6/32*2*pi)) & (input_qtheta_ytheta <
    (-2/32*2*pi));
    even_theta = (-4/32*2*pi);
end

end
```

OSR 16

```
% even
% steps_in_pi=8, NA=5, NP=OSR=16.
function [even_r_pwm, even_R, even_theta]=
          evenquan16(input_qr_yr,input_qtheta_ytheta);
%NA=5;
if (input_qr_yr < 0.2437);
    even_R=0;
    even_r_pwm=0/8;
 elseif (input_qr_yr >= 0.2437)&(input_qr_yr < 0.6939);
    even_R=0.4873;
    even_r_pwm=2/8;

elseif (input_qr_yr >= 0.6939)&(input_qr_yr < 1.4885);
    even_R=0.9003;
    even_r_pwm=4/8;

elseif (input_qr_yr >= 1.4885)&(input_qr_yr < 1.2248);
    even_R=1.1763;
    even_r_pwm=6/8;

else
    even_R=1.2733;
    even_r_pwm=8/8;

end

%% NP=16
% even is in fractional
if (input_qtheta_ytheta >= (-1/32*2*pi)) & (input_qtheta_ytheta <
    2*2*pi));
    n_theta = (0/32*2*pi);
    (input_qtheta_ytheta >= (1/32*2*pi)) & (input_qtheta_ytheta
    3/32*2*pi));
    n_theta = (2/32*2*pi);
```

```
elseif (input_qtheta_ytheta >= (3/32*2*pi)) & (input_qtheta_ytheta
     < (5/32*2*pi));
    even_theta = (4/32*2*pi);
elseif (input_qtheta_ytheta >= (5/32*2*pi)) & (input_qtheta_ytheta
     < (7/32*2*pi));
    even_theta = (6/32*2*pi);
elseif (input_qtheta_ytheta >= (7/32*2*pi)) & (input_qtheta_ytheta
     < (9/32*2*pi));
    even_theta = (8/32*2*pi);
elseif (input_qtheta_ytheta >= (9/32*2*pi)) & (input_qtheta_ytheta
     < (11/32*2*pi));
    even_theta = (10/32*2*pi);
elseif (input_qtheta_ytheta >= (11/32*2*pi))& (input_qtheta_ytheta
     < (13/32*2*pi));
    even_theta = (12/32*2*pi);
elseif (input_qtheta_ytheta >= (13/32*2*pi))& (input_qtheta_ytheta
     <=(15/32*2*pi));
    even_theta = (14/32*2*pi);
elseif (input_qtheta_ytheta >= (15/32*2*pi))& (input_qtheta_ytheta
     < (16/32*2*pi));
    even_theta = (16/32*2*pi);
elseif (input_qtheta_ytheta >= (-15/32*2*pi))&(input_qtheta_ytheta
     < (-16/32*2*pi));
    even_theta = (-16/32*2*pi);
elseif (input_qtheta_ytheta  < (-15/32*2*pi))&(input_qtheta_ytheta
     < (-13/32*2*pi));
    even_theta = (-14/32*2*pi);
elseif (input_qtheta_ytheta >= (-13/32*2*pi))&(input_qtheta_ytheta
     < (-11/32*2*pi));
    even_theta = (-12/32*2*pi);
elseif (input_qtheta_ytheta >= (-11/32*2*pi))&(input_qtheta_ytheta
     < (-9/32*2*pi));
    even_theta = (-10/32*2*pi);
elseif (input_qtheta_ytheta >= (-9/32*2*pi))& (input_qtheta_ytheta
     < (-7/32*2*pi));
    even_theta = (-8/32*2*pi);
elseif (input_qtheta_ytheta >= (-7/32*2*pi))& (input_qtheta_ytheta
     < (-5/32*2*pi));
    even_theta = (-6/32*2*pi);
elseif (input_qtheta_ytheta >= (-5/32*2*pi))& (input_qtheta_ytheta
     < (-3/32*2*pi));
    even_theta = (-4/32*2*pi);
else (input_qtheta_ytheta >= (-3/32*2*pi)) & (input_qtheta_ytheta
     < (-1/32*2*pi));
    even_theta = (-2/32*2*pi);
end

end
```

Cartesian to Polar

```
r_yr = sqrt((vn_i ).^2 + (vn_q).^2);
theta_ytheta = atan2 (vn_q,vn_i);
```

Cartesian

```
y_i = even_R .*cos(even_theta);
```

```
en_delay_q = even_R .*sin(even_theta);
```

Quantisation Even

```
if      (steps_in_pi == 2)
        [even_r_pwm, even_R, even_theta] =
        evenquan4(input_qr_yr,input_qtheta_ytheta);
elseif (steps_in_pi == 4)
        [even_r_pwm, even_R, even_theta] =
        evenquan8 (input_qr_yr,input_qtheta_ytheta);
elseif (steps_in_pi == 8)
        [even_r_pwm, even_R, even_theta] =
        evenquan16(input_qr_yr,input_qtheta_ytheta);
end

R_feedback = even_R;
theta_feedback = even_theta;

R_feedback_array (i)        = R_feedback;          %feedback ke DS
theta_feedback_array (i)    = theta_feedback;      %feedback ke DS

even_R_array (i)             = even_R;
even_theta_array (i)         = even_theta;
even_r_pwm_array (i)         = even_r_pwm;

input_qr_yr_array (i)          = input_qr_yr;
input_qtheta_ytheta_array(i)   = input_qtheta_ytheta;
```

Keseluruhan Code Pemrograman (Even Matlab)

```
%% generate input
fc= 2.355*10^9; %generate input frekuenesi carrier
                choose : 2.355 (2.3 GHz),3.584 (3.5 GHz)
fs= 4.710*10^9; %Samplng Frequency  choose : 4.710,7168
BW= 36.8*10^6;  %choose 56

steps_in_pi =2;  %choose: 2,4,8
NP=2*steps_in_pi; %OSR_r=RF = 4,8,16
fs_rf= NP*fs;

T_sample =4710; %N_sample = OSR   choose: 4710,7168
No_of_periods=1;
sample_max = (T_sample*No_of_periods);
offset=1; %Offset signal from carrier  choose: 1,.....

noise = (randn(1,1)+j*(randn(1,1)))*10^-4;
%noise= [-0.000120748692268504 + 7.17238651328839e-05i]; %test
only

max_num= offset/No_of_periods;


and proses
ak=1:max_num;
ble_max;
```

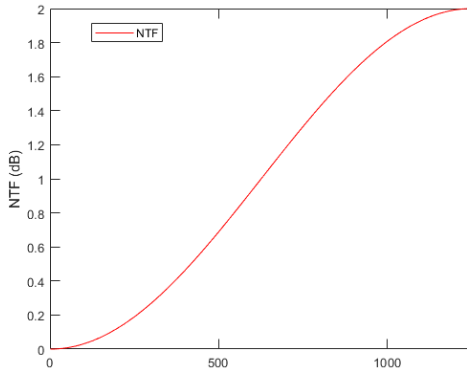
```
p=1;
Amp(p)=0.5;
input_un=Amp(p)*exp(j*2*pi*i/T_sample*jarak)+ noise;  %Input
signal

%% input I dan Q dalam format Cartesian/Rectangular
input_i = real(input_un);
input_q = imag(input_un);

%% Modulasi Delta Sigma Orde 1
yn_delay_i=0;
en_delay_i=0;
yn_delay_q=0;
en_delay_q=0;
     for i=1:sample_max;
%% MOD1
    yn_i = input_i(i)- en_delay_i;
    vn_i = yn_i + yn_delay_i;
    yn_delay_i = vn_i;

    yn_q  = input_q(i)- en_delay_q;
    vn_q = yn_q + yn_delay_q;
    yn_delay_q = vn_q;

    yn_i_array(i)        = yn_i; %Input signal of I-plane
    yn_q_array(i)        = yn_q; %Input signal of Q-plane

    en_delay_i_array(i)  = en_delay_i;
    en_delay_q_array(i)  = en_delay_q;

    vn_i_array (i)       = vn_i;
    vn_q_array (i)       = vn_q;

%% Cartesian to Polar
    input_qr_yr = sqrt((vn_i ).^2 + (vn_q).^2);
    input_qtheta_ytheta = atan2 (vn_q,vn_i);

%% Quantization: EVEN
    if      (steps_in_pi == 2)
            [even_r_pwm, even_R, even_theta] =
            evenquan4(input_qr_yr,input_qtheta_ytheta);
    elseif (steps_in_pi == 4)
            [even_r_pwm, even_R, even_theta] =
            evenquan8 (input_qr_yr,input_qtheta_ytheta);
    elseif (steps_in_pi == 8)
            [even_r_pwm, even_R, even_theta] =
            evenquan16(input_qr_yr,input_qtheta_ytheta);
    end

edback = even_R;
a_feedback = even_theta;

edback_array (i)        = R_feedback;      %feedback ke DS
a_feedback_array (i)    = theta_feedback;  %feedback ke DS
```

```
    even_R_array (i)              = even_R;
    even_theta_array (i)          = even_theta;
    even_r_pwm_array (i)          = even_r_pwm;

    input_qr_yr_array (i)           = input_qr_yr;
    input_qtheta_ytheta_array(i)    = input_qtheta_ytheta;

%% Polar to Cartesian
      en_delay_i = even_R .*cos(even_theta);
      en_delay_q = even_R .*sin(even_theta);
end
%% RF to PWM/PPM
[pwm] = SIRMARF (even_r_pwm_array,even_theta_array,steps_in_pi);
end

%  FIGURE RESULT
figure();
stem(20*log10(abs(fft(pwm))),'.r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum RF Signal');
legend ('Even');
xlim([1000 6500]);
xlim([300 4500]);
%ylim ([-30   100]);

figure();
plot(20*log10(abs(fft(pwm))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum RF Signal');
legend ('Even');

figure();
plot(20*log10(abs(fft(pwm))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum RF Signal');
legend ('Even');
xlim([500 1600]);

figure();
plot(20*log10(abs(fft(vn_i_array))),'r','LineWidth',1)
plot(20*log10(abs(fft(vn_q_array))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum Blok DS');
legend ('Even');

 Filter-DS
out=fft(vn_i_array(sample_max/2+1:sample_max)+j.*vn_q_array
    (sample_max/2+1:sample_max));
out=abs(fft_DS_out);
n_DS_out=20*log10(abs_DS_out);
```

```
figure()
plot (fftshift(spectrum_DS_out), 'r','LineWidth',1)
title ('Output spectrum Blok DS');
legend ('Even');

figure();
plot(20*log10(abs(fft(input_qr_yr_array))),'r','LineWidth',1)
plot(20*log10(abs(fft(input_qtheta_ytheta_array))),'r','LineWidth'
,1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum Blok Q');
legend ('Even');

%output-Q
fft_Q_out=fft(input_qr_yr_array(sample_max/2+1:sample_max)+j.*inpu
t_qtheta_ytheta_array(sample_max/2+1:sample_max));
abs_Q_out=abs(fft_Q_out);
spectrum_Q_out=20*log10(abs_Q_out);
figure()
plot (fftshift(spectrum_Q_out),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum Blok Q');
legend ('Even')

% Input I
figure();
plot(20*log10(abs(fft(yn_i_array))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Input I');
legend ('Even');
% Input Q
figure();
plot(20*log10(abs(fft(yn_q_array))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Input Q');
legend ('Even');

%Feedback Block
figure()
plot(20*log10(abs(fft(R_feedback_array))),'r','LineWidth',1)
plot(20*log10(abs(fft(theta_feedback_array))),'r','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Feedback I, OSR 4,8,16');
title ('Feedback Q, OSR 4,8,16');
legend ('Even');

 Sigma Delta Filter

*log10(abs(fft(en_delay_i_array))),'r','LineWidth',1)
*log10(abs(fft(en_delay_q_array))),'r','LineWidth',1)
'Frequency/fs');
```

```
ylabel('Output Power Spectrum Frequency (dB)');
title ('OSR 4,8,16');
legend ('Even');
```

Figure Result
Ploting Noise Transfer Function Delta Sigma MOD1

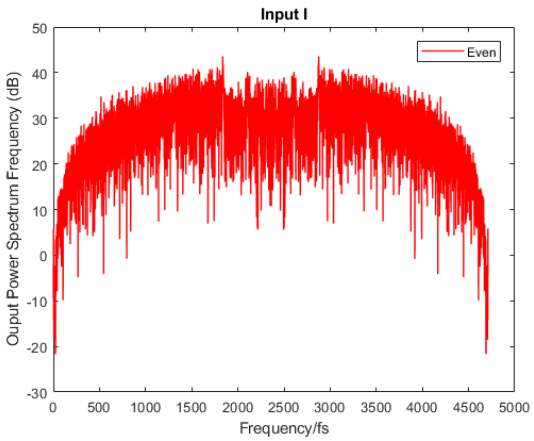


Normalized Frequnecy

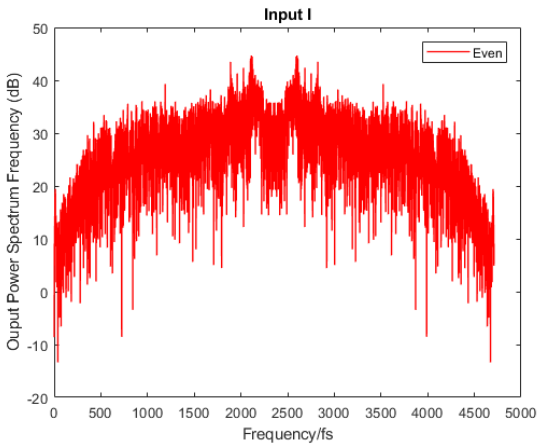
Ploting Output Spectrum Input I dan Q

a) Frekuensi 2,3 GHz

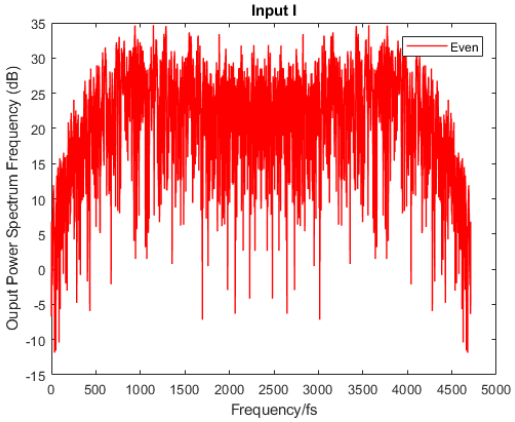
- **Input I**



OSR 4

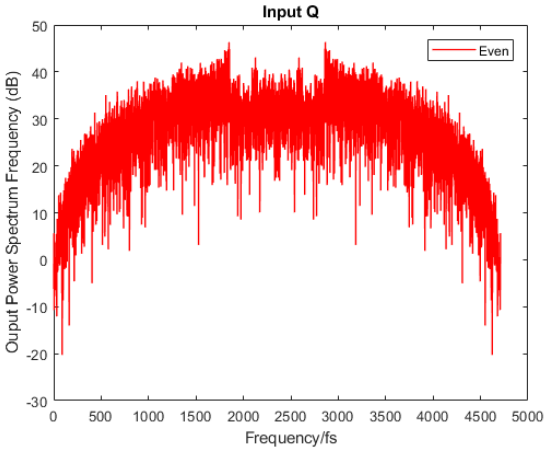


OSR 8

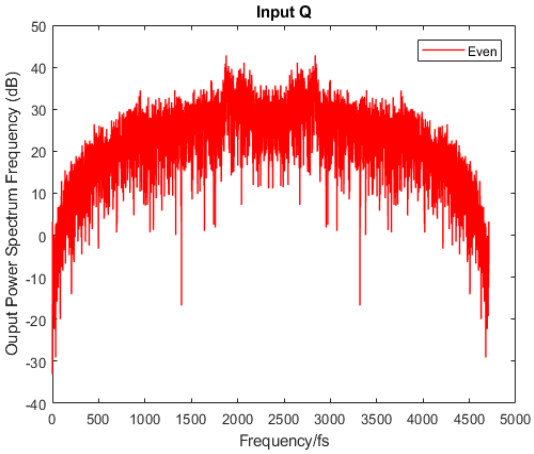


OSR 16

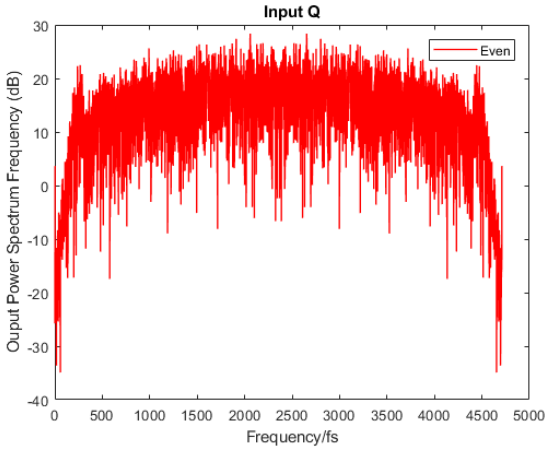
- **Input Q**



OSR 4



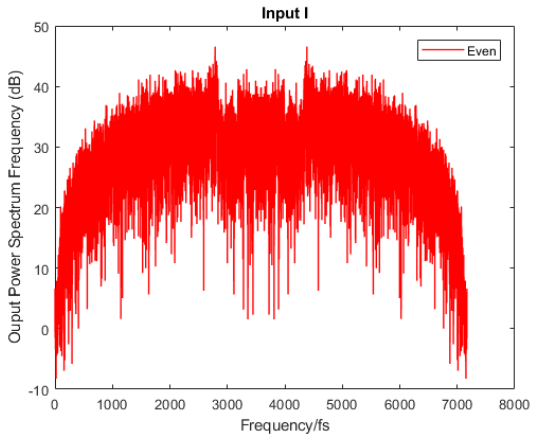
OSR 8



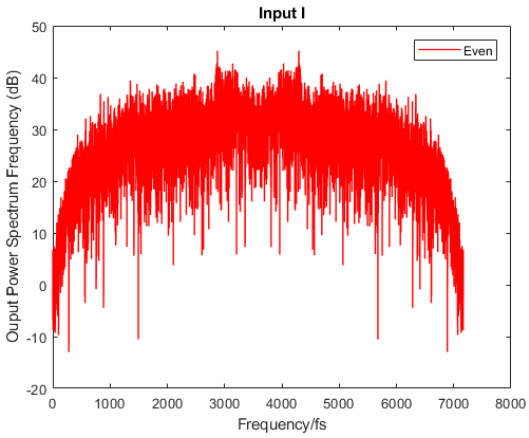
OSR 16

b) Frekuensi 3,5 GHz

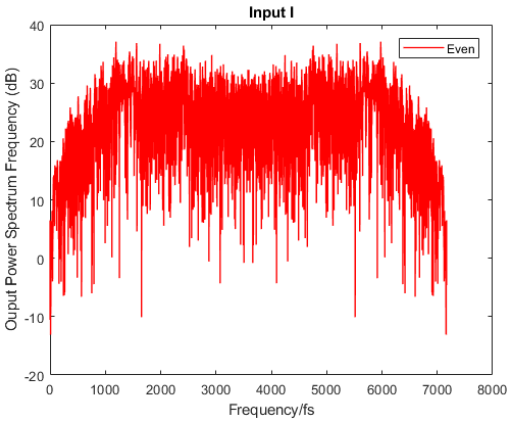
- **Input I**



OSR 4

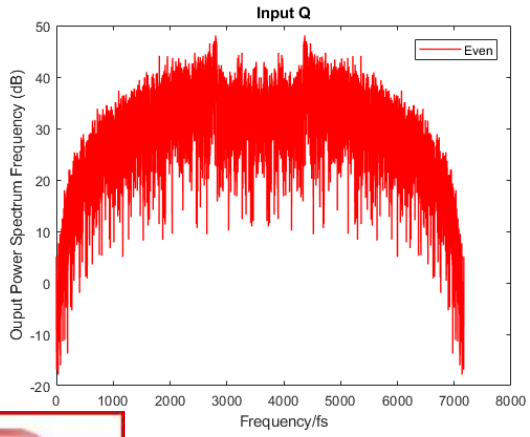


OSR 8

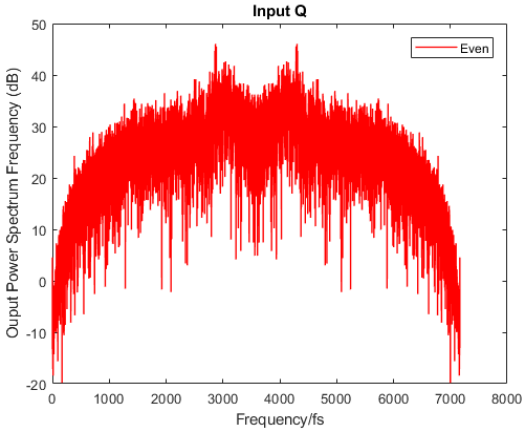


OSR 16

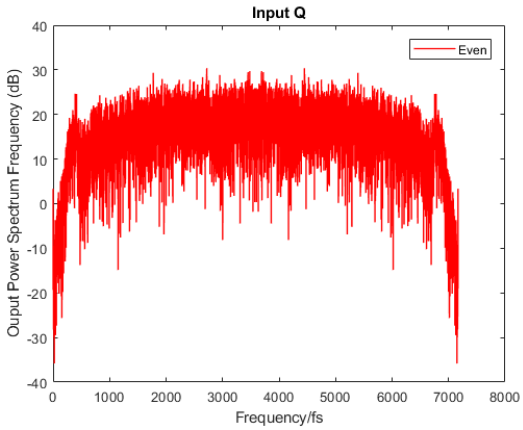
- **Input Q**



OSR 4



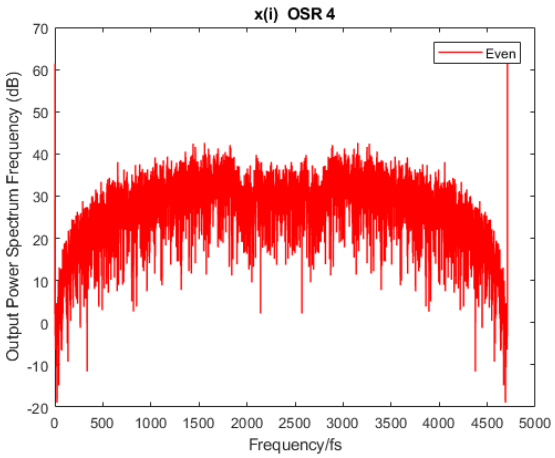
OSR 8

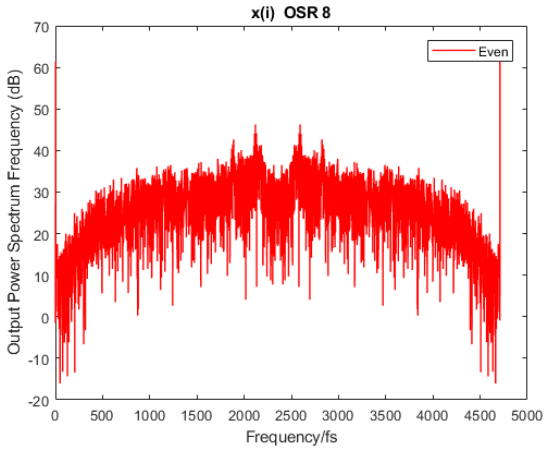


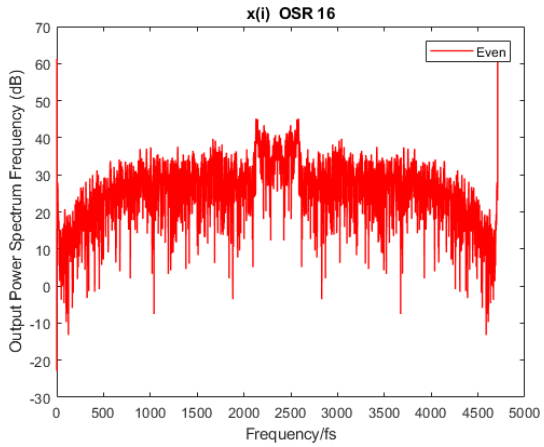
OSR 16

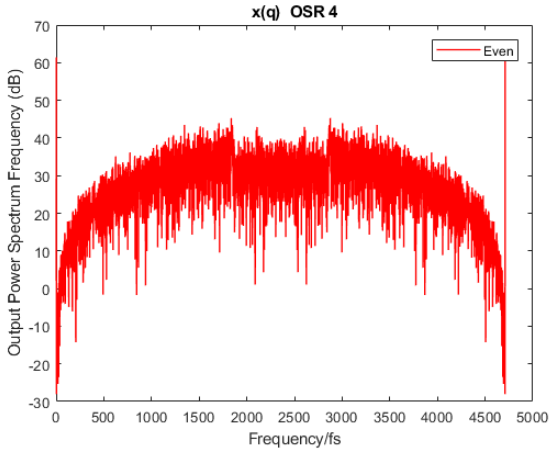
Ploting Input Sigma Delta (x_i dan x_q)

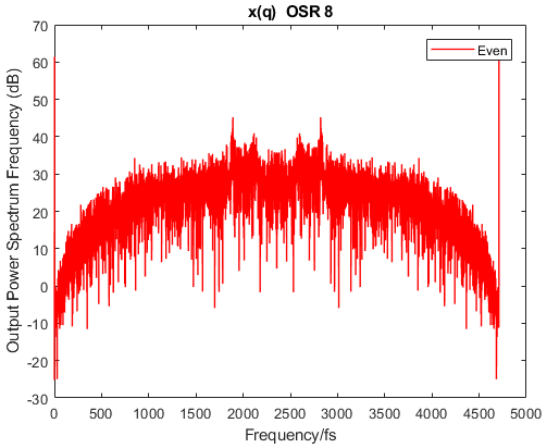
a) Frekunesi 2,3 GHz

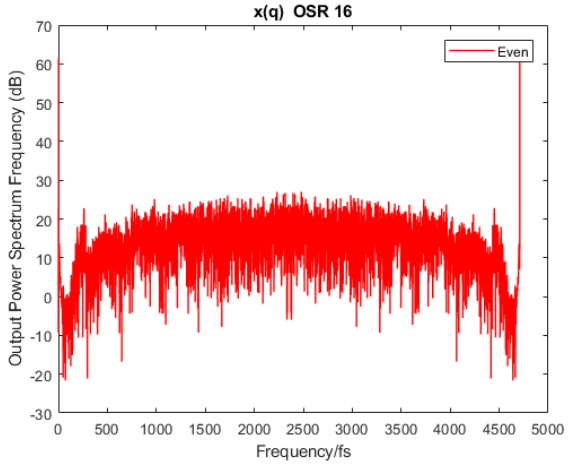




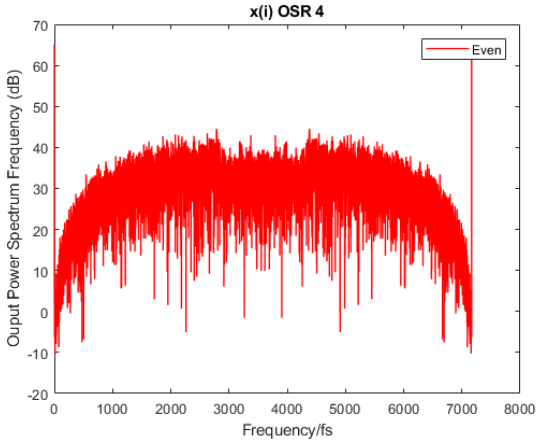


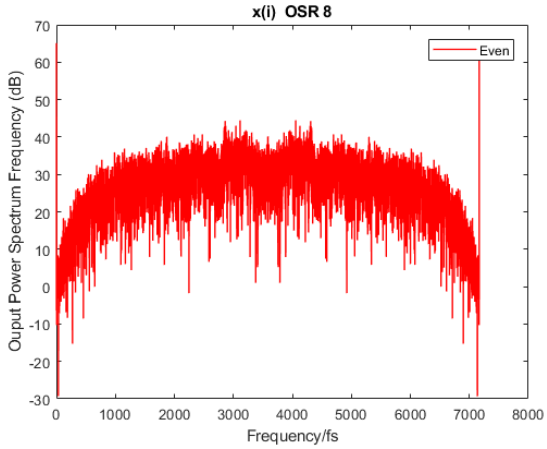


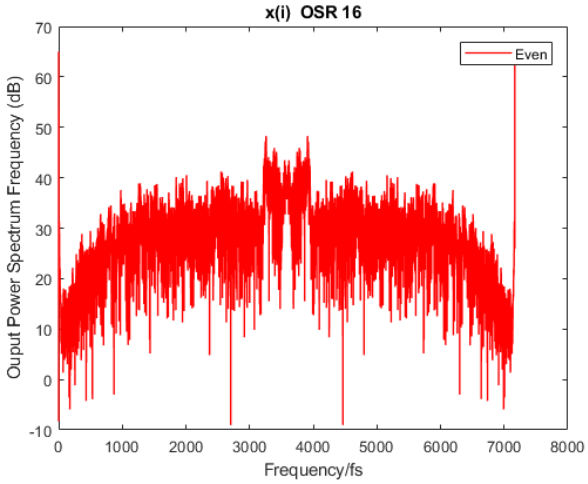


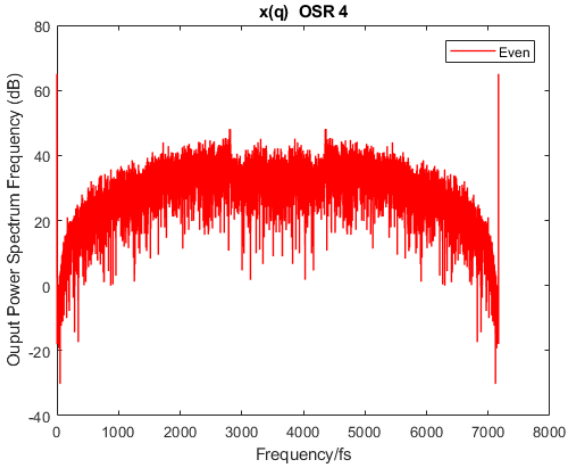


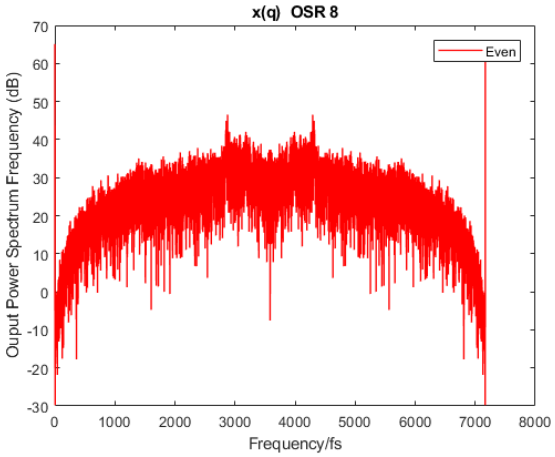
b) Frekuensi 3,5 GHz

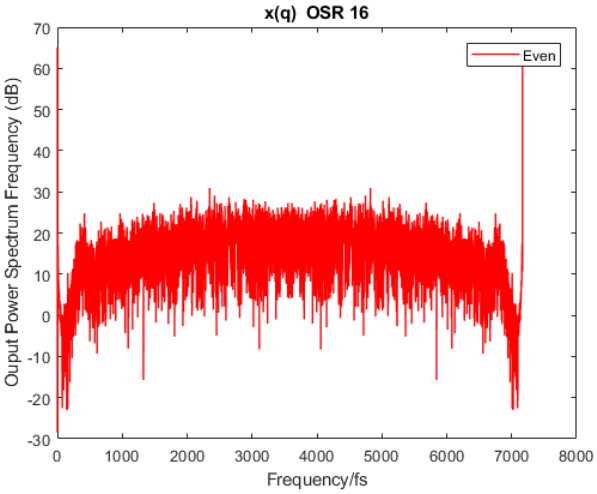






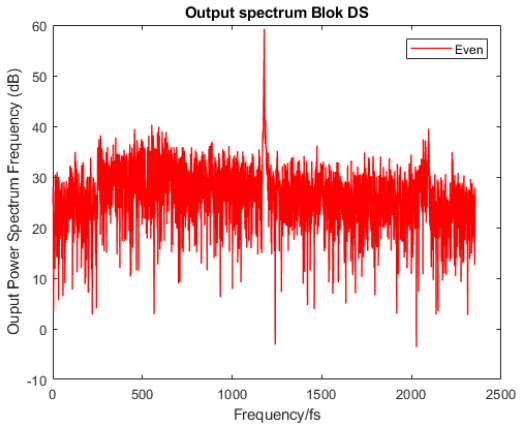




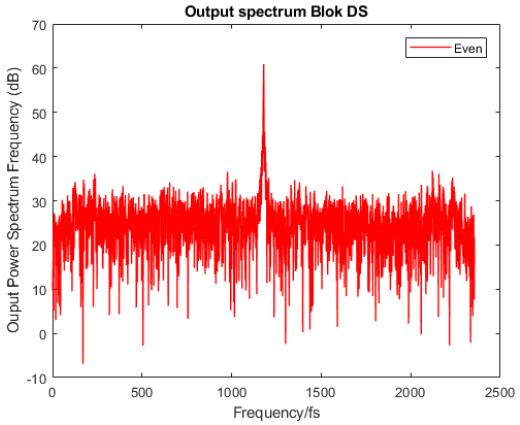


Ploting Output Spectrum Blok Delta Sigma (y_i dan y_q)

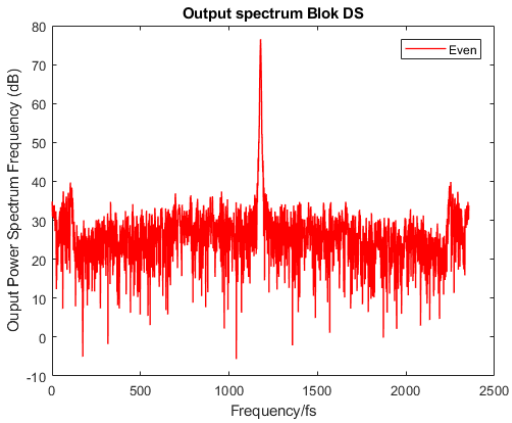
a) Frekuensi 2,3 GHz



OSR 4

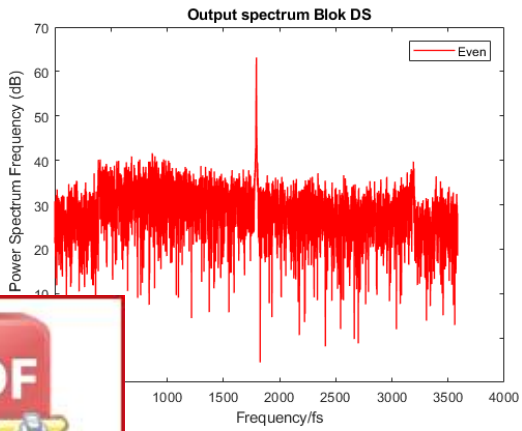


OSR 8

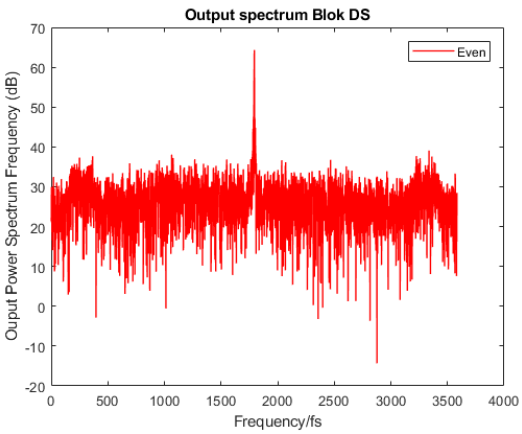


OSR 16

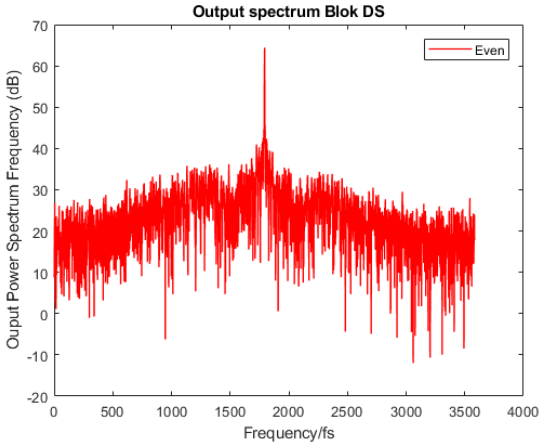
b) Frekuensi 3,5 GHz



OSR 4



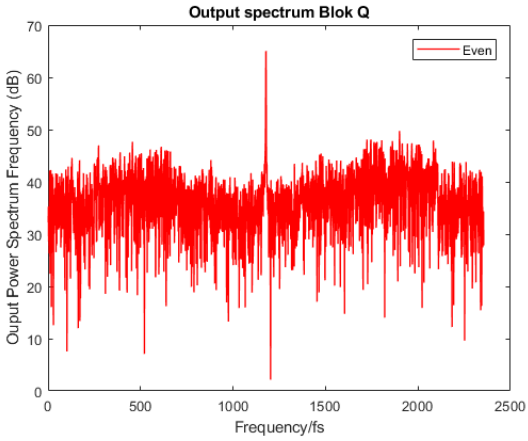
OSR 8



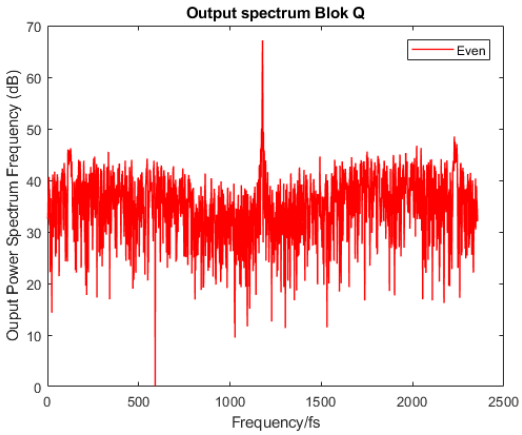
OSR 16

Ploting Output Spectrum Blok Kuantisasi (y_R, y_θ)

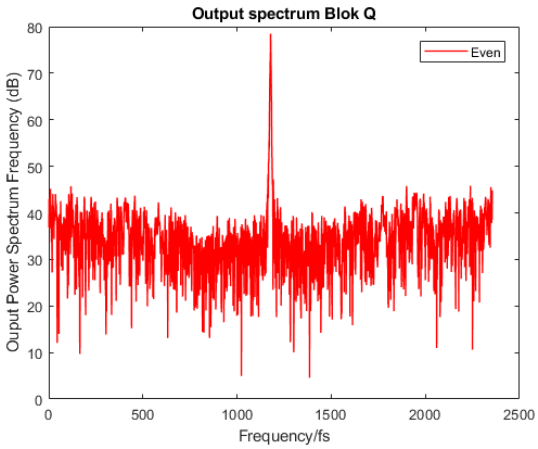
a) Frekuensi 2,3 GHz



OSR 4

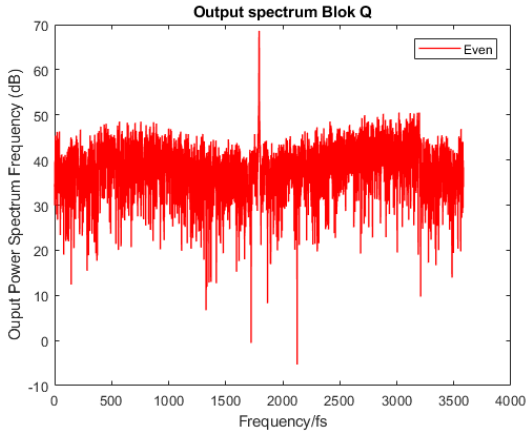


OSR 8

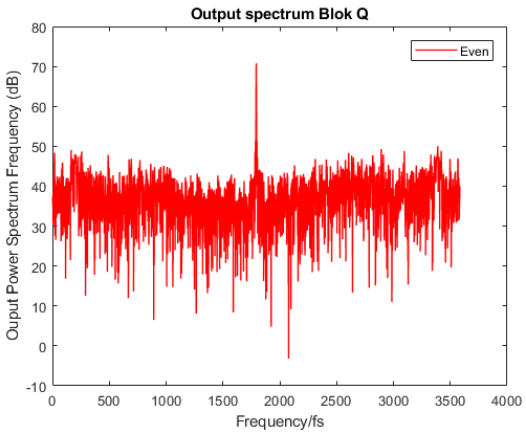


OSR 16

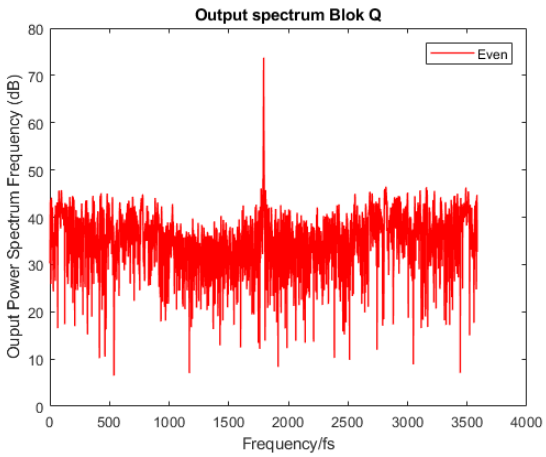
b) Frekuensi 3,5 GHz



OSR 4



OSR 8



OSR 16

Ploting Output Spectrum Feedback ΣΔ Filter ($\hat{y}_i, \hat{y}_\theta$)

a) Frekuensi 2,3 GHz