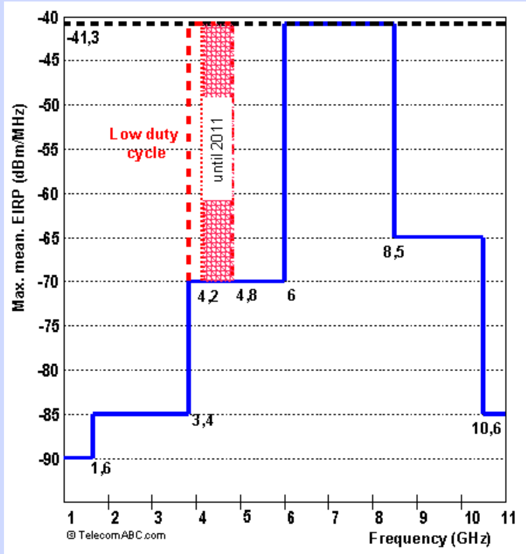
Lecture 4/5: UWB Transmitter

Kommunikations
Technik

Ultra-wideband Radio

- FCC specification: An intentional radiator that, at any point in time, has a fractional bandwidth equal to or greater than 0.20 or has a UWB bandwidth equal to or greater than 500 MHz, regardless of the fractional bandwidth
- GSM: 200KHz
- IEEE 802.11b/g/n: 20MHz, 40MHz
- WiMedia UWB: 528MHz
- Impulse UWB several GHz



Reference: Giancola, Guerino : Understanding ultra wide band radio fundamentals. (D45 YDA1769)

Ultra-wideband Radio

- OFDM based UWB

128 Subcarriers

528MHz Bandwidth/ Sub-band 4.125MHz

Modulation QPSK (Quadrature Phase Shift Keying)

Channel Coding: Convolutional code LDPC code

High data rate, 1024Mbps

Applications: Wireless USB, Wireless video streaming

- Short Impulse based UWB

Carrier-less, very short symbol duration.

Modulation: PPM (Pulse position modulation), BPSK (Binary Phase Shift Keying) or DBPSK (Differential BPSK)

Low data rate, typically several Mbps

Applications: localization and tracking, low data rate transmission

IR-UWB Transmitter Chain

- Digital Communication?

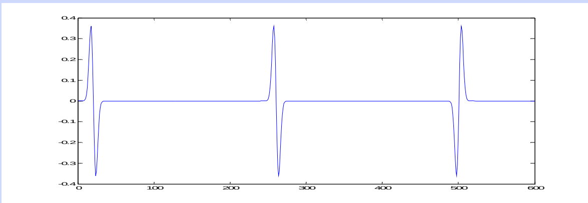
 ...0110001100111011010... 

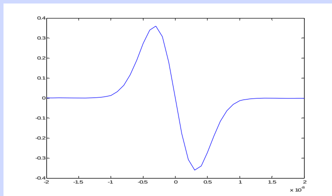
$\boldsymbol{b} = (\cdots, b_0, b_1, \cdots, b_k, b_{k+1}, \cdots,)$

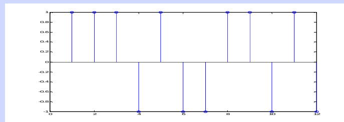
$\boldsymbol{c} = (\cdots, c_0, c_1, \ldots, c_{j-1}, c_j \cdots)$

Information bits → Repetition Coder → Transmission encoder → spreading $\boldsymbol{d} = \boldsymbol{a}\ .\boldsymbol{c}$ → Differential encoder → $\boldsymbol{d}'$ → Pulse Shaper → $s(t)$

$\boldsymbol{a}^* = (\cdots, b_0, b_0, \cdots, b_0, b_1, b_1, \cdots, b_1, \cdots, b_k, b_k, \cdots, b_k, b_{k+1}, b_{k+1}, \cdots, b_{k+1}, \cdots)$







$$s(t) = \sum_{j=-\infty}^{+\infty} d'_j\, p(t - jT_s)$$

$p(t)$

www.KommunikationsTechnik.org

Generation of Information Bits /1

- Information Bits Generation

`function [bits]=info_bits(num_bits)`

`info_bits.m` function create a stream binary values.

number of bits is defined by the input parameters (‘`num_bits`’).

use `rand('state',15)`to reproduce the same output bit stream.

info_bits.m

```
Function[bits]=info_bits(num_bits)

rand('state',15);

bits = rand(1,num_bits)>0.5;
```

Exercise

Design the info_bits with C-Mex, info_bits_mex.cpp

Generation of Information Bits /2

- Write a matlab script to generate the information bits

$$\boldsymbol{b} = \left(\cdots, b_0, b_1, \cdots, b_k, b_{k+1}, \cdots,\right)$$

Define the number of bits

```
num_bits;
```

Call your function

```
bits= info_bits(num_bits);
```

Check the distribution by

```
sum(bits)./length(bits);
```

Question:

Why is the ‘1’ and ‘0’ are equally distributed? How about in a real communication system?

Repetition Coder/1

- Why do we need repetition?

...1111000011110000011110000...

...1111000000110000011110000...

To achieve redundant transmission.

- The length of output sequence is (num_bits * Ns).

$$\boldsymbol{a}^{*}=\left(\cdots,b_0,b_0,\cdots,b_0,b_1,b_1,\cdots,b_1,\cdots,b_k,b_k,\cdots,b_k,b_{k+1},b_{k+1},\cdots,b_{k+1},\cdots\right)$$

- A simple channel coding

Not robust to burst error

Repetition Coder/2

- Define the function

```
Rep_coder.m
Input: generated information bits, Ns
Output: repeated version of input
```

- Example

  Repetition factor Ns = 4

  Input

$$\boldsymbol{b} = \left(\cdots, b_0, b_1, \cdots, b_k, b_{k+1}, \cdots,\right)$$

  Output

$$\boldsymbol{a}^* = \left(\cdots, b_0, b_0, \cdots, b_0, b_1, b_1, \cdots, b_1, \cdots, b_k, b_k, \cdots, b_k, b_{k+1}, b_{k+1}, \cdots, b_{k+1}, \cdots\right)$$

Repetition Coder/3

rep_code.m

```
function [repbits] = rep_coder(bits,Ns)

num_bits    = length(bits); %number of bits
rect_filter = ones(1,Ns);

temp1 = zeros(1,(num_bits*Ns));
temp1(1:Ns:1+Ns*(num_bits-1)) = bits;

temp2    = conv(temp1, rect_filter);
repbits  = temp2(1:Ns*num_bits);
```

Try to do…

Find the repeated sequence if

num_bits = 5 , Ns = 3 and

Transmission Coder /1

- Direct-Sequence Spread Spectrum

Define a function

DS_code.m

Input: the period of the PN

Output: random PN code

Generate a random PN code composed of (+1,-1) values with period of Np=12.

$$\boldsymbol{c} = \left(\cdots, c_0, c_1, \ldots, c_{Np-1}, \cdots \right)$$

DS_code.m

```
function [DScode] = DS_code (Np)
Dscode = ((rand(1,Np) > 0.5).*2) – ones (1,Np);
```

Transmission Coder /2

- Apply the DS code to the repeated information sequence

Define a function

```
DS_PAM.m
Inputs: repeated info bits, generated DS code
Outputs: coded bits with DS
```

Transform the repeated information sequence into positive and negative valued PAM sequence: $a_j = (2a_j^* - 1)$

Apply the DS code to the information sequence $\boldsymbol{a}_j$

Transmission Coder /3

DS_PAM.m

```
function [PAMseq] = DS_PAM(rep_bits,DS_code)

DS_len  =length(DS_code);
seq_len = length(rep_bits);
PAMseq  = zeros(1,seq_len);

for k = 1 : seq_len
    KDS   = DS_code(1+mod(k-1,DS_len));
    PAMseq(k)=((2*rep_bits(k))-1) * KDS;
end
```

Try to do…

Find the DS-PAM sequence when

num_bits=10, Ns=4, Np=4.

Answer

```
num_bits = 10;
Ns = 4;
Np = 4;
bits    = info_bits(num_bits);
Repbits = rep_coder(bits,Ns);
Dscode = DS_code(Np);
PAMseq =DS_PAM(repbits,DScode);
```

Differential Encoder /1

- The output of differential encoder is the logical difference between the current input element and the previous output element.

Define a function

```
DS_PAM.m
Inputs: bits sequence, initial state
Outputs: Differencial code of the input
```

Assuming that x_i is a bit intended for transmission, and y_i is a bit actually transmitted (differentially encoded)

$$y_i = y_{i-1} * x_i$$

input		1	-1	-1	1	1	-1	1	-1
Initial state	1								
output		1	-1	1	1	1	-1	-1	1

Differential Encoder /2

diff_encoder.m

```
function [output] =  diff_enc(input,inputState)
    output    = zeros(size(input));
    output(1) = (inputState * input(1));
    for p = 2:length(input)
        output(p)= (output(p-1)* input(p));
    end
```

Task

- Check your results, set initial state=1 ,input= [1 -1 -1 1 1 -1 1 -1];

Exercise

- design a mex file, diff_encoder_mex.cpp to realize the same functionality.

Pulse Shape Filter /1

- The pulse shape filter generate impulse response of the UWB pulse to be transmitted.
- The impulse response of pulse shape filter is the first derivative of Gaussian waveform.

$$p(t)=\frac{-t}{\sqrt{2\pi}\sigma^3}e^{\left(\frac{-t^2}{2\sigma^2}\right)}$$

Matlab simulation

- Define the parameters time window length of the pulse T_c and sigma σ and the sampling frequency.

```
gausBpam    = gauss_bpam(pulseDuration, time);
```

- The length of the output filter is similar to window time length.

Pulse Shape Filter /2

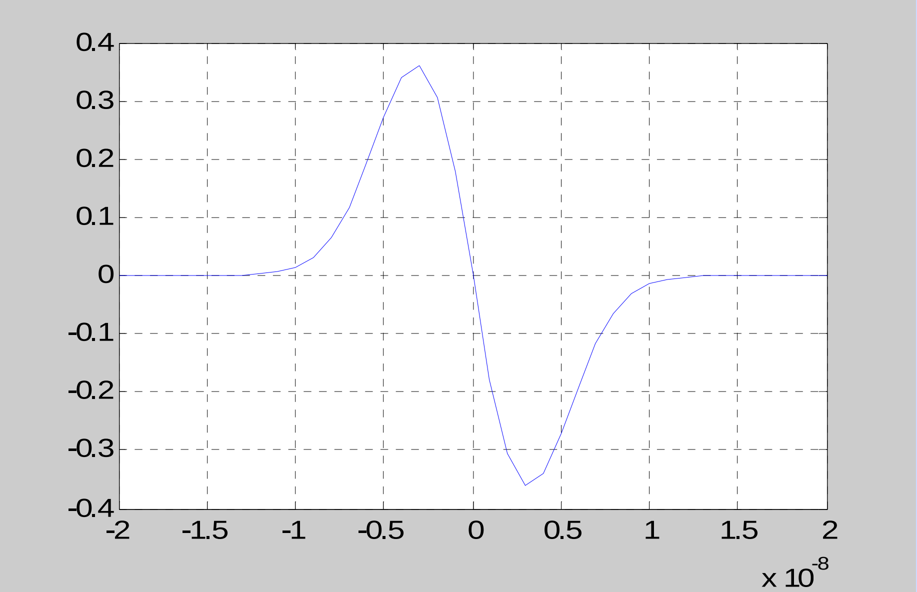
gauss_bpam.m

```
%This function generate the UWB pulse

function[gaussianBpamPulse] =gauss_bpam(pulseDuration, time)

sigma   = pulseDuration/(2*pi); % sigma
gaussianBpamPulse = -time ./ (sqrt(2*pi)*sigma.^3)…
                   .* exp( - time.^2/(2*sigma.^2));
% filter Impulse Response
norm_guass        = sqrt(gaussianBpamPulse*gaussianBpamPulse');
gaussianBpamPulse = gaussianBpamPulse ./ norm_guass);
% normalize the pulse
```

Pulse Shape Filter /3

Try to do…

1-Plot the impulse response of a pulse shape filter if window length=20 and sampling frequency= 1e9Hz.



Plot the Gaussian impulse

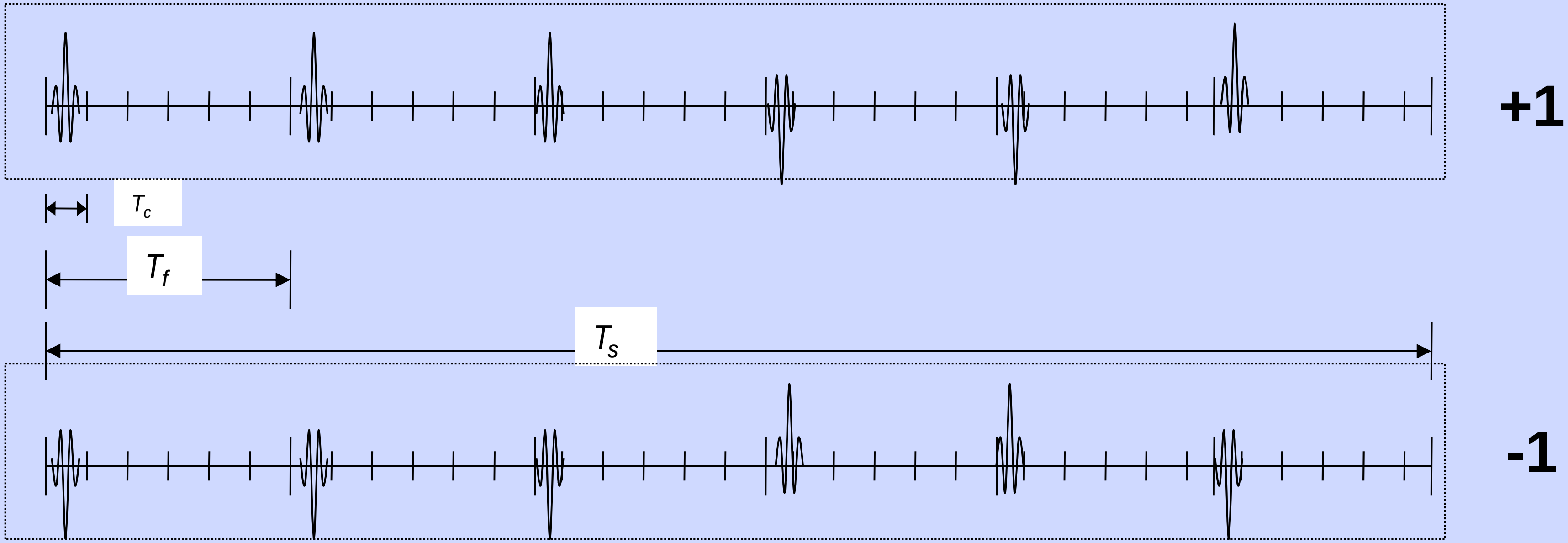
```
sampling_time   = 1e-9;
win_len         = 20;
pulse_len       = sampling_time*win_len;
pulse_time      = (-win_len:1:win_len)* sampling_time;
gausBpam        = gauss_bpam(pulse_len, pulse_time);
plot(pulse_time, gausBpam); grid on;
```

Pulse Shape Filter /4

Exercise

- Design a mex file gauss_bpam_mex.cpp which generate the second derivative Gaussian pulse.

Transmission of IR-UWB Pulses /1



Each symbol represented by sequence of very short pulses

Each user uses different PN sequences (for multiple access)

Bandwidth mostly determined by pulse shape

Transmission of IR-UWB Pulses /2

- The transmitted signal s(t) at the output of transmitter:

$$s(t) = \sum_{j=-\infty}^{+\infty} d_j p(t - jT_s)$$

- d_j : coded transmitted samples represented in(-1,1)
- $p(t - jT_s)$: impulse response of pulse shape filter.
- The bit duration T_s is the time used to transmit one bit.

Transmission of IR-UWB Pulses /3

Simulation of s(t):

- Create the transmitted samples .

```
tx_samples;  %(DS-PAM signal)
```

- Use upsample matlab function to get distinct samples spaced by
  .

```
Tx_signal = upsample(tx_samples,
 int32(bit_duration./sampling_time));
```

- Filter the oversampled signal with the impulse response .

```
UWB_TX = filter(gausBpam ,1, Tx_signal)
```

Transmission of IR-UWB Pulses /4

UWB_transmitter.m

```
function [UWB_Tx] = UWB_transmitter(num_bits,Ns,
         Np,sampling_time ,bit_duration)
 % impulse response of pulse shape filter
win_len     = 20;
pulse_len   = sampling_time*win_len;
pulse_time  = (-win_len:1:win_len)* sampling_time;
gausBpam    = gauss_bpam(pulse_len, pulse_time);
bits       = info_bits(num_bits);  % information bits
repbits    = rep_coder(bits,Ns);  %  repeated bits
DScode     = DS_code (Np);  % code
pam_sig    = DS_PAM(repbits,DScode);
tx_samples = diff_enc (pam_sig, 1 );
Tx_signal  = upsample(tx_samples, int32(bit_duration./ sampling_time));
 UWB_Tx     = filter(gausBpam ,1, Tx_signal);
 end
```

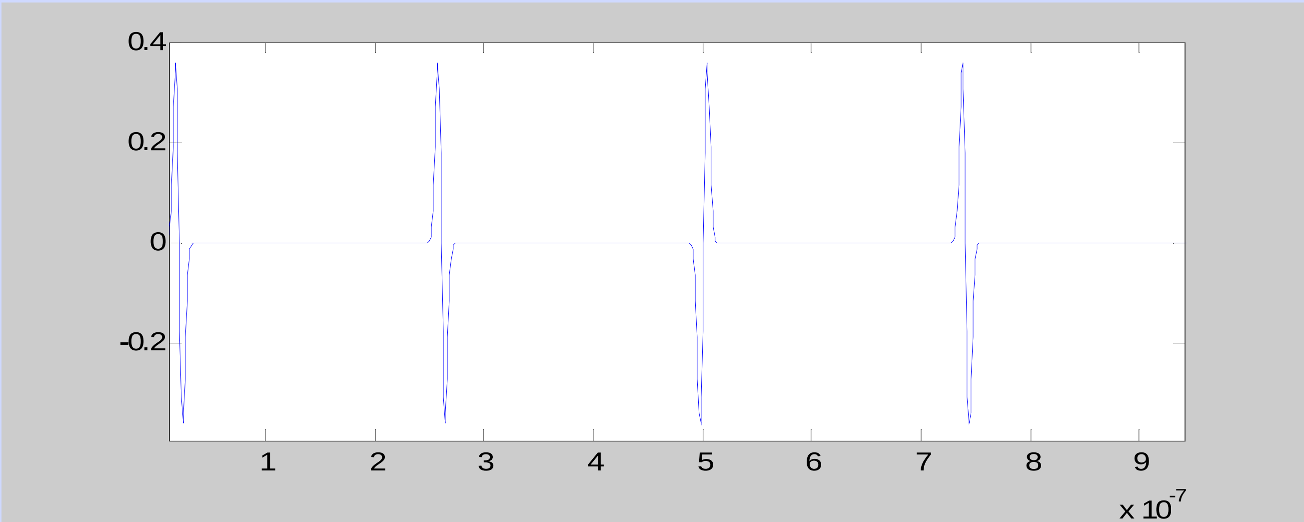
Transmission of IR-UWB Pulses /5

<u>Try to do…</u>

Plot the output of IR-UWB transmitter If number of bits = 5, Ns=3, Np=3, sampling time= 1e-9 and bit duration= 240 e-9.

<u>Simulation</u>

```
UWB_Tx = UWB_transmitter
(5,3,3,1e-9,240e-9);
plot([1:length(UWB_Tx)].*1
e-9,UWB_Tx)
```



Transmission of IR-UWB Pulses /6

Exercise

Modify the UWB_transmitter.m

Replace some modulors with your C-Mex function.

Source bits	bits = gen_bits
Repetition coder	rep_code
Differential encoder	diff_enc
Pulse shaper	guass_bpam