Valsts pētījumu programma
"**Informācijas tehnoloģiju zinātniskā bāze**"

Projekta Nr.3

"Oriģinālu signālu apstrādes paņēmienu izveide un izpēte konkurētspējīgu IT tehnoloģiju radīšanai"

INFORMATĪVĀ ATSKAITE
PAR PROJEKTA SESTĀ ETAPA REALIZĀCIJU
(2009. gada janvāris - decembris)

Atskaite sagatavota: 24.11.2009.g.
IZPILDĪTĀJA līguma uzskaites Nr. 09-VP-1-2
PASŪTĪTĀJA līguma uzskaites Nr. 09-VP-1-2
Programmas koordinators:
Dr.habil.sc.comp. I. Biļinskis, „Elektronikas un datorzinātņu institūts"
Projekta vadītājs: Dr. sc. comp. M. Greitāns, „Elektronikas un datorzinātņu institūts"

Rīga - 2009

Saturs

1. Akustiskā diapazona signālu analīze, izmantojot izveidotos notikumvadīti pārveidotu nestacionāru signālu apstrādes paņēmienus.

Iepriekšējā etapa ietvaros tika aprakstīta signāla laikformas atjaunošana no līmeņu-šķērsojumu nolasēm, izmantojot sinc-funkciju filtru ar laikā mainīgu frekvenču caurlaides joslu. Joslas platums, kas atkarīgs no signāla frekvenču īpašībām, tika atrasts, izmantojot līmeņu-šķērsojumu nolašu laika momentus. Signāla atjaunošanai tika izmantots iteratīvs algoritms, kas balstījās uz nolašu interpolāciju un interpolētā signāla filtrēšanu ar iegūto filtru. Tālāk, līdzīgi apstrādājot kļūdas signālu, tika uzlabota atjaunošanas precizitāte. Iegūtie rezultāti publicēti (M. Greitans and R. Shavelis "*Signal-dependent sampling and reconstruction method of signals with time-varying bandwidth*") un prezentēti starptautiskā konferencē SAMPTA 2009 (Marseļa, Francija).
Šī etapa ietvaros ir izstrādāta pēc līmeņu šķērsojumu principa diskretizēta signāla atjaunošanas metode, kas ņem vērā atjaunotā signāla vērtību ierobežojumus starp atbilstošiem diskretizācijas līmeņiem. Metodes apraksts un signālapstrādes piemēri doti 1.1. nodaļā. Iegūtie rezultāti publicēti (M. Greitans and R. Shavelis *"Signal-dependent techniques for non-stationary signal sampling and reconstruction"*) starptautiskā konferencē EUSIPCO 2009 (Glāzgova, Lielbritānija). Apstrādes praktiskā realizācija parādīta 1.2. nodaļā.

1.1. Signālu atjaunošana no līmeņu-šķērsojuma nolasēm

Saskaņā ar nolašu teorēmu jebkuru frekvenču joslā līdz F_{max} ierobežotu signālu $s(t)$ var atjaunot no tā laikā vienmērīgi izvietotām nolasēm $s(t_n)$, ja diskretizācijas frekvence ir vienāda vai pārsniedz $2F_{max}$ vērtību. Atjaunoto signālu nosaka izteiksme

$$s(t)=\sum_{n=-\infty}^{\infty} s(t_n)\,h_1(t,t_n), \tag{1.1}$$

kur $h_1(t,t_n)$ ir atjaunošanas filtra impulsa reakcija – klasiski sinc-funkcija

$$h_1(t,t_n)=\mathrm{sinc}(2\pi F_{max}(t-t_n)) \tag{1.2}$$

Ja apskata nestacionāru signālu ar laikā mainīgu frekvenču joslas platumu $f_{max}(t)$, tad tā diskretizācijas frekvencei klasiski jābūt vismaz $2F_{max}$, kur $F_{max}\geq\max(f_{max}(t))$. Tātad diskretizācijas procesā signāla nestacionaritāte (laikā mainīgs frekvenču joslas platums) tiek ignorēta. Lai to ņemtu vērā, impulsa reakcijas (1.2) vietā ir piedāvāts [1] izmantot

$$h_2(t,t_n)=\mathrm{sinc}\,(\Phi(t)-\Phi(t_n)), \tag{1.3}$$

kur

$$\Phi(t)=2\pi\int_0^t f_{max}(t)\,dt \tag{1.4}$$

ir svārstību fāze sinusoīdai, kuras frekvence laikā mainās pēc likuma $f_{max}(t)$, bet $t\geq 0$. Šajā gadījumā nolases $s(t_n)$ tiek ņemtas laika momentos t_n, kad $\Phi(t_n)=n\pi$. Stacionāra un frekvenču joslā F_{max} ierobežota signāla gadījumā izteiksmi (1.2) iegūst no (1.3), ja $f_{max}(t)=\mathrm{const}=F_{max}$. Nestacionāra signāla gadījumā atjaunošanas filtra impulsa reakcija $h_2(t,t_n)$ ir atkarīga no signāla momentānās frekvences $f_{max}(t)$ un nolašu izvietojums laikā ir nevienmērīgs ar vidējo diskretizācijas frekvenci mazāku par $2F_{max}$. Nolašu teorēma šajā gadījumā izpildās lokāli – signāla

momentānā diskretizācijas frekvence ir vienāda ar $2 f_{max}(t)$ [1].

Praksē informācija par diskretizējamā signāla maksimālo frekvenci $f_{max}(t)$ parasti nav dota, tāpēc tiek izmantota līmeņu-šķērsojuma diskretizācija. No iegūtajiem līmeņu šķērsojumu laika momentiem t_m var atrast maksimālās frekvences tuvinājumu $\hat{f}_{max}(t)$, kuru tālāk izmanto signāla atjaunošanai [1,2].

Lai no līmeņu-šķērsojumu nolasēm $s(t_m)$ veiktu signāla atjaunošanu saskaņā ar izteiksmi

$$\hat{s}(t)=\sum_{n=1}^{N} s(t_n)h_2(t,t_n), \tag{1.5}$$

jāatrod nolašu $s(t_n)$ vērtības. Kļūdas signāla enerģija

$$E=\sum_{m=1}^{M}[\hat{s}(t_m)-s(t_m)]^2, \tag{1.6}$$

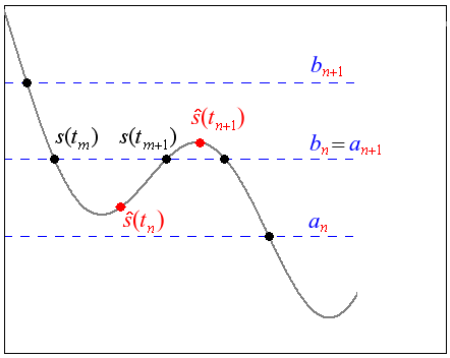
kur

$$\hat{s}(t_m)=\sum_{n=1}^{N}\hat{s}(t_n)h_2(t_m,t_n), \tag{1.7}$$

būs atkarīga no atrastajām $\hat{s}(t_n)$ vērtībām, tāpēc nolases meklē tā, lai kļūdas signāla enerģija būtu minimāla. Pielīdzinot parciālos atvasinājumus $\frac{\partial E}{\partial \hat{s}(t_n)}$, kur $n=1,2,\ldots,N$, nullei, iegūst lineāru vienādojumu sistēmu, kuras atrisinājums ir meklētās $\hat{s}(t_n)$ vērtības. Atrisinājums

$$\hat{\mathbf{S}}=\mathbf{S}\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}, \tag{1.8}$$

kur $\hat{\mathbf{S}}=[\hat{s}(t_1),\hat{s}(t_2),\ldots,\hat{s}(t_N)]^T$, $\mathbf{S}=[s(t_1),s(t_2),\ldots,s(t_M)]$ un $\mathbf{H}$ ir $M\times N$ matrica, kuras elements rindā m un kolonnā n ir $h_2(t_m,t_n)$, var dot tādas $\hat{s}(t_n)$ vērtības, kas neatrodas starp atbilstošiem diskretizācijas līmeņiem. Piemēram, 1.1. zīmējumā nolasei $\hat{s}(t_n)$ jāatrodas starp līmeņiem a_n un b_n, bet nolasei $\hat{s}(t_{n+1})$ – starp līmeņiem a_{n+1} un b_{n+1}.



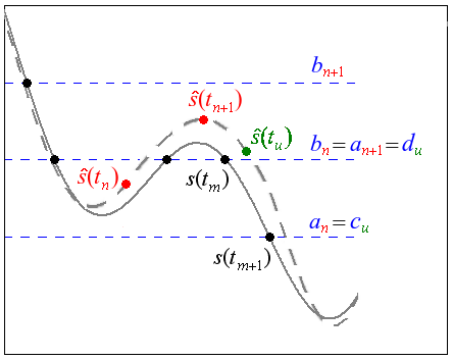
1.1. zīm. Meklējamās signāla nolases $\hat{s}(t_n)$

Lai novērstu iegūto $\hat{s}(t_n)$ vērtību neatbilstību diskretizācijas līmeņiem, kļūdas signāla enerģiju (1.6) minimizē, ievērojot nosacījumu $a_n\le\hat{s}(t_n)\le b_n$ katram n. Uzdevumu var atrisināt, izmanotojot Matlab optimizācijas funkcijas *quadprog* vai *lsqlin*.

Pēc $\hat{s}(t_n)$ aprēķina atrod atjaunotā signāla vērtības laika momentos t_u

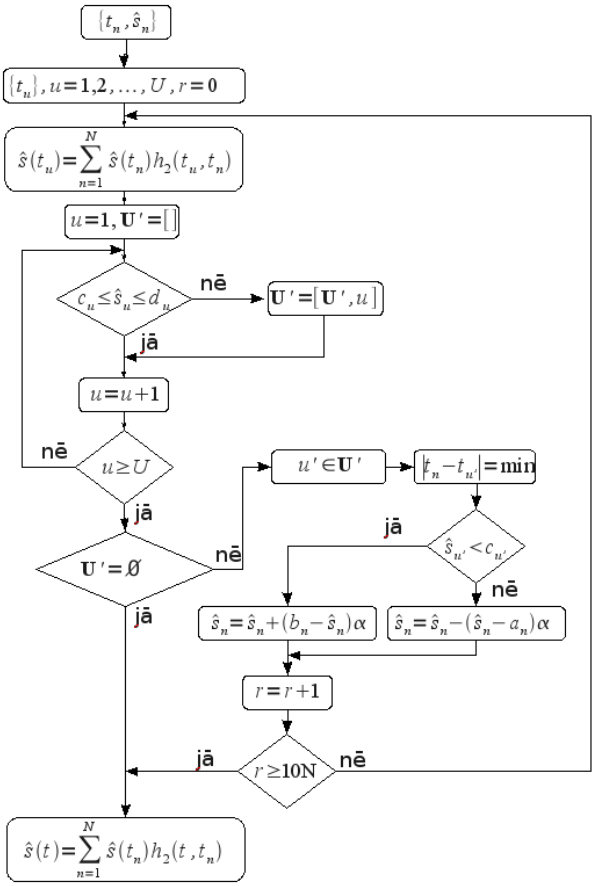
$$\hat{s}(t_u)=\sum_{n=1}^{N}\hat{s}(t_n)h_2(t_u,t_n), \tag{1.9}$$

kurām, līdzīgi kā $\hat{s}(t_n)$, vajadzētu atrasties starp atbilstošiem diskretizācijas līmeņiem $c_u \le \hat{s}(t_u) \le d_u$. Piemēram, 1.2. zīmējumā aprēķinātajām nolasēm $\hat{s}(t_n)$ un $\hat{s}(t_{n+1})$ izpildās nosacījumi $a_n \le \hat{s}(t_n) \le b_n$ un $a_{n+1} \le \hat{s}(t_{n+1}) \le b_{n+1}$, savukārt atjaunotā signāla vērtībai laika momentā t_u nosacījums $c_u \le \hat{s}(t_u) \le d_u$ neizpildās (signāls laika intervālā no t_m līdz t_{m+1} atrodas starp līmeņiem c_u un d_u). Tas nozīmē, ka $\hat{s}(t_{n+1})$ ir jāsamazina.



1.2. zīm. Atjaunotais signāls (svītrlīnija)

Ņemot vērā, ka atjaunotā signāla vērtībām jebkuros laika momentos jāatrodas starp atbilsošiem diskretizācijas līmeņiem, var izveidot nolašu $\hat{s}(t_n)$ korekcijas algoritmu (1.3. zīm.), ar kuru $\hat{s}(t_n)$ vērtības tiek palielinātas vai samazinātas (ievērojot nosacījumu $a_n \le \hat{s}(t_n) \le b_n$), lai uzlabotu atjaunošanas precizitāti [2].



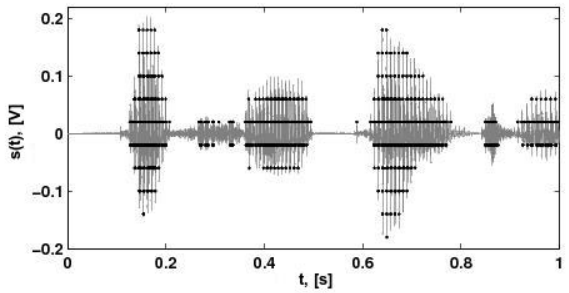
1.3. zīm. Nolašu $\hat{s}(t_n)$ korekcijas algoritms

Algoritma pamatā ir sekojošas darbības:

1) tiek izvēlēti vienmērīgi izvietoti laika momenti $\{t_u\}$, $u = 1,2,\ldots,U$;
2) tiek aprēķinātas signāla vērtības $\hat{s}(t_u)$;
3) atrod visus indeksus u, kuriem neizpildās nosacījums $c_u \le \hat{s}(t_u) \le d_u$;
4) no šiem ideksiem pēc gadījuma rakstura izvēlas vienu, piemēram, u' un atrod laika momentam $t_{u'}$ tuvāko t_n vērtību;
5) ja $\hat{s}(t_{u'}) < c_{u'}$, tad $\hat{s}(t_n)$ tiek palielināts par $(b_n - \hat{s}(t_n))\alpha$, kur $0 < \alpha < 1$ nosaka pieauguma straujumu, bet ja $\hat{s}(t_{u'}) > c_{u'}$, tad $\hat{s}(t_n)$ tiek samazināts par $(\hat{s}(t_n) - a_n)\alpha$;
6) rezultātā mainījusies ir viena $\hat{s}(t_n)$ vērtība, kas ietekmē atjaunoto signālu visā tā garumā, tāpēc signāls tiek pārrēķināts no jauna un procedūra atkārtojas no 2. līdz 6. solim.

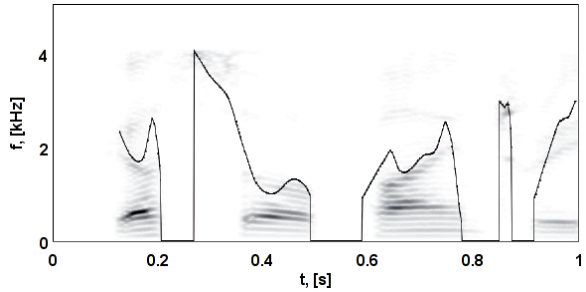
Algoritms tiek pārtraukts, ja katram u izpildās nosacījums $c_u \le \hat{s}(t_u) \le d_u$ vai tiek sasniegts noteikts iterāciju skaits.

Aprakstīto metodi pārbaudīsim MATLAB-ā (programmas 1.pielikumā), veicot runas signāla atjaunošanu no līmeņu-šķērsojumu nolasēm [2]. Signāls tiek ņemts no TIMIT datu bāzes (/timit/train/dr1/ mtpf0/sx335.wav) un ir 3.8 sekundes garš. Signāla diskretizācijas frekvence ir 16 kHz. Lai iegūtu līmeņu-šķērsojuma nolases, tas tiek interpolēts ar sinc-funkcijām, kā rezultātā signāla spektrs tiek ierobežots līdz 4 kHz. Izmantojot 10 diskretizācijas līmeņus, iegūst 3301 līmeņu-šķērsojuma nolases, kas ir aptuveni 10 reizes mazāk nekā vienmērīgas diskretizācijas gadījumā, ja signālu diskretizē ar 8 kHz frekvenci. Nolašu skaita samazinājumu galvenokārt nosaka klusuma pauzes starp vārdiem (1.4. zīmējums).



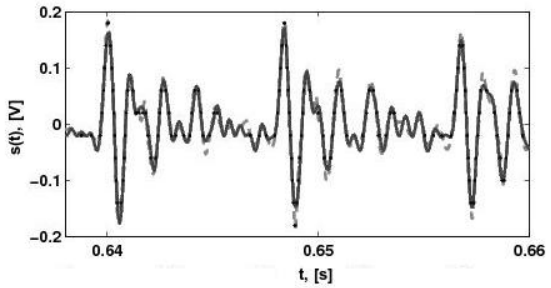
1.4. zīm. Runas signāls un tā līmeņu-šķērsojuma nolases (melni punkti)

Veicot signāla atjaunošanu, sākumā atrod tā momentāno maksimālo frekvenci $\tilde{f}_{max}(t)$, kas nepieciešama matricas **H** aprēķinam. Lai novērtētu līknes atbilstību patiesajām frekvenču vērtībām, iegūst vienmērīgi diskretizēta signāla laika-frekvenču attēlojumu, pielietojot īsintervāla Furjē transformāciju. Rezultāts parādīts 1.5. zīmējumā, kurā signāla spektra komponentes ar lielāku jaudu attēlotas tumšākā krāsā. Kā redzams, tad $\tilde{f}_{max}(t)$ (nepārtraukta līnija) labi izseko signāla spektra augstāko frekvenču vērtībām.

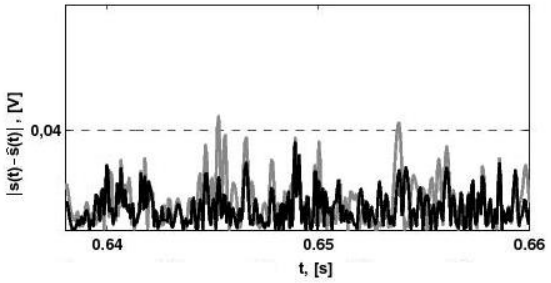


1.5. zīm. Signāla laika-frekvenču attēlojuma salīdzinājums ar aprēķināto $\tilde{f}_{max}(t)$

Pēc $\tilde{f}_{max}(t)$ noteikšanas seko atjaunošanas procedūra. Izteiksme (1.6) tiek minimizēta, ievērojot (1.7) un meklējamo nolašu vērtību ierobežojumu $a_n \le \hat{s}(t_n) \le b_n$. Kad ir atrastas $\hat{s}(t_n)$ vērtības, seko algoritma (1.3. zīm.) izpilde. Laika momenti $\{t_u\}$ tiek izvēlēti vienmērīgi ar diskretizācijas frekvenci 64 kHz. Pēc pirmās pārbaudes 30% $\hat{s}(t_u)$ vērtību neizpildās nosacījums $c_u \le \hat{s}(t_u) \le d_u$ un atjaunošanas kļūda $\sqrt{U^{-1}\sum_{u=1}^{U}(s(t_u)-\hat{s}(t_u))^2}$ ir 22 mV, savukārt pēc $10N$ iterācijām šis skaitlis samazinās līdz 8% un kļūda ir 15 mV. Atjaunotā signāla fragments parādīts 1.6. zīmējumā ar nepārtrauktu līniju, bet kļūdas signāls $|(s(t)-\hat{s}(t))|$ pirms un pēc korekcijas algoritma izpildes parādīts 1.7. zīmējumā ar attiecīgi pelēku un melnu līniju. Kļūdas signāla amplitūda nepārsniedz 40 mV, kas ir attālums starp blakus esošiem diskretizācijas līmeņiem.



1.6. zīm. Sākotnējā (svītrlīnija) un atjaunotā (nepārtraukta līnija) runas signāla fragmenta salīdzinājums

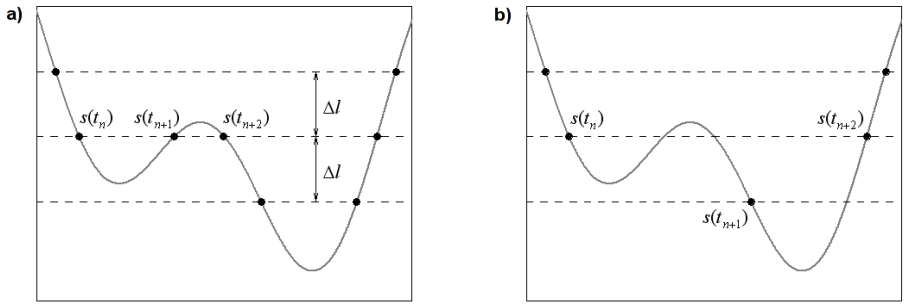


1.7. zīm. Kļūdas signāls pirms (pelēka līnija) un pēc (melna līnija) korekcijas algoritma izpildes.

Atjaunošanas kvalitāte ir atkarīga no apstrādājamo nolašu skaita. Diskretizējot signālu ar 20 līmeņiem, iegūst 8509 nolases un atjaunošanas kļūda samazinās līdz 7 mV.

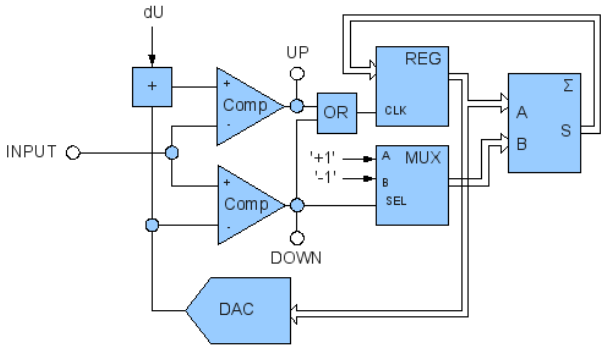
1.2. Signālatkarīgas apstrādes praktiskā realizācija

Praktiski tika realizēta signālatkarīga diskretizācija, kas darbojas pēc delta-imaiņu principa (1.8.b zīmējums). Delta-izmaiņu diskretizācijas gadījumā katra nākamā signāla nolase tiek ņemta laika momentā, kad signāla izmaiņa no iepriekšējās nolases vērtības sasniedz noteiktu delta slieksni Δl.



1.8. zīm. Līmeņu-šķērsojuma (a) un delta-izmaiņu (b) diskretizācija

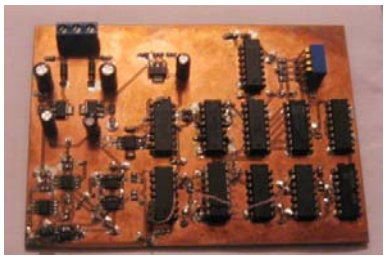
Praktiskās realizācijas blokshēma parādīta 1.9. zīmējumā.



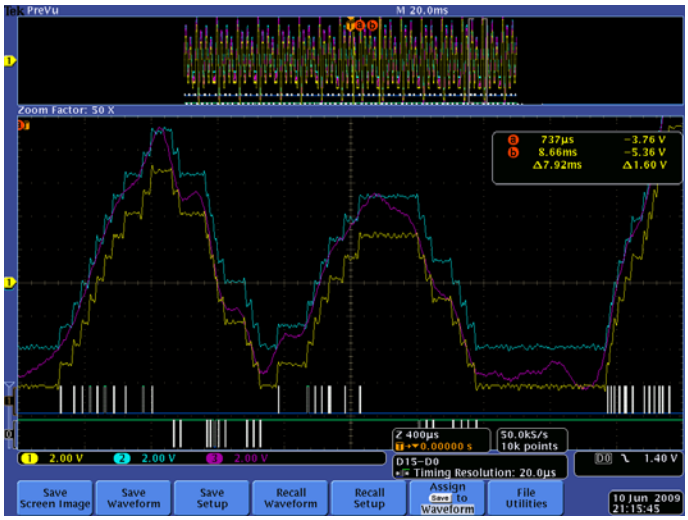
1.9. zīm. Delta-izmaiņu diskretizācijas praktiskās realizācijas blokshēma

Ieejas signāls tiek pievadīts diviem komparatoriem, kur to salīdzina ar sprieguma līmeņiem (diskretizācijas līmeņiem) u_1 un $u_2 = u_1 + du$, turkāt $u_1 < s(t) < u_2$ (ja du vietā izvēlas $du/2$ vērtību, tad delta-izmaiņu diskretizācijas vietā iegūst līmeņu-šķērsojuma diskretizāciju (1.8.a zīmējums)). Kad signāls pieaug un šķērso u_2 līmeni (tātad $s(t) > u_2$), nostrādā augšējais komparators, kā rezultātā DAC ieejas bitu secībai summējas "+1", tā izejas spriegums pieaug par $du/2$ un jaunajām u_1 un u_2 vērtībām izpildās $u_1 < s(t) < u_2$. Kad signāls samazinās un šķērso u_1 līmeni (tātad $s(t) < u_1$), nostrādā apakšējais komparators, kā rezultātā DAC ieejas bitu secībai summējas "–1", tā izejas spriegums samazinās par $du/2$ un jaunajām u_1 un u_2 vērtībām izpildās $u_1 < s(t) < u_2$.

Saskaņā ar 1.9. zīmējumā attēloto blokshēmu izgatavota iespiedplate (1.10. zīmējums), kurā realizēti 16 diskretizācijas līmeņi, kas izvietoti vienmērīgi diapazonā no –10V līdz +10V. Ar osciloskopu uzņemtais runas signāla delta-izmaiņu diskretizācijas rezultāts parādīts 1.11. zīmējumā (runas signāls vieolētā krāsā, DAC izejas signāls u_1 dzeltenā krāsā un signāls $u_2 = u_1 + du$ zilā krāsā).



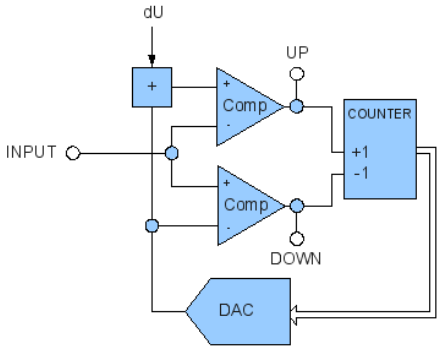
1.10. zīm. Iespiedplate signāla diskretizācijai pēc delta-izmaiņu principa



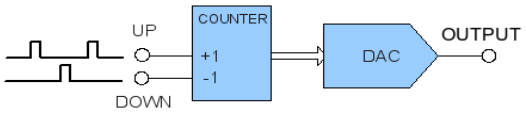
1.11. zīm. Ar osciloskopu uzņemtā runas signāla (violētā krāsā) delta-izmaiņu diskretizācijas

rezultāts (DAC izejas signāls dzeltenā krāsā)

Tā kā līmeņu šķērsojumu notikumos DAC ieejas bitu secība palielinās vai samazinās attiecīgi par "+1" vai "–1", tad 1.9. zīmējumā pārādīto blokshēmu var vienkāršot, aizstājot reģistru, multipleksoru un summatoru ar "+/–" skaitītāju (1.12. zīmējums).



1.12. zīm. Delta-izmaiņu diskretizācijas praktiskās realizācijas vienkāršota blokshēma



1.13. zīm. Pēc delta-izmaiņu principa diskretizēta signāla atjaunotāja bloskhēma

1.14. zīm. Iespiedplate signāla atjaunošana no līmeņu-šķērsojumu nolasēm

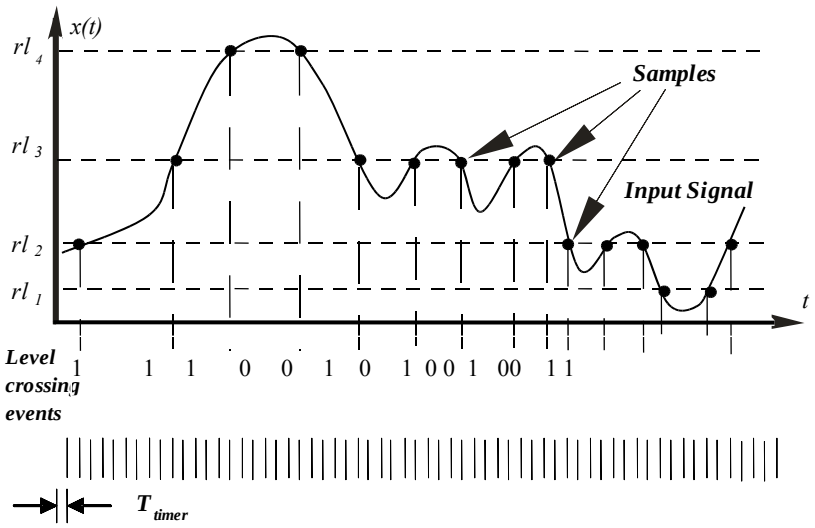
Signāla vienkāršotai atjaunošanai izgatavota 1.14. zīmējumā parādītā iespiedplate, kas darbojas saskaņā ar 1.13. zīmējuma blokshēmu. Kā ieejas signāli tiek ņemti 1.12. zīmējuma blokshēmas komparatoru izejas signāli "up" un "down". Šos signālus pievada "+/–" skaitītājam, kura izejas ciparu signāls tiek pārvērsts analogā signālā. Rezultātā iegūst kāpņveida signālu (sakrīt ar 1.11. zīmējumā parādīto DAC izejas signālu), kuru tālāk nogludinot iegūst sākotnējā signāla tuvināto formu.

Literatūra

1. R.Shavelis and M.Greitans, *"Signal-dependent sampling and reconstruction method of signals with time-varying bandwidth"*, Proceedings of the Int. Conf. on Sampling Theory and Applications SAMPTA 2009, Marseille, France, May (published on CD).
2. M.Greitans and R.Shavelis, *"Signal-dependent techniques for non-stationary signal sampling and reconstruction"*, Proceedings of the European Signal Processing Conference EUSIPCO 2009, Glasgow, UK, August.

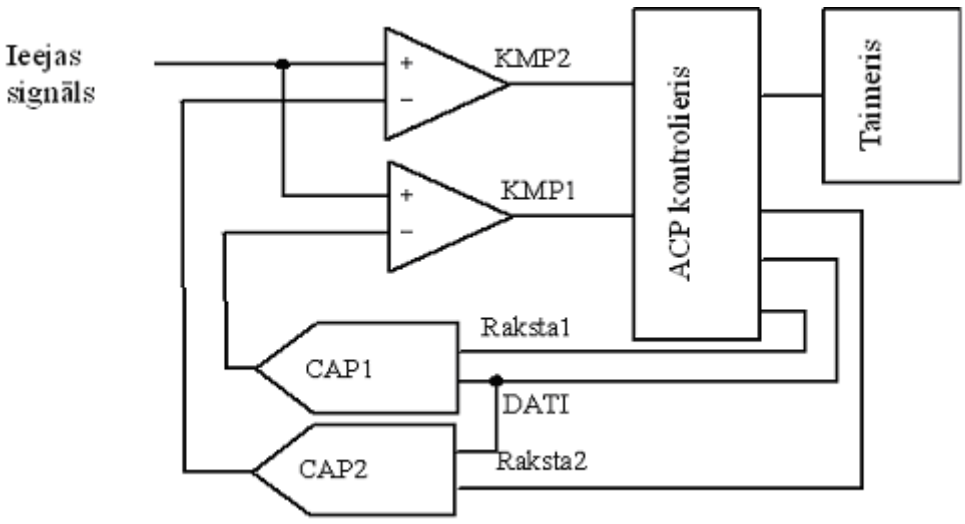
1.3. Adaptētu etalonlīmeņu pielietošana līmeņu šķērsojuma shēmās

Adaptētu etalonlīmeņu pielietošana līmeņu šķērsojuma shēmās ir viena no iespējām, ko var izmantot nestacionāru signālu apstrādei, lai samazinātu nolašu skaitu. Tā izmanto etalonlīmeņu kopu, kas tiek pielāgota pārveidojamā signāla izmaiņām laikā, kā attēlots 1.zīmējumā. Tika izvirzīts pieņēmums, ka adaptētu etalonlīmeņu pielietošanas metode ļauj samazināt etalonlīmeņu, skaitu, nepasliktinot atjaunojamā signāla kvalitāti salīdzinājumā ar vienādā attālumā izvietotu (neadaptētu) etalonlīmeņu pielietošanu.



1.zīm. Ieejas signālam adaptēti etalonlīmeņi $rl_1 \div rl_4$.

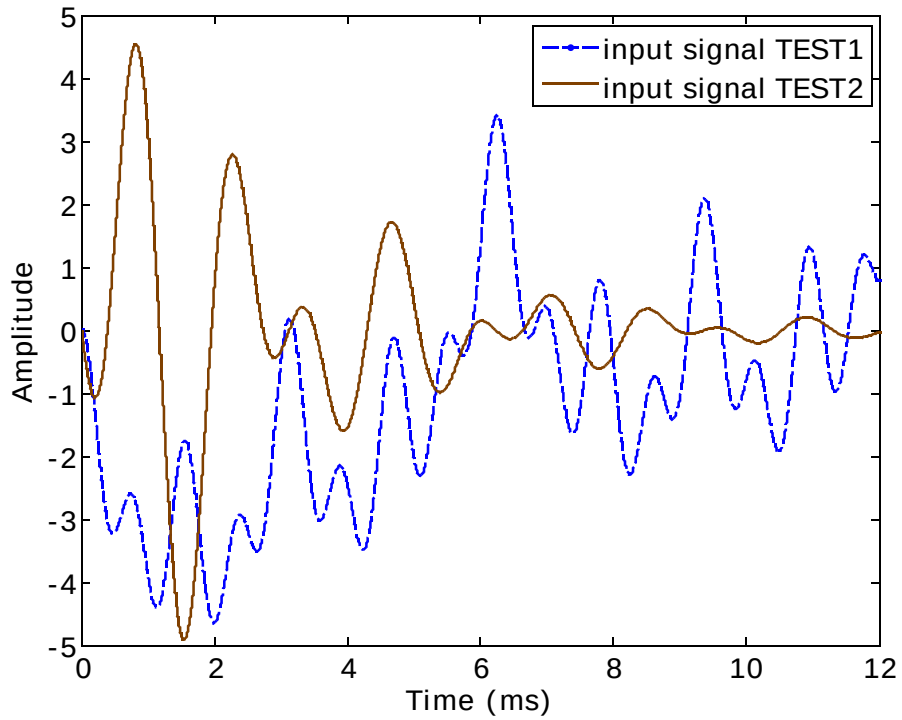
Šāda pieņēmuma pārbaudei tika izmantota 2.zīmējumā attēlotās shēmas modelis, kur cipar-analogo pārveidotāju CAP1 un CAP2 izejās etalonlīmeņu vērtības neizmainās vienmērīgi, bet izmainās saskaņā ar noteiktu sadalījumu, kurš glabājas ACP kontroliera atmiņā. Šajā shēmā pie-



2.zīm. Līmeņu šķērsojuma analogciparu pārveidotāja shēma.

lietotajiem CAP nav augstas prasības to linearitātei, kas vienkāršo izmantojamos shēmas

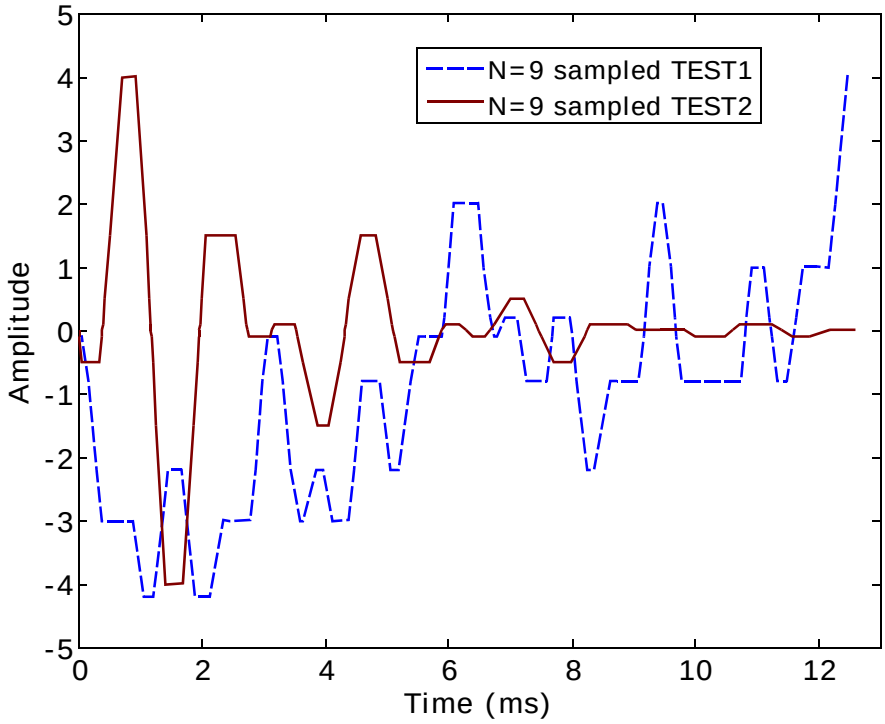
risinājumus.
Metodes pārbaudei izmantoja 2 testa signālus TEST1 un TEST2, kas parādīti 3.zīmējumā.



3.zīm. Ieejas testa signāli TEST1 un TEST2.

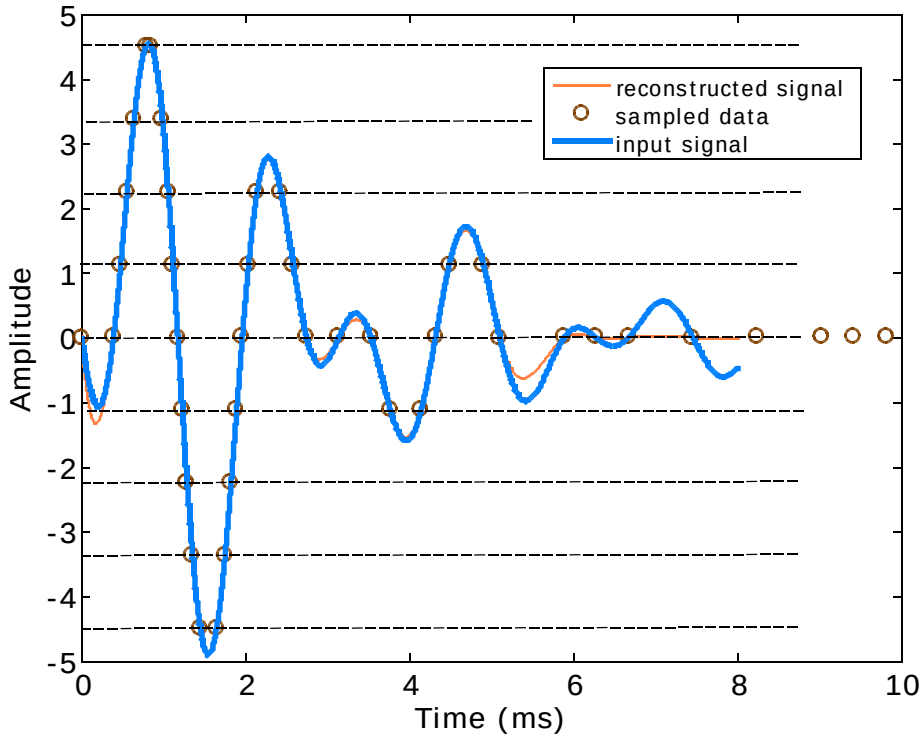
Katrs signāls tika ciparots 10 reizes, izmantojot dažādu etalonlīmeņu skaitu. Tika izmantoti 5, 7, 9, 13 un 16 gan vienādā attālumā izvietoti (neadaptēti), gan adaptēti etalonlīmeņi.

Kā piemērs 4.zīmējumā attēloti TEST1 un TEST2 signālu līmeņu šķērsojuma nolases, pielietojot 9 adaptētus etalonlīmeņus.



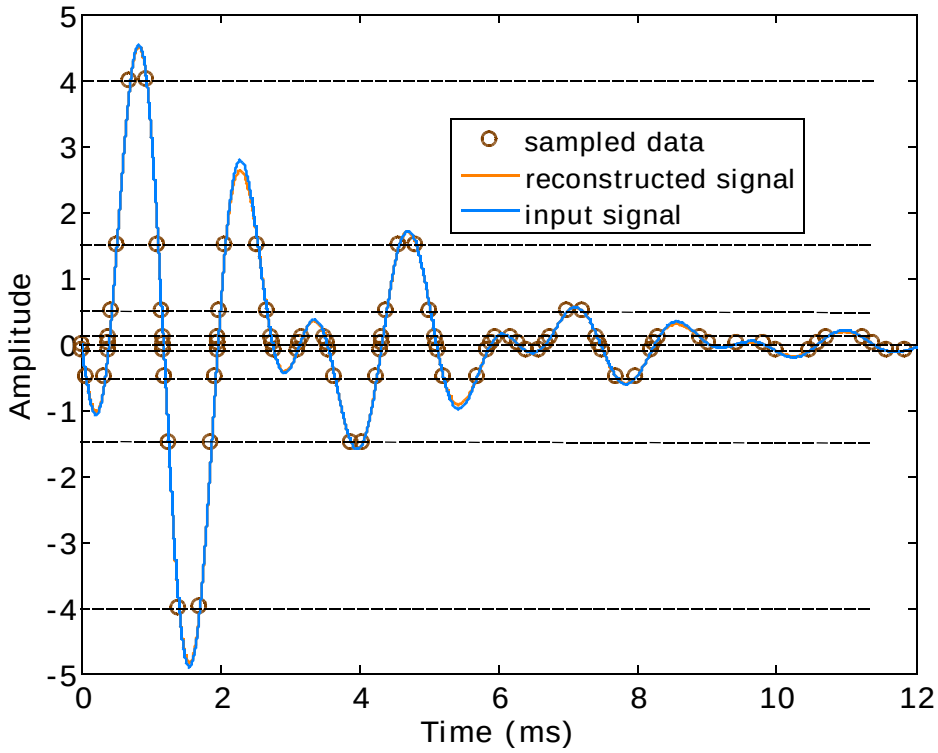
4.zīm. TEST1, TEST2 signālu līmeņu šķērsojuma nolases pie 9 adaptētiem etalonlīmeņiem.

Iegūto signālu nolašu kvalitāte tika novērtēta, salīdzinot ieejas testa signālu ar atjaunoto signālu, kas tika iegūts no testa signālu līmeņu šķērsojuma nolasēm. Signālu atjaunošana notika vienmērīgi diskretizējot testa signālu līmeņu šķērsojuma nolases ar pietiekoši augstu diskretizācijas frekvenci, kas nodrošina pienācīgu virsdiskretizāciju (oversampling) un sekojoši interpolējot tās ar kubiskā splaina funkciju.



5.zīm. TEST2 signāls un tā atjaunotais signāls no līmeņu šķērsojuma nolasēm pie neadaptētiem 9

etalonlīmeņiem.



6.zīm. TEST2 signāls un tā atjaunotais signāls no līmeņu šķērsojuma nolasēm pie adaptētiem 9 etalonlīmeņiem.
Vizuāli salīdzinot 5. un 6. zīmējumus redzams, TEST2 signāls un tā atjaunotais signāls redzami atšķiras neadaptētu etalonlīmeņu gadījumā un nebūtiski atšķiras adaptētu etalonlīmeņu gadījumā.

Kvantitatīvai atšķirību novērtēšanai starp testa signāliem un atjaunotajiem signāliem izmanto korelācijas koeficientu, ko aprēķina izmantojot MatLab funkciju:

$$R = corrcoef(x, y), \tag{1}$$

kur R ir korelācijas koeficients, x ir ieejas un y ir atjaunotais signāls.
Aprēķinot korelācijas koeficientu dažādu etalonlīmeņu skaitam gan neadaptētiem, gan adaptētiem etalonlīmeņiem tika iegūtas sekojošas korelācijas koeficientu vērtības, kas parādītas 1. un 2. tabulā.

Tabula 1. Korelācija starp testa signāliem un atjaunotajiem signāliem dažādam etalonlīmeņu skaitam N pie neadaptētiem etalonlīmeņiem

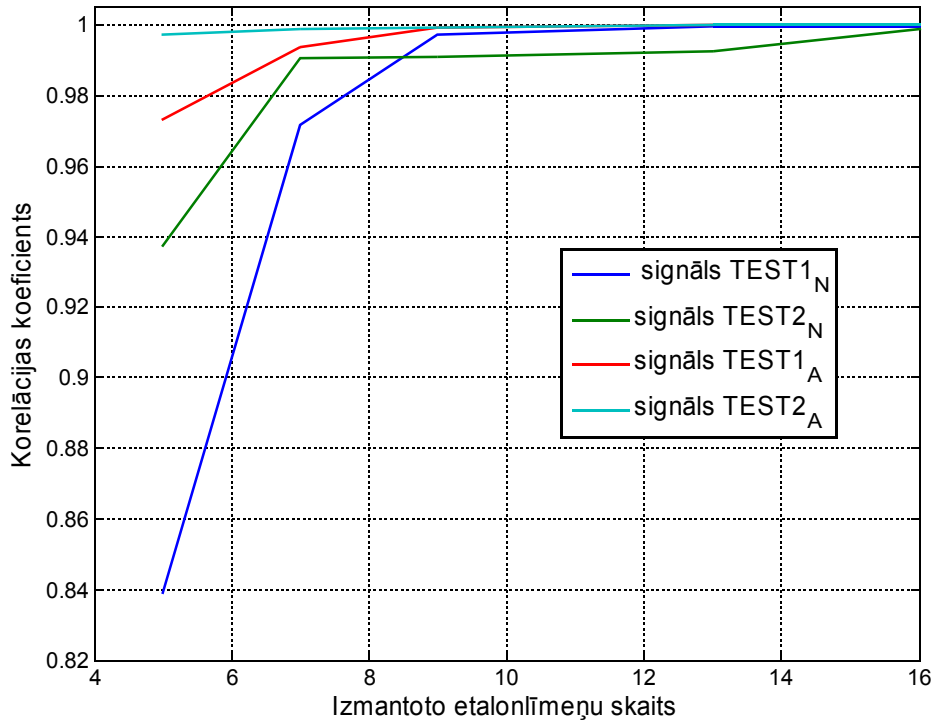
N	TEST1	TEST2
5	0.8392	0.9376
7	0.9716	0.9907
9	0.9975	0.9910
13	0.9997	0.9927
16	0.9998	0.9989

Tabula 2. Korelācija starp testa signāliem un atjaunotajiem signāliem dažādam etalonlīmeņu skaitam N pie adaptētiem etalonlīmeņiem

N	TEST1	TEST2
5	0.9735	0.9974
7	0.9937	0.9988

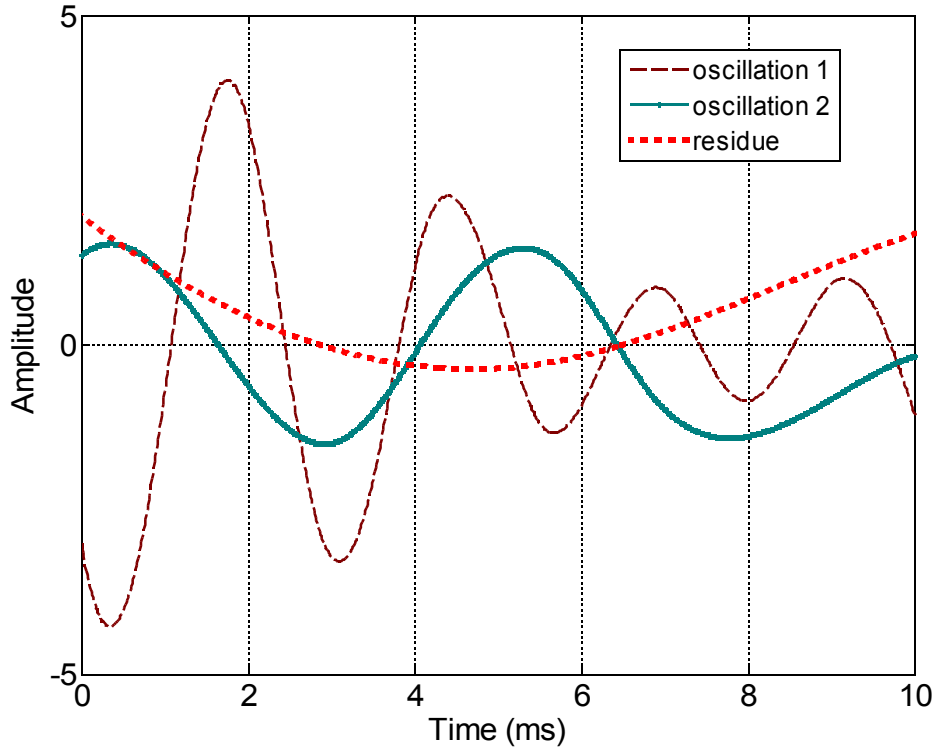
9	0.9991	0.9992
13	0.9999	1.0000
16	1.0000	1.0000

Kā redzams pie maza etalonlīmeņu skaita $N < 9$ korelācija ir ievērojami labāka adaptētiem etalonlīmeņiem.
Grafiskais tabulu 1 un 2 attēlojums ir parādīts 7.zīmējumā.



7.zīm. Ieejas signālu un atjaunoto signālu korelācija.

Lai novērtētu ne vien ieejas signālu un atjaunoto signālu korelāciju bet arī to komponenšu korelāciju, ieejas signāls un atjaunotais signāls tika sadalīti tos veidojošās komponentēs. Pēc sadalīšanas tika novērtēta ieejas signālu un atjaunoto signālu komponenšu korelāciju. Signālu dekompozīcijai tika izmantota empīriska modas dekompozīcija (EMD), kas piemērota nestacionāru signālu apstrādei. 8. zīmējumā attēlota TEST2 atjaunotā signāla empīriskās modas dekompozīcija (EMD) pie $N = 9$, ko veido 2 svārstības un signāla atlikums.



8.zīm. TEST2 atjaunotā signāla empīriskās modas dekompozīcija (EMD) pie $N = 9$.
Signālu apstrādei tika izmantota MatLab EMD programma *rParabEmd*. Aprēķinot korelācijas koeficientu signālu EMD dažādu etalonlīmeņu skaitam gan neadaptētiem, gan adaptētiem etalonlīmeņiem tika iegūtas sekojošas korelācijas koeficientu vērtības, kas parādītas 3. un 4. tabulā

Tabula 3. Korelācija starp testa signālu un atjaunoto signālu EMD dažādam etalonlīmeņu skaitam N neadaptētiem etalonlīmeņiem

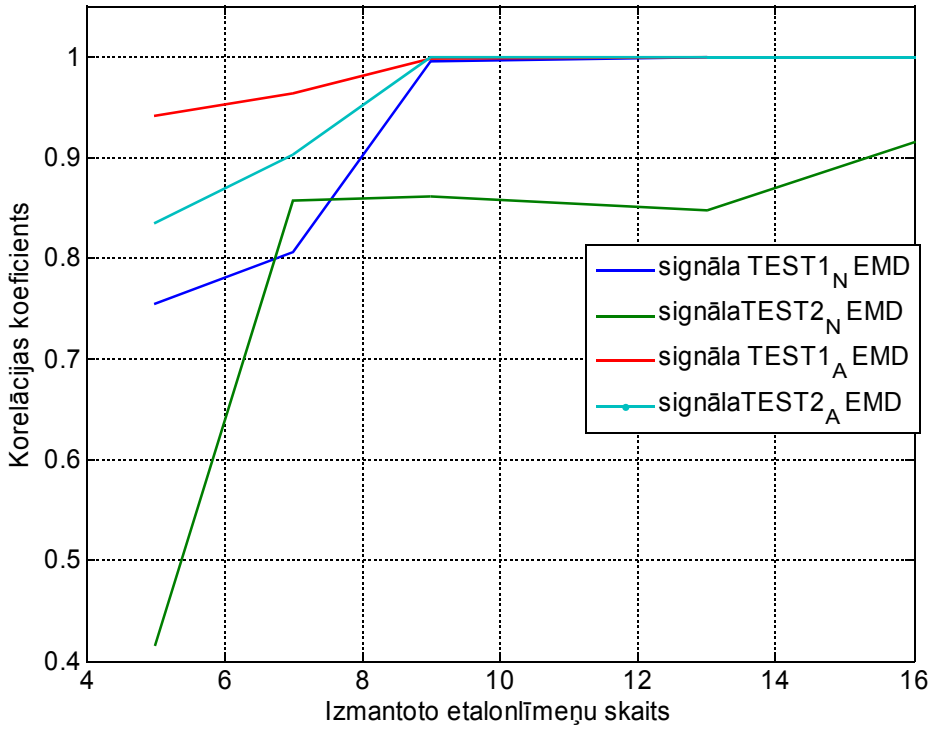
N	TEST1	TEST2
5	0.7543	0.4161*
7	0.8056	0.8567*
9	0.9957	0.8617*
13	0.9994	0.8478*
16	0.9995	0.9152

Tabula 4. Korelācija starp testa signālu un atjaunoto signālu EMD dažādam etalonlīmeņu skaitam N adaptētiem etalonlīmeņiem

N	TEST1	TEST2
5	0.9409	0.8351
7	0.9640	0.9026
9	0.9981	0.9996*
13	0.9999	0.9998
16	1.0000	0.9998

* norāde ka korelācija nav aprēķināta visam signālam.

Kā redzams pie maza etalonlīmeņu skaita $N < 9$ korelācijas atšķirības EMD gadījumā ir daudz uzskatāmākas nekā signālu gadījumā. Grafiskais tabulu 3 un 4 attēlojums ir parādīts 9.zīmējumā



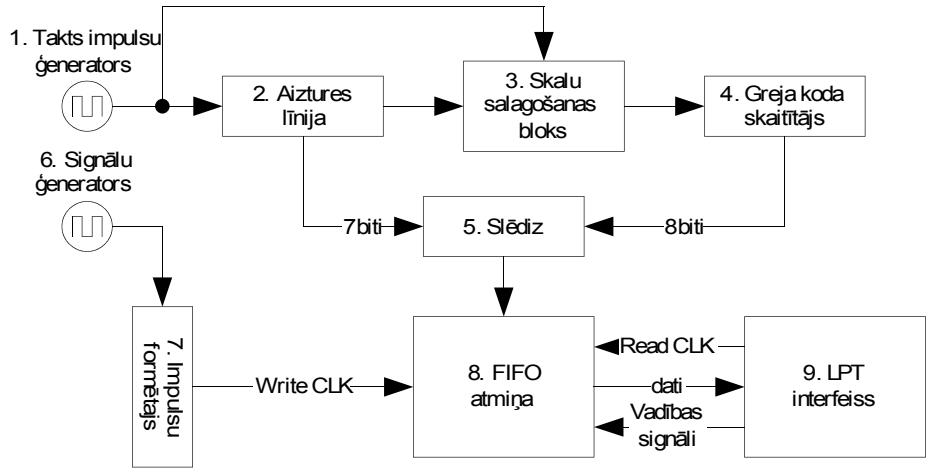
9.zīm. Ieejas signālu un atjaunoto signālu EMD korelācija

Literatūra

1. **Baums A., Greitans M., Grunde U.** Level-crossing sampling using microprocessor based system // Proceedings of the International Conference on Signals and Electronic Systems ICSES'08. –Krakow, Poland.– 2008. – P. 19-23.
2. **Huang N.E., Shen Z., Long S.R., et al.** The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis // Proceedings of the Royal Society. – 1998. – vol.454. –P. 903–995.
3. **Rato, R. T., Ortigueira, M. D., and Batista, A. G.** On the HHT, its problems, and some solutions // Mechanical Systems and Signal Processing. – 2008. – Vol. 22.– P. 1374-1394

Laika intervālu ciparošanai var pielietot dažādus risinājumus. Tie atšķiras ar precizitāti, shēmas risinājumiem u.c. parametriem. Sekojošā darbā apskatīts laika intervālu ciparošanas risinājums, pielietojot Greja skaitītāju un aiztures līnīju.

1.4. Laiks-cipars pārveidotāja maketa apraksts



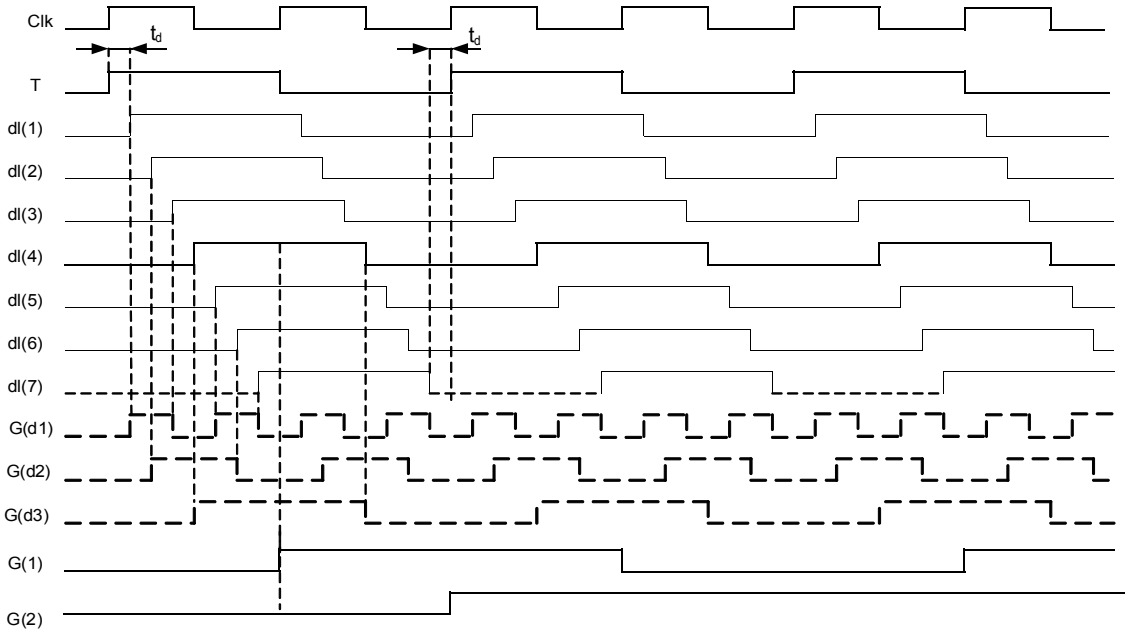
1. zīm. Izveidotā laiks-cipars pārveidotāja blokshēma.

Izveidotā laiks-cipars pārveidotāja maketa blokshēma ir attēlota 1. zīm. Atskaites pielikumā ir dotas bloku loģiskās shēmas. Kopumā sistēmas darbs sastāv no trijiem posmiem: skalu izveidošana, datu nolase FIFO atmiņā impulsa pienākšanas momentā, datu nolase no FIFO datorā tālākai apstrādei.

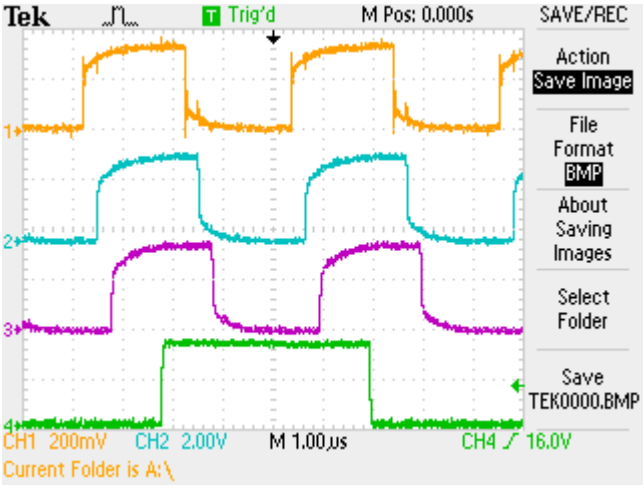
1.4.1. Skalu izveidošana un salāgošana

1. Takts impulss no takts impulsu ģeneratora (1) vienlaicīgi pienāk uz aiztures līnijas (2) un skalu salāgošanas bloka ieejām (3). Rezultātā aiztures līnijas izejās veidojas 7 bitu precīzā skala, dalot katru pienākošu takts impulsu ar 7 aiztures elementiem.
2. No aiztures līnijas vidējas izejas (4. izeja) signāls pienāk uz salāgošanas bloka (3) sinhronizācijas ieeju. Šis signāls atbloķē sinhronizācijas elementu kā rezultātā nākamais takts impulss no takts impulsu ģeneratora (1), pienākot uz salāgošanas bloka sinhronizācijas elementa ieeju, tiek izlaists tam cauri.
3. Takts impulss no salāgošanas bloka (3) pienāk uz Greja koda skaitītāju (4), kā rezultāta uz izejām tiek veidota 8 bitu rupja skala, kodēta Greja kodā.

2. zīm. ir parādītas precīzas skalas 7 pozīcijas (dl(1) – dl(7)), 3 bitu Greja kods (G(d1) – G(d3)), kas tiek veidots, apstrādājot nolasītus datus no aiztures līnijas un divas jaunākas Greja koda skaitītāja pozīcijas G(1) un G(2). Pateicoties salāgošanas blokam, Greja koda jaunākā pozīcija G1 sākas no aiztures līnija vidēja elementa (d4) vidus. 3. zīm. ir attēlots skalu izveidošanas un salāgošanas rezultāts. Pirmais signāls ir aiztures līnijas 2. pozīcija (d2), 2 signāls – 3. pozīcija (d3), 3 signāls – 4. pozīcija, un 4 signāls – Greja koda skaitītāja jaunākā pozīcija (G1).



2. zīm. Laiks-cipars pārveidotāja skalas un to salāgošana.



3. zīm. Skalu salāgošanas rezultāts.

Lai pārveidotu aiztures līnijas pozīcijas Greja kodā ir jāievēro sekojošo noteikumu: aiztures līnijas pozīciju skaitam jābūt vienādam ar

$$m = 2^k - 1 \qquad (1)$$

kur m – aiztures līnijas pozīciju skaits un k – veidota Greja koda pozīciju skaits.
Pārveidot nolasītu vērtību no aiztures līnija Greja kodā var sekojoši. Pieņemsim, ka tika nolasīta sekojošā vērtība : 0011111. (d6, d5..d0) Tad Greja koda (G3, G2, G1) pozīcijas var noteikt sekojoši:

$$G1 = d0 \text{ XOR } d2 \text{ XOR } d4 \text{ XOR } d6 \qquad (2)$$
$$G2 = d1 \text{ XOR } d5$$
$$G3 = d3$$

Rezultātā iegūsim vērtību – 111.

Precīzās skalas izmantošana dot acīmredzamu priekšrocību: laiks-cipars pārveidotāja

izšķirtspēja kļūst vienāda aiztures līnijas viena elementa aizturei t_d . Izmantojot tikai Greja koda skaitītāju laiks-cipars pārveidotāja izšķirtspēja ir vienāda ar takts impulsu periodu T_{clk} . Šī priekšrocība ir derīga gadījumos kad :

Aiztures līnija ir stabila un precīza. Tā kā mūsu gadījumā aiztures līnijā izmanto RC ķēdes, pietiekamo stabilitāti un precizitāti sasniegt neizdevās.
Ar Greja koda skaitītāju nevar sasniegt augstāku izšķirtspēju. Aiztures līnijas struktūra salīdzinājumā ar Greja koda skaitītāju ir vienkāršāka, bet izveidot stabilo aiztures elementu ir grūti un daudz grūtāk (vai arī neiespējami) mainīt aiztures lielumu laiks-cipars pārveidotāja darbā. Tādējādi nevar izmainīt pārveidotāja mērīšanas diapazonu, gadījumā, kad, piemēram, ir jānomēra lielāks laika intervāls ar ne tik augstu izšķirtspēju. Bet Greja koda skaitītāja mērīšanas diapazonu var izmainīt vienkārši mainot takts impulsu frekvenci: palielinot frekvenci mērīšanas diapazons palielinās, bet precizitāte samazinās un attiecīgi samazinot frekvenci mērīšanas diapazons samazinās, bet precizitāte palielinās.

1.4.2. Skalu stāvokļu nolase atmiņā

Laiks-cipars pārveidotāja testēšanas laikā par impulsu avotu kalpoja signālu ģenerators (6) (1.zīm.) Takts signāls pienāk uz impulsu formētāju ieeju (7) un no signāla tiek veidots 30-40 ns garš impulss. Šis impulss pienāk uz FIFO atmiņas (8) WCLK ieeju, un atkarībā no slēdža (5) stāvokļa tiek nolasīts atmiņā aiztures līnijas vai Greja koda skaitītāja stāvoklis. Ieraksta atļauja/aizliegums (WEN) tiek vadīts no datora caur LPT interfeisu (9). Tabula 1. attēlo darba gaitā izvēlēto FIFO atmiņas konfigurāciju:

Tabula 1. FIFO atmiņas konfigurācija

Izvads	Nosaukums	Stāvoklis
79	$\overline{PRS}$	0
77	$\overline{LD}$	1
76	$\overline{FWFT}$	0
73	OW	1
69	$\overline{BE}$	0
68	IP	0
65	PFM	0
63	RM	0
60	$\overline{RT}$	0
59	$\overline{OE}$	0
6	IW	1
2	$\overline{SEN}$	0

1.4.3. Datu nolase no atmiņas uz datoru.

Nolases procesu kontrolē dators, izmantojot LPT interfeisu. Dators veido takts impulsus ar periodu T = 1ms, kas pienāk uz FIFO atmiņas nolases takts ieeju RCLK. Datu nolase no atmiņas tiek atļauta/aizliegta no datora caur LPT interfeisu.

1.5. Greja koda skaitītāja un aiztures līnija atsevišķa testēšana

1.5.1. Greja koda skaitītājs

Maksimālā takts frekvence, ar kuru izveidotais skaitītājs var strādāt ir 20 MHz (periods 50ns), Testēšanai tika izmantota takts impulsu frekvence 10 MHz (T = 100ns – skaitītāja izšķirtspēja), jo impulsa garums ir apmēram 40-50 ns. Testēšanas gaita bija sekojoša. Tika mērīts jau precīzi zināmais intervāls starp diviem impulsiem. Signālu ģeneratora pirmā izeja bija konfigurēta kā takts impulsu avots, un otra izeja – kā signāla impulsu avots. Takts impulsi tika padoti vienlaikus ar signāla impulsiem. Rezultāta pirmā nolase vienmēr ir bijusi vienāda ar nulli. Tātad laika intervāls būs vienāds ar

$$(n2 - n1) \cdot \tau \qquad (3)$$

kur n2 un n1 ir otrā un pirmā impulsa pienākšanas momentu vērtības, un τ - pārveidotāja izšķirtspēja.

Tabula 1. ir attēlots 5 intervālu mērīšanas rezultāts. Ir doti tikai otrā signālu impulsa vērtības (jo pirmā impulsa vērtība visos gadījumos ir 0). Katrs intervāls tika mērīts 3 reizes.

Tabula 2. Greja koda skaitītāja nolasītas vērtības.

N.p.k	Ģenerēts laika intervāls	Nolasītas vērtības no Greja koda skaitītāja		
		1. reize	2. reize	3. reize
1	226ns	00000011 (3)	00000011 (3)	00000011 (3)
2	956ns	00001101 (13)	00001101 (13)	00001101 (13)
3	9.5us	01110000 (112)	01110000 (112)	01110000 (112)
4	13.3us	11000111 (199)	11000111 (199)	11000111 (199)
5	22.7us	10010010 (146)	10010010 (146)	10010010 (146)

Tabulā 2 tiek aprēķināti laika intervāli un salīdzināti ar etalona laika intervālu.

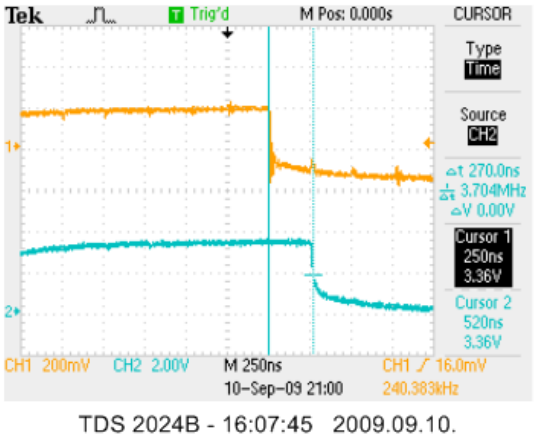
Tabula 3. Aprēķināti intervāli.

N.p.k	Ģenerēts laika intervāls	Decimālas vērtības			Aprēķināts intervāls	**Kļūda**
		1. rez.	2. rez.	3. rez.		
1	226ns	2	2	2	200ns	**226-200=26ns**
2	956ns	9	9	9	900ns	**56ns**
3	9.57us	95	95	95	9,500 us	**70ns**
4	13.32us	133	133	133	13,300 us	**20ns**
5	22.71us	227	227	227	22,700 us	**10ns**

No tabulas 3 ir redzams, ka kļūda nepārsniedz vērtību 100ns – Greja koda skaitītājs strādā pietiekami stabili.

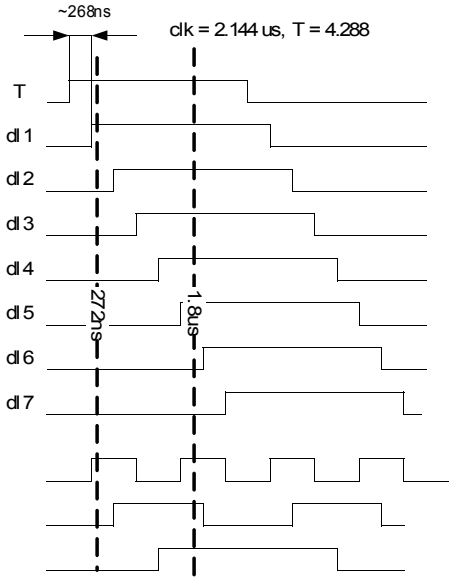
1.5.2. Aiztures līnija

Aiztures līnijas viena elementa aizture ir ~270ns (4.zīm.), tātad aiztures līnijas izšķirtspēja ir vienāda ar ~270ns. Eksperimentāli tika noteikts, ka viena elementa aizture ir 268ns. Tā kā aiztures līnija ir izveidota no RC elementiem, viena elementa aizture ir diezgan nestabila. Šī nestabilitāte neietekmē dažas mikrosekundes garus laika intervālu mērījumus. Tomēr, kā būs redzams atskaites 3. daļā, mērot 100us un garākus laika intervālus, nestabilitātes ietekme pieaug.



4. att. Aiztures līnijas viena elementa aiztures laika mērīšana.

Uz 5. zīm. ir shematiski attēlota laika intervāla mērīšanas būtība ar aiztures līniju. Lai aprēķinātu takts impulsa periodu, kuru veiksmīgi varētu sadalīt 7 pozīcijās, ir jāsareizina viena aiztures elementa vērtību ar 8. Tādējādi nepieciešamais takts impulsa periods ir 2.144us.



5. zīm. Laika intervāla mērīšana ar aiztures līniju.

Aiztures līnija testēšanas gaita bija tādā pati kā Greja koda skaitītāja testēšanas laikā. Izmainījās tikai takts impulsa periods, kas šajā gadījumā ir vienāds ar 2.144us. Tabula 4 satur nolasītas no aiztures līnijas vērtības katram laika intervālam, savukārt tabula 5 satur aprēķinātus laika intervālus un mērīšanas kļūdu.

Tabula 4. Nolasītas vērtības no aiztures līnijas pie dažādiem laika intervāliem.

N.p.k,	Ģenerēts laika intervāls	Nolasīti stāvokli no aiztures līnijas elementiem		
		1. eksp.	2. eksp.	3.eksp.
1	268ns-271.9ns	0000000	0000000	0000000
2	272ns	0000001	0000001	0000001
3	758ns	0000011	0000011	0000011
4	1.8us	0011111	0011111	0011111
5	1.86us	1111111	1111111	1111111

Tabula 5. Aprēķināti laika intervāli.

N.p.k,	Ģen. laika int.	Greja kods/Decimālais kods			Apr. laika intervāls	Kļūda
		1. reize	2. reize	3.reize		
1	268ns-271.9ns	0/0	0/0	0/0	0	268-271.9 ns
2	272ns	1/1	1/1	1/1	268	4ns
3	758ns	3/2	3/2	3/2	536ns	222ns
4	1.45us	7/5	7/5	7/5	1.340us	110ns
5	1.92us	4/7	4/7	4/7	1.876us	44ns

Kā redzams kļūda nepārsniedz sistēmas izšķirtspēju – 268ns.

1.6. Laiks-cipars pārveidotāja maketa testēšana.

Laiks-cipars pārveidotājs tika testēts sekojoši. Tāpat kā Greja koda skaitītāja un aiztures līnijas testēšanas gadījumā par takts impulsa un par laika intervālu avotu kalpoja signālu ģenerators. Tika mērīts laika intervāls starp diviem impulsiem. Tā kā maketam ir tikai viena atmiņa, testēšanas gaita bija sekojoša:
1) Ar slēdža palīdzību vispirms atmiņas datu ieejām tika pieslēgts Greja koda skaitītājs.
2) Tad no signālu ģeneratora tika padoti takts impulsi un reģistrējamie impulsi.
3) Impulsu reģistrācijas rezultāts tika nolasīts no atmiņas.
4) Ar slēdža palīdzību pie atmiņas datu ieejām tika pieslēgta aiztures līnija
5) Tika atkārtoti 2, 3 posmi.
6) Katram mērītam laika intervālam bija atkārtoti posmi no 1 līdz 5.
Tabulā 6 parādīti dažādu laika intervālu mērīšanas rezultāti. Tā kā pirmā impulsa vērtība ir vienāda ar 0, tabula satur tikai otrā impulsa vērtības pie dažādiem laika intervāliem.

Tabula 6. Laika-cipars pārveidotāja testēšanas rezultāts.
Takts impulsa frekvence = 2.144us.

N.p.k	Laika intervāls	G.k. sk. vērtība	A.līnijas vērtība	Rezultāts Greja kodā	Decimālā vertība	Apreķ. laika int.	Kļūda
1	25.89	00001010	001	00001010001	97	25,996	106ns
2	35.12us	00011000	010	00011000010	131	35,108	12ns
3	87.23	00111100	100	00111100100	327	87,636	406ns
4	94.6us	00111010	011	00111010011	354	94,872	872ns
5	100	00111001	101	00111001101	374	100,232	232ns
6	123.56us	00100101	000	00100101000	463	124,084	524ns
7	167.7us	01101001	110	01101001110	628	168,304	396ns
8	203.56us	01110000	110	01110000110	763	204,484	924ns
9	287.80us	11000101	010	11000101010	1075	288,1	300ns
10	323,9us	11011100	110	11011100110	1211	324,548	648ns
11	402.15us	11100110	000	11100110000	1503	402,804	654ns
12	470,6us	10110110	001	10110110001	1758	471,144	544ns
13	545.1us	10000001	010	10000001010	2035	545,38	280ns

Laiks cipars pārveidotāja stabilās darbības gadījuma kļūda nedrīkst pārsniegt aiztures līnijas viena elementa aiztures vērtību, kas mūsu gadījuma ir vienāda ar 268ns. Bet tā kā RC ķēde dod nestabilu aizturi, kļūda gandrīz visos gadījumos pārsniedz 268ns robežu. Tomēr aiztures līnija dotā priekšrocība – *sistēmas izšķirtspēja pie nemainīgas takts impulsa frekvences palielinās*, kaut arī ne tik, cik bija teorētiski aprēķināts. Apskatīsim tabulu 7, kurā tiek apskatītas tikai Greja koda skaitītāja dažas vērtības. Šajā gadījumā, lai no Greja koda skaitītāja vērtībām iegūtu laika intervālu, ir jāsareizina tos ar $T_{clk} = 2.144us$.

Tabula 7. Greja koda skaitītāja vērtību salīdzinājums ar sistēmas kopēju rezultātu

N.p.k	Laika intervāls	G.k.sk. vērtība	Decimālā vērtība	Aprēķ. laika int.	Kļūda g.k.sk. gadījumā	Laiks-cipars pārv. kļūda
1	25.89	00001010	12	25,728	162ns	106 ns
4	87.23	00111100	40	85,76	1,47us	406ns
5	100	00111001	46	98,624	1,376us	232ns

9	287.80us	11000101	134	287,296	531ns	300ns
10	323,9us	11011100	151	323,744	156ns	648ns
12	470,6us	10110110	219	469,536	1,064us	544ns
13	545.1us	10000001	254	544,576	524ns	280ns

No tabulas 7 ir redzams, ka aiztures līnijas izmantošana palielina laiks-cipars pārveidotāja precizitāti, bet tomēr reālā mērīšanas kļūda pārsniedz teorētiski plānotu robežu 268ns aiztures līnijas elementu nestabilitātes dēļ. Bet, kā redzams, Greja koda skaitītājs strādā stabili, un kļūda nepārsniedz 2.144us robežu.

2. Programmvadāmā radio (SDR) darbības izpēte, ieskaitot nevienmērīgas diskretizācijas un atbilstošas signālu apstrādes izmantošanu.

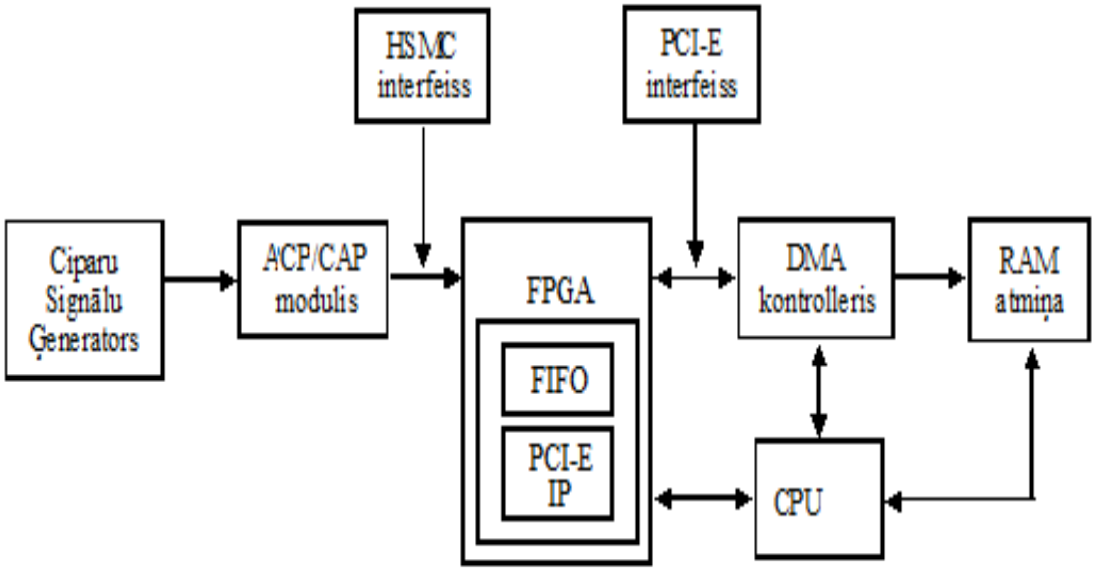
SDR – software defined radio jeb programmvadāms radio. Programmradio pamatā ir īpaša aparatūra, kas darbojas kā uztvērējs ļoti plašā frekvenču diapazonā.Platjoslu RF signālu ciparošana ir sarežģīts process, ko var balstīt uz :

1) RF signāla spektra lejuppārnesi:
   a) nokonvertēts signāls tiek saciparots ar zemu diskretizācijas frekvenci,
   b) frekvenču uzklāšanās (d)efekta tiek izmantots ciparveida lejuppārnesei.
2) ciparošanu ar ļoti augstu diskretizācijas frekvenci (vairāki GHz), taču ātrdarbīgiem ACP izšķirtspēja parasti ir ierobežota līdz 6-8 bitiem. Ļoti lielā datu plūsma rada problēmas to pārsūtīt un apstrādāt.
3) Nevienmērīgu diskretizāciju ar samazinātu datu plusmas apjomu un atbilstošu signālapstrādi.

Sistēmas funkcionalitāti var izmainīt pāris minūšu laikā, lejupielādējot attiecīgo programmatūru, neizmainot izmantotās iekārtas struktūru.

Kombinējot programmradio un virtuālo instrumentu tehnoloģijas, var ātri prototipēt datu apstrādes un mērīšanas ierīces, izmantojot ātru interfeisu, datortehnikas jaudu un programmnodrošinājuma sniegtās iespējas.

Izstrādātā prototipa blokshēma



2.1. Datu saciparošanas veidi.

2.1.1. Regulārā diskretizācija.

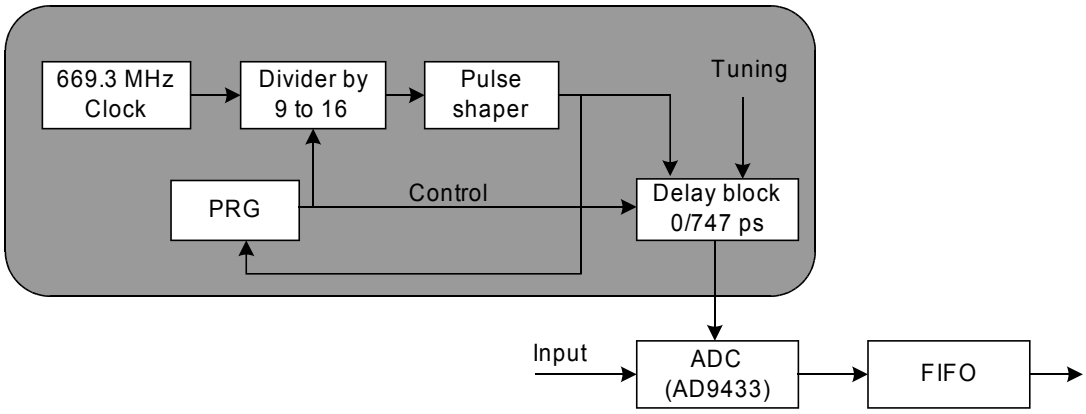
Regulārās diskretizācija tika izmantota sākotnējās sistēmas testēšanā. Datu savākšanas platē atrodas analogais-ciparu pārveidotājs (AD9254 ar 14 bitu izšķirtspēju un 150 MSPS maksimālo diskretizācijas frekvenci). HSMC (*High Speed* Mezzanine Connector) spraudnis tiek izmantots, lai pieslēgtu šo plati pie citām iekārtām. Šajā projektā datu savākšanas plate tiek pieslēgta pie Stratix II GX plates ar PCI Express kopni.

Fig.1. Datu savākšanas plate.

Regulārā diskretizācijas plate tika aizvietota ar neregulārās diskretizācijas plati, tās zemās frekvenču joslas caurlaides dēļ.Pēc klasiskās Naikvista teorijas maksimālā signāla frekvence būtu 75Mhz, kas ir vismaz desmit reizes zemāka par nepieciešamo. Tamdēļ tika izvēlēta neregulārā diskretizācija.

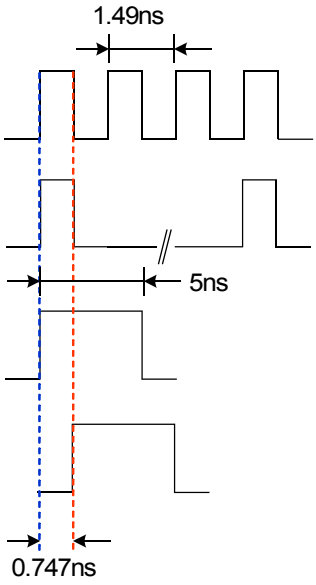
2.1.2. Neregulārā diskretizācija.

Ar nevienmērīgas diskretizācijas palīdzību ir iespēja apstrādāt plašāku frekvenču diapazonu izmantojot zemu vidējo diskretizācijas frekvenci. Tomēr analogs-ciparu pārveidotājam ir jābūt ar plašu ieejas joslu. Pēc iespējamu risinājumu analīzes tika izvēlēts *ANALOG DEVICE* firmas 12 bitu analogs-ciparu pārveidotājs *AD9433*. Šī mikroshēma ļauj strādāt frekvenču diapazonā līdz pat 750 Mhz. Ir jāpievērš uzmanība, ka neskatoties uz tik plašu ieejas joslu šim ACP diskretizācijas frekvence ir ierobežota līdz 125 MHz. Tas nozīme, ka lietojot klasisku vienmērīgu diskretizāciju būtu iespējams apstrādāt signālus, ievērojot Naikvista kritēriju, tikai līdz 62 MHz. Nenoliedzami būtu iespēja lejuppārnest spektru analoga veidā, bet tad apstrāde notiktu noteiktajā šaurajā diapazonā nevis pilnā 750MHz joslā. Tieši tāpēc, lai izvairītos no šī ierobežojuma tika izmantota nevienmērīga diskretizācija. Attēlā 4.2 ir paradīts nevienmērīgas diskretizācijas realizācijas blokshēma, kurā tiek izmantots 669.3 MHz takts impulsu ģenerators no firmas *FORDAHL*.



4.2. att. Nevienmērīgas diskretizācijas realizācijas blokshēma

Augstfrekvences taktsimpulss (669 MHz) nonāk uz dalītāju ar mainīgo dalīšanas koeficientu. Šīs dalītājs tiek vadīts ar preido-gadījumskaitļu ģeneratoru, kas nosaka, ka dalīšanas koeficienta maiņā notiks preido-gadījuma veidā. Šis impulsa garums nav pietiekoši ilgs, lai ACP funkcionētu pareizi. Šim konkrētam *ADC* taktsimpulsa garumam ir jābūt ne mazāk par 5ns. Šim nolūkam iegūtais taktsimpulss tiek pagarināts līdz garumam 5ns. Izmantojot iegūto taktsimpulsu nebūtu iespējas realizēt platjoslu signālu apstrādi. Tāpēc, šis signāls tiek padots uz aiztures līniju, kas nobīda viņu par 0.747 ns. Rezultātā tiek iegūta ļoti maza starpība starp oriģinālo signālu un nobīdīto. Abi šie signāli tiek padoti uz multipleksoru, kas vadās ar preido-gadījumskaitļu ģeneratoru. Pārslēdzot multipleksoru notiek izvēlē, kurš no signāliem tiks padots uz *ACP*. Rezultātā iegūstam nolases kas atrodas uz nevienmērīga režģa ar ekvivalento diskretizācijas frekvenci 1338.653 MHz. Tas dod iespēju apstrādāt signālus līdz pat 669.326 MHz. Ir jāpatur prātā, ka reāli vidēja frekvence ir ap 50 MHz kas padara shemtehnisko risinājumu ne tik sarežģītu ka tas būtu izmantojot 1.3 GHz frekvenci.



4.3. att. Nevienmērīgas diskretizācijas realizācijas laika diagramma.

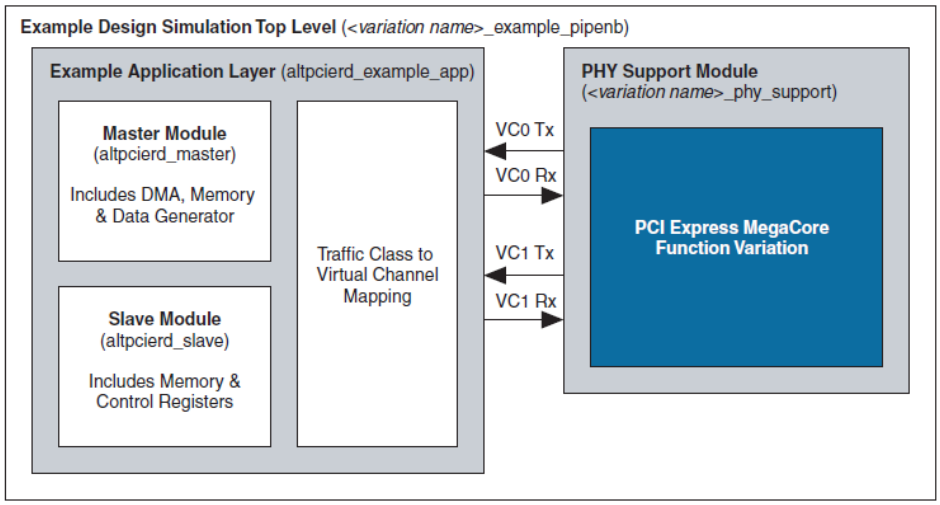
Pielikumā ir paradīts VHDL programmas kods, ar kuru palīdzību ir realizēts pseido-random skaitļu ģenerators (*rnd reģistrs*). DL0-DL3 signāli, kas mainās laikā pseido-random veidā, tiek padoti uz skaitītāju.

2.2. PCI-e kopnes realizācija izmantojot Stratix II GX Edition izstrādes plati

Lai nodrošinātu datu sūtīšanu reālā laikā uz datoru, tika izvēlēta PCI-e kopne, izmantojot vienu diferenciālo līniju var pārraidīt 250 MB/s, palielinot diferenciālo līniju skaitu pārraides ātrums proporcionāli pieaug. Šim mērķim tika iegādāta Alteras PCI-e Stratix II GX Edition izstrādes plate. Plates tehniskā informācija ir atrodama www.altera.com mājaslapā.

Fig.2. PCI Express interfeisa plate.

Izmantojot Quartus II vidē MegaWizard Plug-in Manager var noģenerēt PCI Express kodola pamatu. Tas ietver sevī nepieciešamās sastāvdaļas, kas būtu nepieciešamas, lai izveidotu savu sistēmu. Detalizētāku informāciju par šo var atrast PCI Express Compiler User Guide. Modificējot noģenerēto VHDL kodu PCI Express kodolam var ielikt papildus funkcionalitāti.



Galvenais fails, kurš tiks modificēts projekta gaitā ir *pex_example_top.vhd*. Tajā atrodas divi svarīgie moduļi: MASTER, kas atbild par DMA datu apmaiņu un SLAVE, par parasto datu pārsūtīšanu.

SLAVE datu rakstīšanas/sūtīšanas režīmā tiek iesaistīts CPU (centrālais processors), tādā veidā nevar tik panākts maksimālais datu sūtīšanas ātrums, šo problēmu atrisina izmantojot MASTER moduli, jeb DMA (Direct Memory Acess), CPU tiek iesaistīts tikai DMA inicializācijas

sākumā.

Lai izvairītos no datu pazušanas ACP diskretizētie dati nāk uz FIFO atmiņu, kur tiek formēti no 16 bitiem 64 bitos, to parāda sekojošs programmas kods:

```
process(hrstn, acp_clk_in)
  begin
    if hrstn = '0' then
      my_req_sm      <= S0;
      data64_out     <= (others => '0');
      data_ready <= '0';
    elsif rising_edge(acp_clk_in) then
     case my_req_sm is
       when S0 =>
                            data_ready <= '0';
                            data64_out(11 downto 0)  <= acp_data;
                            data64_out(63 downto 12) <= (others => '0');
                            my_req_sm <= S1;
       when S1 =>
                           data_ready <= '0';
                            data64_out(27 downto 16) <= acp_data;
                           my_req_sm <= S2;
       when S2 =>
                            data_ready <= '0';
                            data64_out(43 downto 32)<= acp_data;
                           my_req_sm <= S3;
       when S3 =>
                            data_ready <= '1';
                            data64_out(59 downto 48) <= acp_data;
                            my_req_sm <= S0;

    end case;
end if;
end process;
```

FIFO atmiņa realizē arī citas funkcijas, tas ir, ģenerē pārtraukumu (my_var ir pieslēgts pie app_int_sts), tad kad atmiņa ir aizpildījusies līdz pusei (par to atbild m_wrusedw signāls)

```
process(rstn, clk_in)
  begin
    if rstn = '0' then
      my_var <='0';
      my_var_tmp <= '0';
    elsif rising_edge(clk_in) then
      if m_wrusedw = "100000000000000" and tmp_var = '0' and my_var_tmp <= '0' then
      my_var <='1';
     elsif m_wrusedw = "100000000000000" and tmp_var = '1' and my_var_tmp <= '0' then
      my_var <='0';
      my_var_tmp <= '1';
     elsif m_wrusedw = "100000000000001" and my_var_tmp = '1' then
      my_var_tmp <= '0';
```

```
    else
    my_var <='0';
end if;
end if;
end process;
```

PCI-e kodols, kas tika realizēts Stratix II GX mikroshēmā, var ģenerēt MSI (Message signaled interrupts) un Legacy PCI pārtraukumus. FPGA mikroshēmas programmā, kurā ir realizēts šis kodols, ir atbilstoši izvadi, jeb signāli, kas atbild par pārtraukumu realizāciju. Šajā projekta gaitā tika izmantota Windows XP vide. Atšķirībā no Linux un Windows Vista, šī vide uztur tikai Legacy PCI pārtraukums. Līdz ar to tālāk ies runa tieši par šo pārtraukumu izmantošanu, kuri ļauj nodrošināt nepieciešamu funkcionalitāti.

Pēc diskretizācijas dati nonāk FIFO atmiņā, kuras apjoms ir ierobežots. Lai novērstu datu zudumus, pirms šī atmiņa būtu aizpildīta ir nepieciešams pārsūtīt datus caur interfeisu uz datoru. Tā atbrīvojot atmiņu var turpināt ierakstīt jaunus datus nepazaudējot nevienu nolasi. Lielo un nepārtraukto datu plūsmu ilglaicīga pārsūtīšana no FIFO atmiņas uz datoratmiņu nav viegls uzdevums. Tas prasa pareizi rūpīgi izveidot algoritmu, kas nodrošinās reāla laika režīmu. Tieši šim mērķim darbā tika izmantoti pārtraukumi.

FIFO atmiņā ir karodziņš, kas tiek noraustīts, kad atmiņā ir aizpildīta līdz pusei (half-full flag). Šīs signāls ir piesaistīts pie pārtraukuma signāla (app_int_sts). Tas nozīme, ka noraustoties, šis signāls izsauks pārtraukuma pieprasījumu signālu. Datorprogramma nepārtraukti gaida šo signālu un saņemot to, tā var sākt datu pārsūtīšanu caur kopni. Laikā, kad notiek datu pārsūtīšana, ienākošie dati turpina ierakstīties FIFO atmiņa un līdz ar to dati netiek pazaudēti.

Pastāv arī iespēja izmantot tā saucamo pieprasījumu stāvokļu reģistru (interrupt status register - INTCSR) un tādā veidā realizēt vairākus papildus iespējas. Piemēram, izveidot pieprasījumus ar augstāku vai zemāku prioritāti. Šajā reģistrā, kurš tiek realizēts FPGA programmā, var ierakstīt pieprasījuma numuru. Datora programma saņemot pieprasījuma signālu var nolasīt šī reģistrā ierakstīto vērtību un izpildīt noteiktu darbību. Ierakstot pieprasījumu stāvokļa reģistrā citu vērtību var izsaukt citu darbību tā padarot programmatūru daudz funkcionālāku.

PCI-E kodolā INTCSR reģistrs ir realizēts samērā vienkāršā veidā. Katru reizi, kad atnāk pārtraukums (kad FIFO atmiņa aizpildās līdz pusei un noraustas app_int_sts signāls) tiek ierakstīta vērtība BAR 0 reģistrā adresē 0x04. Datora aplikācija saņemot pieprasījuma signālu nolasa šī reģistra vērtību un salīdzina ar masku. Ja tā sakrīt ar masku, tad pieprasījums tika apstiprināts un INCSR reģistrs tiek notīrīts, t.i. tajā ierakstīta nulle. INCSR reģistra vērtības salīdzināšana ar masku ir nepieciešams lai izveidotu vairāku prioritāšu pieprasījumus. Tas nozīme, ka atnākot dažādiem pārtraukumiem tiks izpildītas dažādas darbības.

2.3. Datu sūtīšana izmantojot C aplikāciju

Draivera izveidei tikai pielietotas sekojošas programmas:

- Microsoft Visual C++ 2008 Express edition (http://www.microsoft.com/express/vc/) lietotāja aplikāciju izstrādei
- Jungo Windriver (http://www.jungo.com/st/windriver_usb_pci_driver_development_software.html) PCI-e draivera izveidošana. Šis rīks arī piedāvā lietotājam gatavas funkcijas, kuras var lietot, lai sadarbotos ar iekārtām, kas pieslēgtas pie PCI vai PCI-e kopnēm.

Programmatūra tika izveidota Linux un Windows vidēs. Tomēr, Linux vidē netika realizēti pārtraukumu apstrāde un DMA datu pārraides režīms. Tas ir saistīts ar to, ka funkcijas, kas atbild par pārtraukumu apstrādi un DMA datu pārraides režīmu ir specifiskas katrai iekārtai. Lai izveidotu savu draiveru, kas uzturētu iepriekšminētas funkcijas tas prasa OS Linux vides padziļinātas zināšanas. Līdz ar to tika pielietots Windriver draiveru izstrādes rīks, kas uztur Stratix II GX čipā realizēto PCI-e interfeisu. Tas nozīme, ka lietotajam ir pieejamas gatavas funkcijas (datu pārsūtīšana/saņemšana, pārtraukumu apstrāde, DMA datu pārraide, utt.).

2.3.1. C aplikācija, kas realizē SLAVE sūtīšanu caur PCI-e kopni

Iepriekš bija minēts, ka ir divi veidi, kā var realizēt datu pārsūtīšanu no ārējas plates uz datoru caur PCI-e kopni. Pirmais veids (IO Memory access), iesaista datora centrālo procesoru (CPU) un līdz ar to nevar iegūt maksimāli iespējamo datu pārsūtīšanas ātrumu. Izmantojot WINDRIVER funkcijas šis veids tiek realizēts ļoti vienkārši. C kods izskatās sekojoši:

```
hDev = DeviceFindAndOpen (VENDOR_ID, DEVICE_ID);
ALTERA_WriteDword (hDev, BAR0, 0x00, Data);  // Datu ierakstīšana
ALTERA_ReadDword (hDev, BAR0, 0x00, Data);  // Datu nolasīšana
DeviceClose(hDev, NULL);
```

, kur *Data* ir masīvs ar datiem.

Lai realizētu lielāku datu pārsūtīšanas ātrumu tiek pielietos DMA (Direct Memory Access) režīms. Šis režīms izveido kanālu starp datora operatīvo atmiņu un izveidotās iekārtas atmiņu. Tādā veidā arēja iekārta var veikt datu nolasīšanu un ierakstīšanu operatīvā atmiņā neatkarīgi no procesora(CPU). Lietojot DMA režīmu, datora procesors inicializē datu pārsūtīšanu un pēc tam var veikt citas darbības, kamēr visi dati tiek pārkopēti un tika saņemts pārtraukums no DMA kontroliera, par to, ka dati tika pārsūtīti. Tāds režīms ir īpaši nepieciešams reālā laikā sistēmu izveidošanā.

2.3.2. C aplikācija, kas realizē MASTER (DMA) sūtīšanu caur PCI-e kopni

Lai izveidotu DMA pārraides kanālu ir nepieciešamas DMA kontrolieris, kas tika realizēts Stratix II GX mikroshēma nodot vairākus parametrus. Talāk tiek parādīti nepieciešamie soļi:

4. Ierakstīt PCI-e iekārtas addresses reģistros 0x00 un 0x04;
5. Ierakstīt operatīvas atmiņas sākuma addressi reģistrā 0x14, kur tiks saglabāti pārsutītie dati;
6. Reģistrā 0x08 ierakstīt DMA operācijas garumu, t.i. cik liels apjoms tiks parsutīts vai nolasīts;
7. Reģistrā 0x0C tiek ierakstīti papildus parametri, detalizētāku informāciju var atrast PCI-e Megacore dokumentācijā. Ierakastot datus šajā reģistrā sākas DMA kanāla darbība;
8. Lasot reģistra 0x0C progresa bita vērtību var notekt, kad DMA darbība būs pabeigta.

Šeit ir paradīts C programmas kods:

```
hDev = DeviceFindAndOpen (VENDOR_ID, DEVICE_ID);
ALTERA_DMAOpen(hDev, &pDma->pBuf, dwOptions, 2048, &pDma->hDma);
ALTERA_WriteReg32(hDev, 0x00, LowerAddr);
```

```
ALTERA_WriteReg32(hDev, 0x04, UpperAddr);
ALTERA_WriteReg32(hDev, 0x14, hDma->pDma->pBuf);
ALTERA_WriteReg32(hDev, 0x08, 10000*2048);
/* Start DMA Transfer */
ALTERA_WriteReg32(hDma->hDev, 0x0c, fIsRead ? 0x00 : 0x40);
DMAcomplete = ALTERA_ReadReg32(hDev, 0x0c);
printf ("data: %d\n", data);
DeviceClose(hDev, NULL);
```

2.3.3. C programma pārtraukumu apstrādei.

C programma aplikācija pārtraukumu apstrāde notiek sekojoša veidā, kad iekārta ģenerē pārtraukumu (noraustas app_int_sts signās PCIE Megacore programmā) ir nepieciešams veikt sekojošas darbības:

- nolasīt INTCSR reģistra vērtību;
- salīdzināt to ar masku;
- ierakstīt nulli INTCSR reģistrā (iztirīt reģistru).

Zemāk ir parādīts programmas kods, kas veic šīs darbības:

```
DWORD VIRTEX5_IntEnable(WDC_DEVICE_HANDLE hDev,
    VIRTEX5_INT_HANDLER funcIntHandler)
{
    #define NUM_TRANS_CMDS 3

  /* #1: Read from the Register0 register */
    pAddrDesc = &pDev->pAddrDesc[AD_PCI_BAR0];
    trans[0].dwPort = pAddrDesc->kptAddr + 0x04;
    trans[0].cmdTrans = WDC_ADDR_IS_MEM(pAddrDesc) ? RM_DWORD : RP_DWORD;

    /* #2: Mask interrupt status from the Register0 register */
    trans[1].cmdTrans = CMD_MASK;
    trans[1].Data.Dword = 1;

  /* #3: Write to the Register0 register */
    pAddrDesc = &pDev->pAddrDesc[AD_PCI_BAR0];
    trans[2].dwPort = pAddrDesc->kptAddr + 0x04;
    trans[2].cmdTrans = WDC_ADDR_IS_MEM(pAddrDesc) ? WM_DWORD : WP_DWORD;
    trans[2].Data.Dword = 0;

    /* Store the diag interrupt handler routine, which will be executed by
       VIRTEX5_IntHandler() when an interrupt is received */
    pDevCtx->funcDiagIntHandler = funcIntHandler;

    /* Enable the interrupts */
    dwStatus = WDC_IntEnable(hDev, &trans, NUM_TRANS_CMDS, INTERRUPT_CMD_COPY,
VIRTEX5_IntHandler,
            (PVOID)pDev, WDC_IS_KP(hDev));
```

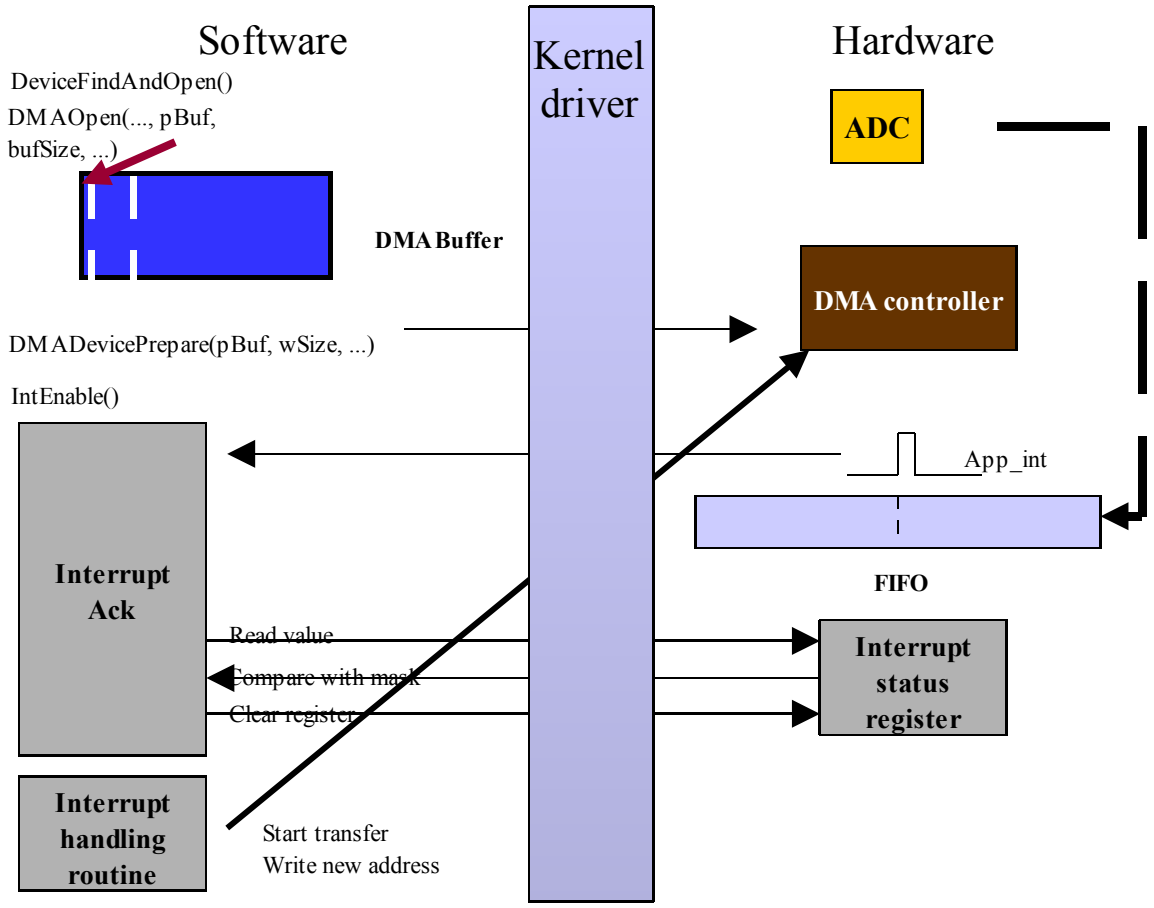
2.3.4. Linux C aplikācija

Pielikumā ir paradīta C programma, kas veic sekojošas darbības:

- atrod iekārtu, kas pieslēgta pie PCI-Express interfeisa;
- nolasa tas konfigurāciju;
- veic datu nolasīšanu;

Lai darbotos ar PCI-e interfeisu ir nepieciešams uzstādīt PCIUTILS un PCIUTILS-DEVEL rīkus. Sakarā ar to, ka pietrūka zināšanas un pieredze darbā ar pārtraukumiem un DMA datu pārsūtīšanas zem Linux vides, tika izvelēta programmatūra WINDRIVER(JUNGO), kas ļauj samērā vienkārši veikt tās darbības.

2.3.5. Real - Time mode



Pielikums

```
#include <stdio.h>
#include <pci/pci.h>
#include "/usr/include/linux/pci.h"
#define PCITAG struct pci_filter *
#include <sys/mman.h>  /* PROT_READ etc.*/

int main (int argc, char *argv[])
{
struct pci_access *pacc;
struct pci_dev *dev;
struct pci_filter ltag;
struct pci_dev *d;
struct pci_dev *config_space = NULL; /* pointer to the PCI device that will be configered */
unsigned int c;

/* Check IO permissions to be able to open /dev/mem */
if (iopl (3))
{
printf ("Cannot get I/O permissions (being root helps)\n");
return 0;
}

pacc = pci_alloc();
pci_init(pacc);
pci_scan_bus(pacc);

for (dev = pacc->devices; dev; dev = dev->next)
{
 pci_fill_info(dev, PCI_FILL_IDENT | PCI_FILL_BASES);
//c = pci_read_byte(dev, PCI_INTERRUPT_PIN);   /* Read config register directly */
}

dev = pacc->devices;
printf("Dev = %d\n", dev);
unsigned char *ptr;
  while (dev != NULL)
  {
printf("[%d:%d:%d] VendorID: %04x DeviceID: %04x\n", dev->bus, dev->dev, dev->func,    dev-
>vendor_id, dev->device_id);
printf ("Class %X Revision %X ", pci_read_word (dev, 0x0a), pci_read_word (dev, 0x08));


     if ((dev->vendor_id != 0x1172) || (dev->device_id != 0x0007))
                                        dev = dev->next;
     else
{
printf("\nAltera Development kit was found\n");
printf ("Vendor %X Device %X\n ", pci_read_word (dev, 0x00), pci_read_word (dev, 0x02));
```

```
printf ("Command %X Status %X\n ", pci_read_word (dev, 0x04), pci_read_word (dev, 0x06));

int pos, len=4;
void *buf;
read(dev, pos, ptr, len);
printf("Pos=%d, Len=%d\n", pos, len);
printf("Size of buf=%d\n", sizeof(buf));
printf("Size of Cache=%d\n", sizeof(dev->cache));
printf("Size of Cache=%d\n", dev->cache_len);
printf("Cache=%d\n", dev->cache);

u32 pci_mem_addr, pci_io_addr;
char *ptr_to_pci_mem;

pci_mem_addr = pci_read_long(dev, 0x10) & PCI_BASE_ADDRESS_MEM_MASK;
printf("\npci_mem_addr = %x\n", pci_mem_addr);

int fd = open("/dev/mem", 2);
printf( "fd_mem %d\n", fd);
#define PCI_MEM_LEN (1<<10) /* This must be <= than the actual amount of PCI memory */
ptr_to_pci_mem = (char *) mmap(NULL, PCI_MEM_LEN, PROT_READ|PROT_WRITE,
MAP_SHARED, fd, (off_t)pci_mem_addr);

 printf("Memory pointer: %p\n", ptr_to_pci_mem);
 printf("PCI memory @%#x\n", pci_mem_addr);
 printf("PCI memory is :\n %x", *(ptr_to_pci_mem+0) );
 printf(" %x", *(ptr_to_pci_mem+1) );

munmap(ptr_to_pci_mem, PCI_MEM_LEN);

break;
}
  }

#define PCIIOC_BASE             ('P' << 24 | 'C' << 16 | 'I' << 8)
  char path[128];
  char buf[1024];
  int fd;
  FILE *fptr;
  unsigned int bus, device, function, devfn;
  unsigned int sbus=0, sdevfn=0, svend;
  unsigned long bar, sbar =0;
//  char *ptr;
  unsigned int *data;
u16 ven, devv;

sprintf( path, "/proc/bus/pci/03/00.0");
printf( "path is :%s\n", path );
```

```
  fd = open("/proc/bus/pci/03/00.0", 0); // 0 - read only; 2 - read and write
  if( fd == -1 ) perror( "Couldn't open device file" );


pread(fd, &ven, 2, PCI_VENDOR_ID);
pread(fd, &devv, 2, PCI_DEVICE_ID);
printf( "ven is :%x\n", ven);
printf( "dev is :%x\n", devv);

u32 buf2;

lseek(fd, 0x00, SEEK_SET);
read(fd, buf2, 8);
printf( "size of buf2 :%d\n", sizeof(buf2));

u16 bara;

pread(fd, &bara, 2, PCI_BASE_ADDRESS_0); // printf ("BAR0 %X\n ", pci_read_word(dev,
PCI_BASE_ADDRESS_0));
printf( "bar is :%x\n", bara);
pread(fd, &bara, 2, PCI_BASE_ADDRESS_1);
printf( "bar is :%x\n", bara);
pread(fd, &bara, 2, PCI_BASE_ADDRESS_2);
printf( "bar is :%x\n", bara);


printf ("MEMORY LIMIT %X\n ", pci_read_word (dev, PCI_MEMORY_LIMIT));


close(fd);

pci_cleanup(pacc);
return 0;

}
```

3. Augstas jutības signālu pārveidošana – metodes un aparatūra

3.1. Balansa komparatora shēmas un konstrukcijas pilnveidošana.

Pārskata periodā izstrādāti divi balansa komparatora varianti. Viens no variantiem ir jauns shemotehnisks risinājums, kas atšķiras ar to, ka komparatora slieksnis nav atkarīgs no signāla avota izejas pretestības. Otrs variants atšķiras ar to, ka balansa komparators tiek mazāk slogots ar pieslēguma ķēdēm – pie komparatora ieejas pienāk tikai viena ķēde, pa kuru, pielietojot oriģinālu shemotehnisku un konstruktīvu risinājumu, komparatora ieejai tiek pievadīts gan ieejas, gan komkpensācijas signāls, kā arī tiek noņemts komparatora nostrādāšanas signāls. Rezultātā palielinās komparatora ātrdarbība, jo komparatora ieeja netiek noslogota ar pieslēgumu ķēžu parazītiskajām kapacitātēm, kas salīdzinājumā ar komparatora ieejas kapacitāti (mūsu gadījumā tā ir tikai 0,8pF) sastāda visai būtisku noslogojumu. Minēto shemotehnisko un konstruktīvo risinājumu autori uzskata, ka būtu jāapsver iespēja patentēt šos risinājumus, kā arī projekta iepriekšējā etapā izstrādāto komparatora variantu ar atbalsta pretestību, kas palielina komparatora stabilitāti.

3.2. Efektīvu transformētā laika signālapstrādes metožu izstrāde un izpēte.

Adaptīvās metodes funkcionāli paplašina pārveidotāja iespējas, ļaujot reģistrēt signālus plašākā amplitūdu diapazonā. Tas it sevišķi svarīgi ir pārveidotāju pielietojumos superplatjoslas radiolokācijā, kur vienlaikus ar vājiem signāliem ir jāreģistrē ļoti spēcīgus signālus. Pārskata periodā ir izstrādātas vairākas adaptīvās metodes, kas balstās uz atšķirīgiem adaptācijas principiem:

1. Princips: adaptācijai izmanto informāciju, kas tiek iegūta signāla mērīšanas laikā dotajā signāla fāzes punktā.
2. Princips: adaptācijai izmanto informāciju, kas tika iegūta signāla mērīšanas iepriekšējā fāzes punktā.

Adaptīvā statistiskā metode.

Metode izmanto informāciju, kas iegūta signāla mērīšanas laikā dotajā fāzes punktā. Klasiskās statistiskās metodes gadījumā signāls tiek *n* reizes salīdzināts ar slieksni un pēc salīdzināšanas rezultātu iegūšanas signāla pieaugumu dotajā signāla fāzes punktā procesors aprēķina pēc formulas:

$$\Delta u_{2i} = \eta \Phi^{-1}(\hat{p}_i) = \eta \sqrt{2}\ erf^{-1}(\frac{2n_i^+ + \varepsilon(n_i^+)}{n} - 1),$$

kur

n_i^+ -sliekšņa pārsniegšanas gadījumu skaits;

$\eta < 1$ - modifikācijas koeficients;

$\varepsilon(n_i^+)$ - speciāli ievests koeficients, lai gadījumos, kad $n_i^+ = n$ vai $n_i^+ = 0$ funkcija $erf^{-1}(\frac{2n_i^+}{n} - 1)$ nepārvērstos par "+" vai "-" bezgalību.

Signāla pārveidojumu iegūst kā momentāno vērtību mērījumu secību:

$$u_{2i} = u_{2\,i-1} + \Delta u_{2i}$$

Adaptīvās metodes gadījumā funkcija $g(\eta, n_i^+, n) = \eta\sqrt{2}\ erf^{-1}(\frac{2n_i^+ + \varepsilon(n_i^+)}{n} - 1)$ tiek aizstāta ar sekojošu funkciju:

$$f(q) = \sqrt{2}(q + k\ sign(q)|q|^r),$$

kur $q = (\frac{2n_i^+}{n} - 1)$,

$k = k_1\ erf^{-1}(1 - 1/n) - 1$, $r = \frac{\ln c - \ln k}{\ln q_0}$.

Parametri k_1, c un q_0 tiek izvēlēti tā lai panāktu dinamiskā diapazona paplašināšanos, nesamazinot signāla/trokšņa attiecību $h_2 = \frac{u_2}{\sigma_2}$.

Visai labus rezultātus iegūst pie šādiem nosacījumiem:

$$c = 0{,}01;$$

$$q_0 \approx 0.6.$$

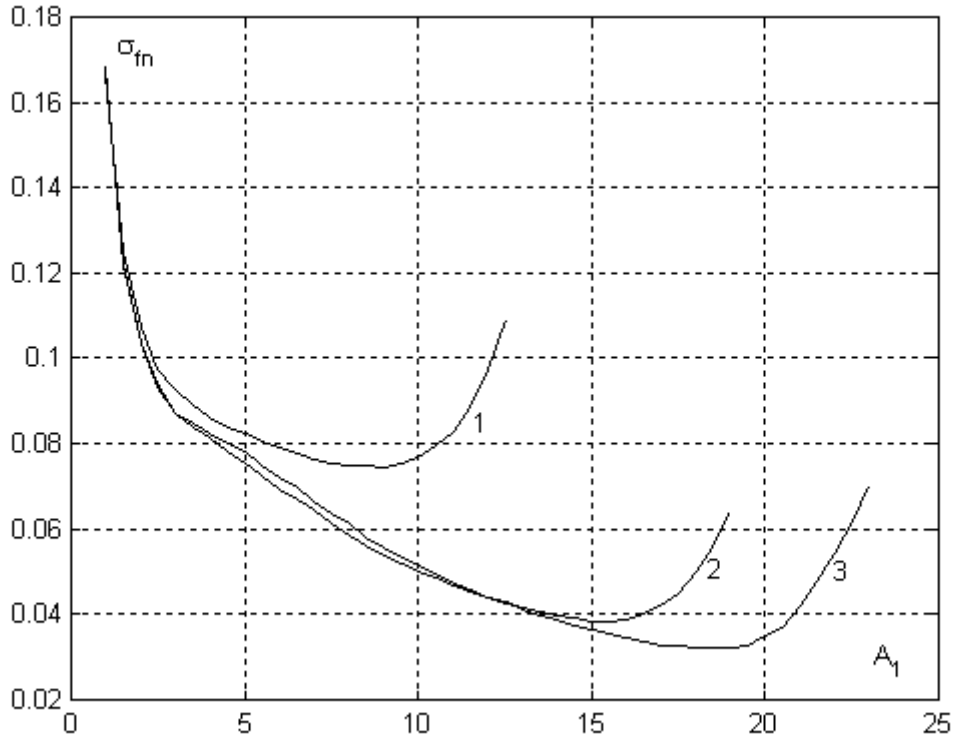
Lai izvērtētu šādi veidotas adaptīvās metodes efektivitāti kā pārveidojamais signāls tika izvēlēts superplatjoslas radiolokācijas signāla modelis — harmoniska monosvārstība:

$$u_s(t_i) = A_1 \sin\frac{2\pi}{n_T} i,$$

Pārveidojuma kvalitāte tika vērtēta kā pārveidojuma rezultāta vidējā kvadrātiskā novirze no patiesās signāla vērtības:

$$\sigma_{fn} = \frac{1}{A_1}\sqrt{\frac{1}{n_T}\sum_{i=0}^{n_T}(u_{2i} - u_{si})^2}$$

Ilustrācijai zīmējumā (1) ir parādīta ar troksni $\sigma_1 = 1$ maskētas monosvārstības pārveidojuma kļūda atkarībā no monosvārstības amplitūdas pie dažādām parametra k_1 vērtībām. Salīdzinājumam zīmējumā attēlota pārveidojuma kļūda, lietojot neadaptīvo statistisko metodi (līkne 1), adaptīvo metodi pie $k_1 = 2$, (līkne 2) un pie $k_1 = 3$ (līkne 3). Strobu skaits šajos eksperimentos bija n=25, fāzes punktu skaits uz monosvārstības periodu $n_T = 50$.



Zīm. 1. Pārveidojuma rezultāta kļūda atkarībā no signāla amplitūdas: statistiskā metode (līkne1); adaptīvā metode pie k_1=2 (līkne 2) un adaptīvā metode pie k_1 =3 (līkne 3).

Kā redzam no pārveidojuma rezultātiem, adaptīvā metode nodrošina ievērojamu dinamiskā diapazona paplašinājumu: līknei 1 minimums ir pie amplitūdas $A_1 \approx 8.5$, līknei 2 — pie amplitūdas $A_1 \approx 16$, bet līknei 3 — pie $A_1 \approx 18.5$. Bez tam adaptīvās metodes gadījumā šis minimums ir zemāks nekā parastajai statistiskajai metodei.

Uz iegūto rezultātu bāzes var formulēt sekojošu tālāko pētījumu loku: kādai ir jābūt optimālajai informācijas apstrādei, lai pie uzdotā maskējošā trokšņa σ_1, uzdotā iztvērumu (strobu) skaita *n*, fāzes punktu skaita n_T un nepieciešamās signāla/trokšņa attiecības $h_2 = \frac{u_2}{\sigma_2}$, iegūtu maksimāli iespējamo dinamisko diapazonu.

Adaptīvā s-metode.

Šī metode izmanto informāciju, kas iegūta signāla mērīšanas laikā iepriekšējā fāzes punktā. Metode izstrādāta uz klasiskās «up-and-down» metodes bāzes. Saskaņā ar klasisko «up-and-down» metodi,

signāls $u_1(t)$ laika momentā t_i tiek salīdzināts ar komparatora slieksni e_{i-1}, kura lielums tiek izmainīts ar soli $\pm s$ saskaņā ar procedūru:

$$e_i = e_{i-1} + s\ sign(u_1(t_i) - e_{i-1}).$$

Pēc n salīdzināšanas operācijām iegūstam signāla momentānās vērtības mērījuma rezultātu kā sliekšņa pēdējo vērtību $u_{2i} = e_{in}$, . Šādi iegūto momentāno vērtību secība veido laikā transformēto signālu u_2 .

Esam izstrādājuši adaptīvu «up-and-down» metodi, kas izmanto signāla iepriekšējā fāzes punktā iegūto informāciju par strobējamā komparatora nostrādāšanas reižu skaitu. Saskaņā ar šo metodi, nākošajā signāla fāzes punktā signāls tiek mērīts ar soli, ko pirms mērīšanas iestāda vienādu ar šādi aprēķināmu lielumu:

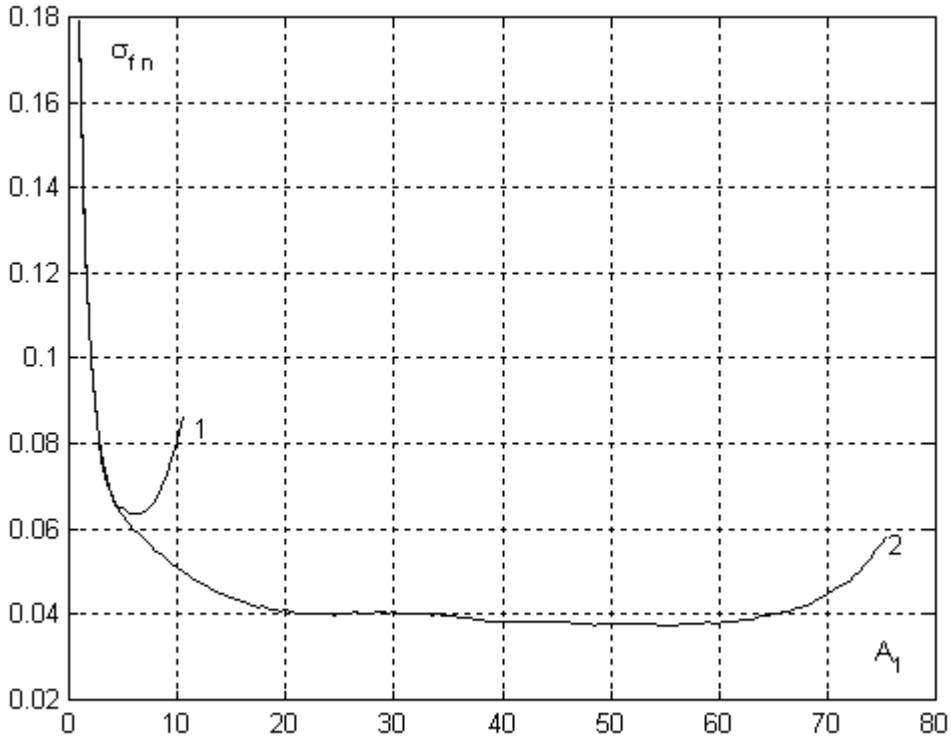
$$s_{i+1} = s_0(1 + k_0 \left| q_{1i} \right|^{\beta}),$$

kur

$q_{1i} = 2n_i^+ / n - 1,$

k_0 un β - adaptācijas koeficienti.

Zīmējumā 2. ir parādīta pārveidojuma rezultāta kļūda σ_f «up-and-down» metodes gadījumā (līkne 1) un ar adaptīvo «up-and-down» metodi iegūtā rezultāta kļūda (līkne 2) pie šādiem nosacījumiem: $\sigma_1 = 1,\ n = 25, n_T = 50, s = 0.05.,\ s_0 = 0.05, k_0 = 6, \beta = 9$. No iegūtiem rezultātiem redzams, ka adaptīvā metode nodrošina ievērojamu dinamiskā diapazona paplašinājumu. Bez tam arī signāla pārveidošanas minimālā kļūda ir būtiski mazāka nekā klasiskās «up-and-down» metodes gadījumā.



Zīm. 2. Pārveidojuma rezultāta kļūda atkarībā no signāla amplitūdas: «up-and-down» metode (līkne 1); adaptīvā «up-and-down» metode (līkne 2).

Adaptīvā α - metode.

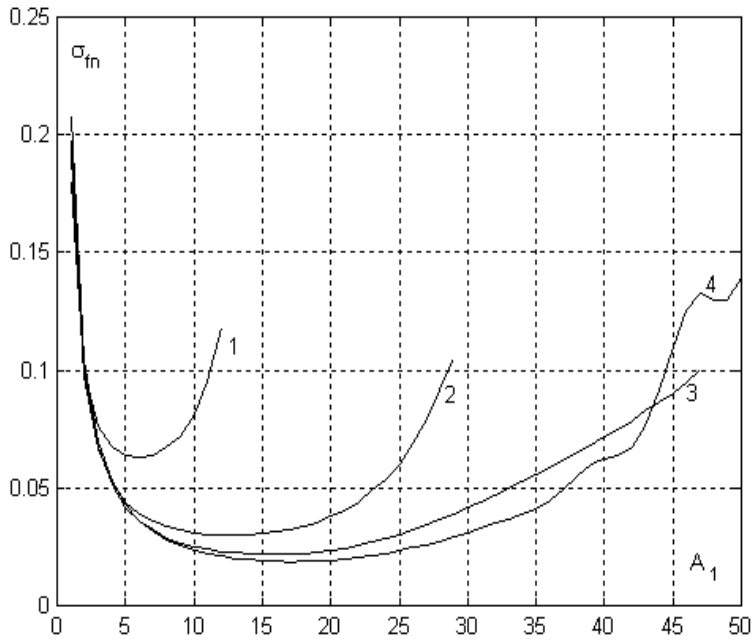
Šī metode izmanto informāciju, kas iegūta signāla mērīšanas laikā iepriekšējā fāzes punktā. Metode izstrādāta uz klasiskās «up-and-down» metodes bāzes. Saskaņā ar šo metodi, slieksni nākošajā signāla fāzes punktā iestāda vienādu ar:

$$e_{i+1} = e_i + \alpha(e_i - e_{i-1}),$$

kur $\alpha > 0$ - adaptācijas koeficients.

Signāla pārveidošanas kļūda σ_f pie dažādām α vērtībām parādīta Zīmējumā 3. No iegūtiem rezultātiem redzams, ka α-metode, salīdzinājumā ar klasisko «up-and-down» metodi, dod būtisku dinamiskā diapazona paplašinājumu. Savukārt, salīdzinājumā ar iepriekš apskatīto s-metodi, α-metode uzrāda mazāku kļūdu σ_f.

No visa augstāk ilustrētā redzams, ka adaptīvo metožu jomā pastāv lielas iespējas uzlabot signālapstrādi un paveras plašs pētījumu lauks.



Zīm. 3. Pārveidojuma rezultāta kļūda atkarībā no signāla amplitūdas: «up-and-down» metodes gadījumā (līkne 1) un adaptīvās α -metodes gadījumā pie dažādām koeficienta α vērtībām: α =0,6 (līkne 2); α =0,8 (līkne 3) un α =1,0 (līkne 4).

3. Signālu pārveidotāja bāzes bloka modernizācija

Stroboskopiskā signālu pārveidotāja (oscilogrāfa) bāzes bloka modernizācija tika veidota kā bloka sasaiste ar datoru, nodrošinot bloka vadību no datora. Šāds tehniskais risinājums it sevišķi svarīgs ir pārveidotāja pielietojumos superplatjoslas radiolokacijā, kad pārveidotājs no datora tiek gan vadīts, gan uz datora notiek informācijas papildapstrāde un indikācija. Konkrētajā gadījumā pārveidotāju vada programmējams kontrolieris ATMEL ATMEGA 128.

Papildus iegūstamās iespējas:

- Saglabāt oscilogrammas (vai radiolokatora reflektogrammas) failos datora diskos to vēlākai apstrādei un attēlošanai dažādos griezumos;
- Uzkrāt oscilogrammas pārveidotāju vadošās programmas atmiņā to attēlošanai pēc vajadzības un bez atkārtotas oscilogrāfēšanas;
- Viduvēt zināmu skaitu oscilogrammu trokšņa līmeņa pazemināšanai;
- Pielietot to vai citu algoritmu oscilogrammas apstrādei pēc operatora izvēles;
- Attēlot vairākas oscilogrammas vienlaikus.

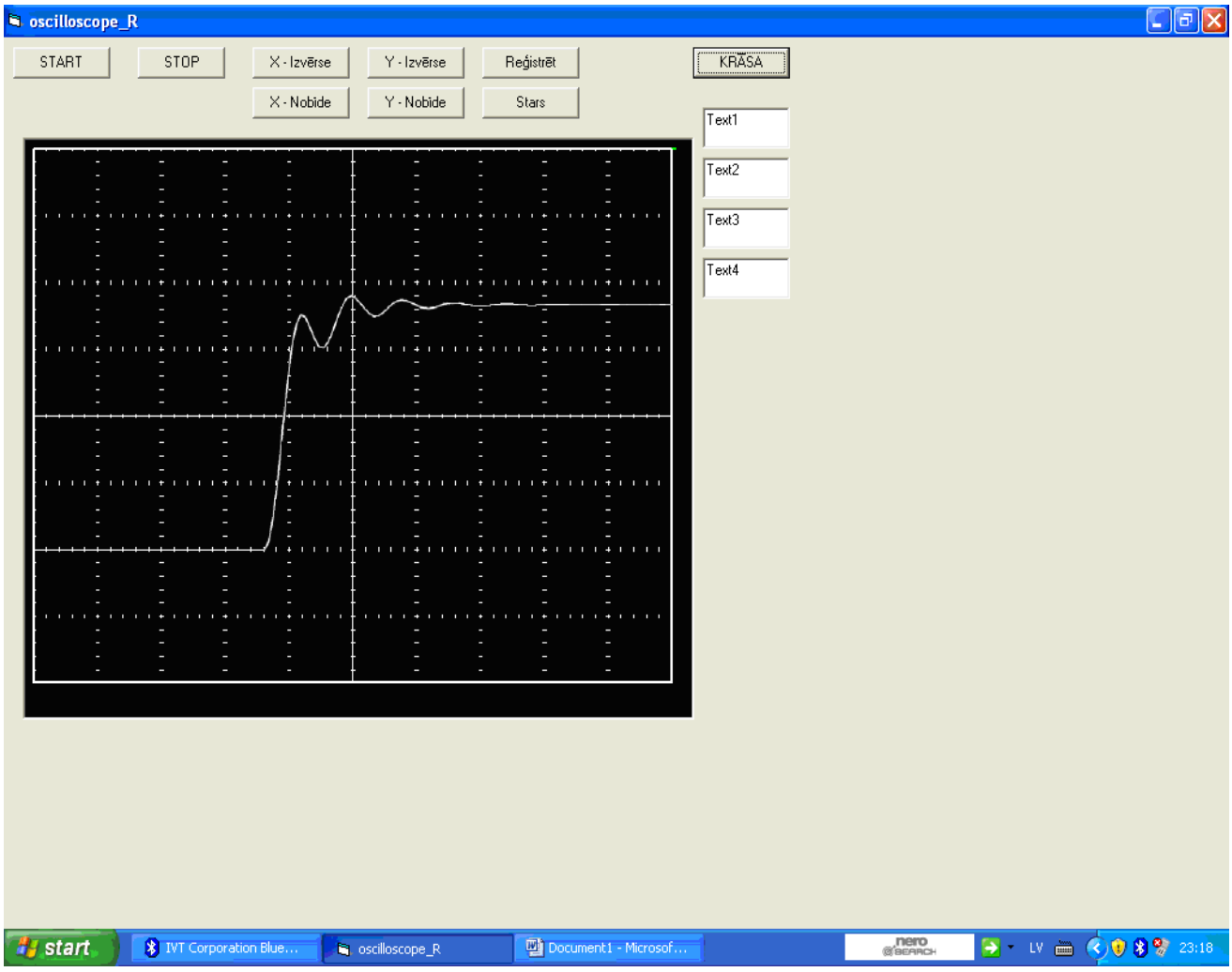
Lai pārveidotāju varētu vadīt ar datora palīdzību, ir nepieciešama speciāla programma, kura spēj strādāt Windows vidē un kurai būtu lietotāja grafiskais interfeiss, kas nodrošina oscilogrāfa vadību un darbības rezultātu attēlošanu grafiskā veidā (skat. Zīm. 1).

Zīmējumā ir parādītas galvenās oscilogrāfa vadības pogas:

1. *Start*, kura aktivizē dialogu ar operatoru, lai ievadītu interfeisa programma – oscilogrāfs parametrus

Stop, kas ir komanda programmas darba pārtraukšanai;

2. X-izvērse, kura aktivizē dialogu ar operatoru horizontālās izvērses ievadīšanai oscilogrāfā;



Zīm. 1 Lietotāja grafiskais interfeiss

3. X-nobīde, kura aktivizē dialogu ar operatoru horizontālās nobīdes ievadīšanai oscilogrāfā;

4. Y-izvērse, kura aktivizē dialogu ar operatoru vertikālās izvērses ievadīšanai oscilogrāfā;

5. Y-nobīde, kura aktivizē dialogu ar operatoru vertikālās nobīdes ievadīšanai oscilogrāfā;

6. Reģistrēt, kura aktivizē dialogu ar operatoru par oscilogrammu reģistrēšanu datora diska failā;

7. Stars, kura dod rīkojumu oscilogrāfam atrast staru;

8. Krāsa, kura ļauj izvēlēties ekrāna rūtiņu krāsu.

Vairākas no vadības pogām atver jaunus dialoga logus. Rūtiņās *Text* programma izvada informāciju par notiekošo darbību.

Secinājumi:

1. Izmantojot projekta ietvaros iegūtos tehniskos risinājumus, kā arī signālapstrādes algoritmus, ir iespējams izgatavot signālu diskrētos stroboskopiskos pārveidotājus ar jutību 15mkV (RMS). To apstiprina divi izgatavotie pārveidotāju eksperimentālie paraugi, kas atšķiras ar dažādas efektivitātes signālapstrādes algoritmiem.
2. Teorētiskie aprēķini un pārveidotāju datormodelēšana rāda, ka šāda tipa pārveidotājos būtu iespējams sasniegt 10 GHz joslu (un pat vairāk), ja strobējamo komparatoru izgatavotu SAF mikroshēmas veidā, kas samazinātu shēmas elementu ģeometriskos izmērus, parazītiskās induktivitātes un kapacitātes.
3. Ir iztrādātas jaunas efektīvas signālapstrādes metodes, kas samazina signāla pārveidošanai nepieciešamo strobu (iztvērumu) skaitu un vairākkārtīgi paplašina pārveidojamo signālu dinamisko diapazonu. Tas it sevišķi svarīgi ir tādos pārveidotāju pielietojumos kā superplatjoslas radiolokācija, jo būtiski palielina lokatora ātrdarbību un ļauj līdztekus ar ļoti vājiem signāliem, nepārslēdzot diapazonu, reģistrēt arī jaudīgus signālus.
4. Ir iegūti trīs strobējamā komparatora tehniski uzlabojumi:
   - Strobējamais komparators ar atbalsta pretestību — uzlabo komparatora stabilitāti;
   - Strobējamais komparators, kura slieksnis nav atkarīgs no signāla avota izejas pretestības;
   - Komparators, kura shemotehniski konstruktīvais risinājums mazāk šuntē komparatora ieeju un tādā veidā ļauj pilnīgāk izmantot komparatora elektronisko elementu (konkrētajā gadījumā tuneļdiožu) ātrdarbību un tā rezultātā pārveidotāja frekvenču joslu.

Augstāk minēto shemotehnisko risinājumu autori uzskata, ka būtu jāapsver iespēja šos risinājumus patentēt.

Publikāciju saraksts:

1. V.Plociņš. “Statistical Method of Signal – Noise Ratio Maximization”, “Electronics and Electrical Engineering” - Kaunas: Technologija, Vol. 94, No. 6, 2009, pp. 3-8. (Pētījums saistīts arī ar grantu 04-1127)

2. K.Krūmiņš, V.Pētersons, V.Plociņš. Features of Implementation of the Modified “up-and-down” Method”, “Electronics and Electrical Engineering” - Kaunas: Technologija, Vol. 93, No. 5, 2009, pp. 51-54. (Pētījums saistīts arī ar grantu 04-1127)

3. E. Beiners, K.Kruminsh. “Simulation and calculation of a balanced tunnel diode comparator”, “AUTOMATIC CONTROL AND COMPUTER SCIENCES”, 2009, Vol.43, Issue 1, pp. 17-21. (Pētījums visā pilnībā attiecas uz VPP)

4. E. Beiners, K.Kruminsh. “Simulation and calculation of an asymmetrical balanced tunnel diode comparator”, “AUTOMATIC CONTROL AND COMPUTER SCIENCES”, 2009, Vol.43, Issue 2, pp. 109-112. (Pētījums visā pilnībā attiecas uz VPP)

5. V.Karklinsh, K. Kruminsh. “Comparison of Signal Detection Methods under Conditions of Discrete Stroboscopic Transformation”, “AUTOMATIC CONTROL AND COMPUTER SCIENCES”, 2009, Vol.43, Issue 5, pp. 227-232. (Pētījums saistīts arī ar grantu 04-1127)

6. Tiek gatavota publikācija par adaptīvajām metodēm.

Augstāk minētos darbus veica:

Vad. Pētn., Dr.sc.comp. Kārlis Krūmiņš;

Vad. Pētn. Dr.sc.comp. Elmārs Beiners;

Pētn., Dr.sc.comp. Valdis Kārkliņš;

Asistents Vilnis Pētersons;

Progr. Tehn. Valdemārs Plociņš;

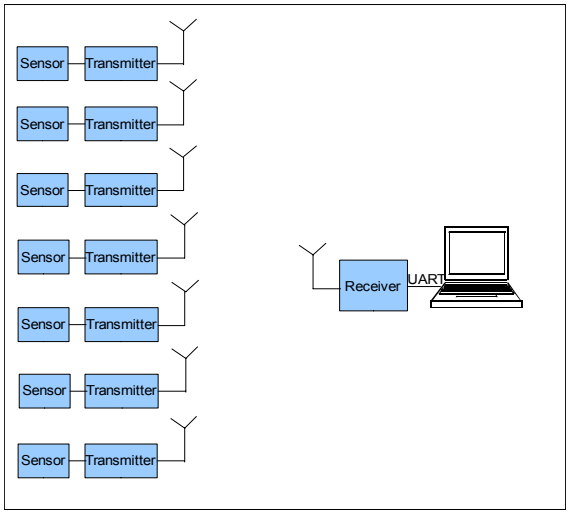
Inž. Madis Menke.

4. Multi-sensoru bezvadu tīkla izveide elektriskā lauka mērījumiem.

Ievads. Elektro pārvades līniju diagnostika ir apgrūtinoša, jo tai pa tiešo ar mēriekārtu nevar pieslēgt. Līdz ar to ir visai apgrūtināta līniju disbalansa kontrole. Disbalansu rada nevienmērīga noslodze pa fāzēm, kas ir svarīgs parametrs. Līdz ar to nepieciešama iekārta, ar kuras palīdzību var nomērīt līnijas parametrus nepieslēdzoties pašai līnijai. Var mērīt vadu radīto elektrisko lauku, kur nepieciešami elektriskā lauka sensori un pārraides sistēma, kas mērījumus pārsūtītu uz datoru. Turklāt, lai rezultāti būtu precīzi nepieciešami mērījumi vairākos punktos zem līnijas, līdz ar to pārraides sistēmai jābūt realizētai ar bezvadu pārraides sistēmas palīdzību, jo vairāku desmitu sensoru datu savākšana pa vadu sakariem ir neracionāla. Sensori zem līnijas jāizvieto vienādā attālumā līdzenā vietā.

4.1. Bezvadu sensoru tīkla risinājums.

Piedāvātais risinājums ir izveidot sistēmu no savācēj-iekārtas, kura pieslēgta datoram ar USB uz UART pārveidotāju un septiņiem elektriskā lauka sensoriem, katram no tiem ir savs bezvadu pārraides modulis datu pārraidei uz datoru (zīm.1.1). Elektriskā lauka sensori darbojas 'slave' režīmā, tie gaida komandas no savācējiekārtas sākt lasīt datus. Starta komanda kalpo kā sinhronizācijas komanda vienlaicīgi aktivizējot ACP, tādā veidā realizējot visu sensoru vienlaicīgu datu nolasīšanu. Visu sistēmas darbību vada no datora ar izveidoto LabView virtuālo instrumentu. Virtuālā instrumenta grafiskais lietotāja interfeiss nodrošina vadības funkciju, kā arī savākto datu apstrādi un attēlošanu lietotājam.
Starta komanda sākt lasīt datus tiek sūtīta atkārtoti tikai pēc attiecīgas pogas nospiešanas uz grafiskā lietotāja interfeisa. Katra sensora nolasīto datu bezvadu pārsūtīšana uz savācējiekārtu tiek realizēta pēc laika dalīšanas principa. Katram sensoram ir savs laika logs, kurš paredzēts mērījumu datu pārsūtīšanai. Šāda pieeja atvieglo sensoru tīkla pārraides protokolu. Visi bezvadu moduļi, kuri atbild par pārsūtīšanu atrodas uz viena frekvenču kanāla un izmanto vienu un to pašu adreses. Uzrakstītā programmatūra nodrošina desmit sensoru vienlaicīgu pieslēgumu savācējiekārtai. Pārveidojot programmnodrošinājumu un izstrādājot citu savācējiekartu iespējams palielināt sensoru skaitu līdz divdesmit, kas būtu nepieciešamais skaits.



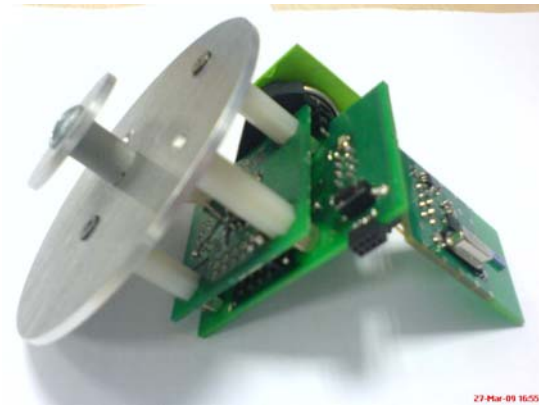
Zīm.4.1 Piedāvātā risinājuma blokshēma.

4.2 Elektriskā lauka bezvadu sensors.

Elektriskā lauka bezvadu sensors (zīm.4.2) sastāv no elektriskā lauka sensora – kondensatora(divas metāla ripas), pastiprinātāja, vadības mikrokontroliera un raiduztvērēja nRF24L01. Barošanai izmantotas divas AA tipa baterijas. Kā redzams zīmējumā iespiedplates veidotas moduļu veidā, lai varētu tās izmantot arī citiem mērķiem un arī lai vienkāršāk papildināt jau esošo sensoru ar jaunu moduli.

Analogā ieejas daļā ir izmantots augstas ieejas pretestības operācijpastiprinātās, kurš darbojas kā buferpastiprinātājs starp augstomīgo sensoru un zemomīgo mikrokontrolierī integrēto ACP. Pastiprinātājs ir ar nemaināmu pastiprinājuma koeficientu. Atkarībā no mērāmā elektriskā lauka intensitātes ir iespējams arī mainīt sensora jutību. To var panākt mainot pastiprinājuma koeficientu mikrokontrolierī integrētajam operācijpastiprinātājam, kurš slēgts kaskādē ar sensora buferpastiprinātāju. Šādu jutības maiņas iespēja ir ieviesta, lai varētu mērīt elektriskā lauka intesitāti vairākos augstsprieguma līniju apakštīklos. Papildus analogajā daļā ir izmantots no mikrokontroliera ACP iegūtā atbalsta sprieguma buferis, kurš atsaista mazlogojamo mikrokontroliera ACP atbalsta spriegumu no sensora plates.

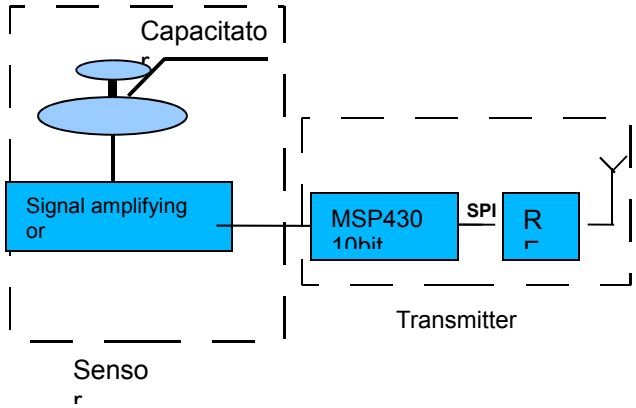
Atbalsta spriegums tiek summēts ar sensora signālu, tādā veidā ACP pievadītais signāls tiek nobīdīts uz ACP dinamiskā diapazona vidusdaļu. Tālāk šo signālu ir iespējams pastiprinā/vājināt mainot mikrokontrolierī integrēto operācipastiprinātāju diferenciālā pastiprinātāja slēguma pastiprinājuma koeficientu.



Zīm. 4.2 Elektriskā lauka bezvadu sensors.

Pēc analogā signāla apstrādes ciparu nolases tiek saglabātas mikrokontroliera atmiņā, kopā uzkrājot 96 nolases. ACP darbojas DMA režīmā ar diskretizācijas frekvenci 1kHz. Pēc diskretizācijas sensors gaida savu laika intervālu lai nosūtītu nolasītos datus uz savācējiekārtu. Nākošais etaps ir datu pakešu formēšana un pārsūtīšana pa bezvadu sakaru kanālu uz datoru. Raiduztvērējs nokonfigurēts režīmā, kur datu pakas garums ir 32 baiti. Garums izvēlēts vadoties no kopēja pārsūtāmā nolašu skaita. Pēc tam, kad mikrokontrolieris pa SPI interfeisu nosūtījis mērījumu datus uz nRF24L01 raiduztvērēja FIFO atmiņu, tiek aktivizēta bezvadu pārraide attiecīgajam sensoram attiecīgajā laika momentā. Katram sensoram pārraidei dots 3ms laika intervāls, kas ir ar rezervi, ja rodas kļūdas pārraidē, kā arī lai nodrošinātu, lai mikrokontrolieris paspēj apstrādāt pārtraukumus. Pēc 3ms laika, kad dati nosūtīti veiksmīgi, katrs sensors gaida kamēr visi pārējie nosūtīs datus uz savācējiekārtu. Tas kopā aizņem 30ms, pēc tam visi sensori tiek nokonfigurēti uztveršanas režīmā un gaida komandu no datora sākt lasīt jaunus datus.

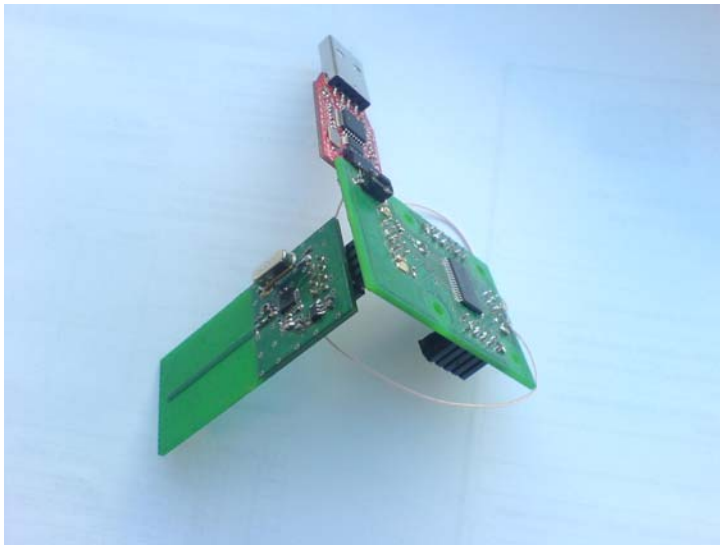
Mikrokontrolieris nodrošina raiduztvērēja konfigurēšanu un vadību. Konfigurācijas tiek realizēta ar SPI interfeisa starpniecību. SPI interfeiss darbojas ar 4MHz frekvenci. Lai nokonfigurētu raiduztvērēju nepieciešams sagatavot sūtāmo konfigurācijas paketi, ko veic mikrokontrolieris. Konfigurācijas pakete arī nosaka to, kāds būs pārraides protokols(datu pakas garums, pārraides ātrums, pārraidītā signāla jauda, frekvenču kanāls).
Zīmējumā 4.3 parādīta sensora blokshēma. Galvenais nosacījums sensoru konstrukcijas veidošanā bija izmēri un funkcionalitāte. Ar funkcionalitāti šeit domājot to, lai sensoru būtu vienkārši uzlabot no mehāniskās konstrukcijas viedokļa, ja rodas nepieciešamība. Tāpēc sensors salikts no atsevišķiem moduļiem. Svarīga loma ir sensora svaram un izmēriem, lai konstrukcija to atļautu ērti nostiprināt zem līnijas. Mūsu gadījumā sensori nostiprināti uz plastmasas caurulēm 1,90m augstumā no zemes.



Zīm. 4.3 Elektriskā lauka bezvadu sensora blokshēma.

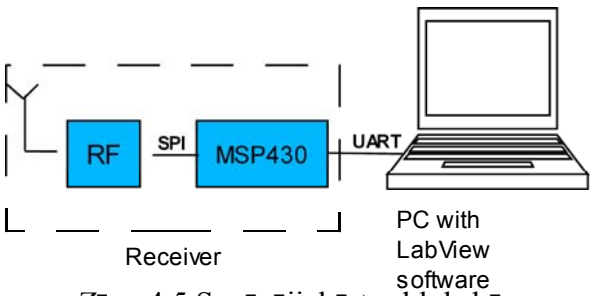
4.3 Savācējiekārta.

Savācējiekārta(zīm 4.4) sastāv no trīs atsevišķām iespiedplatēm, kuras savienotas kopā. Iespiedplates izstrādātas un izgatavotas Elektronikas un datorzinātņu institūtā. Kontroliera iespiedplate ir identiska sensora platei. Savienojums ar datoru nodrošināts izmantojot USB uz UART pārveidotāju. Savācējiekārtas uzdevums ir nodrošināt savienojumu ar datoru un sadarbību ar virtuālo instrumentu, kā arī sensoru datu savākšanu un pārsūtīšanu uz datoru.
Savācējiekārtā izmantots MSP430F2274 mikrokontrolieris ar 1k flash atmiņu un takts frkvenci 16MHz. Tas darbojas, kā vadības loģika raiduztvērējam un vienlaikus kā datu savācējs un datu pārsūtītājs uz datoru pa UART. Šāds risinājums izvēlēts jo atvieglo sākotnējo programmatūras izstrādi iekārtas vadībai. Vēlāk plānots UART aizstāt ar ātrāku interfeisu, lai varētu palielināt datu plūsmas ātrumu. Šobrīd datu pārraides ātrums starp datoru un savācējiekārtu ir 9,6kb/s.
Par UART pārveidotāju izmantots Texas Instruments MSPeZ430U, kurš arī kalpo kā mikrokontroliera programmators. Savācējiekārtas blokshēma parādīta 4.5 zīmējumā.



Zīm. 4.4 Savācējiekārta.

Savācējiekārta saglabā atmiņā visu septiņu sensoru saņemtos datus, pēc tam tos kā vienu datu masīvu nosūta uz datoru, kur tālāk datus apstrādā izveidotais LabView virtuālais instruments. Svarīgs parametrs ir atmiņas lielums, kas šajā gadījumā arī ierobežo sensoru skaitu.

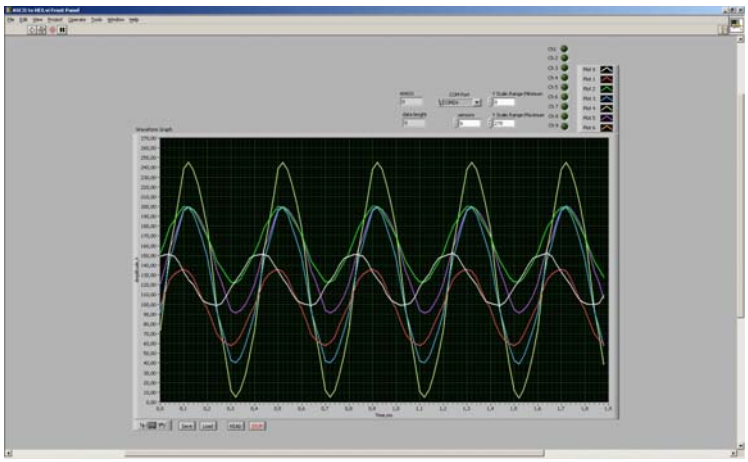


Zīm. 4.5 Savācējiekārtas blokshēma.

4.4 Programmatūras izstrāde.

Mikrokontroliera programmas rakstītas C valodā izmantojot IAR Workbench. Sākumā tiek nokonfigurēts kontrolieris un pa SPI interfeisu aizsūtīta konfigurācija raiduztvērējam nRF24L01. Konfigurācijas pakete satur tādus uzstādījumus, kā raiduztvērēja pārraidāmās datu pakas garums, pārraides ātrums, frekvenču kanāls u.c. Pēc konfigurācijas raiduztvērējs iestājas attiecīgajā režīmā un gaida tālākās komandas no kontroliera. 1. pielikumā un pievienotas sensora un 2. pielikumā savācējiekārtas C valodā uzrakstītās programmas.

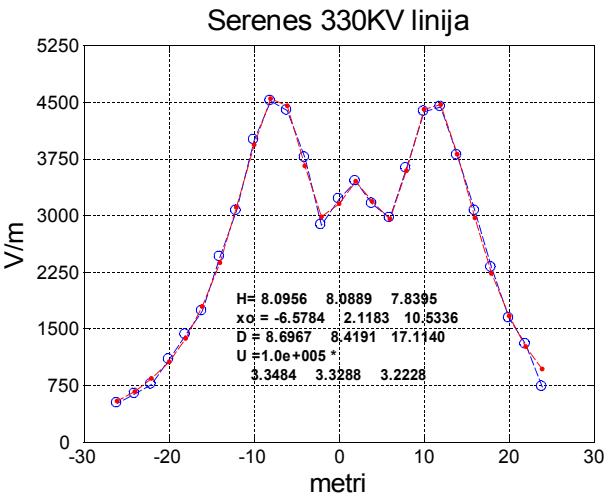
Sadarbība ar sensoriem tiek nodrošināta izmantojot Labview programmēšanas vidi. Labview ir grafiskā programmēšanas valoda ar kuras palīdzību ir iespējams izveidot grafisko interfeisu (logu) datu ievadīšanai un attēlošanu. Projektā Labview tiek izmantots sensoru skaita ievadīšanai, kā arī sadarbības porta ar raidītāju-uztvērēju norādīšanai. Izmantojot grafisko logu (zīm 4.6), tiek attēloti no sensoriem saņemtie dati. Saņemot datus no sensoriem tie ir ASCII koda izpildījumā, kurš nav tieši attēlojams, jo satur dažādus simbolus, kas nesniedz vizuālu informāciju par spriegumu augstsprieguma līnijā. Izmantojot Labview dati tiek pārveidoti uz heksadecimālo ekvivalentu, kuru ir iespējams attēlot grafika veidā un saglabāt failā kā saprotamus lielumus. Projekta laikā tika izskatīts variants šo pārveidojumu veikt, izmantojot MSP430xxxx mikrokontrolieri, nevis datoru. Taču izveidojot attiecīgas programmas tika konstatēts, ka nepieciešamā kontroliera atmiņa ir lielāka par 1k , kas nozīmē, ka lai realizētu šādu ideju, nepieciešams izmantot citu kontrolieri. Pie tam datu pārveidošana kontrolierī prasa zināmu laika resursu (jebkura darbība kontrolierī izpildās vienas vai vairāku (atkarībā no darbības sarežģītības pakāpes) takts laikā), kas šajā gadījumā palielina no kontrolieriem saņemtās informācijas nolasīšanas laiku un apjomu (viens ASCII baits satur 2 HEX baitus). Tādēļ kā optimāls risinājums tika izvēlēts datu paveidošanu izmantojot personālo datoru. No katra sensora saņemtais datu apjoms tiek apzīmēts ar unikālu simbolu, kas ir ietverta datu pirmajā baitā. Atpazīstot šo simbolu, ir zināms, no kura sensora dati tika saņemti. Saņemtos datus iespējams saglabāt 'txt' failā, kuru vēlāk apstrādā ar matemātiskajām metodēm, iegūstot augstsprieguma līnijas sprieguma sadalījumu fāzēs.



Zīm. 4.6 Grafiskais lietotāja interfeiss.

4.6 Rezultāti

Izveidotā bezvadu datu pārraides sistēma nodrošina mērījumu pārraidi no sensoriem uz datoru. Piedāvāta sistēma pilnībā izstrādāta Elektronikas un datorzinātņu institūtā, kas ietver sevī iespiedplašu izstrādi, kā arī programmatūru.
Ar izveidoto sistēmu veikti mērījumi zem reālas elektro pārvades līnijas. Sensoru skaits būtiski iespaido mērījumu precizitāti. Nepieciešams lielāks skaits sensoru, lai vienlaicīgi veiktu mērījumus vismaz 15 punktos zem līnijas. Uz doto brīdi sensoru skaits ir septiņi, kas ir par maz. Līdz ar to nepieciešams palielināt sensoru skaitu. Bezvadu sensori nodrošina datu pārraidi 40m attālumā, kas ir pietiekoši veicot mērījumus. Zemāk parādīti eksperimentālie mērījumi zem elektro pārvades līnijas pie Sērenes. Zilā līkne ir mērījumi, sarkanā aproksimācija. Mērījumi veikti pārvietojot sensorus un veicot jaunu mērījumu, iemesls tam ir nepietiekams sensoru skaits. Līdz ar to uz doto brīdi nav mērījumu, kuri būtu veikti vienlaicīgi visos punktos.



5. Multimodālas biometrijas paņēmienu attīstība izmantojot redzamo un infrasarkano attēlu apstrādi.

Šī etapa ietvaros tika izstrādāti vairāki attēlu iegūšanas paņēmieni cilvēka biometrijas datu reģistrēšanai, kā arī attīstītas attēlu apstrādes metodikas, kas ļauj no iegūtajiem attēliem izdalīt nepieciešamos biometrijas datus.

Darbs, galvenokārt, tika veikts šādos virzienos:

9. Plaukstas biometrija

j. Plaukstas attēlu iegūšana

xi. Izveidota eksperimentāla attēlu iegūšanas sistēma;
xii. Novērtēta iegūto attēlu kvalitāte pie dažādām sistēmas konfigurācijām, un izvēlēta labāka;*
xiii. Izveidota attēlu iegūšanas sistēma ar parālēlās programmējamās loģikas (FPGA) attīstības rīku palīdzību.

n. Plaukstas attēlu apstrāde

*xv. Iepriekš izveidotā 2D kompleksā salāgotā filtra teorētiskā bāze tika pamatota un pierādīta, veicot eksperimentus ar testa attēliem. Rezultāti publicēti**;*
xvi. Halo efekta noņēmšanai ir uzlabots 2D kompleksā filtra algoritms;
xvii. Realizēts Furjē deskriptoru iegūšanas algoritms plaukstas ģeometrijas analīzei.

18. Sejas biometrija

s. Sejas attēla iegūšana

xx. Izveidots CMOS sejas attēlu iegūšanas modulis;
xxi. Izveidots sejas atpazīšanas sistēmas makets;
xxii. Sastādīta DSP programma sejas attēlu iegūšanai.

w. Sejas atpazīšana

xxiv. Realizēts sejas detektēšanas algoritms izmantojot Hausdorfa attālumu;
xxv. Realizēts Eigenfaces algoritms seju atpazīšanai;
xxvi. Algoritmi realizēti DSP sistēmā.

*Veikti eksperimenti infrasarkano attēlu iegūšanai dažādos viļņa garumos, izmantojot dažādas metodes. Iegūtie eksperimentālie rezultāti sagatavoti un iesniegti publicēšanai:
R.Fuksis, M.Greitans, O.Nikisins, M.Pudzs „Infrared Imaging System for Analysis of Blood Vessel Structure" starptautiskā konferencē ELECTRONICS 2010 (Lietuvā).

**Rezultāti par komplekso 2D salāgoto filtru apkopoti publikācijā un ziņots starptautiskajā konferencē:
R.Fuksis, M.Greitans, M.Pudzs, "Object Analysis in Images Using Complex 2D Matched Filters", Proceedings of IEEE Region 8 EUROCON 2009 Conference, May 2009, Saint-Petersburg, Russia, pp. 1394 –1399.

5.1. Biometrijas sistēmas

Uzticama identitātes noteikšanas sistēma ir ļoti svarīga dažādu darbību veikšanai, piemēram, iekļūšanai ēkās, finansiālu darbību veikšanai, robežu šķērsošanai u.c. Pašreiz lietotajām metodēm – parolēm, ID kartēm, pasēm u.c. Ir trūkumi – paroles var aizmirst, karti vai pasi var nozaudēt, vai nozagt, tādēļ rodas nepieciešamība pēc drošas biometrijas ierīces, kas spēs identificēt cilvēku pēc tā bioloģiskajām īpašībām. Protams, arī biometrijas metodes iespējams viltot, piemēram, pirkstu nospiedumus.

Mūsu piedāvājums ir izmantot cilvēku plaukstas biometrijas datus, lai pārliecinātos par personu identitāti. Kā arī veiktie pētījumi sejas atpazīšanas jomā apstiprina, ka izmantojot cilvēka sejas parametrus mēs varam iegūt drošu biometrijas sistēmu.

5.2. Plaukstas biometrija

Cilvēka plauksta satur daudz informācijas, kas būtu noderīga personu identifikācijai. Pirmajā acu skatā mēs varam pamanīt tikai plaukstas formu un rievas, kas atrodas uz plaukstas pamatnes, taču pielietojot modernus attēlveidošanas paņēmienus un iegūstot attēlus infrasarkanajā diapazonā mēs varam arī iegūt plaukstas asinsvadu izvietojuma attēlu. Tādā veidā plauksta sniedz informāciju par trīs dažādiem biometrijas parametriem.

Asinis transportē skābekli ar hemoglobīna palīdzību, kas tajās atrodas; Hemoglobīns kļūst piesātināts ar skābekli, kad tam pievienojas skābeklis no plaušām. Tas kļūst nepiesātināts, kad tiek izmantots ķermeņa asinsvados. Artērijas satur skābekļa piesātinātu hemoglobīnu, bet vēnas, asinsvadi skābekļa nepiesātinātu hemoglobīnu.

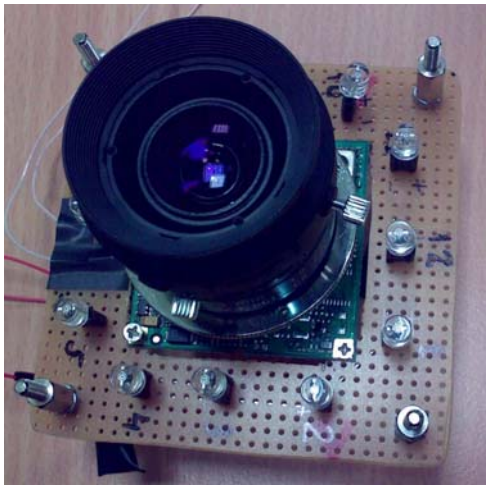
Divu veidu hemoglobīnam ir dažādi absorbcijas spektri. Skābekļa nepiesātināts hemoglobīns sevišķi labi absorbē gaismu ar viļņa garumu ap 760 nm, tuvo infrasarkano staru apgabalā. Kad tiek uzņemts plaukstas attēls izmantojot tuvo infrasarkano starojumu, asinsvadi būs tumšāki par apkārtējo plaukstas daļu, jo tikai asinsvadi absorbē šos starus. Artērijas ir dziļāk, tāpēc tās praktiski nevar redzēt.

Tiek lietotas divas asinsvadu attēlveidošanas metodes: atstarošanās un pārraides metode. Atstarošanās metodē plauksta tiek apgaismota no apakšas, bet pārraides metodē plaukstu apgaismo no aizmugures, sāna vai virsmas ap to.

Atstarošanās metodē izstarojošais gaismas avots un uztverošā ierīce var tikt apvienoti, jo virziens kādā notiek izstarošana un atstarošana ir vienāds. Tomēr pārraides metodē ierīces jālieto atsevišķi, jo mainās izstarošanas un uztveršanas ierīču novietojums.

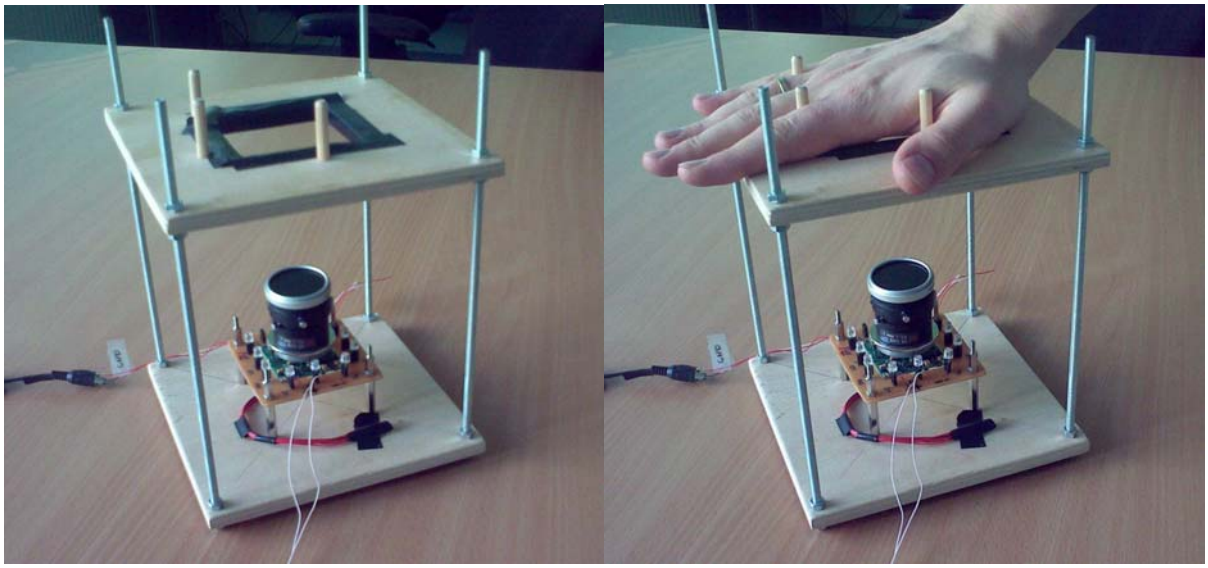
5.3. Plaukstas attēla iegūšanas sistēma

Tika izveidota eksperimentāla attēlu iegūšanas sistēma redzamajā un infrasarkanajā gaismā. Sistēmas pamatā ir kamera ar spektrālo jutību infrasarkanajā diapazonā, infrasarkano lēcu un infrasarkanajām diodēm (5.1. zīm.).



5.1. zīm. Kameras makets

Lai salīdzinātu attēlu kvalitāti un to kā asinsvadi tiek uzņemti pie dažādiem viļņa garumiem, tika izveidots speciāls statīvs plaukstas novietošanai virs kameras (5.2. zīm). Rezultātā tika iegūti 10 plaukstas asinsvadu izvietojuma attēli, pie katra infrasarkanā apgaismojuma viļņa garuma.



5.2. zīm. Eksperimentālā attēlu iegūšanas sistēma ar plaukstas statīvu

Pēc attēlu iegūšanas, kamera tos nosūta uz datoru caur USB datu savākšanas terminālu. Nākamais solis ietver attēlu apstrādi, kas tiek veikta MATLAB vidē.

Tika salīdzināti atstarošanās un pārraides metožu rezultāti savā starpā.

5.4. Attēlu kvalitātes novērtēšana sistēmas parametru izvēlei

Ir vairāki veidi kā novērtēt attēlu kvalitāti, piemēram, pēc histogrammas. Taču parastās novērtēšanas metodes mūsu gadījumā nedod pietiekamu informāciju, lai secinātu pie kura infrasarkanā viļņa garuma tiek iegūti labākie attēli.

Attēlu iegūšanas sistēmas galvenais uzdevums ir iegūt pēc iespējas kvalitatīvākus plaukstas asinsvadu attēlus, kas nozīmē vairāk asinsvadu un mazāk trokšņa. Tādēļ tiek ieviesti divi parametri, kas nosaka sistēmas efektivitāti:

1. Detektējamo asinsvadu skaits: n_s

2. Tīrības pakāpe: $$P = \frac{n_s}{n_s + n_N}$$

kur n_N – dektējamo trokšņu skaits.

Salīdzinot iegūtos attēlus ir grūti izvērtēt abus parametrus vienlaicīgi, tādēļ tiek apskatīts to reizinājums, kas tiek nosaukts par efektivitātes vērtību:

$$E_{ff}(T) = P \cdot n_S = \frac{n_S^2}{n_S + n_N}$$

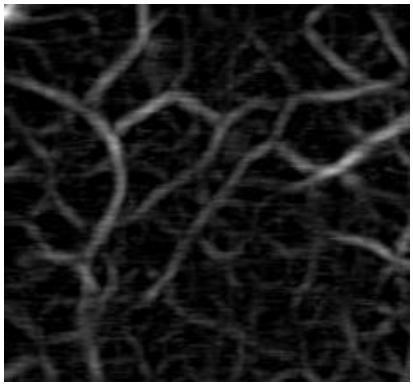
Efektivitātes vērtība ir atkārīga no apstrādes procesā izmantotās sliekšņa vērtības T: pie

augstākām sliekšņa vērtībām būs mazāk trokšņu, kā arī mazāk detektēto asinsvadu, kas nav pietiekami personu identificēšanai; turpretīm, pie zemākām sliekšņa vērtībām detektēto asinsvadu skaits pieaug, bet pieaug arī traucējošo objektu skaits, kas nav droši atpazīšanai. Piedāvātais efektivitātes rādītājs ir paņēmiens kā atrast optimālo sliekšņa vērtību un pēc tās maksimuma vērtības savstarpēji novērtēt uzņēmtos attēlus.

Iegūtais plaukstas attēls pirms apstrādes ir redzams5.3. zīm:

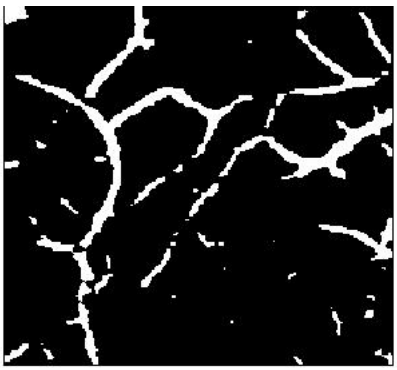
5.3. zīm. Plaukstas attēls pirms apstrādes

Veicot 2D salāgoto filtrāciju ar Gausa filtra masku, tiek iegūts asinsvadu attēls, taču kā redzam5.4. zīm., tas satur daudz trokšņus, kurus ir iespējams no attēla izslēgt.



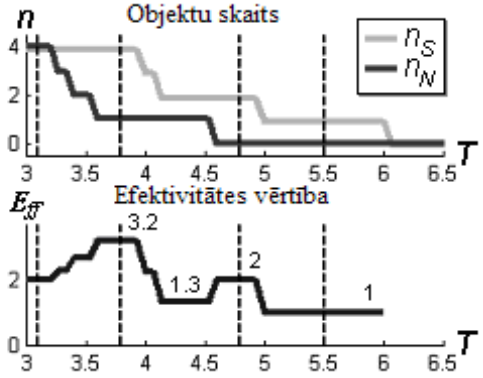
5.4. zīm. Plaukstas attēls pēc 2D
salāgotās filtrācijas ar Gausa masku

Lai atvieglotu tālāko attēla segmentēšanu tiek veikta sliekšņoperācija, 20% no maksimālās intensitātes vērtības. Nosakot operācijas sliekšņa vērtību, visi objekti ar intensitāti zemāku par sliekšņa vērtību tiek atmesti, bet pārējie atstāti. Attēls pēc sliekšņoperācijas ir redzams 5.5. zīmējumā.



5.5. zīm. Attēls pēc sliekšņoperācijas

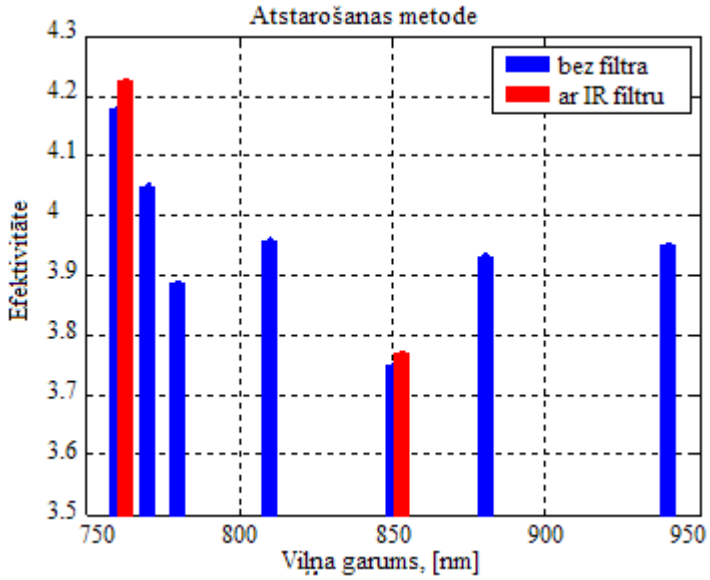
Algortms sadala attēlu segmentos, meklējot līniju fragmentus. Katra segmenta maksimālā intensitātes vērtība tiek saglabāta – tā nosaka maksimālo sliekšņa vērtību, lai segments tiktu detektēts. Attiecīgi, ja sliekšņa līmenis būs augstāks par saglabāto sliekšņa vērtību, tad segments netiks detektēts. Šīs iegūtās vērtības ļauj saskaitīt segmentu skaitus pie dažādām sliekšņa vērtībām. Tālāk tika izveidota maska, kas pusautomātiskā režīmā sadala segmentus divās kopās: pirmajā kopā tiek ietverti tie segmenti, kas pieder asinsvadiem, otrā – tie, kas detektēti kā troksnis. Rezultātā tiek iegūts grafiks, kurš attēlo objektu skaitu un efektivitātes vērtību atkarībā no sliekšņa vērtības T (5.6. zīm.):



5.6. zīm. Segmentu skaits un efektivitātes vērtības pie dažādām sliekšņa vērtībām

Pirmajā grafikā redzams objektu skaits: pelēkā līnija attēlo detektētos asinsvadu segmentus un melnā līnija parāda detektētos trokšņu segmentus. Otrais grafiks parāda kā izmainās efektivitātes vērtība, atkarība no detektēto objektu skaita – tai ir noteikts maksimums, kas arī nosaka optimālo sliekšņa vērtību. Katrs iegūtais attēls tika novērtēts, izmantojot minēto efektivitātes maksimālo vērtību. Veicot eksperimentus pie vairākiem infrasarkanā starojuma viļņa garumiem, šie novērtējumi izrādījās dažādi.

Attēlu novērtēšanas rezultāti tika apkopoti 5.7. zīm:

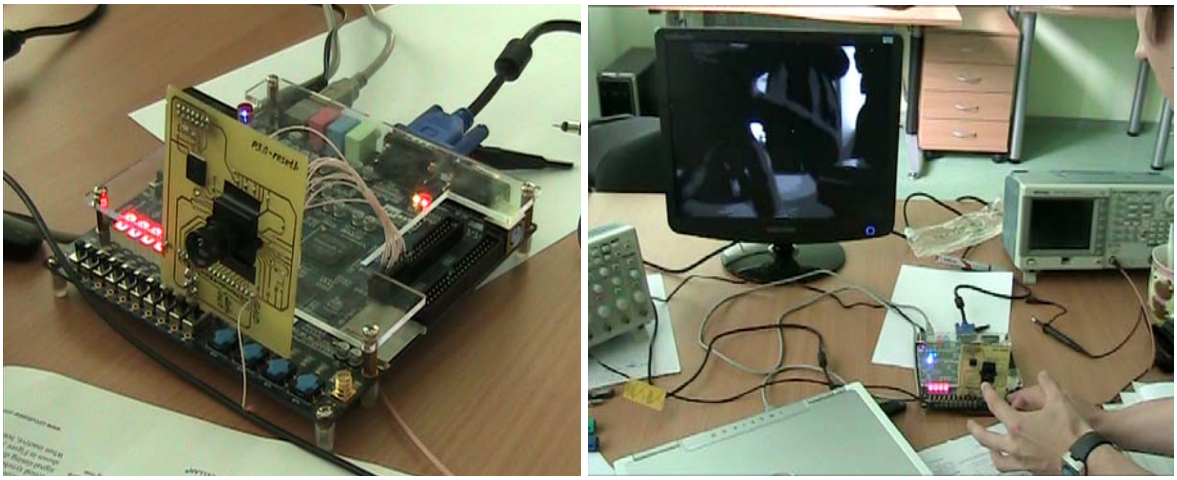


5.7. zīm. Eksperimentālie rezultāti atstarošanas metodei

Kā redzams, no eksperimentālo rezultātu grafika, vislabākos attēlus iespējams iegūt pie sistēmas konfigurācijas, kad tiek izmantotas 760nm infrasarkanās gaismas diodes ar 760nm±10nm infrasarkanās gaismas optisko joslas filtru. Tieši šādi sistēmas konfigurācijas parametri tika uzstādīti, lai iegūtu vairāku cilvēku plaukstas asinsvadu izvietojuma zīmējumus tālākai attēlu apstrādes paņēmienu attīstīšanai.

5.5. Attēlu iegūšanas sistēma, izmantojot FPGA

Pēc optimālo sistēmas parametru izvēles, darbs tika veikts automatizētās sistēmas rādīšanai – lai ar vienas kameras palīdzību ātri (automātiskajā režīmā) iegūt plaukstas attēlus gan infrasarkanajā gaismā, gan redzamajā. Iegūtie attēli sniedz informāciju par plaukstas ģeometriju (gan atpazīšanai, gan arī lai pārliecinātos, ka sistēmai bija uzrādīta plaukstа, nevis viltots asinsvadu zīmējums), asinsvadiem un rievām.
FPGA paralēli ļauj gan uzņemt attēlus, gan arī veikt to apstrādi ar nepieciešamiem filtriem.

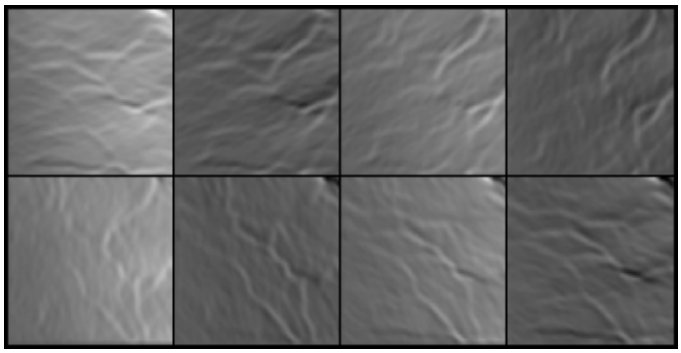


5.8. zīm. Eksperimentālie rezultāti atstarošanas metodei

Bija izveidots attēlu iegūšanas sistēmas prototips, ko var redzēt attēlā (5.8.). Šī ierīce izmanto savu attēlu sensoru un tajā esošais FPGA var arī aizvietot sensoram nepieciešamas palaišanas un konfigurēšanas mikroshēmas, kas samazina nepieciešamu detaļu skaitu.

5.5. Kompleksais 2D salāgotais filtrs asinsvadu izdalīšanai

Salāgotā filtrācija ir visefektīvākais veids kā izdalīt vajadzīgo signālu uz trokšņa fona – tas ir pierādīts matemātiski. Taču veicot 2D salāgotu filtrāciju, atšķirībā no 1D gadījuma (piemēram, laika funkcijām) – filtra masku ir iespējams arī rotēt. Pie katra maskas leņķa filtrs izdalīs tikai tās detaļas (asinsvadus), kas būs ko-orientētas ar to. Jo vairāk leņķu tiek izvēlēti, jo precīzāka ir izdalīšana, taču arī lielāks operāciju skaits. Attēlā redzami filtrācijas rezultāti 8 leņķiem:



5.9. zīm. Salāgotās filtrācijas rezultāti pie 8 maskas leņķiem

Rodas jautājums: kā pēc tam šos rezultātus apstrādāt? Tradicionālā salāgotā filtrācija paredz rezultātu apvienošanu, ņēmot maksimālas filtra reakciju vērtības – tadējādi "savācot" kopā visas reakcijas. Taču asinsvadu izdalīšanā mūs interesē tikai skaidri orientēti līniju fragmenti.
Bija pamanīts ka šādos gadījumos filtra reakcijas būs vairāk izteiktās tieši asinsvada virzienā. Objektiem, kuri nav līdzīgi līnijām (trokšņiem), reakcijas būs ar haotisku raksturu, vai visos virzienos aptuveni vienādas. Filtra ideja bija piešķirt reakcijām orientāciju, vienādu ar maskas dubultotu leņķi, pārvēršot tās par kompleksiem vektoriem; saskaitīt kopā un samazināt leņķi. Šos procesus apraksta sekojošas formulas:

$$\vec{c}(x_0, y_0) = \sum_{\varphi} \left[e^{j2\varphi} \cdot \int\int_D f(x,y) \cdot M(x_0, y_0; x, y; \varphi) \cdot dxdy \right]$$

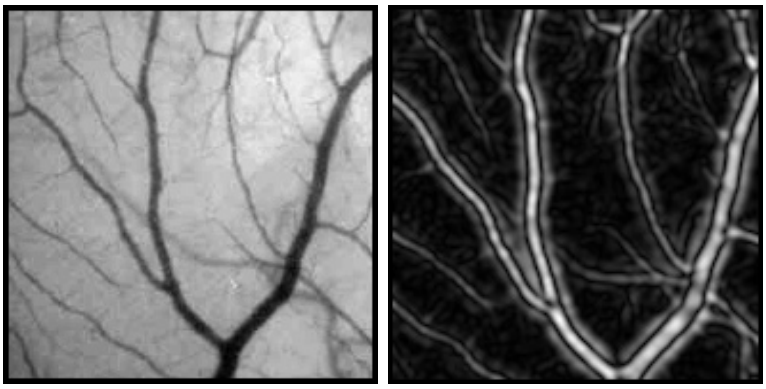
$$\vec{c}(x_0, y_0) = c(x_0, y_0) \cdot e^{j2\psi} \quad \Rightarrow \quad \vec{v}(x_0, y_0) = c(x_0, y_0) \cdot e^{j\psi}$$

Visus procesus, izņemot pēdējo, linearitātes dēļ izdevās apvienot kopā – vienā filtra maskā. Šāds filtrs tika nosaukts par Komplekso 2D Salāgoto Filtru. Šis filtrs ir spējīgs izdalīt asinsvadus ar vienu filtrāciju (īstenībā divas – reālo un imagināro maskas daļu), bet tam piemīt leņķa izšķiršanas spēja, kas ir labāka par augstāk aprakstītiem 8 leņķiem. Pie tam, filtrācijas rezultāts ir nevis skalāras vērtības, bet vektori (korelācijas vektori), kas nosaka ne tikai asinsvadu intensitāti, bet arī orientāciju, kas palīdz attēla segmentēšanas procesā.
Kompleksais 2D salāgotais filtrs bija veiksmīgi prezentēts starptautiskajā Eurocon 2009 konferencē Sanktpēterbugā, Krievijā.

Ar šo filtru iegūtie korelācijas vektori ir līdzīgi īpašvektoriem, ko iegūst ar Hessian matricas palīdzību.

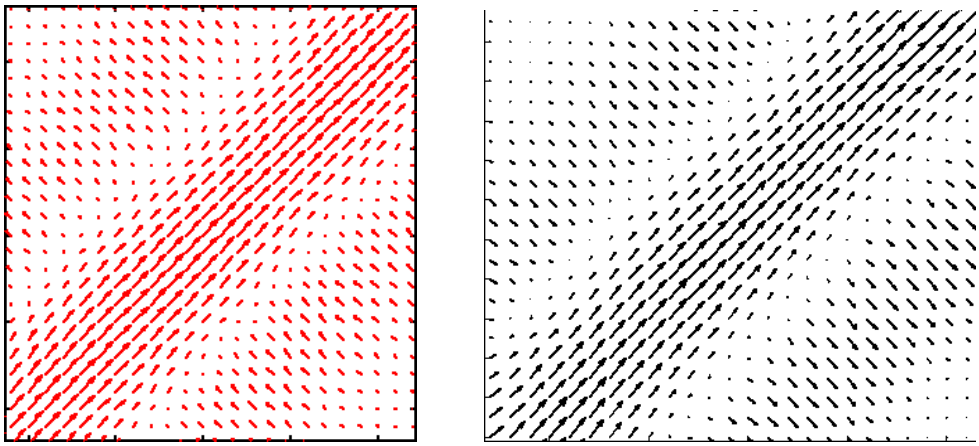
5.6. Kompleksais 2D salāgotais filtrs bez Halo efekta

Kompleksam 2D salāgotam filtram piemīt nevēlama īpašība rādīt tā saukto Halo efektu ap izdalītiem objektiem.



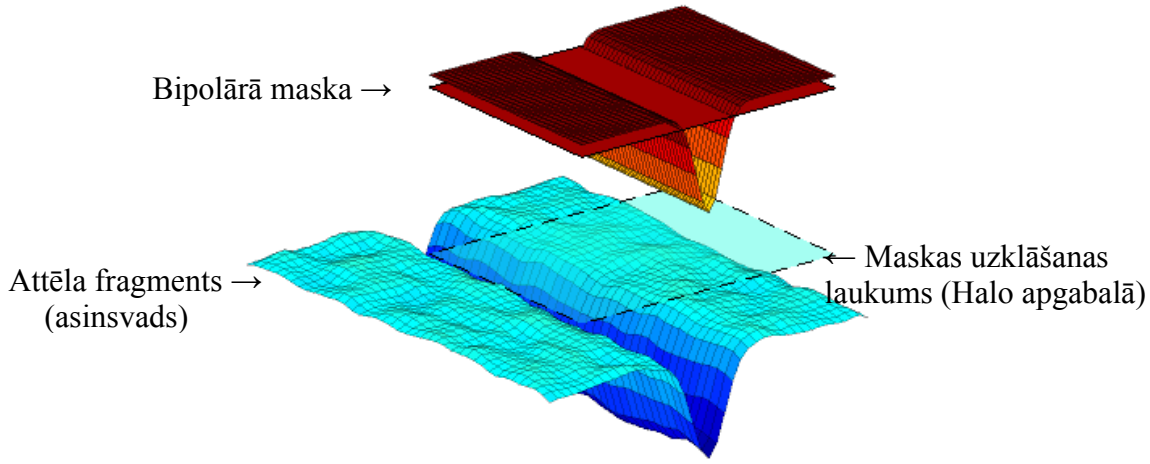
5.10. zīm. Halo efekta piemērs

Arī Hessian Matricas paņēmiens rada līdzīgo efektu – apkārtnē ap izdalītiem asinsvadiem (Halo apgabalos) veidojās "mākoņi" ar perpendikulāri orientētiem korelācijas vektoriem.



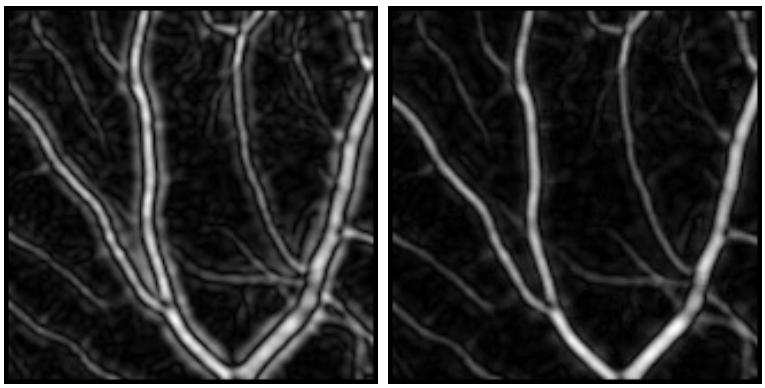
5.11. zīm. Korelācijas vektori un Hessian matricas īpašvektori – izdalītais asinsvads pa vidu, un Halo apgabals sānos

Bija izpētīts kāpēc kompleksais filtrs tā reaģē. Izrādījās, ka izmantojot bipolāru masku (parādīta attēlā) ir iespējams izvairīties no šāda efekta, ko, piemēram, nespēj Hessian matricas paņēmiens.



5.12. zīm. Bipolārā salāgotā filtra maska un negatīvo reakciju veidošanās Halo apgabalā

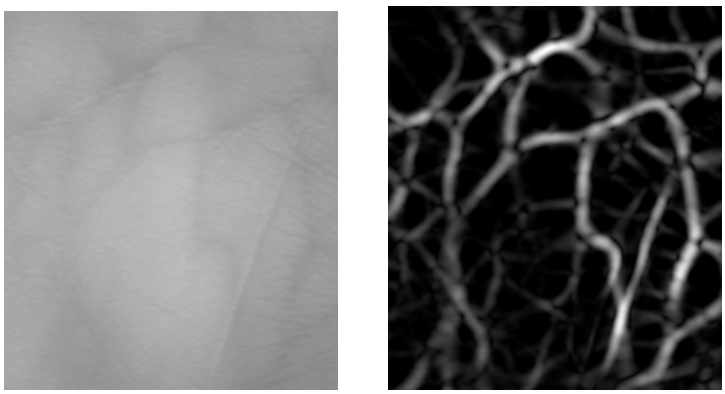
Šāda filtra maska rada negatīvas reakcijas asinsvada virzienā Halo apgabalā, kas dubultotiem leņķiem ir ekvivalents pozitīvām filtra reakcijām perpendikulāri asinsvadam. Piemērojot vienkāršu taisngriešanu (negatīvo reakciju atmešanu) izdevās pilnībā izvairīties no Halo efekta.



5.13. zīm. Kompleksais 2D salāgotais filtrs ar un bez Halo efekta

Šāds paņēmiens ir būtisks filtra kvalitātes uzlabojums, taču uz ātruma rēķina – filtrācija ir jāveic atkal pie katra maskas leņķa.

Kompleksais 2D salāgotais filtrs bez Halo efekta ļauj efektīvi apstrādāt iegūtos infrasarkanos asinsvadu attēlus. Filtrs izdala tikai līnijām līdzīgas detaļas, ko var redzēt attēlos:

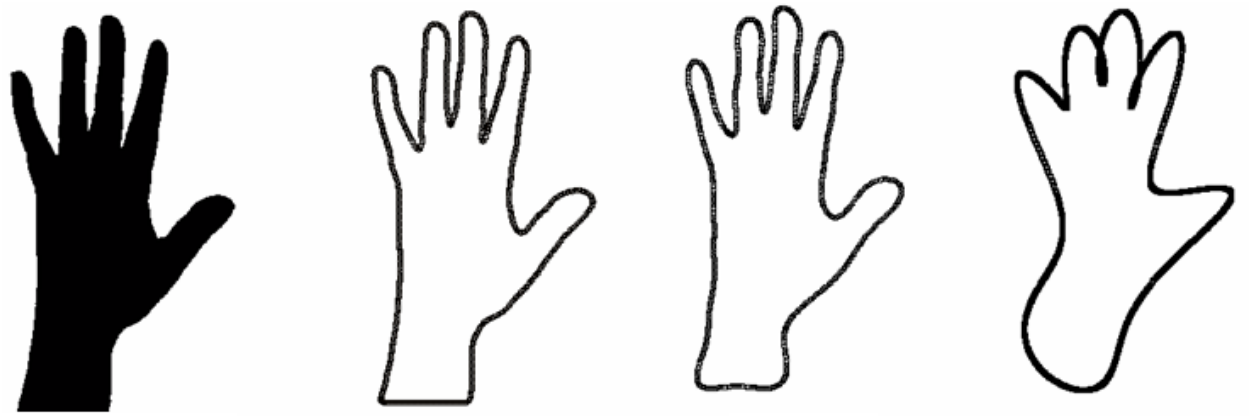


5.14. zīm. Reālais attēls un 2D kompleksā salāgotā filtra reakcija (bez Halo efekta)

Pateicoties kompleksiem korelācijas vektoriem, attēlu ir iespējams viegli segmentēt un biometrisko atpazīšanu veikt pēc personas plaukstas asinsvadu tīkla sazarojumu punktu koordinātēm, kas ir rotācijas un mērogošanas invariants salīdzināšanas veids. Pie tam, sazarojuma punktu koordinātu glabāšana datubāzē ir efektīva no izmantojamās atmiņas apjoma viedokļa.

5.7. Plaukstas ģeometrijas analīze ar Furjē deskriptoriem

Tika realizēts plaukstas ģeometrijas izdalīšanas un optimizācijas algoritms, kas balstās uz Furjē deskriptoru metodes. Šāda algoritma izveidošanai ir liela nozīme datu glabāšanas procesā. Tas dod iespēju neglabāt visu plaukstas attēlu, bet gan tikai nelielu daļu koeficientu, kas apraksta konkrēto plaukstas ģeometriju. Veiktie eksperimenti pierāda, ka iegūstot plaukstas attēlu un izdalot tā kontūru, mēs to varam aprakstīt ar Furjē deskriptoriem. Iegūtie Furjē koeficienti parasti ir ļoti daudz, taču eksperimentu rezultātā izrādījās, ka plaukstas formu iespējams atjaunot bez nozīmīgiem kropļojumiem, ja mēs atstājam tikai 10% no visiem aprēķinātajiem Furjē koeficientiem. Eksperimentu rezultāti ir parādīti 1.3.5.15. zīmējumā:



5.15. zīm. No kreisās, oriģinālais attēls, 10% Furjē koeficientu, 4% un 1%.

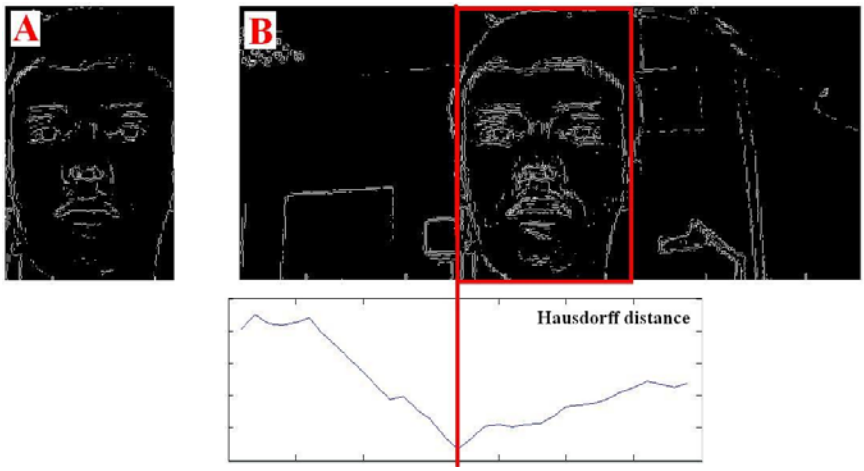
5.8. Sejas detektēšanas un atpazīšanas sistēma

Viens no svarīgākajiem biometriskiem parametriem ir cilvēka seja. Šajā projektā tika veikti plaši pētījumi sejas atpazīšanas jomā, kuru mērķis ir portatīvas, energoefektīvas, datorneatkarīgas biometrijas atpazīšanas sistēmas izveidošana.
Šādas sistēmas izveidošana ir multidisciplinārs uzdevums, kuru atrisināšanas procesu var sadalīt sekojošos posmos:

- sejas apstrādes un atpazīšanas algoritmu izveidošana
- algoritmu realizācija un testēšana datorā
- datorneatkarīgas, reāla laika signālu apstrādes sistēmas izvēlēšana (DSP sistēma)
- attēlu iegūšanas bloka izveidošana
- attēlu apstrādes un atpazīšanas algoritmu implementācija DSP sistēmā
- uz DSP bāzētas sējas atpazīšanas sistēmas testēšana

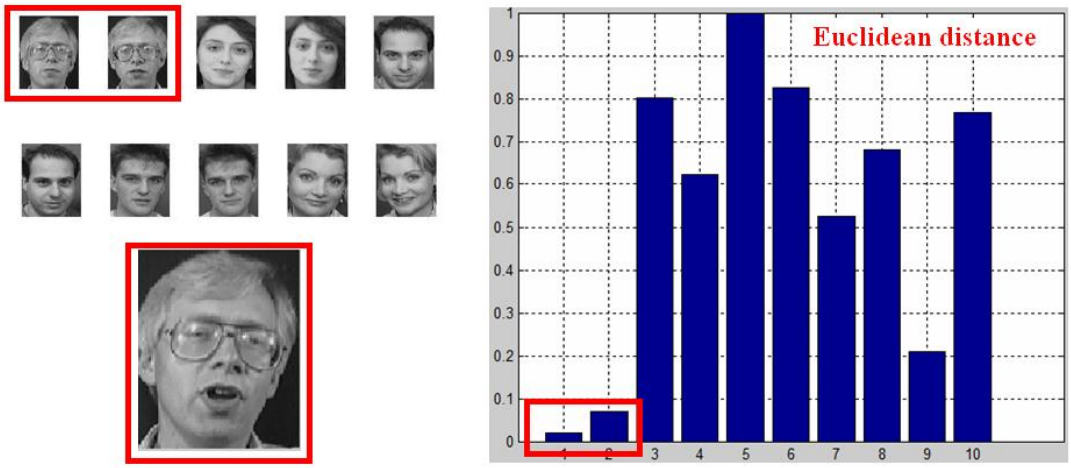
Sejas atpazīšana mūsu gadījumā tiek balstīta uz diviem algoritmiem: uz Hausdorfa attāluma bāzētais sejas detektēšanas algoritms un „Eigenface" jeb „īpašseju" sejas atpazīšanas algoritms.
Īsumā Hausdorfa sejas detektēšanas algoritmu var aprakstīt sekojoši:
Sejas attēls (sejas modelis) tiek pārvietots divās dimensijās gar pētāmo attēlu ar noteiktu soli. Katrā pārvietošanās punktā tiek aprēķināts Hausdorfa attālums starp sejas modeli un atbilstošo attēla daļu.

Hausdorfa attāluma minimums nosāka vislielāko sakritību starp sejas modeli un pētāmā attēla daļu, citiem vārdiem, sejas atrašanās vietu attēlā.



5.16. zīm. Sejas modeļa A pārvietošana gar attēlu B horizontālajā virzienā

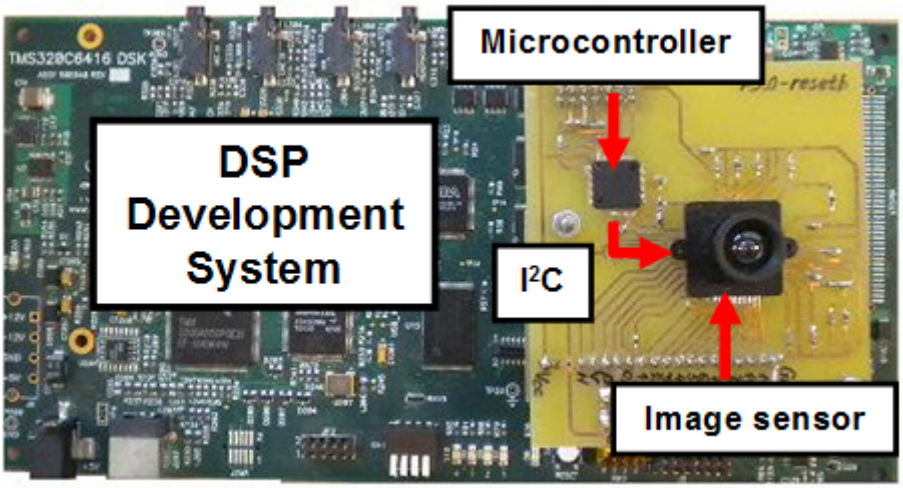
„Eigenface" sejas atpazīšanas algoritms balstās uz „Principiālo komponenšu analīzes" paņēmieniem un jaunas sejas salīdzināšanu ar seju datu bāzi. Algoritma darbības rezultātā tiek izrēķināts Eiklīda attālums starp jauno seju un katru seju no datu bāzes. Attāluma minimums nosaka vislabāko sakritību.



5.17. zīm. Eigenface algoritma darbības rezultāts datu bāzei ar 10 attēliem

Šie algoritmi pilda divas fundamentālās funkcijas: sejas detektēšanu attēlā un nodetektētas sejas atpazīšanu. Algoritmi tika realizēti un notestēti datorā, ka arī implementēti datorneatkarīgā signālu apstrādes sistēmā.
Reālā laikā seju atpazīšanas sistēmas maketa izveidošanai tika izvēlēts specializēts TMS320C6000 sērijas signālapstrādes procesors.
Seju atpazīšanas sistēmas pamatelementi:

1. Plate ar signālprocesoru (TMS320C6416), kuras uzdevumi ir attēlu sensora datu savākšana un attēlu apstrāde (sejas detektēšana un atpazīšana).
2. Mikrokontrolieris (MSP430F169), kura uzdevums ir attēlu sensora parametru uzstādīšana caur I^2C interfeisu.
3. Attēlu sensors (KAC-9618), kurš tiek paredzēts digitāla attēla iegūšanai.



5.18. zīm. Seju atpazīšanas sistēmas makets

Uz doto brīdi ir veiksmīgi izpildīta nozīmīga sejas atpazīšanas uzdevuma daļa.
Tika izveidota uz signālprocesora balstīta aparatūra sejas atpazīšanai, sastādīta signālprocesora programma datu iegūšanai no attēlu sensora. Izpētīti sejas detektēšanas un atpazīšanas algoritmi MATLAB vidē, ka arī implementēti signālprocesorā.

1. Pielikums

MATLAB programma signāla atjaunošanai no līmeņu-šķērsojumu nolasēm, ņemot vērā meklējamo nolašu vērtību ierobežojumu

```
clear all
a=load('/home/rolands/Desktop/work/ShannTFD/sx335interp.mat'); a=a.Satj;
t=0:1/80000:length(a)*1/80000-1/80000;
[Sn2,tn2,xn,txn,u]=lvcrossing2(a,80000,5,0.6);tic
tnint=[0.125,0.21;0.26,0.34;0.36,0.5;0.6,0.8;0.84,0.88;0.9,1.1;1.13,1.19;1.28,1.4;1.46,1.6;1.7,1.88;
1.89,1.915;1.96,2.1;2.25,2.45;2.55,2.8;2.84,3.05;3.09,3.16;3.25,3.5];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fs=8000;
tatj=tn2(1):1/Fs:tn2(end);  %atjaunotais signals
Satj = zeros(size(tatj));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for z = 1:length(tnint(:,1))
ind = tn2>=tnint(z,1)&tn2<=tnint(z,2);  %signals
Sn = Sn2(ind);
tn = tn2(ind);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[fup,tup,tint]=fupFn(Sn,tn);%frekvences noteiksana
tup=(tint(:,1)+tint(:,2))/2;

Fs=64000; dtt=1/Fs;
tt=tn(1):dtt:tn(end);
ft = zeros(size(tt));
for ma = 25:45
    ft = ft+Aproksimacija(fup,tup',ma,tt);
end;
ft = ft/21;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ff=0; faze=ff;              %fazes noteikshana
for g=1:length(ft)-1
    ff=ff+ft(g)*1/Fs;
    faze=[faze,ff];
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tk=[]; f0=0; kk=1.45;     %atrod laika momentus tk
for g=1:length(faze)-1
    if faze(g)<=f0&&faze(g+1)>=f0
        tk=[tk,(f0-faze(g))/(faze(g+1)-faze(g))*(tt(g+1)-tt(g))+tt(g)];
        f0=f0+0.5/kk;
    end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
indz=tatj>=tn(1)&tatj<=tn(end);
tatjz=tatj(indz);
faze=2*pi*faze*kk;
fazen=interp1(tt,faze,tn,'linear','extrap');
fazez=interp1(tt,faze,tatjz,'linear','extrap');
sig=sin(faze);
sign=sin(fazen);
sigz=sin(fazez);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FF=zeros(length(tn),length(tk));
fii=0;
for k=1:length(tk)
    FF(:,k)=sinc(fazen-fii)';
    fii=fii+pi;
end;

S11=FF'*Sn';
S22=FF'*FF;
C=S22\S11;
Xn=FF*C;

[tnk,ind]=sort([tn,tk]);
on=ind>length(tn);
on=find(on);
ind=on-[1:length(on)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ds=Sn(2:end)-Sn(1:end-1);
ds(ds>0)=1;
```

```
ds(ds<0)=-1;

kk=0;
while ds(end-kk)==0
    if ds(end-kk-1)==1
        ds(end-kk:end)=(-1).^[2:kk+2];
        break;
    elseif ds(end-kk-1)==-1
        ds(end-kk:end)=(-1).^[1:kk+1];
        break;
    else
        kk=kk+1;
    end;
end;

while isempty(find(ds==0,1))<1
    for k=1:length(ds)-1
        if ds(k)==0
            if ds(k+1)==1
                ds(k)=-1;
            elseif ds(k+1)==-1
                ds(k)=1;
            end;
        end;
    end;
end;

koef=zeros(size(C));

for k=1:length(C)
    kk=find(u==Sn(ind(k)));
    uu=sort([u(kk),u(kk+ds(ind(k)))]);
    koef(k)=(C(k)-uu(1))/(uu(2)-uu(1));
end;
koef(koef>1)=1;
koef(koef<0)=0;

for k=1:length(C)
    kk=find(u==Sn(ind(k)));
    uu=sort([u(kk),u(kk+ds(ind(k)))]);
    C(k)=uu(1)+(uu(2)-uu(1))*koef(k);
end;

Fatj=zeros(length(tatjz),length(tk));
fii=0;
for k=1:length(tk)
    Fatj(:,k)=sinc(fazez-fii)';
    fii=fii+pi;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SSatj=Fatj*C;

[tnatj,ind2]=sort([tn,tatjz]);
on=ind2>length(tn);
on=find(on);
ind2=on-[1:length(on)];

bord=zeros(length(tatjz),2);
for k=1:length(tatjz)
    kk=find(u==Sn(ind2(k)));
    bord(k,:)=sort([u(kk),u(kk+ds(ind2(k)))]);

end;
du=bord(:,2)-bord(:,1);

step=0.05; dsl=0.05; step2=0.03;
p=0;
while p<length(tk)*10
    cond=SSatj>bord(:,2)+dsl*du|SSatj<bord(:,1)-dsl*du;
    g=find(cond);
    if length(g)<2
        break;
    else
        nn=ceil(length(g).*rand(1,1));
        k=g(nn);
        [aa,bb]=sort(abs(tk-tatjz(k)));
        inddt=bb(1);
        kk=find(u==Sn(ind(inddt)));
        if SSatj(k)>bord(k,2)+dsl*du(k)
            koef(inddt)=koef(inddt)*(1-step);
```

```
        else
            koef(inddt)=koef(inddt)*(1-step)+step;
        end;
        uu=sort([u(kk),u(kk+ds(ind(inddt)))]);
        C(inddt)=uu(1)+(uu(2)-uu(1))*koef(inddt);
        inddt=bb(2);
        kk=find(u==Sn(ind(inddt)));
        if SSatj(k)>bord(k,2)+dsl*du(k)
            koef(inddt)=koef(inddt)*(1-step2);
        else
            koef(inddt)=koef(inddt)*(1-step2)+step2;
        end;
        uu=sort([u(kk),u(kk+ds(ind(inddt)))]);
        C(inddt)=uu(1)+(uu(2)-uu(1))*koef(inddt);
        SSatj=Fatj*C;
        p=p+1;
    end;
end;

cond=SSatj>bord(:,2)+dsl*du|SSatj<bord(:,1)-dsl*du;

Satj(indz)=SSatj;

inda=t>=tn(1)&t<=tn(end);
aha=a(inda);
an=interp1(t(inda),aha,tatjz,'linear','extrap');

kluda = sqrt(1/length(an)*sum((an-SSatj').^2))

end; toc
figure;
plot(t,a,tn2,Sn2,'.',tatj,Satj,tk,C,'o');
hold on
plot(tatj,Satj,'+k');
```

No. 4.
2. Pielikums

```
#include "msp430x22x4.h"
#include "nRF_deff.h"

//volatile unsigned int i;
unsigned char ADC[576];
int a=64, ready=0, i=0, RxMode=0, TxMode=0,readyTX=3,count=0, lim=576, c, b, skait=0, rob=0;


 unsigned char CSN_Hi=0x02, CSN_Lo=0x02;


//Deffinitions for nRF config register write**********************************

unsigned char nRF_addrTX1[5] = {0xE7,0xE7,0xE7,0xE7,0xE7};
/
unsigned char Command = 0x00;
unsigned char Dati = 0x00;
//*****************************************************************************

//  unsigned char volatile *CSN;
//  CSN = &P2OUT;
//  unsigned char Hi=0x02, Lo=0x00;



// Function prototipes*********************************************************
void nRF_Config (void);
void nRF_Config_addr (unsigned char *);
void Flush_Tx_buff (void);
void Flush_Rx_buff (void);
void Clear_IRQTX (void);
void Clear_IRQRX (void);
void LoadAndSendData (void);
void ReadData (void);
void GetStartCommand (void);
void Set_TX_Mode (void);
void Set_RX_Mode (void);
void StartSampling (void);
void UART_Send (void);
void Read_nRF24L01 (void);
//*****************************************************************************

void main(void)
{



  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  BCSCTL1 |= XTS + DIVA_1;                  // LFXT1 = HF XTAL
  BCSCTL3 |= LFXT1S_2;                      // LFXT1S1 = 3-16Mhz
  do
  {
    IFG1 &= ~OFIFG;                         // Clear OSC fault flag
    i = 0xFF;                               // i = Delay
    while (i--);                            // Additional delay to ensure start
  }
  while (OFIFG & IFG1);                     // OSC fault flag set?
  BCSCTL2 |= SELM_3 + DIVM_1  + SELS  + DIVS_3;  // MCLK = LFXT1   DIVM_1 master clock devider 8MHz


  P1DIR |= 0x01;                            // Set P1.0 output direction
  P1SEL |= 0x01;                            // Set P1.0 option select
  P2DIR = 0x26;                              // P2.0,2 output
  P2REN |= 0x01;                              // resistor enabled


//*****************************************************************************
// for timer B
  P4SEL |= 0x07;                          // P4.x - P4.2 option select
  P4DIR |= 0x07;                          // P4.x = outputs
//*****************************************************************************
```

```
  P3SEL |= 0xE;                             // P3.0,4,5 USCI_B0 option select
  UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC;   // 3-pin, 8-bit SPI master  UCCKPH +
  UCB0CTL1 |= UCSSEL_1;                     //1= ACLK  2=SMCLK
  UCB0BR0 |= 0x02;
  UCB0BR1 = 0;
  UCA0MCTL = 0;
  UCB0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine*
//  IE2 |= UCB0RXIE;

//*********************************************************************
//UART configuracija

  P3SEL |= 0x30;                // P3.4, 3.5 = USCI_A0 TXD/RXD
  UCA0CTL1 = UCSSEL_2;          // SMCLK for UART
  UCA0BR0 = 208;                //8MHz 9600==>65,  4MHz 9600==>161   2MHz 9600==> 208
  UCA0BR1 = 0;                  //8MHz 9600==>3,  4MHz 9600==>1      2MHz 9600==>
  UCA0MCTL = UCBRS0;            // Modulation UCBRSx = 1
  IE2 |= UCA0RXIE;              // Enable USCI_A0 RX interrupt

//*********************************************************************


  for (i = 100; i; i--);


  Command=WRITE_REG|EN_AA;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|EN_RXADDR;
  Dati=0x01;
  nRF_Config();
  Command=WRITE_REG|SETUP_AW;
  Dati=0x03;
  nRF_Config();
  Command=WRITE_REG|SETUP_RETR;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|RF_CH;
  Dati=0x50;
  nRF_Config();
  Command=WRITE_REG|RF_SETUP;
  Dati=0x0F;                        // 0x0F 2Mb/s   0x07 1Mb/s
  nRF_Config();
  Command=WRITE_REG|STATUS;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|RX_ADDR_P0;
  nRF_Config_addr(nRF_addrTX1);
  Command=WRITE_REG|TX_ADDR;
  nRF_Config_addr(nRF_addrTX1);
  Command=WRITE_REG|RX_PW_P0;
  Dati=0x01;                        //bija 0x20 testam meginu ar 16bitu garu start paku...
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x4E;
  nRF_Config();
  TxMode=1;

  for (i = 100; i; i--);


  P2OUT = 0x02;   //for TX


//***********************************************************
// Timer_B settings

  TBCCTL0 = OUTMOD_2 + CCIE;
  TBCCTL1 = OUTMOD_2 + CCIE;              // TBCCR1 interrupt enabled
  TBCTL = TBSSEL_1;//  + TBIE;            // ACLK (8MHz)
  TBCCR1 = 12000;                       //
  TBCCR0 = 65000;

//***********************************************************

  __bis_SR_register(GIE);


  while(1)
```

```
  {

    if (ready==1)
    {
     ready=0;
//      TBCTL |= MC_1;                          // Start Timer_B, up mode
     skait=0;
     rob=32;
     Read_nRF24L01();
     skait=32;
     rob=64;
     Read_nRF24L01();
     skait=64;
     rob=96;
     Read_nRF24L01();
     skait=96;
     rob=128;
     Read_nRF24L01();
     skait=128;
     rob=160;
     Read_nRF24L01();
     skait=160;
     rob=192;
     Read_nRF24L01();
     skait=192;
     rob=224;
     Read_nRF24L01();
     skait=224;
     rob=256;
     Read_nRF24L01();
     skait=256;
     rob=288;
     Read_nRF24L01();
     skait=288;
     rob=320;
     Read_nRF24L01();
     skait=320;
     rob=352;
     Read_nRF24L01();
     skait=352;
     rob=384;
     Read_nRF24L01();
     skait=384;
     rob=416;
     Read_nRF24L01();
     skait=416;
     rob=448;
     Read_nRF24L01();
     skait=448;
     rob=480;
     Read_nRF24L01();
     skait=480;
     rob=512;
     Read_nRF24L01();
     skait=512;
     rob=544;
     Read_nRF24L01();
     skait=544;
     rob=576;
     Read_nRF24L01();
     lim=576;
    UART_Send();
    Set_TX_Mode();
    }

    }
  }

//*********************************************************************
// UART sanem komandu lai palaistu nRF24L01

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  if (UCA0RXBUF==57) //ja atsutits 9 (acsii 57),iet talak3
  {
    Set_TX_Mode();
    LoadAndSendData();
    Flush_Tx_buff();
```

```
    Clear_IRQTX();
    Set_RX_Mode();
    ready=1;
   }
}

void UART_Send (void)
{
   while (count<lim)                    //sutamo elementu skaits
    {
//       ptr_1 = &ascii_el_1[ ADC[count]];
       while (!(IFG2 & UCA0TXIFG));
       UCA0TXBUF = ADC[count];
//       UCA0TXBUF = *ptr_1;
//       ptr = &ascii_el[ADC[count]];
//       while (!(IFG2 & UCA0TXIFG));
//       UCA0TXBUF = *ptr;
       count=count+1;
    }
    if (count==lim)
    {
   //   while (!(IFG2 & UCA0TXIFG));
    //  UCA0TXBUF = 10; //newline
    //  while (!(IFG2 & UCA0TXIFG));
    //  UCA0TXBUF = 13; //carriage return
      count=0;

    }
}


#pragma vector=TIMERB0_VECTOR
__interrupt void TB0_ISR(void)
{
//      P2OUT |= 0x20;
//      TBCTL &= ~MC_1;

//      Set_TX_Mode();
//      P2OUT &= ~0x20;
}

void Read_nRF24L01(void)
{
     while ((0x01 & P2IN));                    // Data received??
     P2OUT |= 0x20;
     ReadData();
     Flush_Rx_buff();
     Clear_IRQRX();
     P2OUT &= ~0x20;
}



void nRF_Config_addr (unsigned char* addr)
{
   P2OUT &= ~CSN_Lo;
   UCB0TXBUF = Command;
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[0];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[1];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[2];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[3];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[4];
   while (!(IFG2 & UCB0TXIFG));
   for (i = 1; i; i--);
   P2OUT |= CSN_Hi;
}


void nRF_Config (void)
{
   P2OUT &= ~CSN_Lo;
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = Command;
   while (!(IFG2 & UCB0TXIFG));
```

```
  UCB0TXBUF = Dati;
  while (!(IFG2 & UCB0TXIFG));
  for (i = 1; i; i--);
  P2OUT |= CSN_Hi;
}


void ReadData (void)
{
//    IFG2 &= ~UCB0RXIFG;
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=RD_RX_PLOAD;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x00;
    for(i = skait; i < rob; i++){
    UCB0TXBUF=0x00;
    while (!(IFG2 & UCB0RXIFG));
    ADC[i] = UCB0RXBUF;

                        }
    P2OUT |= CSN_Hi;

}


void LoadAndSendData (void)
{
//***Start Data upload to nRF*****************************************
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WR_TX_PLOAD;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0xAB;
    while (!(IFG2 & UCB0TXIFG));


//***End of Data upload to nRF*****************************************

    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
    P2OUT |= 0x04;
    for (i = 1; i; i--);                    // ensures at least 10us CSN high
    P2OUT = 0x02;
    while ((0x01 & P2IN));                  // ACK received or overflow??
}

void GetStartCommand (void)
{
    while ((0x01 & P2IN));                  // Data received??
//    ReadData();
    P2OUT &= ~0x20;
    Flush_Rx_buff();
    Clear_IRQRX();
    P2OUT |= 0x20;
}


void Set_TX_Mode (void)
{
  Command=WRITE_REG|RX_PW_P0;               // set packet size to 32 bytes
  Dati=0x01;                                //******************************
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x4E;
  nRF_Config();
  P2OUT = 0x02;
  RxMode=0;
}


void Set_RX_Mode (void)
{
  Command=WRITE_REG|RX_PW_P0;               // set packet size to 32 bytes
  Dati=0x20;                                //******************************
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x3F;
  nRF_Config();
```

```
  P2OUT = 0x06;
  RxMode=1;
}


void Flush_Tx_buff (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=FLUSH_TX;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}

void Flush_Rx_buff (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=FLUSH_RX;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}


void Clear_IRQTX (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WRITE_REG|0x07;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x30;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}

void Clear_IRQRX (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WRITE_REG|0x07;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x40;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}
```

Nod. 4. 1. Pielikums

```
#include "msp430x22x4.h"
#include "nRF_deff.h"

//volatile unsigned int i;
unsigned int ADC[96];
unsigned int a=64, ready=0, i=0, RxMode=0, TxMode=0, F0=100,
F1=6000,F2=12000,F3=18000,F4=24000,F5=30000,F6=36000,F7=42000,F8=48000,F9=54000,skait=0, rob=0;

  unsigned char CSN_Hi=0x02, CSN_Lo=0x02, RXbuf=0;


//Deffinitions for nRF config register write***********************************
unsigned char nRF_addrP0[5] =  {0xE7,0xE7,0xE7,0xE7,0xE7};
unsigned char nRF_addrTX1[5] = {0xE7,0xE7,0xE7,0xE7,0xE7};
unsigned char nRF_addrTX2[5] = {0xC2,0xC2,0xC2,0xC2,0xC2};
unsigned char nRF_addrTX3[5] = {0xF1,0xC2,0xC2,0xC2,0xC2};
unsigned char nRF_addrTX4[5] = {0xCD,0xC2,0xC2,0xC2,0xC2};
unsigned char nRF_addrTX5[5] = {0xA3,0xC2,0xC2,0xC2,0xC2};
unsigned char nRF_addrTX6[5] = {0x05,0xC2,0xC2,0xC2,0xC2};
unsigned char Command = 0x00;
unsigned char Dati = 0x00;
//*****************************************************************************

//  unsigned char volatile *CSN;
//  CSN = &P2OUT;
//  unsigned char Hi=0x02, Lo=0x00;



// Function prototipes*********************************************************
void nRF_Config (void);
void nRF_Config_addr (unsigned char *);
void Flush_Tx_buff (void);
void Flush_Rx_buff (void);
void Clear_IRQTX (void);
void Clear_IRQRX (void);
void LoadAndSendData (void);
void LoadAndSendDataN (void);
void GetStartCommand (void);
void ReadData (void);
void Set_TX_Mode (void);
void Set_RX_Mode (void);
void StartSampling (void);
//*****************************************************************************

void main(void)
{

  WDTCTL = WDTPW + WDTHOLD;                   // Stop WDT
  BCSCTL1 |= XTS + DIVA_1;                    // LFXT1 = HF XTAL
  BCSCTL3 |= LFXT1S_2;                        // LFXT1S1 = 3-16Mhz
  do
  {
    IFG1 &= ~OFIFG;                           // Clear OSC fault flag
    i = 0xFF;                                 // i = Delay
    while (i--);                              // Additional delay to ensure start
  }
  while (OFIFG & IFG1);                       // OSC fault flag set?
  BCSCTL2 |= SELM_3 + DIVM_1  + SELS  + DIVS_3;   // MCLK = LFXT1   DIVM_1 master clock devider 8MHz
  ADC10CTL1 = INCH_3 + SHS_1 + CONSEQ_2;            // TA1 trigger sample start
  ADC10CTL0 = SREF_1 + ADC10SHT_2 + REFON + REFOUT + REF2_5V + ADC10ON + ADC10IE;
  TACCR0 = 30;                              // Delay to allow Ref to settle
  TACCTL0 |= CCIE;                          // Compare-mode interrupt
  TACTL = TASSEL_2 + MC_1;                  // TACLK = SMCLK, Up mode
  __bis_SR_register(CPUOFF + GIE);          // LPM0, TA0_ISR will force exit
  TACCTL0 &= ~CCIE;                         // Disable timer Interrupt
  ADC10CTL0 |= ENC;                         // ADC10 Enable
//  ADC10AE0 |= 0x80;                         // P2.0 ADC10 option select

//*****************************************************************************
// opinu config

  OA0CTL0 = OAP_2 + OAPM_3;// + OAADC0;  //P3.6 input signal
  OA0CTL1 = OAFBR_0 + OAFC_7;

  OA1CTL0 = OAN_3 + OAP_2 + OAPM_2 + OAADC1;// + OAADC0;  //P3.7 input signal
```

```
  OA1CTL1 = OAFBR_2 + OAFC_6;    // gain 1,66
//*****************************************************************************


  P1DIR |= 0x01;                            // Set P1.0 output direction
  P1SEL |= 0x01;                            // Set P1.0 option select
  P2DIR = 0x26;                              // P2.0,2 output
  P2REN |= 0x01;                             // resistor enabled


//*****************************************************************************
// for timer B
//  P4SEL |= 0x07;                      // P4.x - P4.2 option select
//  P4DIR |= 0x07;                      // P4.x = outputs
//*****************************************************************************


  P3SEL |= 0xE;                             // P3.0,4,5 USCI_A0 option select
  UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC;           // 3-pin, 8-bit SPI master
  UCB0CTL1 |= UCSSEL_1;                             //1= ACLK  2=SMCLK
  UCB0BR0 |= 0x02;
  UCB0BR1 = 0;
  UCA0MCTL = 0;
  UCB0CTL1 &= ~UCSWRST;                             // **Initialize USCI state machine*


  for (i = 100; i; i--);

  Command=WRITE_REG|EN_AA;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|EN_RXADDR;
  Dati=0x01;
  nRF_Config();
  Command=WRITE_REG|SETUP_AW;
  Dati=0x03;
  nRF_Config();
  Command=WRITE_REG|SETUP_RETR;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|RF_CH;
  Dati=0x50;
  nRF_Config();
  Command=WRITE_REG|RF_SETUP;
  Dati=0x0F;                        // 0x0F 2Mb/s   0x07 1Mb/s
  nRF_Config();
  Command=WRITE_REG|STATUS;
  Dati=0x00;
  nRF_Config();
  Command=WRITE_REG|RX_ADDR_P0;
//  Dati=0x9A;                        // only for testing TX to FTDI RX
//  nRF_Config();                     // only for testing TX to FTDI RX
  nRF_Config_addr(nRF_addrTX1);
  Command=WRITE_REG|TX_ADDR;
//  Dati=0x9A;                        // only for testing TX to FTDI RX
//  nRF_Config();                     // only for testing TX to FTDI RX
  nRF_Config_addr(nRF_addrTX1);
  Command=WRITE_REG|RX_PW_P0;
  Dati=0x01;                        //bija 0x20 testam meginu ar 16bitu garu start paku...
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x3F;
  nRF_Config();
  RxMode=1;

  for (i = 100; i; i--);

  P2OUT = 0x06;

  TACCR0 = 8001-1;                  // PWM Period
  TACCTL1 = OUTMOD_3;                       // TACCR1 set/reset
  TACCR1 = 8000;                    // TACCR1 PWM Duty Cycle
  TACTL = TASSEL_1;                     // ACLK(8MHz), up mode
  ADC10DTC1 = 0x60;                     // 32 conversions
  ADC10SA = (unsigned int )ADC;

//*****************************************************************
// Timer_B settings
```

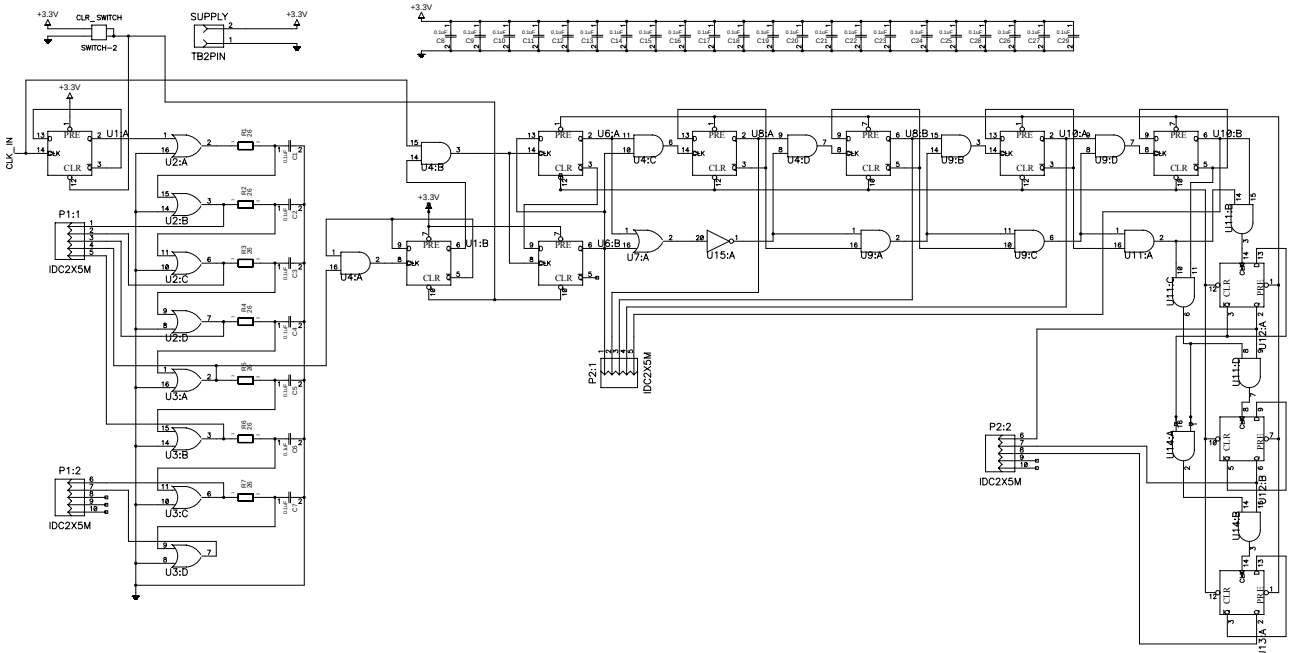
```
  TBCCTL0 = OUTMOD_2 + CCIE;
  TBCCTL1 = OUTMOD_2 + CCIE;                // TBCCR1 interrupt enabled
  TBCTL = TBSSEL_2;//  + TBIE;                 // SMCLK (2MHz)
  TBCCR1 = F7;                          //
  TBCCR0 = 65535;


//*********************************************************


//  __bis_SR_register(LPM3_bits + GIE);       // Enter LPM3 w/ interrupts


  while(1)
  {

    if (RxMode==1)
    {
      GetStartCommand();
//      if(RXbuf==0xAB){
      StartSampling();
//      Clear_IRQRX();
      RxMode=0;
      Set_TX_Mode();
//                    }

    }

    if (ready==1)
    {

//      LoadAndSendData();
//      Flush_Tx_buff();
//      Clear_IRQTX();
//      ready =0;
//      Set_RX_Mode();
    }
  }

}

// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    TACTL &= ~MC_1;
    P2OUT &= ~0x20;
    ADC10SA = (unsigned int )ADC;             // Data buffer start
    ready=1;
    P2OUT |= 0x20;
    RxMode=0;
    TBCTL |= MC_1;                          // ACLK, up mode
//    TBCCTL1 |= CCIE;
//    TBCCTL0 |= CCIE;
}


#pragma vector=TIMERA0_VECTOR
__interrupt void TA0_ISR(void)
{

  TACTL = 0;                            // Clear Timer_A control registers

  __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
}

// Common ISR for TBCCR1-2 and overflow
#pragma vector=TIMERB1_VECTOR
__interrupt void TBX_ISR(void)
{
  switch (TBIV)       // Efficient switch-implementation
  {
    case  2:                            // TBCCR1
//      TBCCR1 += 400;                      // Offset until next interrupt
      P2OUT &= ~0x20;
      LoadAndSendData();
      Flush_Tx_buff();
      Clear_IRQTX();
      skait=31;
```

```
      rob=63;
      LoadAndSendDataN();
      Flush_Tx_buff();
      Clear_IRQTX();
      skait=63;
      rob=95;
      LoadAndSendDataN();
      Flush_Tx_buff();
      Clear_IRQTX();
      ready =0;
      P2OUT |= 0x20;
      break;
    case  4:                          // TBCCR2

      break;
    case 14:                           // Overflow

      break;
  }
}

#pragma vector=TIMERB0_VECTOR
__interrupt void TB0_ISR(void)
{
      TBCTL &= ~MC_1;
//      TBCCTL0 &= ~CCIE;                   // Timer_B int ieslegshana
      Set_RX_Mode();

}

void nRF_Config_addr (unsigned char* addr)
{
   P2OUT &= ~CSN_Lo;
   UCB0TXBUF = Command;
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[0];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[1];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[2];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[3];
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = addr[4];
   while (!(IFG2 & UCB0TXIFG));
   for (i = 1; i; i--);
   P2OUT |= CSN_Hi;
}


void nRF_Config (void)
{
   P2OUT &= ~CSN_Lo;
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = Command;
   while (!(IFG2 & UCB0TXIFG));
   UCB0TXBUF = Dati;
   while (!(IFG2 & UCB0TXIFG));
   for (i = 1; i; i--);
   P2OUT |= CSN_Hi;
}

void LoadAndSendDataN (void)
{
  //***Start Data upload to nRF*****************************************
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WR_TX_PLOAD;
    while (!(IFG2 & UCB0TXIFG));
    for(i = skait; i < rob; i++){
    ADC[i]>>=2;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=ADC[i];
                        }
//***End of Data upload to nRF*****************************************

    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
//    for (i = 2; i; i--);                        //delay to CSN
```

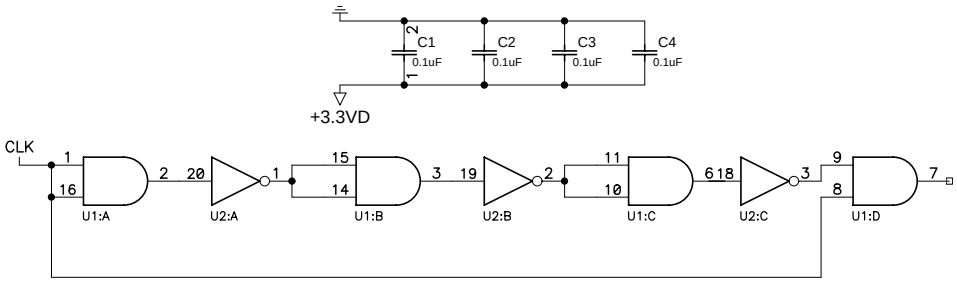
```
    P2OUT |= 0x04;
    for (i = 1; i; i--);                    // ensures at least 10us CSN high
    P2OUT = 0x02;
//    for (i = 20; i; i--);                 // bija 100 cikli
    while ((0x01 & P2IN));                  // ACK received or overflow??
}


void LoadAndSendData (void)
{
  //***Start Data upload to nRF*****************************************
    P2OUT &= ~CSN_Lo;

    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WR_TX_PLOAD;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0xF7;
    for(i = 0; i < 31; i++){
    ADC[i]>>=2;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=ADC[i];
                    }
//***End of Data upload to nRF*****************************************

    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
//    for (i = 2; i; i--);                  //delay to CSN
    P2OUT |= 0x04;
    for (i = 1; i; i--);                    // ensures at least 10us CSN high
    P2OUT = 0x02;

    while ((0x01 & P2IN));                  // ACK received or overflow??
}

void GetStartCommand (void)
{
    while ((0x01 & P2IN));                  // Data received??
//    ReadData();
    P2OUT &= ~0x20;
    Flush_Rx_buff();
    Clear_IRQRX();
    P2OUT |= 0x20;
}

void ReadData (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=RD_RX_PLOAD;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x00;
//    while (!(IFG2 & UCB0RXIFG));
    RXbuf = UCB0RXBUF;

    P2OUT |= CSN_Hi;

}
void Set_TX_Mode (void)
{
  Command=WRITE_REG|RX_PW_P0;
  Dati=0x20;
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x4E;
  nRF_Config();
  P2OUT = 0x02;
  RxMode=0;
}


void Set_RX_Mode (void)
{
  Command=WRITE_REG|RX_PW_P0;
  Dati=0x01;
  nRF_Config();
  Command=WRITE_REG|CONFIG;
  Dati=0x3F;
  nRF_Config();
  P2OUT = 0x06;
```

```
  RxMode=1;
}

void StartSampling (void)
{

  TACTL |= MC_1;                        // ACLK, up mode
  P2OUT &= ~0x20;
  P2OUT |= 0x20;
}


void Flush_Tx_buff (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=FLUSH_TX;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}

void Flush_Rx_buff (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=FLUSH_RX;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}


void Clear_IRQTX (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WRITE_REG|0x07;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x30;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}

void Clear_IRQRX (void)
{
    P2OUT &= ~CSN_Lo;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=WRITE_REG|0x07;
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF=0x40;
    while (!(IFG2 & UCB0TXIFG));
    for (i = 1; i; i--);
    P2OUT |= CSN_Hi;
}
```

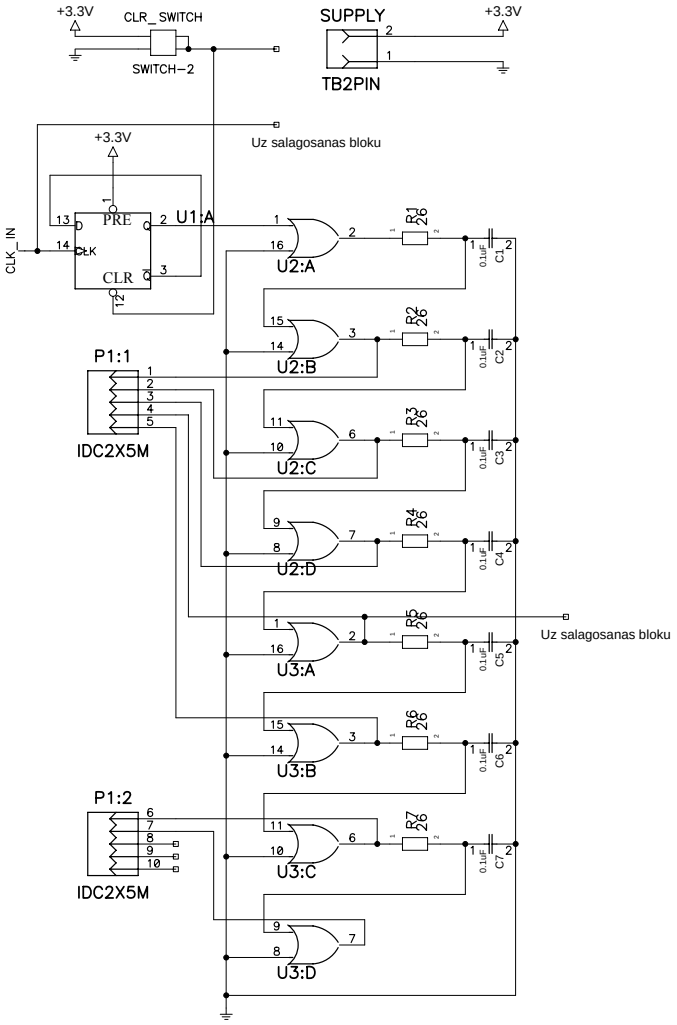
Nod. 1
4. Pielikums
4.1. Shēmas



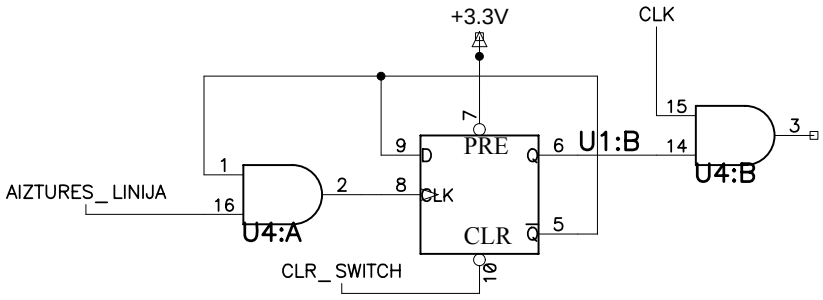
6. zīm. Aiztures līnijas un Greja koda skaitītāja kopēja shēma.



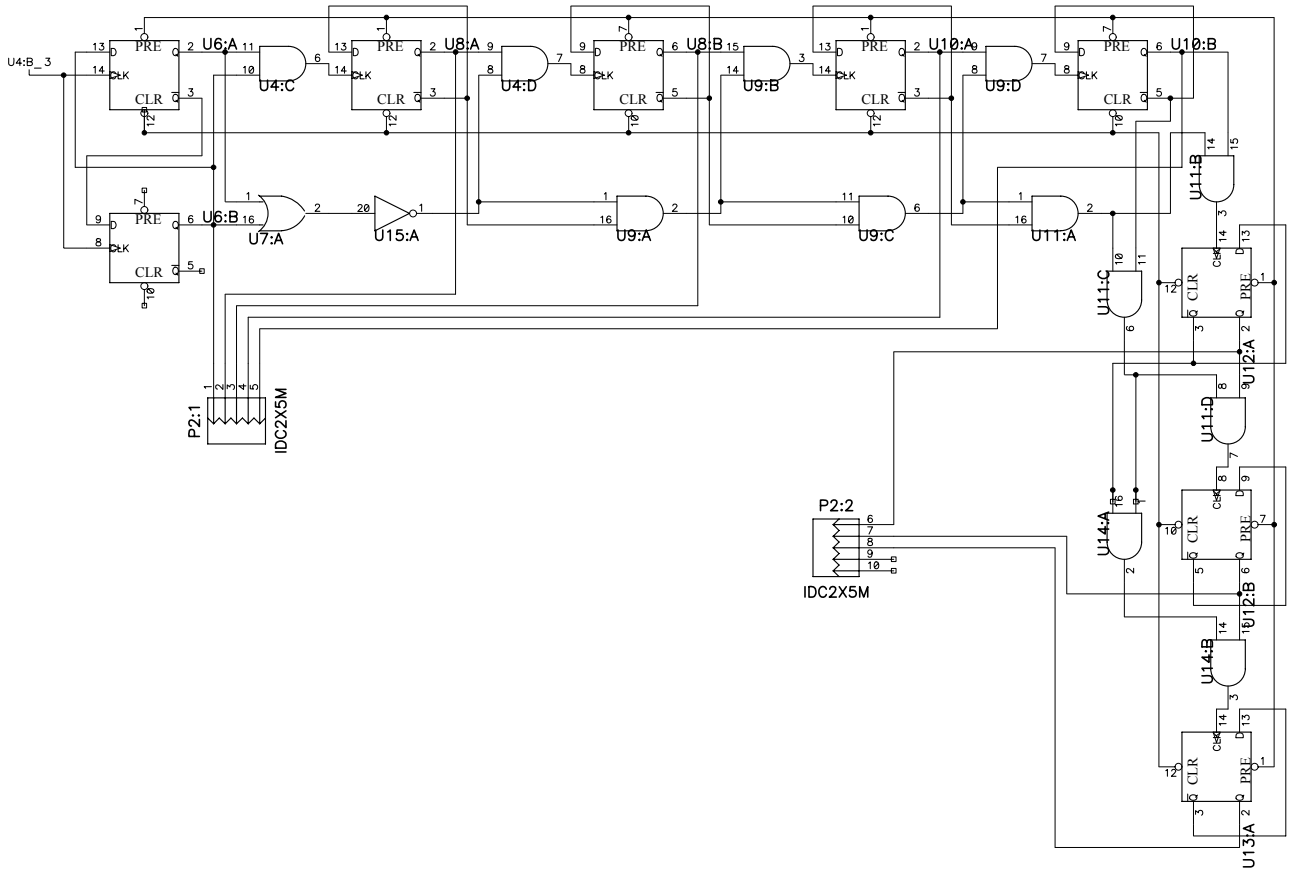
7. Impulsu formētājs



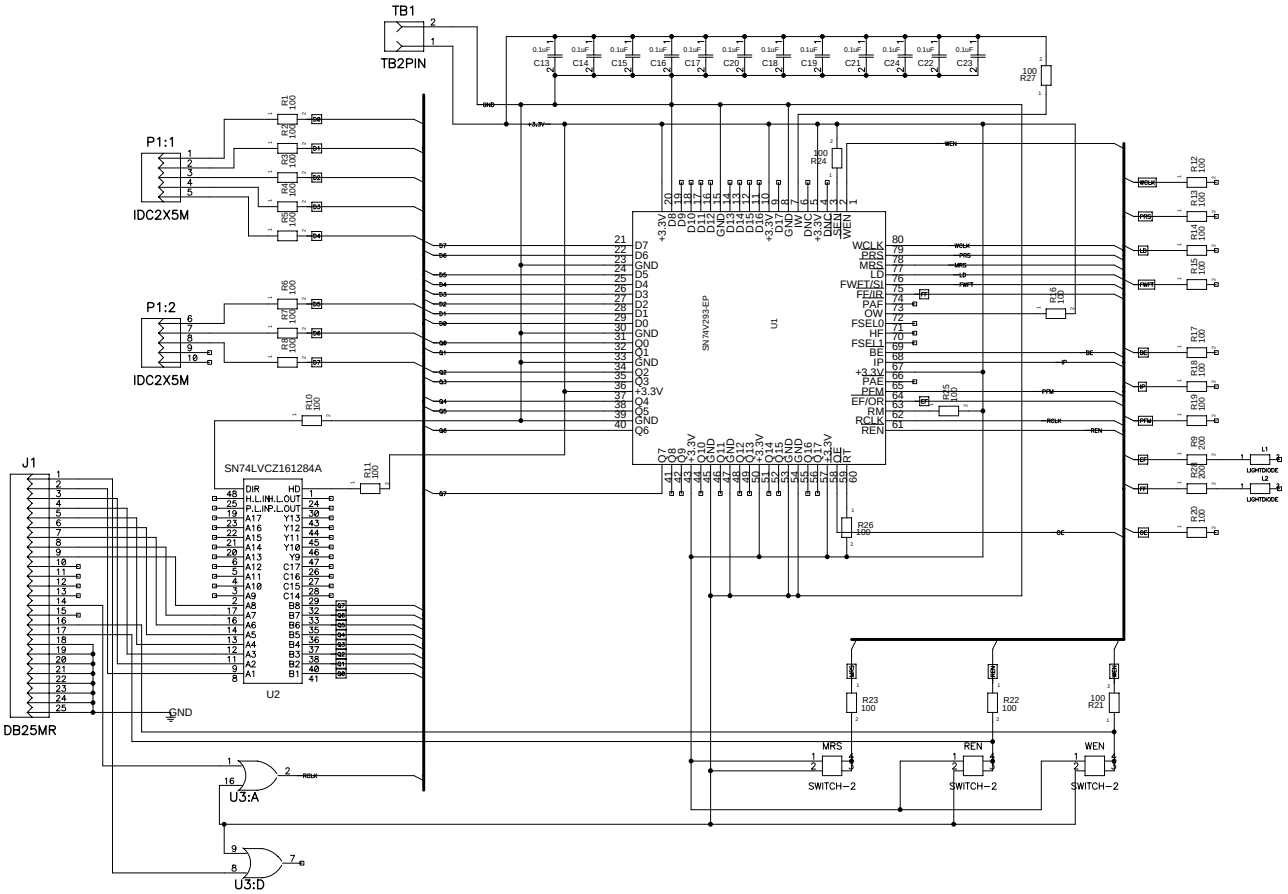
8. Aiztures līnija



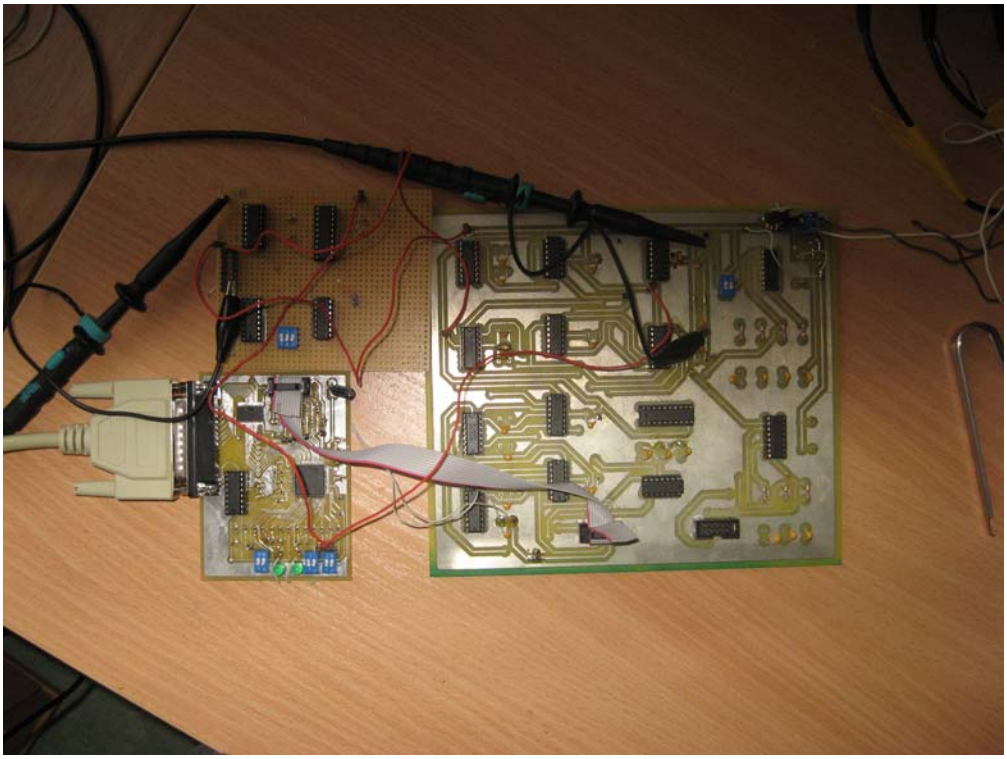
9. Skalu salāgošanas bloks



10. Greja koda skaitītājs.



11. FIFO atmiņas saslēgums.

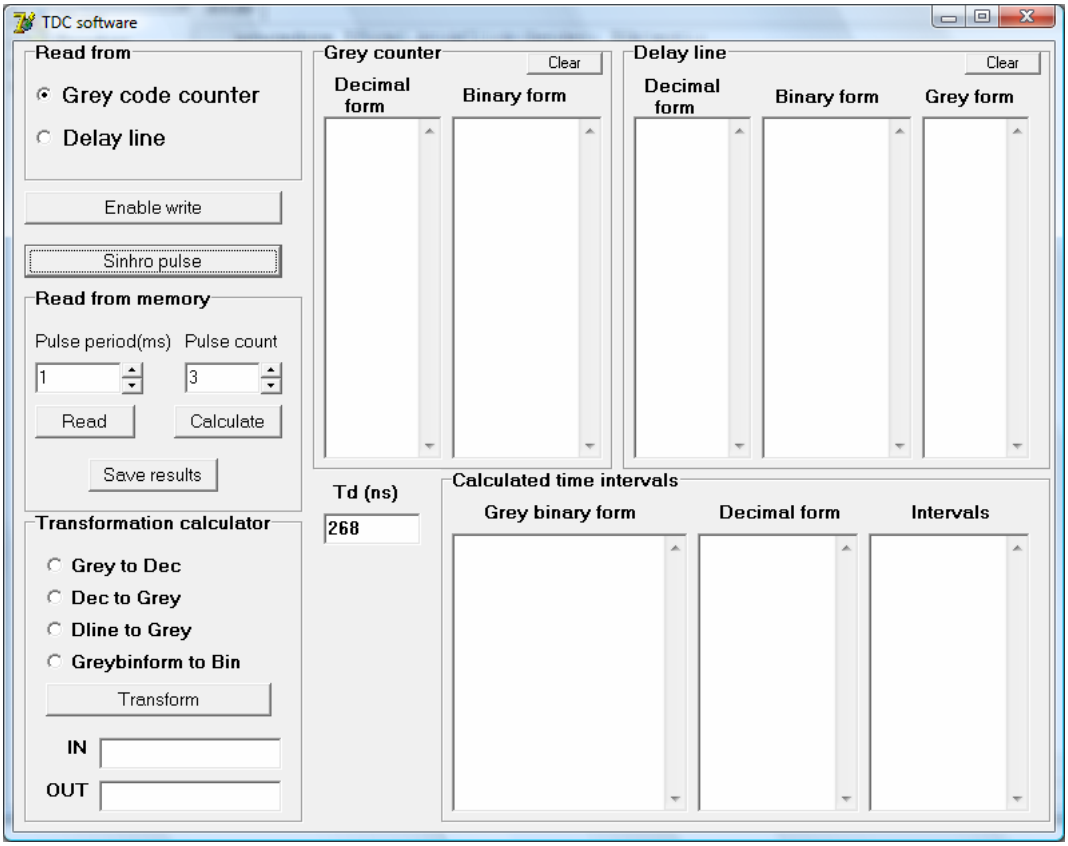


12. Laiks-cipars pārveidotāja makets

4.2. Laiks-cipars pārveidotāja programma „TDC software”

4.2.1. Lietotāja rokasgrāmata

Programma laiks-cipars pārveidotajam tika izstrādāta Delphi 6.0 vidē. Tā kā Windows XP neļauj tieši piekļūt LPT portam, tika izmantots speciāls draiveris LPT portam inpout32.dll, kas ir pieejams Internetā. Palaižot programmu TDC software, parādās programmas galvenais logs (12. zīm.).



13. Laiks-cipars pārveidotāja programmas logs.

1. Laika intervālu mērīšanas sākumā izvēlas, no kurienes tiks lasīti dati (*Read from*) – vai no Greja koda skaitītāja (*Grey code counter*), vai no aiztures līnijas (*Delay line*).
2. Nākamais posms – ir jānospiež pogu *Enable write*, kas atļauj ierakstu atmiņā (uz FIFO atmiņas WEN izvadu tiek padota loģiska 0).
3. Pēc 2. posma atmiņa laiks-cipars pārveidotājs ir gatavs darbam. Tā kā pārveidotāja testēšanas laikā par takts impulsu un reģistrējamu impulsu avotu kalpoja signālu ģenerators, nospiežot pogu *Sinhro pulse*, uz signāla ģeneratora sinhronizācijas ieeju tiek padots impulss, kas atbloķē ģeneratora izejas. Gadījumā, ja tiek izmantots cits neatkarīgs impulsu avots, šī poga nav vajadzīga.
4. Kad reģistrējami impulsi tika piereģistrēti atmiņā, tos ir jānolasa programmā. Tekstā logā *Pulse period*, var ierakstīt nolases impulsu periodu milisekundēs, minimālā vērtība ir 1ms. Tekstā logā *Pulse count* tiek norādīts nolases impulsu skaits. Nospiežot pogu *Read* vērtības no atmiņas tiks nolasītas programmas atmiņā. Atkarībā, no kurienes tika lasīti dati (Greja koda skaitītājs vai aiztures līnijas), nolasītie dati tiks atspoguļoti attiecīgajā sekcijā *(Grey couner* vai *Delay line*).

5. Kad ir nolasīti visi dati no Greja koda skaitītāja un aiztures līnijas, var aprēķināt laika intervālus, nospiežot pogu *Calculate*. Rezultāti tiks atspoguļoti sekcijā *Calculates time intervals*. Teksta logā *Td (ns)* alternatīvas aiztures līnijas izmantošanas gadījumā var izmainīt laiks-cipars pārveidotāja izšķirtspēju.
6. Programmā ir papildus funkcija - iespēja konvertēt dažas vērtības:
   - *Grey to Dec* – no Greja koda decimālajā
   - *Dec to Grey* – no decimālā koda Greja kodā.
   - *Dline to Grey* – aiztures līnijas vērtību pārveidošana Greja kodā
   - *Greybinform to Bin* – Greja koda binārās formas pārveidošana binārā kodā.

   Tekstā logā *IN* tiek ievadīti dati, un pēc pogas *Transform* nospiešanas pārveidošanas rezultāts tiks attēlots tekstā logā *OUT*.
7. Programmā ir iespēja saglabāt rezultātus tekstā failā. To var izdarīt, nospiežot pogu *Save*.

4.2.1. Programmas kods

```
var
  Form1: TForm1;
  RCLKcnt : Integer;
  Grey, Dline : Array of Integer;
  intervals : Array of Single;
  Greybin, Dlinebin : Array of String;

implementation
{$R *.dfm}

//-----------------------------------------------------------------------------------------------------------------
Pārveidošanas funkcijas//
function BinToInt(Value: string) : Integer; Pārveidošana no binārās formas decimālā.
var
   i, iValueSize: Integer;
begin
  Result := 0;
  iValueSize := Length(Value);
  for i := iValueSize downto 1 do
    if Value[i] = '1' then Result := Result + (1 shl (iValueSize - i));
end;

function IntToBin (int : Integer; dig : Integer) : String;  Pārveidošana no decimālās binārā.
var
  i : Integer;
begin
    Result := '';
    for i := dig-1 downto 0 do
      if int and (1 shl i) <> 0 then
        Result := Result + '1'
      else Result := Result + '0';
end;

function GreyToDec (value : integer; dig : Integer) : Integer;  Pārveidošana no Greja koda decimālā.
var
  i, t, j : Integer;
  binary, binaryT : Array of Integer;
  binstring : string;
begin
  SetLength (binary, dig);
  SetLength (binaryT, dig);
  binstring := IntToBin(value,dig); //Decimal Grey - Binary Grei
```

```
//-----Binary string - banary array
  t := dig-1;
  For i := 0 to dig -1 do begin
    binaryT [t] := StrToInt (binstring[i+1]);
    dec (t);
  end;
//----Binary Grei - Binary decimal
  for i := 0 to dig-1 do begin
    binary[i] := binaryT[i];
    for j := i+1 to dig-1 do
      binary[i] := binary[i] XOR binaryT[j];
  end;
//----Binary decimal - decimal value
  binstring := '';
    for i := dig-1 downto 0 do //
      binstring := binstring + inttostr (binary[i]);
  Result := BinToInt(binstring);
end;
```

function DecToGrey (value : Integer; dig : Integer) : Integer; *Pārveidošana no decimālā koda Greja kodā*

```
var
  binstring : String;
  t, i, j : Integer;
  binary, binaryT : Array of Integer;
begin
  SetLength (binary, dig);
  SetLength (binaryT, dig);
  binstring := IntToBin(value,dig); //Decimal - Binary
//-----Binary string - banary array
  t := dig -1;
  For i := 0 to dig -1 do begin
    binaryT [t] := StrToInt (binstring[i+1]);
    dec (t);
  end;
//----Binary - Binary Grey
  for i := 0 to dig-2 do begin
    binary[i] := binaryT[i] xor binaryT[i+1];
  end;
  binary[dig-1] := binaryT[dig-1];
//----Binary decimal - decimal value
  binstring := '';
    for i := dig-1 downto 0 do //
      binstring := binstring + inttostr (binary[i]);
  Result := BinToInt(binstring);
end;
```

function GreybinToBin (value : string; dig : integer) : string; *Pārveidošana no Greja koda bināras formas binārajā kodā*

```
var
  t, i, j : integer;
  binary, binaryT : Array of Integer;
begin
  SetLength (binary, dig);
  SetLength (binaryT, dig);
  t := dig-1;
  For i := 0 to dig -1 do begin
    binaryT [t] := StrToInt (value[i+1]);
    dec (t);
  end;
  for i := 0 to dig-1 do begin
    binary[i] := binaryT[i];
    for j := i+1 to dig-1 do
      binary[i] := binary[i] XOR binaryT[j];
```

```
  end;
  result := '';
  for i := dig-1 downto 0 do
    result := result + inttostr (binary[i]);
end;

function BinTogreyBin (value : string; dig : integer) : string; Binārā koda pārveidošana  Greja koda binārā formā.
var
  t, i, j : integer;
  binary, binaryT : Array of Integer;
begin
  SetLength (binary, dig);
  SetLength (binaryT, dig);
  t := dig-1;
  For i := 0 to dig -1 do begin
    binaryT [t] := StrToInt (value[i+1]);
    dec (t);
  end;
  for i := 0 to dig-2 do begin
    binary[i] := binaryT[i] xor binaryT[i+1];
  end;
  binary[dig-1] := binaryT[dig-1];

  result := '';
    for i := dig-1 downto 0 do
      result := result + inttostr (binary[i]);
end;

function DlineToGrey (value : string; dig : Integer): string;   Aiztures līnijas vērtību pārveidošana Greja kodā.
var
  binstring : String;
  t, i, j : Integer;
  binary, binaryT : Array of Integer;
begin
  SetLength (binary, dig);
  SetLength (binaryT, dig);
//-----Binary string - banary array
  t := dig -1;
  For i := 0 to dig -1 do begin
    binaryT [t] := StrToInt (value[i+1]);
    dec (t);
  end;
//Dline result -> Grey code
  binary[2] := binaryT[3];
  binary[1] := binaryT[1] xor binaryT[5];
  binary[0] := binaryT[0] xor binaryT[2] xor binaryT[4] xor binaryT[6];

  binstring := '';
    for i := 2 downto 0 do //
      binstring := binstring + inttostr (binary[i]);
  Result := binstring;
end;
//-------------------------------------------------------------------------------------------------------------------------------------------------------------//

procedure TForm1.FormCreate(Sender: TObject);
var
  cnt : Integer;
  temp : string;
begin
  Out32 (890, 231); Programmas darbības sakumā uz atmiņas WEN un REN ieejam tiek padots loģiskais 1, tāda veida bloķējot
```

```
end;                    ierakstu/nolasi

procedure TForm1.btReadClick(Sender: TObject);  Vērtību nolase no atmiņas
var
  temp, i, inter : integer;
  binstring, binstring2 : string;
begin
  inter := StrToInt (edInterval.Text);
  RCLKcnt := StrToInt (edRCLKcount.Text);
  SetLength (Grey, RCLKcnt);
  SetLength (Dline, RCLKcnt);
  SetLength (Greybin, RCLKCnt);
  SetLength (Dlinebin, RCLKCnt);
  out32 (890, 239); //Enable REN
  sleep (1);
  for i:= 0 to 1 do begin
    out32 (890, 239);
    sleep (1);
    out32 (890, 237);
    sleep (1);
  end;
  out32 (890, 239);

  for i:= 0 to RCLKcnt -1 do begin
    out32 (890, 239);
    sleep (inter);
    out32 (890, 237);
    sleep (inter);
    temp := inp32 (888);

    If rbGrey.Checked = true then begin
      Grey[i] := temp;
      mmgrey.Lines.add (IntToStr (i+1) + '. ' + IntToStr (grey[i]));
      binstring := IntToBin(grey[i], 8);
      Greybin[i] := binstring;
      mmgreybin.Lines.Add(IntToStr (i+1) + '. ' + binstring);
    end
   else if rbDline.Checked = true then begin
      DLine[i] := temp;
      mmdline.Lines.add (IntToStr (i+1) + '. ' + IntToStr (DLine[i]));
      binstring := IntToBin(Dline[i], 7);
      mmdlinebin.Lines.Add(IntToStr (i+1) + '. ' + binstring);
      binstring2 := DlineToGrey (binstring, 7);
      Dlinebin [i] := binstring2;
      mmdlinegrey.Lines.Add(IntToStr (i+1) + '. ' + binstring2)
    end;
  end;
  out32 (890, 239);//RCLK - low
  out32 (890, 231);//REN - high
end;

procedure TForm1.bttranslateClick(Sender: TObject);  Laika intervālu aprēķināšana.
var
  i, td : Integer;
  temp : Single;
  resultgrey : Array of string;
  resultdec : Array of Integer;
  binstring : String;
begin
  mmresultgrey.Clear;
  mmresultdec.Clear;
  mmintervals.Clear;
  SetLength (resultgrey, RCLKcnt);
```

```
  SetLength (resultdec, RCLKcnt);
  SetLength (intervals, RCLKcnt);
  td := StrToInt (edtd.Text);
  for i := 0 to RCLKcnt-1 do begin
   resultgrey [i]:= Greybin [i] + dlinebin[i];
   mmresultgrey.Lines.Add (IntToStr (i+1) + '. ' + resultgrey[i]);

   binstring := GreybinToBin(resultgrey [i], 11);
   resultdec[i] := BinToInt (binstring);

   mmresultdec.lines.Add(IntToStr (i+1) + '. ' + IntToStr(resultdec [i]));
   intervals [i] := resultdec [i] * td / 1000;
  end;
  for i := 0 to RCLKcnt-2 do begin
   temp := intervals [i+1] - intervals [i];
   mmintervals.Lines.Add (IntToStr (i+2) + '-' + InttoStr (i+1) + ' = '
     + floattoStrf (temp, fffixed, 8, 2));
  end;
  end;

procedure TForm1.btsaveClick(Sender: TObject);  Datu saglabāšana teksta failā.
var
  fname : string;
  afile : TextFile;
  i : integer;
  temp : string;
begin
  saveDialog1.Filter := 'Text file|*.txt|Word file|*.doc';
  saveDialog1.Title := 'Result saving in text file';
  saveDialog1.InitialDir := 'C:\Documents and Settings\maros\Desktop';
  saveDialog1.DefaultExt := 'txt';
  saveDialog1.FilterIndex := 1;
  if saveDialog1.Execute then begin
    AssignFile (afile, SaveDialog1.Filename);
    Rewrite (afile);
    write (afile, 'Intervals');
    writeln (afile);
    writeln (afile);
    For i := 0 to RCLkcnt -1  do begin
      temp := mmintervals.Lines[i];
      writeln (afile, temp );
    end;
    CloseFile(afile);
  end;
end;

procedure TForm1.btenablewriteClick(Sender: TObject); Ieraksta atļauja atmiņā
begin
  Out32 (890, 227);
  btenablewrite.Visible := False;
  btdisablewrite.Visible := True;
end;

procedure TForm1.btdisablewriteClick(Sender: TObject);  Ieraksta bloķēšana atmiņā
begin
  Out32 (890, 231);
  btenablewrite.Visible := True;
  btdisablewrite.Visible := False;
end;

procedure TForm1.btokClick(Sender: TObject);  Datu pārveidošana sekcijā Transformation calculator
var
  intext, temp1, dig : integer;
```

```
  dstring : string;
begin
  intext := StrToInt (edin.Text);
  dstring := edin.Text;
  If rb1.Checked = true then
    edout.Text := IntToStr (GreyToDec(intext, 10));
  If rb2.Checked = true then
    edout.Text := IntToStr (DecToGrey(intext, 10));
  IF rb3.Checked = true then begin
    dstring := DlineToGrey(IntToBin(intext,10),10);
    temp1 := BinToInt(dstring);
    edout.Text := IntToStr (temp1);
  end;
  If rb4.Checked = true then begin
    dig := Length (dstring);
    edout.Text :=GreybinToBin(dstring, dig);
  end;
end;

procedure TForm1.btcleargreyClick(Sender: TObject);  Greja koda skaitītāja sekcijas notīrīšana
begin
  mmgrey.Clear;
  mmgreybin.Clear;
end;

procedure TForm1.Button2Click(Sender: TObject);  Aiztures līnijas sekcijas notīrīšana
begin
  mmdline.Clear;
  mmdlinebin.Clear;
  mmdlinegrey.Clear;
end;
end.
```

SIGNAL-DEPENDENT TECHNIQUES FOR NON-STATIONARY SIGNAL SAMPLING AND RECONSTRUCTION

M. Greitans and R. Shavelis

Institute of Electronics and Computer Science
14 Dzerbenes Str., Riga LV-1006, Latvia
phone: + (371) 67558110, fax: + (371) 67555337, email: greitans@edi.lv, shavelis@edi.lv
web: www.edi.lv

ABSTRACT

The paper describes the processing of non-stationary signals, which takes the advantages offered by the use of signal-dependent techniques in sampling and analysis procedures. The level-crossing approach is exploited for signal sampling, whereby the local sampling density provides information about the local maximum spectral frequency of the signal. The frequency is used to build signal-dependent reconstruction functions required for signal recovery by solving a least squares problem. The results of simulation are presented using speech as an example. The approach developed can be implemented using asynchronous design techniques and could be aimed at application in speech transmission over wireless networks.

1. INTRODUCTION

The spectral contents of signals of practical interest often change with time. Generally, a signal with time-varying spectral bandwidth can be approximated with fewer samples per interval using appropriate non-equidistantly spaced samples than using uniform sampling procedure, where the sampling rate is chosen taking into account the highest signal frequency. Intuitively speaking, the non-stationarity of the signal should be reflected in the process of analog-to digital conversion – the low frequency regions should be sampled at a lower rate than the high frequency regions.

A special class of non-uniform sampling is derived if the sampling process is driven by the signal itself – it is so called signal dependent sampling. Popular types are based on zero-crossing, reference signal crossing, level crossing and send-on-delta concepts. Particular attention should be paid to cases, where the local sampling density derives from the local properties of the signal. One of the sampling approaches with such a quality is level-crossing sampling, which will be used further in the paper as a tool for digital data capture from a continuous time signal.

2. LEVEL-CROSSING SAMPLING

The idea of level-crossing sampling (LCS) is based on the principle that samples are captured when the input signal crosses predefined levels. Such a sampling strategy has quite long history and is exploited for various applications [1, 2]. The quantization levels can be located arbitrarily, however, if there is no special reason, the typical solution is to dispose them uniformly along the amplitude range of the signal.

It has been shown that level-crossing sampling has several interesting properties and is more efficient than traditional sampling in many respects [3]. In particular, it can be related to the processing of non-stationary signals, because the local density of samples reflects the local characteristics of the signal [4, 5]. If a waveform is changing rapidly, the samples are spaced more closely, and conversely – if a signal is varying slowly, the samples are spaced sparsely. This property allows tracking of the local maximum frequency in the signal spectrum in order to use it for data analysis. Since the level-crossing sampling scheme provides non-equidistantly spaced samples, appropriate processing methods must be developed.

3. RECONSTRUCTION OF SIGNAL WITH TIME-VARYING BANDWIDTH

Several methods for reconstruction of non-uniformly sampled band-limited signals are used. For correct recovery, they typically require that the maximal length of the gaps between the sampling instants does not exceed the Nyquist [6]. If a signal is non-stationary with time-varying spectral bandwidth, the global satisfying of this requirement is not an appropriate decision, because that provides redundant data. The use of level-crossing sampling scheme can reduce the amount of samples, because the intervals between samples are determined by signal local properties and by the number of quantization levels. The quality of processing can be improved if the recovery procedure takes into account the local bandwidth of the signal [7]. In the following subsections will be discussed proposed idea and methods for reconstruction using filters with time-varying bandwidth and for estimation of local maximum frequency of signal from its level-crossing samples.

3.1 Signal-dependent reconstruction functions

The sampling theorem states that every bandlimited signal $s(t)$ can be reconstructed from its equidistantly spaced samples if the sampling rate equals or exceeds the Nyquist rate $2F_{max}$, where F_{max} is the maximum frequency in the signal spectrum. The reconstruction in time domain can be expressed as

$$\hat{s}(t) = \sum_{n=0}^{N-1} s(t_n) h(t - t_n), \tag{1}$$

where $\widehat{s}(t)$ denotes reconstructed signal, N is the number of the original signal samples $s(t_n)$ and $h(t)$ is an appropriate impulse response of the reconstruction filter, classically, sinc-function

$$h_1(t) = \mathrm{sinc}(2\pi F_{max} t) \tag{2}$$

As the sampling instants $t_n = \frac{n}{2F_{max}}$, then the impulse response

$$h_1(t-t_n) = h_1(t,t_n) = \mathrm{sinc}(2\pi F_{max}t - n\pi), \qquad (3)$$

where $h_1(t-t_n) = h(t,t_n)$ is written as the function of two arguments. The reconstructed signal becomes

$$\hat{s}(t) = \sum_{n=0}^{N-1} s(t_n)h_1(t,t_n) \qquad (4)$$

If the signal with time-varying frequency bandwidth $f_{max}(t)$ is considered, then the sampling rate of the signal according to Nyquist must be at least $2F_{max}$, where $F_{max} = \max(f_{max}(t))$. In this case any information about the local spectral bandwidth is ignored during the sampling process. To take it into account, it is proposed instead of $h_1(t,t_n)$ to use more general function

$$h_2(t,t_n) = \mathrm{sinc}(\Phi(t) - \Phi(t_n)) = \mathrm{sinc}(\Phi(t) - n\pi), \qquad (5)$$

where $\Phi(t) = 2\pi\int_0^t f_{max}(t)dt$ is the phase of the sinusoid, whose frequency changes in time as $f_{max}(t)$, $t \geq 0$ and sampling instants t_n are chosen such that $\Phi(t_n) = n\pi$. If the signal is stationary and band-limited $f_{max}(t) = \mathrm{const} = F_{max}$, equations (3) and (5) become equivalent. In case of non-constant $f_{max}(t)$ waveform of the reconstruction function $h_2(t,t_n)$ and the desired sampling instants t_n are determined by $f_{max}(t)$. Samples are spaced non-equidistantly and the mean sampling frequency can be less than it is required by Nyquist criterion, which, in this case, should be satisfied rather in local than in global sense.

3.2 Reconstruction algorithm

To apply the formula (4) for reconstructing the signal from its level-crossing samples $s(t_m)$, the recovery procedure involves signal resampling from sampling set $\{t_m\}$ to $\{t_n\}$. The new sampling values $\hat{s}(t_n)$ are found by the method of least squares to ensure the minimal error

$$\sum_{m=0}^{M-1} \left(s(t_m) - \hat{s}(t_m)\right)^2 = \min, \qquad (6)$$

where

$$\hat{s}(t_m) = \sum_{n=0}^{N-1} \hat{s}(t_n)h(t_m,t_n) \qquad (7)$$

Considering (6) and (7) the solution in matrix notation is obtained

$$\hat{\mathbf{S}} = \mathbf{S}\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}, \qquad (8)$$

where $\hat{\mathbf{S}} = [\hat{s}(t_0), \hat{s}(t_1), \ldots, \hat{s}(t_{N-1})]$, $\mathbf{H}$ is $M \times N$ matrix whose element in row m and column n is $h(t_m,t_n)$ and $\mathbf{S} = [s(t_0), s(t_1), \ldots, s(t_{M-1})]$.

In the level-crossing sampling case the values of $\hat{s}(t_n)$ are limited by two corresponding adjacent quantization levels $a_n \leq \hat{s}(t_n) \leq b_n$, where $a_n \in Q$, $b_n \in Q$ and Q is the set of all quantization levels. This restriction can be written by substituting

$$\hat{s}(t_n) = a_n + (b_n - a_n)k_n, \qquad (9)$$

where the coefficient $0 \leq k_n \leq 1$. The equation (9) for all samples $\hat{s}(t_n)$ can be written as

$$\hat{\mathbf{S}} = \mathbf{A} + (\mathbf{B} - \mathbf{A}) \circ \mathbf{K}, \qquad (10)$$

where $\mathbf{A} = [a_0, a_1, \ldots, a_{N-1}]$, $\mathbf{B} = [b_0, b_1, \ldots, b_{N-1}]$, $\mathbf{K} = [k_0, k_1, \ldots, k_{N-1}]$ and $(\circ)$ denotes Hadamard product of two matrices. From (8) and (10) it follows

$$\mathbf{K} = \frac{\mathbf{S}\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1} - \mathbf{A}}{\mathbf{B} - \mathbf{A}} \qquad (11)$$

The solution (11) may provide coefficient values that lie outside the allowed interval limits of $[0,1]$. To prevent this the minimization task (6) considering (7) and (9) should be solved for k_n values $0 \leq k_n \leq 1$. As it can be very time-consuming, the coefficients obtained by (11) are roughly limited by (12)

$$k_n = \begin{cases} 0, & \text{if} \quad k_n < 0 \\ k_n, & \text{if} \quad 0 \leq k_n \leq 1 \\ 1, & \text{if} \quad k_n > 1 \end{cases} \qquad (12)$$

Further the coefficients are made more precise considering that the reconstructed signal between two successive level-crossings is also limited by two corresponding quantization levels. If we choose the uniform sampling set $\{t_u\}$ with high enough density and indices $u = 0,1,2,\ldots,U-1$, then the reconstructed signal according to (4) and (9) is

$$\hat{s}(t_u) = \sum_{n=0}^{N-1} \left(a_n + (b_n - a_n)k_n\right) h(t_u,t_n) \qquad (13)$$

Every recovered sample $\hat{s}(t_u)$ must lie between two corresponding quantization levels $c_u \leq \hat{s}(t_u) \leq d_u$, where $c_u \in Q$ and $d_u \in Q$. For all samples this condition can be written as

$$\mathbf{C} \leq \left(\mathbf{A} + (\mathbf{B} - \mathbf{A}) \circ \mathbf{K}\right)\mathbf{G}^T \leq \mathbf{D}, \qquad (14)$$

where $\mathbf{D} = [d_0, d_1, \ldots, d_{U-1}]$, $\mathbf{C} = [c_0, c_1, \ldots, c_{U-1}]$ and $\mathbf{G}$ is $U \times N$ matrix whose element in row u and column n is $h(t_u,t_n)$. The inequality in (14) is applied element-wise. After the estimation of $\mathbf{K}$ according to (11) and (12) the verification of condition (14) follows, providing indices u' that do not satisfy the requirement. By randomly choosing one of the indices u' the index n is found for which the distance $|t_{u'} - t_n|$ is minimal. Then the coefficient k_n is changed as follows

$$k_n = \begin{cases} k_n(1-\alpha), & \text{if} \quad \hat{s}(t_{u'}) > d_u \\ k_n(1-\alpha) + \alpha, & \text{if} \quad \hat{s}(t_{u'}) < c_u \end{cases} \qquad (15)$$

where $0 \leq \alpha \leq 1$ determines how fast the coefficient is decreased towards zero or increased towards one (we choose $\alpha = 0.05$). Thereafter steps (13), (14) and (15) are repeated until condition (14) is satisfied for all u or fixed number of iterations is reached. The fixed number should be set (we choose $10N$) because such technique can not guarantee the fulfilment of (14). However, the number of indices u' can be reduced significantly by random selection of u' and change of corresponding coefficient k_n at each iteration. Random selection is preferred since the new coefficient influences all $\hat{s}(t_u)$ values. The number of indices u' can also be reduced if instead of one coefficient two coefficients k_n and k_{n+1} corresponding to $\hat{s}(t_n)$ and $\hat{s}(t_{n+1})$ with t_n and t_{n+1} located most closely to $t_{u'}$ are changed and the condition (14) is made softer by replacing the values c_u and d_u in matrices $\mathbf{C}$ and

D with $c_u-(d_u-c_u)\beta$ and $d_u+(d_u-c_u)\beta$, where $\beta>0$ (we choose $\beta=0.05$).

For calculation of **H** and **G** either impulse response (3) or (5) can be used. In level-crossing sampling case better reconstruction result is achieved by $h_2(t,t_n)$ as it depends on instantaneous maximum frequency of the signal. The resampling instants t_n are determined by $f_{max}(t)$, that in general case is not known in advance. To solve this problem, an algorithm is developed, which estimates the time-varying instantaneous maximum frequency using information about locations of level-crossings. Please note that instantaneous maximum frequency stands for local bandwidth of the signal and is not the same as instantaneous frequency defined through Hilbert transform.

3.3 Estimation of instantaneous maximum frequency

The local bandwidth of the signal can be estimated by finding its time-frequency representation (TFR) using, for example, short-time Fourier transform, wavelet transform or Wigner-Ville distribution. These methods are developed for uniformly sampled signals, however, there are some modifications in order to find the TFR of non-uniformly sampled signals [8]. The use of such approach is time consuming, thus a simpler method should be considered. In [9] it is shown how to obtain instantaneous frequency of the phase signal sampled by level-crossings. However, signals of practical interest are not so simple, thus the method based on empirical evaluations is proposed.

To estimate the function $\widehat{f}_{max}(t)$ from samples $s(t_m)$, starting with the initial index value $m=0$ two pairs of successive level-crossing samples $s(t_{m'_j})=s(t_{m'_j+1})$ and $s(t_{m''_j})=s(t_{m''_j+1})$ are found such that $m''_j>m'_j$ and the difference $m''_j-m'_j$ is minimal. Thereafter the next two pairs are found considering that $m'_{j+1}=m''_j$. For each $j=1,2,\ldots$ the value $f(t_j)$ is calculated as

$$f(t_j)=\left(t_{m''_j}+t_{m''_j+1}-t_{m'_j}-t_{m'_j+1}\right)^{-1}, \tag{16}$$

where

$$t_j=\frac{1}{4}\left(t_{m''_j}+t_{m''_j+1}-t_{m'_j}-t_{m'_j+1}\right) \tag{17}$$

If a single sinusoid is sampled, then $f(t_j)=f(t_{j+1})$ for all j and it equals the frequency of the sinusoid. If the signal consists of more harmonics, then $f(t_j)$ for different j vary around the average value of $\bar{f}=\frac{1}{J}\sum_{j=1}^{J}f(t_j)$, where J is the total number of detected pairs within the observation time of the signal. Experiments show that $\bar{f}$ is close to the frequency of the highest component. Thus, the estimate of function of instantaneous maximum frequency $\widehat{f}_{max}(t)$ can be obtained by $\{f(t_j)\}$ approximation with piecewise polynomials $p_v^r(t)$ of order r. By choosing the number $L>1$ the observation interval of signal is divided into subintervals

$$\Delta T_v: t\in[t_{v,1};t_{v,2}], \tag{18}$$

where $v=0,1,\ldots$ is the number of subinterval and

$$t_{v,1}=\frac{t_{j=vL}+t_{j=vL+1}}{2}, \tag{19}$$
$$t_{v,2}=\frac{t_{j=(v+1)L}+t_{j=(v+1)L+1}}{2}$$

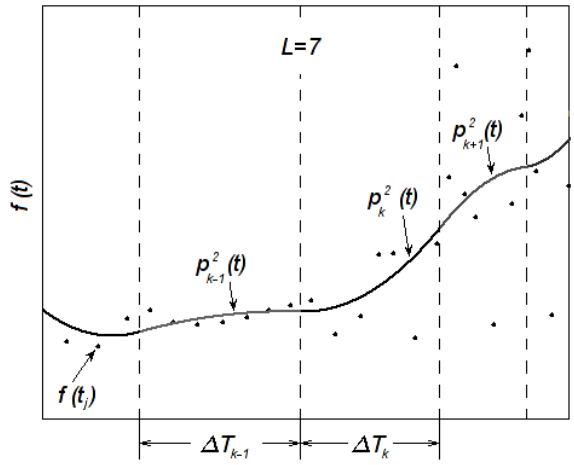


Figure 1: Piecewise polynomial $p_k^2(t)$ approximation (the number of samples per subinterval is $L=7$).

For each subinterval ΔT_v the coefficients $e_{v,r},e_{v,r-1},\ldots,e_{v,1},e_{v,0}$ of polynomial $p_v^r(t)=e_{v,r}t^r+e_{v,r-1}t^{r-1}+\cdots+e_{v,1}t+e_{v,0}$ are found to ensure

$$\begin{aligned}
&p^r_{v-1}(t_{v,1})^{(0)}=p^r_v(t_{v,1})^{(0)}\ ,\ p^r_v(t_{v,2})^{(0)}=p^r_{v+1}(t_{v,2})^{(0)}\\
&p^r_{v-1}(t_{v,1})^{(1)}=p^r_v(t_{v,1})^{(1)}\ ,\ p^r_v(t_{v,2})^{(1)}=p^r_{v+1}(t_{v,2})^{(1)}\\
&\qquad\qquad\vdots\\
&p^r_{v-1}(t_{v,1})^{(r)}=p^r_v(t_{v,1})^{(r)}\ ,\ p^r_v(t_{v,2})^{(r)}=p^r_{v+1}(t_{v,2})^{(r)}
\end{aligned}$$

and the value of expression

$$\sum_{v=0}^{V-1}\sum_{j=vL+1}^{(v+1)L}[f(t_j)-p_v^r(t_j)]^2=\min \tag{20}$$

is minimal. The denotation $(\ldots)^{(r)}$ means the derivative of order r and V is the total number of subintervals. After solving the minimization task using the method of least squares, the coefficients of polynomials $p_v^r(t)$ are obtained and the estimate of instantaneous maximum frequency

$$\widehat{f}_{max}(t)=p_v^r(t),\ \text{if}\ t_{v,1}\le t\le t_{v,2} \tag{21}$$

depends on the number L of samples $f(t_j)$ per subinterval. To reduce the dependency the final frequency estimate is obtained by averaging $\widehat{f}_{max}(t)$ calculated for different L values. The example of piecewise polynomial of order $r=2$ approximation when $L=7$ is shown in Fig. 1.

4. APPLICATION TO SPEECH PROCESSING

Speech transmission is one of the most important and common services in telecommunication networks. One of the basic prerequisites for successful speech transmission over data channels is the use of an effective speech encoding technique. It should compress the speech signal at the sender's end and decompress the digital codes to reconstruct the speech with satisfactory quality at the receiver's end. The main concern for system designers is to preserve the best speech quality and at the same time reduce the necessary bit rate of data

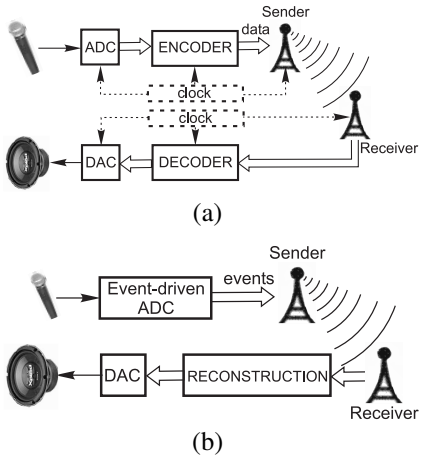


Figure 2: Different structures of speech transmission systems: (a) traditional clock-based; (b) proposed event-driven.

transmission. To achieve such efficiency more and more sophisticated speech-coding algorithms are used that need more memory and computational load.

On the other hand, it is attractive if electronic devices, which perform speech transmission, can be miniaturized with low power consumption, especially in wireless equipment. A simplified block diagram of the "classical" speech transmission approach is illustrated in Fig. 2a. Speech digitizing is based on clock-driven analog-to-digital (A/D) converter, which is followed by a digital signal processing (DSP) block. As a result, the speech data can be compressed approximately ten times, which considerably diminishes the power consumption of the sender (important for wireless system) as well as the load on the data transmission channel (important for VoIP system).

The algorithm described above can be implemented in an alternative structure of speech processing and transmission system, which is based on event-driven A/D conversion and is proposed in [7]. The block diagram of the system is shown in Fig. 2b. It can be seen that this structure provides substantial simplification of the sender part of the system. Application of the method developed to speech processing is motivated by the properties of speech signals. Within the naturally spoken language, pauses and disfluencies occur quite often, which do not provide useful information from the point of view of speech coding. In classical case they are extracted after the A/D conversion during the speech encoding procedure, while level-crossing sampling based A/D converter simply does not capture samples during pauses. In such a way, it is possible to considerably decrease the data flow on the ADC output without additional data encoding. A different application of the idea of signal encoding using level crossings is discussed in [10], where signal is initially prefiltered using a filterbank and then each output is sampled by level-crossings. Also in this case, the speech signal is taken as an example for illustration of the method.

5. SIMULATION RESULTS

The performance of the signal-dependent algorithm was tested on a speech signal taken from the TIMIT database (/timit/train/dr1/mtpf0 /sx335.wav; sampling frequency 16

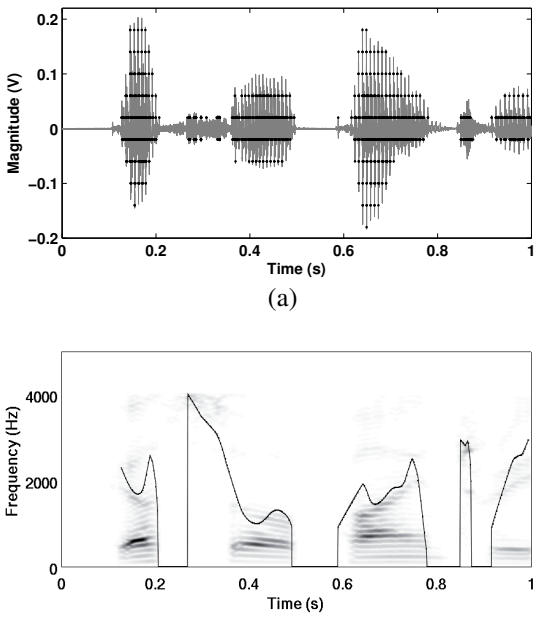


Figure 3: Fragment of the test speech sentence (samples as black points) (a), and its STFT (black line shows the estimated time-varying maximum frequency) (b).

kHz). The signal had been low-pass filtered with a cut-off frequency of 4 kHz, and interpolated by sinc functions to obtain level-crossing samples. Using 10 quantization levels 3301 samples were obtained during 3.8 seconds of the test phrase (mean sampling rate is about 870 samples per second). Uniform sampling at rate of 8 kHz provides 30400 samples. The waveform and the samples captured by LCS are illustrated in Fig. 3a. The time-frequency representation of the signal obtained by STFT is shown in Fig. 3b. The black bold line represents the instantaneous maximum frequency $\widehat{f}_{max}(t)$ of the signal estimated according to (21). From the figure follows that the bandwidth of reconstruction filter will vary in the spectral range up to 4 kHz.

After the estimation of $\widehat{f}_{max}(t)$ the calculation of k_n according to (11) follows. Only 20% of the coefficients obtained lie inside the allowed interval limits of $[0,1]$, while the rest 80% are limited by (12). To verify the condition (14) the uniform sampling set $\{t_u\}$ of 64 kHz is chosen. In total, 30% of reconstructed samples $\hat{s}(t_u)$ calculated by (13) do not satisfy (14) and reconstruction error $\sqrt{\frac{1}{U}\sum_{u=0}^{U-1}(s(t_u)-\hat{s}(t_u))^2}$ is 22 mV. However, after $10N$ repetitions of steps (13), (14) and (15) the number is reduced to 8% and the error becomes 15 mV. The fragment of reconstructed speech is shown in Fig. 4a as a solid line, while the dashed line represents the original signal. The error signals $|s(t_u)-\hat{s}(t_u)|$ before and after iterative adjustment of coefficients are shown in Fig. 4b as gray and black solid lines. It can be noticed that the amplitude of the error signal is decreased and does not exceed the value of 40 mV, which is the distance between two quantization levels. The average simulation time used by an ordinary personal computer (CPU frequency 2.66 GHz) for reconstructing the signal is 10 times larger than the length of

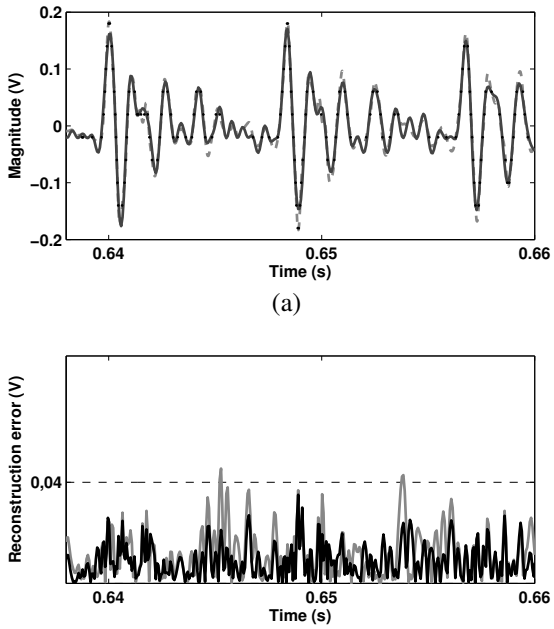


Figure 4: Reconstructed speech signal as black solid line (a), and reconstruction error signals before (gray solid line) and after (black solid line) iterative update of coefficients.

the signal and most of the time (up to 80%) is taken up by iterative adjustment of coefficients.

The reconstruction result improves as the number of quantization levels increases providing more level-crossing samples. If there are 20 quantization levels, then during 3.8 seconds of the test phrase 8509 level-crossing samples are obtained. When $\widehat{f}_{max}(t)$ is estimated the calculation of k_n follows. Now 70% of the coefficients obtained by (11) are limited according to (12). In total, 31% of reconstructed samples $\hat{s}(t_u)$ do not satisfy the condition (14) and the reconstruction error is 10 mV. After $10N$ repetitions of steps (13), (14) and (15) the number is reduced to 9% and the error becomes 6.9 mV.

6. CONCLUSIONS

The proposed approach for non-stationary signal processing uses signal dependent techniques: level crossing sampling for data acquisition and applying of time-varying bandwidth filter for signal reconstruction. The information carried by level-crossing samples is employed in two ways – time instants of samples are used to estimate the instantaneous maximum frequency of the signal, while the amplitude values of samples are used in reconstruction algorithm. The reconstruction procedure is based on solving a least squares problem to find the new samples of the signal at time instants, which are determined by evaluated instantaneous maximum frequency. The adjustment of new sampling values follows by verifying if the reconstructed signal lies between corresponding quantization levels.

Speech signal processing is demonstrated as one of the application areas. Simulation results show advantages of proposed method, which are related to the exclusion of pauses and disfluencies from processing before A/D conversion as well as to the possibility of decrease in sampling density. In case of 10 quantization levels audio perception remains good, while the number of samples is reduced 9 times in comparison with standard uniform processing.

REFERENCES

[1] P. Ellis, "Extension of phase plane analysis to quantized systems," *IRE Transactions on Automatic Control*, vol. 4(2), pp. 43–54, 1959.

[2] M. Miskowicz, "Send-On-Delta Concept: An Event-Based Data Reporting Strategy," *Sensors*, vol. 6, pp. 49–63, 2006.

[3] E. Allier, and G. Sicard, "A new class of asynchronous A/D converters based on time quantization," in *Proc. ASYNC 2003*, Vancouver, BC, Canada, May 12-16. 2003, pp. 196–205.

[4] M. Greitans, "Processing of Non-Stationary Signal Using Level-Crossing Sampling," in *Proc. SIGMAP 2006*, Setubal, Portugal, August 7-10. 2006, pp. 170–177.

[5] S. M. Qaisar, L. Fesquet, and M. Renaudin, "An Improved Quality Adaptive Rate Filtering Technique Based on the Level Crossing Sampling," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 31, pp. 79–84, 2008.

[6] H. G. Feichtinger, and K. Grochening, "Theory and practice of irregular sampling," in *Wavelets: Mathematics and Applications*, J. Benedetto, M. Frazier, editors, pp. 305-363, CRC Press, 1993.

[7] M. Greitans, and R. Shavelis, "Speech sampling by level-crossing and its reconstruction using spline-based filtering," in *Proc. IWSSIP 2007*, Maribor, Slovenia, June 27-30. 2007, pp. 305–308.

[8] M. Greitans, "Time-Frequency Representation Based Chirp-Like Signal Analysis Using Multiple Level Crossings," in *Proc. EUSIPCO 2007*, Poznan, Poland, September 3-7. 2007.

[9] S. Chandrasekhar, and T. V. Sreenivas, "Instantaneous frequency estimation using level-crossing information," in *Proc. ICASSP 2003*, Hong Kong, China, April 6-10. 2003, pp. 141–144.

[10] R. Kumaresan, and N. Panchal, "Encoding Bandpass Signals Using Level Crossings: A Model-based Approach," in *Audio Engineering Society 123rd Convention*, New York, NY, USA, October 5-8. 2007.

Signal-dependent sampling and reconstruction method of signals with time-varying bandwidth

Modris Greitans and Rolands Shavelis

Institute of Electronics and Computer Science, 14 Dzerbenes str., Riga LV-1006, Latvia.
greitans@edi.lv, shavelis@edi.lv

Abstract:

The paper describes the sampling method of nonstationary signals with time-varying spectral bandwidth. The reconstruction procedure exploiting the low-pass filter with time-varying cut-off frequency is derived. The filter application in signal reconstruction from its level-crossing samples is shown. The results of computer simulations are presented.

1. Introduction

The spectral characteristics of signals of practical interest often change with time. Generally, a signal with time-varying spectral bandwidth can be approximated with fewer samples per interval using appropriate non-equidistantly spaced samples than using uniform sampling procedure, where the sampling rate is chosen taking into account the highest signal frequency. For example, let us inspect a signal with wide bandwidth regions and narrow spectral bandwidth in the rest of signal observation. It is more efficient to sample the narrow bandwidth regions at a lower rate than the regions, where spectral bandwidth is wide. Solving this problem correctly requires the knowledge of the function of the instantaneous maximum frequency of signal. The paper will show two typical situations. First, information about the time-varying bandwidth is known a priori. In this case the deliberately non-uniform sampling instants can be calculated in advance, and reconstruction is based on application of filter with appropriate time-varying impulse response function. Second, the signal-dependent sampling scheme - level crossing sampling (LCS) is used for analog-to-digital (A/D) conversion. The idea of level-crossing sampling is based on the principle that samples are captured when the input signal crosses predefined levels. Such a sampling strategy has quite long history and is exploited for various applications [1, 2]. It has been shown that LCS has several interesting properties and is more efficient than traditional sampling in many respects [3]. In particular, it can be related to the processing of non-stationary signals, because if a waveform is changing rapidly, the samples are spaced more closely, and conversely – if a signal is varying slowly, the samples are spaced sparsely [4]. This property allows to calculate the estimate of the function of the instantaneous maximum frequency of signal from the positions of samples. In this case to reconstruct the waveform of signal, an additional resampling procedure is needed before the use of time-varying reconstruction filter, which will be described in next section.

Note that in both cases the local sampling density reflects the local bandwidth of the signal, therefore samples are spaced non-uniformly and advanced algorithms are required for digital signal processing.

2. Reconstruction of signal with time-varying bandwidth

There are several methods used for reconstruction of non-uniformly sampled band-limited signals. For correct recovery, they typically require that the maximal length of the gaps between the sampling instants does not exceed the Nyquist rate [5]. If the signal is non-stationary with time-varying spectral bandwidth, satisfying globally this requirement is not an appropriate decision, because this provides redundant data. The use of level-crossing sampling scheme can reduce the amount of samples, because the intervals between samples are determined by signal local properties and by the number of quantization levels. The quality of processing can be improved if the recovery procedure takes into account the local bandwidth of the signal [6]. In the following subsections the proposed idea and methods for reconstruction using filters with time-varying bandwidth and for the estimation of local maximum frequency of signal from its level-crossing samples will be discussed.

2.1 Idea of signal-dependent reconstruction functions

The sampling theorem states that every bandlimited signal $s(t)$ can be reconstructed from its equidistantly spaced samples if the sampling rate equals or exceeds the Nyquist rate $2F_{max}$, where F_{max} is the maximum frequency in the signal spectrum. The reconstruction in time domain can be expressed as

$$\hat{s}(t) = \sum_{n=0}^{N-1} s(t_n) h(t - t_n), \qquad (1)$$

where $\widehat{s}(t)$ denotes reconstructed signal, N is the number of the original signal samples $s(t_n)$ and $h(t)$ is an appropriate impulse response of the reconstruction filter, classi-

cally, sinc-function

$$h_1(t) = \text{sinc}(2\pi F_{max} t) \tag{2}$$

As the sampling instants $t_n = \frac{n}{2F_{max}}$, then the impulse response

$$h_1(t - t_n) = h_1(t, t_n) = \text{sinc}(2\pi F_{max} t - n\pi), \tag{3}$$

where $h_1(t - t_n) = h(t, t_n)$ is written as the function of two arguments. The reconstructed signal becomes

$$\hat{s}(t) = \sum_{n=0}^{N-1} s(t_n) h_1(t, t_n) \tag{4}$$

If the signal with time-varying frequency bandwidth $f_{max}(t)$ is considered, then the sampling rate of the signal according to Nyquist must be at least $2F_{max}$, where $F_{max} = \max(f_{max}(t))$. In this case any information about the local spectral bandwidth is ignored during the sampling process. To take it into account, it is proposed instead of $h_1(t, t_n)$ to use more general function

$$h_2(t, t_n) = \text{sinc}(\Phi(t) - \Phi(t_n)) = \text{sinc}(\Phi(t) - n\pi), \tag{5}$$

where $\Phi(t) = 2\pi \int_0^t f_{max}(t)dt$ is the phase of the sinusoid, whose frequency changes in time as $f_{max}(t)$, $t \geq 0$ and sampling instants t_n are chosen such that $\Phi(t_n) = n\pi$. If the signal is stationary and band-limited $f_{max}(t) = \text{const} = F_{max}$, Eq. (3) and (5) become equivalent. In case of non-constant $f_{max}(t)$ waveform of the reconstruction function $h_2(t, t_n)$ and the desired sampling instants t_n are determined by $f_{max}(t)$. Samples are spaced non-equidistantly and the mean sampling frequency can be less than it is required by Nyquist criterion, which, in this case, should be satisfied rather in local than in global sense.

2.2 Reconstruction algorithm

To reconstruct the non-uniformly sampled signal according to equation (1), the reconstruction procedure involves signal resampling to the equidistantly spaced sampling set $\{t_n\}$ with sampling period $\Delta t = t_n - t_{n-1} = \frac{1}{2F_{max}}$. The estimation of $\hat{s}(t_n)$ is possible according to the simple iterative algorithm [5] the idea of which is to interpolate the sampled band-limited signal $s(t)$ by the sum $\check{s}_{s(t_m)}(t) = \sum_m s(t_m)\psi_m$ and filter it in order to remove high frequencies. Piecewise linear interpolation, which is well suited to level-crossing samples, uses ψ_m consisting of the triangular functions

$$\psi_m(t) = \begin{cases} \frac{t - t_{m-1}}{t_m - t_{m-1}} & \text{for} \quad t_{m-1} \leq t < t_m, \\ \frac{t_{m+1} - t}{t_{m+1} - t_m} & \text{for} \quad t_m \leq t < t_{m+1}, \\ 0 & \text{elsewhere.} \end{cases} \tag{6}$$

It is proved [5] that if the maximum length of the gaps between the sampling instants $\tau_{max} \leq \frac{1}{2F_{max}}$, then every $s(t)$ can be reconstructed from the values $s(t_m)$ of an arbitrary τ_{max}-dense sampling set $\{t_m\}$ iteratively. The recovery algorithm can be written as:

$$\begin{aligned} &\hat{s}_0(t_n) = \check{s}_{s(t_m)}(t_n); \\ &\hat{s}_0(t) = C\left[\hat{s}_0(t_n)\right]; \\ &\hat{s}_i(t_n) = \hat{s}_{i-1}(t_n) + \check{s}_{(s - s_{i-1})(t_m)}(t_n); \\ &\hat{s}_i(t) = C\left[\hat{s}_i(t_n)\right], \end{aligned} \tag{7}$$

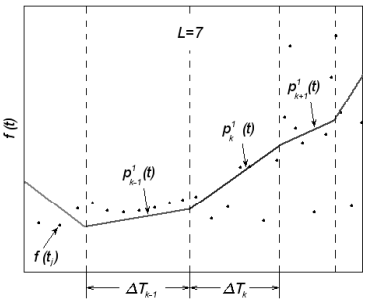


Figure 1: Piecewise polynomial $p_k^1(t)$ approximation.

where i indicates the number of iteration. The linear operator C denotes filtering as the convolution of samples $s(t_n)$ with impulse response $h_1(t, t_n)$ of the filter according to Eq. (4)

$$C\left[s(t_n)\right] = \sum_{n=0}^{N-1} s(t_n) h_1(t, t_n) \tag{8}$$

The sampling of non-stationary signal using level-crossing scheme does not ensure the satisfaction of the requirement $\tau_{max} \leq \frac{1}{2F_{max}}$. Direct application of the above described algorithm leads to a considerable reconstruction error, therefore two substantial enhancements are introduced to the algorithm - performing resampling to the non-equidistantly spaced values and the use of filter with impulse response $h_2(t, t_n)$ instead of classical $h_1(t, t_n)$. The resampling instants t_n are determined by $\Phi(t)$, which depends on $f_{max}(t)$, that in general case is not known in advance. To solve this problem, an algorithm is developed, which estimates the time-varying instantaneous maximum frequency using information about locations of level-crossings.

2.3 Estimation of instantaneous maximum frequency

The obvious ways to estimate the local bandwidth of the signal is by finding its time-frequency representation (TFR) using, for example, short-time Fourier transform, wavelet transform or Wigner-Ville distribution. These methods are developed for uniformly sampled signals, however, there are some modifications in order to find the TFR of non-uniformly sampled signals [7]. The use of such approach is time consuming, therefore a simpler method is considered that is based on empirical evaluations.

To estimate the function $\widehat{f}_{max}(t)$ from samples $s(t_m)$, starting with the initial index value $m = 0$ two pairs of successive level-crossing samples $s(t_{m'_j}) = s(t_{m'_j+1})$ and $s(t_{m''_j}) = s(t_{m''_j+1})$ are found such that $m''_j > m'_j$ and the difference $m''_j - m'_j$ is minimal. Thereafter the next two pairs are found considering that $m'_{j+1} = m''_j$. For each $j = 1, 2, \ldots$ the value $f(t_j)$ is calculated as

$$f(t_j) = \left(t_{m''_j} + t_{m''_j+1} - t_{m'_j} - t_{m'_j+1}\right)^{-1}, \tag{9}$$

where

$$t_j = \frac{1}{4}\left(t_{m''_j} + t_{m''_j+1} - t_{m'_j} - t_{m'_j+1}\right) \tag{10}$$

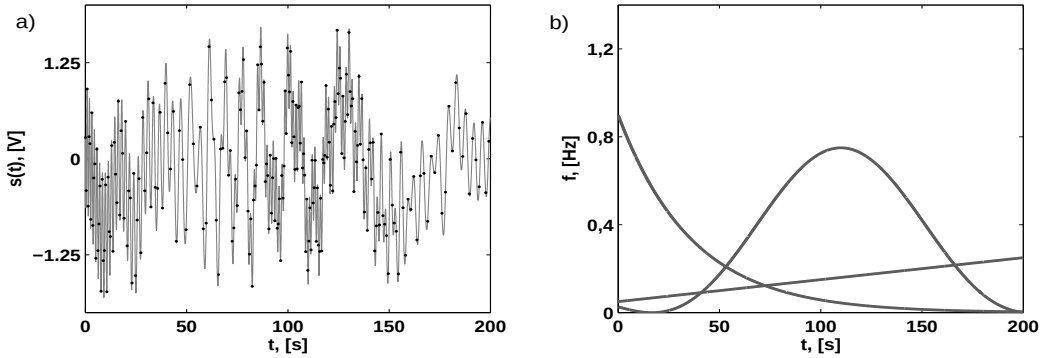


Figure 2: (a) Test signal sampled by $\Phi(t_n) = n\pi$ and (b) frequency traces of its components.

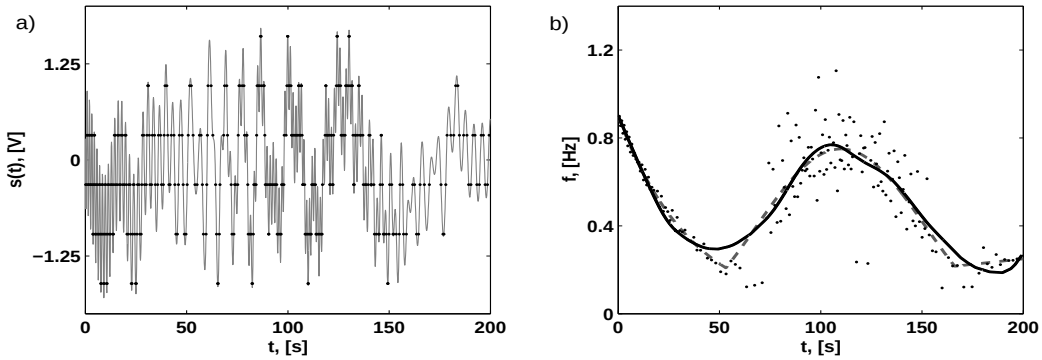


Figure 3: (a) Test signal sampled by level-crossings and (b) estimated instantaneous maximum frequency $\widehat{f}_{max}(t)$ as solid line, true instantaneous maximum frequency as dashed line and $f(t_j)$ as black points.

If a single sinusoid is sampled, then $f(t_j) = f(t_{j+1})$ for all j and it equals the frequency of the sinusoid. If the signal consists of more harmonics, then $f(t_j)$ for different j vary around the average value of $\bar{f} = \frac{1}{J}\sum_{j=1}^{J} f(t_j)$, where J is the total number of detected pairs within the observation time of the signal. Experiments show that $\bar{f}$ is close to the frequency of the highest component. Thus, the estimate of function of instantaneous maximum frequency $\widehat{f}_{max}(t)$ can be obtained by $\{f(t_j)\}$ approximation with piecewise polynomials $p_k^r(t)$ of order r. By choosing the number $L > 1$ the observation interval of signal is divided into subintervals

$$\Delta T_k : t \in [t_{k,1}; t_{k,2}]\,, \tag{11}$$

where $k = 0, 1, \ldots$ is the number of subinterval and

$$\begin{aligned} t_{k,1} &= \frac{t_{j=kL} + t_{j=kL+1}}{2}, \\ t_{k,2} &= \frac{t_{j=(k+1)L} + t_{j=(k+1)L+1}}{2} \end{aligned} \tag{12}$$

For each subinterval ΔT_k the coefficients $a_{k,r}, a_{k,r-1}, \ldots, a_{k,1}, a_{k,0}$ of polynomial $p_k^r(t) = a_{k,r}t^r + a_{k,r-1}t^{r-1} + \cdots + a_{k,1}t + a_{k,0}$ are found to ensure

$$\begin{aligned} &p_{k-1}^r(t_{k,1})^{(0)} = p_k^r(t_{k,1})^{(0)}\ ,\ p_k^r(t_{k,2})^{(0)} = p_{k+1}^r(t_{k,2})^{(0)} \\ &p_{k-1}^r(t_{k,1})^{(1)} = p_k^r(t_{k,1})^{(1)}\ ,\ p_k^r(t_{k,2})^{(1)} = p_{k+1}^r(t_{k,2})^{(1)} \\ &\qquad\qquad\vdots \\ &p_{k-1}^r(t_{k,1})^{(r)} = p_k^r(t_{k,1})^{(r)}\ ,\ p_k^r(t_{k,2})^{(r)} = p_{k+1}^r(t_{k,2})^{(r)} \end{aligned}$$

and the value of expression

$$\sum_{k=0}^{K-1}\sum_{j=kL+1}^{(k+1)L} [f(t_j) - p_k^r(t_j)]^2 = \min \tag{13}$$

is minimal. The denotation $(\ldots)^{(r)}$ means the derivative of order r and K is the total number of subintervals. After solving the minimization task using the method of least squares, the coefficients of polynomials $p_k^r(t)$ are obtained and the estimate of instantaneous maximum frequency

$$\widehat{f}_{max}(t) = p_k^r(t),\ \text{if } t_{k,1} \le t \le t_{k,2} \tag{14}$$

depends on the number L of samples $f(t_j)$ per subinterval. To reduce the dependency the final frequency estimate is obtained by averaging $\widehat{f}_{max}(t)$ calculated for different L values. The example of piecewise polynomial of order $r = 1$ approximation when $L = 7$ is shown in Fig. 1

3. Simulation results

The methods described in previous section are applied to reconstruct nonstationary signal from its nonuniform sam-

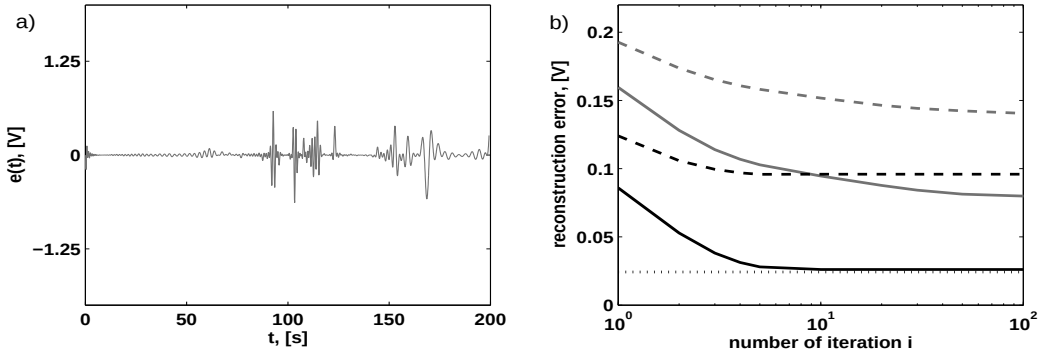


Figure 4: (a) The difference between original and recovered signal from its 349 level-crossing samples after 10 iterations and (b) reconstruction error (solid lines - reconstruction from level-crossings using $h_2(t,t_n)$, dashed lines - reconstruction from level-crossings using $h_1(t,t_n)$, dotted line - reconstruction from samples obtained by $\Phi(t_n)=n\pi$).

ples $s(t_n)$ obtained in two different ways. The first one is when $f_{max}(t)$ is given and sampling instants t_n satisfy $\Phi(t_n)=n\pi$ (Fig. 2). The second way is by level-crossing sampling and $f_{max}(t)$ is not known in advance (Fig. 3).
In the first case 239 nonequidistantly spaced samples were obtained during 200 seconds of the test signal, which consists of three sinusoids with constant amplitudes and time-varying frequencies as shown in Fig. 2b. As the reconstructed signal according to Eq. (4) using $h_2(t,t_n)$ differs insignificantly from the original one, it is not illustrated here. In order to obtain similar result in uniform sampling case, at least 360 samples would be required since the maximum frequency of the signal is $F_{max}=0.9$ Hz.
In the level-crossing sampling case 349 samples were captured using 6 quantization levels (Fig. 3a). To recover the signal the first task was to find the values $f(t_j)$ according to Eq. (9) in order to estimate the instantaneous maximum frequency (14). In Fig. 3b $f(t_j)$ are shown as black points, true $f_{max}(t)$ as dashed line and calculated $\widehat{f}_{max}(t)$ as solid line. The similarity between frequency traces is obvious. The second step was to recover the original signal according to Eq. (7) using level-crossing samples and estimated $\widehat{f}_{max}(t)$. The difference signal $e_i(t)=s(t)-s_i(t)$ after 10 iterations $i=10$ is illustrated in Fig. 4a. The reconstruction error $\sqrt{\frac{1}{T}\int_0^T e_i(t)^2dt}$ reduces as the number of iterations i increases. It is shown in Fig. 4b as a grey solid line. The grey dashed line corresponds to reconstruction error, when instead of time-varying bandwidth filter $h_2(t,t_n)$ the filter with constant cut-off frequency of $F_{max}=0.9$ Hz and impulse response $h_1(t,t_n)$ is used. In this case the achieved result is not so good as the reconstruction quality remains only in intervals, where the sampling density is sufficient. The reconstruction error can be reduced by decreasing the distance between quantization levels giving 437 level-crossing samples. It is shown in Fig. 4b as black solid and dashed lines. The dotted line corresponds to the first case when $f_{max}(t)$ is given and sampling instants t_n satisfy $\Phi(t_n)=n\pi$.

4. Conclusions

The proposed approach for non-stationary signal processing uses signal dependent techniques: level crossing sampling for data acquisition and filtering with time-varying bandwidth for signal reconstruction. The information carried by level-crossing samples is employed in two ways – time instants of samples are used to estimate the instantaneous maximum frequency of the signal, while the amplitude values of samples are used in reconstruction algorithm. The reconstruction procedure is based on the use of iterative filtering with time-varying bandwidth filter. The enhancement of classical signal reconstruction approach is made by introducing signal-dependent, "non-stationary" impulse response and resampling to the corresponding, nonuniform sampling set.
Speech signal processing can be quoted as one of the potential application areas of the proposed algorithm. The level-crossing sampling technique reduces the number of samples and leads to effective signal coding approaches.

References:

[1] P. Ellis. Extension of phase plane analysis to quantized systems. *IRE Transactions on Automatic Control*, 4(2):43–54, 1959.

[2] M. Miskowicz. Send-on-delta concept: An event-based data reporting strategy. *Sensors*, 6:49–63, 2006.

[3] E. Allier and G. Sicard. A new class of asynchronous a/d converters based on time quantization. In *Proc. of International Symposium on Asynchronous Circuits and Systems ASYNC'03*, pages 196–205, 2003.

[4] M. Greitans. Processing of non-stationary signal using level-crossing sampling. In *Proc. of the International Conference on Signal Processing and Multimedia Applications SIGMAP'06*, pages 170–177, 2006.

[5] H. G. Feichtinger and K. Grochening. Theory and practice of irregular sampling. 1994.

[6] M. Greitans and R. Shavelis. Speech sampling by level-crossing and its reconstruction using spline-based filtering. In *Proceedings of the 14th International Conference IWSSIP 2007*, pages 305–308, 2007.

[7] M. Greitans. Time-frequency representation based chirp-like signal analysis using multiple level crossings. In *Proceedings of the 15th European Signal Processing Conference EUSIPCO 2007*, 2007.

ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2009. Nr. 8(96)*
ELEKTRONIKA IR ELETROTECHNIKA

SIGNAL TECHNOLOGY
T121
SIGNALŲ TECHNOLOGIJA

Adaptive Level-Crossing Sampling Based DSP Systems

A. Baums, M. Greitans, U. Grunde
Institute of Electronics and Computer Science,
Dzerbenes st. 14, LV-1006, Riga, Latvia, phone: +371 6755813, e-mail: baum@edi.lv

Introduction

Wireless sensor networks (WSN) are effective tool for data acquisition and processing. Adaptive sampling is widely used in WSN [1] to provide energy-efficient data acquisition. Usually adaptive sensor sampling scheme supports some adaptive sampling algorithms. The algorithms modify sampling rate either controlling sampling frequency or letting sensors to skip sampling.

Further energy savings in a WSN sensor node may be achieved by adding some digital signal processing (DSP) capabilities to the sensor node. That may decrease the amount of transmitted data and may save energy because the primary source of energy consumption in the sensor node is the operation of the radio transceiver. The implementation of adaptive level-crossing (LC) sampling based DSP system is a solution for the energy-efficient data acquisition, and for the decrease of collected data. LC sampling method is chosen due to its built-in capability to adapt to the speed of signal changes instead of controlling sampling rate.

Recently several applications of non-conventional processing DSP systems [2] that are using non-uniform analog-to-digital converters (ADC) based on LC sampling are proposed. Unlike the uniform sampling method, LC sampling compares the signal with a set of reference levels and samples the time interval during the moments when the signal crosses predetermined reference levels. In certain applications that results in decreased number of data to compare with uniform sampling and allows use of DSP system resources only when it is necessary [4].

Adaptive LC sampling based DSP system is different from non-conventional processing DSP systems mentioned above. It is using reference levels that are adaptively spaced to an input signal in order to decrease the number of used reference levels. Adaptive spacing leads to non-equidistantly spaced reference levels. Such a DSP system may be used for acquisition and pre-processing of non-stationary [3,4] and non-linear signals, which are localized in time.

Adaptive level-crossing sampling

Usually LC sampling methods [5,6] are used in non-adaptive way. They assume that a difference between two subsequent reference level values is fixed and reference levels are spaced equidistantly. Equidistantly spaced reference levels are not adapted to the input signal. Non-adaptive approach requires the high linearity source for reference levels, but simplifies calculations of sampled data. Different approach is adaptive LC sampling that is using non-uniformly spaced reference levels as is shown in Fig. 1. Reference levels $rl_1 \div rl_4$ are adapted to the input signal by taking into account the signal probability distribution function and the signal power spectral density. Time intervals are sampled during the moments when the input signal $x(t)$ crosses the predetermined reference level values $rl_1 \div rl_4$. They are marked as level-crossing events. The subsequent quantization of the time interval between two consecutive level-crossing events is performed with a clock.

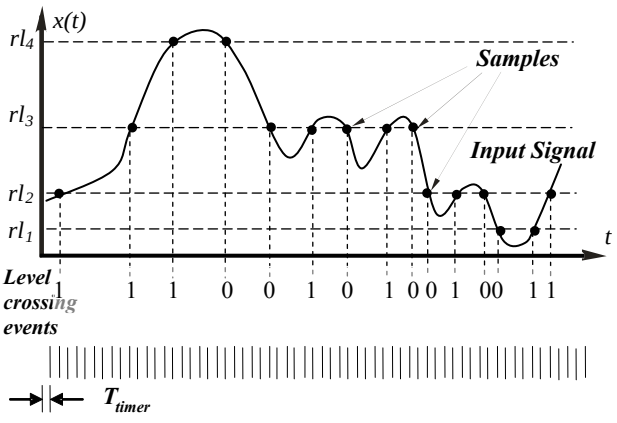


Fig. 1. Adaptive level-crossing sampling

Existing implementations of non-adaptive LC

sampling [5,6] are using different numbers of reference levels. Limited number of reference levels requires their optimal usage. It is shown in [7] that non-equidistantly spaced reference levels are more preferable. We are expecting that the adaptation of reference levels to the input signal will result in the decreased number of levels compare with the number of equidistantly spaced reference levels. Our assumption is that at least two level-crossing events on the signal slope are necessary to obtain fairly accurate reconstructed signal. Number and spacing of reference levels can vary depending from type of the input signal. That requires the implementation of dedicated control of LC sampling and resources that are able to provide such control.

Implementation

Recent hardware implementations of LC sampling ADC are presented in [6]. In [7,8] we offered to use a microprocessor as an ADC controller for LC sampling ADC. An example of such implementation is shown in Fig.2. ADC controller with the microprocessor included is programmable. It is easy to implement adaptive LC sampling by spacing reference level values accordingly to the input signal probability distribution function and by changing timer clock frequency to adjust to the required ADC resolution. ADC controller is testing the outputs of comparators CMP1, CMP2 and reloading digital-to-analog converters DAC1, DAC2. Unlike [6] DAC1 and DAC2 outputs may be non-linear. Depending from the type of signal ADC controller allows use of several subsets of reference levels that are adapted to the specific signal.

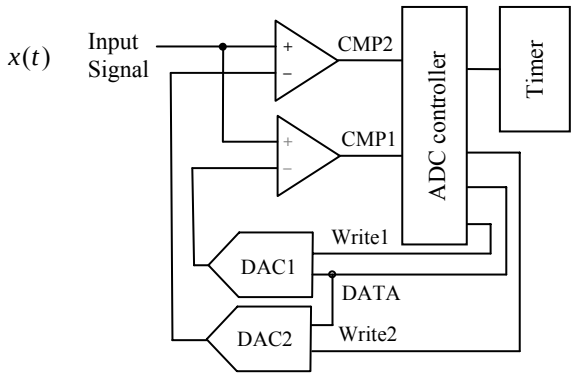


Fig. 2. Adaptive LC sampling scheme implementation

The performance of this architecture is limited by the value of the maximum loop delay $\delta_{\max}$. It determines how fast the loop delivers output changes of CMP1 and CMP2 to the inputs of CMP1 and CMP2. $\delta_{\max}$ is total of delays of ADC controller, DAC and CMP. For correct operation of non-adaptive sampling $\delta_{\max}$ should satisfy condition:

$$\delta_{\max} < 1/(2\pi f_{\max} N), \quad (1)$$

where N is the number of sampling levels and $f_{\max}$ is the maximum input signal $x(t)$ frequency. At given $\delta_{\max}$ increase of N leads to the decrease of the bandwidth of the input signal $x(t)$ that can be sampled. To satisfy (1) a compromise between N and $f_{\max}$ should be find. In case of adaptive sampling $\delta_{\max}$ should satisfy condition:

$$\delta_{\max} < 1/(2\pi f_{\max} M), \quad (2)$$

where M is the calculated number of sampling levels for the smallest used quantization step. Because of $M > N$ frequency limitations for adaptive LC sampling are stronger.

As shown in [6] the ADC resolution based on no-adaptive LC sampling scheme is determined with effective number of bits (ENOB) that includes the hardware resolution L:

$$L = \log_2 N . \quad (3)$$

and the timer clock resolution.

The finite resolution of the timer clock introduces quantization noise in time. According to [5] the quantization noise depends on the input signal amplitude and the resolution ratio R:

$$R = f_{timer} / f_{signal} , \quad (4)$$

where f_{timer} is a timer clock frequency and f_{signal} is a signal frequency. For a sinusoidal signal the signal-to-noise ratio (SNR) is independent of the signal amplitude and proportional only to the resolution ratio:

$$SNR = 20\log R - 11.2dB . \quad (5)$$

This means that doubling of the timer clock frequency f_{timer} results in an increase in SNR by one effective bit [5]. ADC resolution for adaptive LC sampling is defined similarly to ADC resolution for non-adaptive LC sampling.

The reconstruction of the sampled signal is using interpolation procedure and that can limit the maximal achievable resolution. In order to evaluate the impact of the interpolation procedure to the reconstruction of the sampled signal several LC sampling schemes are simulated with different number of reference levels N using uniformly and non-uniformly spaced reference levels.

Simulation

Simulation of the non-adaptive and adaptive LC sampling that is shown in Fig. 3 and in Fig. 4 is performed

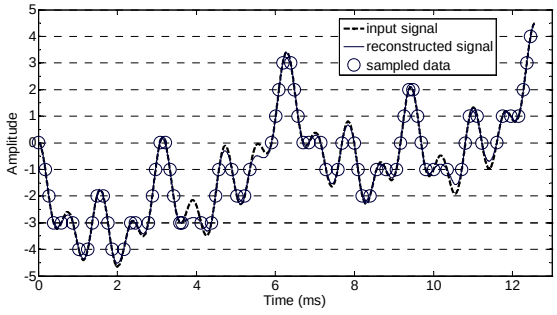


Fig. 3. Non-adaptive LC sampling and reconstruction of signal Test1

to evaluate the quality of the sampled signal after reconstruction and to estimate feasible values of reference levels N .

Two signals Test1 and Test2 were sampled using both methods of LC sampling. Afterwards LC sampling signals Test1 and Test2 were reconstructed using sampled data and MatLab cubic spline data interpolation function. The reconstructed signals were analyzed using the empirical mode decomposition (EMD) to evaluate the quality of the signal LC sampling for both sampling methods.

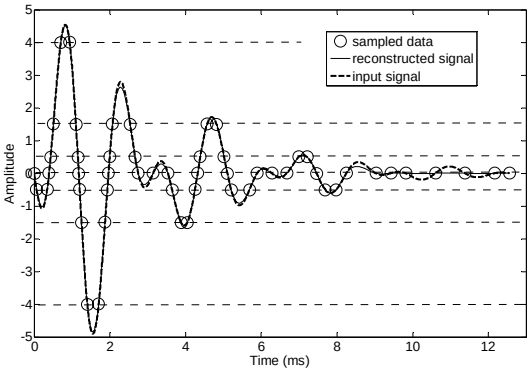


Fig. 4. Adaptive LC sampling and reconstruction of signal Test2

The EMD is proposed by Huang et al. as a new signal decomposition method for nonlinear and non-stationary signals [9]. It is used as an alternative to traditional time-frequency analysis methods. The EMD decomposes a signal into a collection of oscillatory modes, called intrinsic mode functions (IMF), which represent fast to slow oscillations in the signal. We are using the EMD based routine *rParabEmd* that performs the EMD accordingly to the paper [10]. Fig. 5 presents the EMD of the reconstructed signal Test2 that shows two decaying oscillations and some residue value.

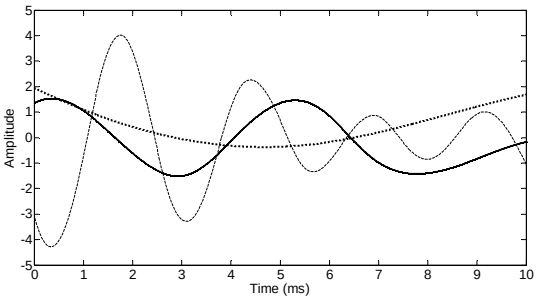


Fig. 5. Empirical mode decomposition of reconstructed signal Test2 for $N = 9$

The correlation between the input signals and the reconstructed signals as well as the correlation between the EMD of input signals and the EMD of reconstructed signal is used to compare non-adaptive and adaptive LC sampling methods. The calculations of the signal correlation were performed using the expression (6) that represents a MatLab function:

$$R = corrcoef(x, y), \qquad (6)$$

where R are correlation coefficients, x is an input and y is a reconstructed signal.

Table 1 and Table 2 show that the correlation between input signals and reconstructed signals is depending from the number of sampling levels N and the method of LC sampling. With increase of N the differences of correlation between both LC sampling methods is decreasing. In Table 3 and Table 4 correlation between EMD of input signals and EMD of reconstructed signals show poor correlation for the non-adaptive LC sampling of the signal Test2. The signal Test2 presents an exponentially decaying signal. Table 1 shows that in the case of non-uniformly spaced reference levels good correlation between an input signal and a reconstructed signal is achievable at 9 reference levels. Comparison of Table 1 with Table 2 shows that quality of the EMD of reconstructed signals essentially depends from the number of levels and their spacing. Reconstructed signals that were sampled using the adaptive spacing of reference levels have better correlation at the small number of reference levels compare to signals that were sampled using uniformly spaced reference levels. It is necessary to notice that use of non-uniformly spaced reference levels requires the knowledge of the input signal properties.

Table 1. Correlation between input signals and reconstructed signals for different N with non-adaptive spacing

N	Test 1	Test 2
5	0.8392	0.9376
7	0.9716	0.9907
9	0.9975	0.9910
13	0.9997	0.9927
16	0.9998	0.9989

Table 2. Correlation between input signals and reconstructed signals for different N with adaptive spacing

N	Test 1	Test 2
5	0.9735	0.9974
7	0.9937	0.9988
9	0.9991	0.9992
13	0.9999	1.0000
16	1.0000	1.0000

Table 3. Correlation between EMD of input signals and EMD of reconstructed signals for different N with non-adaptive spacing

N	Test 1	Test 2
5	0.7543	0.4161*
7	0.8056	0.8567*
9	0.9957	0.8617*
13	0.9994	0.8478*
16	0.9995	0.9152

*correlation is for a part of the reconstructed signal

Table 4. Correlation between EMD of input signals and EMD of reconstructed signals for different N with adaptive spacing

N	Test 1	Test 2
5	0.9409	0.8351
7	0.9640	0.9026
9	0.9981	0.9996*
13	0.9999	0.9998
16	1.0000	0.9998

Conclusions

Adaptive LC sampling ADC with the number of reference levels $N = 9 - 16$ is demonstrating high correlation between values of input signals and reconstructed signals. The EMD of reconstructed test signals confirms that using non-uniformly spaced reference levels the decomposition of reconstructed signals is improving essentially.

Limiting factors of the system are a complexity of the methods of the level-sampled signal reconstruction and of the decomposition of reconstructed signals.

References

1. **Jain A., Chang E.Y.** Adaptive sampling for sensor networks // Proceedings of the 1st Workshop on Data Management for Sensor Networks, DMSN2004. – 2004. – P. 10-16.
2. **Li Y.W., Shepard K.L., Tsividis Y.P**. Continuous time Digital Signal Processors // Proceedings of International Symposium on Asynchronous Circuits and Systems ASYNC'05. – 2005. – P. 138 – 145.
3. **Greitans M.** Processing of Non-Stationary Signal Using Level-Crossing Sampling // Proceedings of the International Conference on Signal Processing and Multimedia Applications SIGMAP2006. – Setubal, Portugal. – 2006.– P. 170-177.
4. **Greitans M., Homjakovs I.** Enhanced Digital Signal Processing of Signal-Dependently Sampled Signals // Electronics and Electrical Engineering.– Kaunas:Technologija. – 2006. – P. 9 – 14.
5. **Sayiner N., Sorensen H. N., Viswanathan T. R.** A level-crossing sampling scheme for A/D conversion.// IEEE Trans. Circuits and Systems. – 1996. – No.4(43). -P. 335 - 339.
6. **Allier E., Goulier J., Sicard G., Dezzani A., André E., Renaudin M.** A 120 nm Low Power Asynchronous ADC // International Symposium on Low Power Electronics and Design ISLPED'05. – 2005. – P. 60 -65.
7. **Baums A., Greitans M., Grunde U.** Developement of asynchronous data processing system by using general-purpose microprocessors // Electronics and Electrical Engineering.– Kaunas:Technologija. –2007. – P. 9-14.
8. **Baums A., Greitans M., Grunde U.** Level-crossing sampling using microprocessor based system // Proceedings of the International Conference on Signals and Electronic Systems ICSES'08. –Krakow, Poland.– 2008. – P. 19-23.
9. **Huang N.E., Shen Z., Long S.R., et al.** The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis // Proceedings of the Royal Society. – 1998. – vol.454. –P. 903–995.
10. **Rato, R. T., Ortigueira, M. D., and Batista, A. G.** On the HHT, its problems, and some solutions // Mechanical Systems and Signal Processing. – 2008. – Vol. 22.– P. 1374-1394.

Received 2009 05 28

Baums A., Greitans M., Grunde U. Adaptive Level-Crossing Sampling Based DSP Systems // Electronics and Electrical Engineering. - Kaunas: Technologija, 2009. - No. x(xx). - P. x-xx.

Adaptive level-crossing sampling based DSP systems are able to provide energy-efficient data acquisition and transmission. The proposed sampling mechanism is a generalization of the sampling with equidistantly spaced reference levels. Two different sampling implementations are offered. Simulation of the adaptive level-crossing sampling is performed to estimate the number of reference levels and evaluate the quality of signal reconstruction. The approach is targeting non-stationary and nonlinear signals using non-conventional digital signal processing methods. High correlation between input and reconstructed signal is supporting use of small number 9-16 sampling levels. Ill. 5, tab. 4, bibl. 10 (in English; summaries in Lithuanian, English, Russian).

Баумс А., Грейтанс М., Грунде У. Адаптивные DSP системы на основе дискретизации пересеченим порога // Электроника и электротехника.- Каунас: Технология, 2009.-№ x(xx). – С.x-xx.

Адаптивные DSP системы на основе дискретизации пересечением порога позволяют энерго-экономичный сбор данных и сокращают объем передаваемых данных. Предлагаемый метод дискретизации является обобщением метода дискретизации с равномерно расположенными порогами. Проводилось моделирование адаптивной дискретизации пересечением порога для определения числа порогов и для оценки качества восстановленного сигнала. Подход предназначен для обработки нестационарных и нелинейных сигналов с использованием цифровых методов обработки сигналов. Высокая степень корреляции подтверждает достаточность использования 9-16 порогов при дискретизации. Ил. 5, таб. 4, библ. 10 (на английском языке; рефераты на литовском, английском, русском).