DIGITALNI PROCESORI SIGNALA – Implementacija FIR filtara na DSP platformi

Vladimir Petrović, Aleksandra Lekić

Univerzitet u Beogradu – Elektrotehnički fakultet

2018/2019

FIR filtar

- Predstavlja se sa L odbiraka impulsnog odziva $h[n]$. Za svako $n \geq L$ važi da je $h[n] = 0$.
- Uvek je stabilan što se postiže odsustvom povratne sprege i prisustvom polova funkcije prenosa samo u nuli.
- Ima konačnu memoriju od L odbiraka.
- Mogu imati linearnu faznu karakteristiku (nema fazne distorzije) – za simetričan ili antisimetričan impulsni odziv.
- Manje izražene *finite-precision* greške nego kod IIR filtra.
- Lako se implementira u DSP.
- Potreban je značajno veći red FIR filtra za postizanje istih karakteristika kao kod IIR filtra.

Karakteristike filtra

LTI filtar karakterišu:

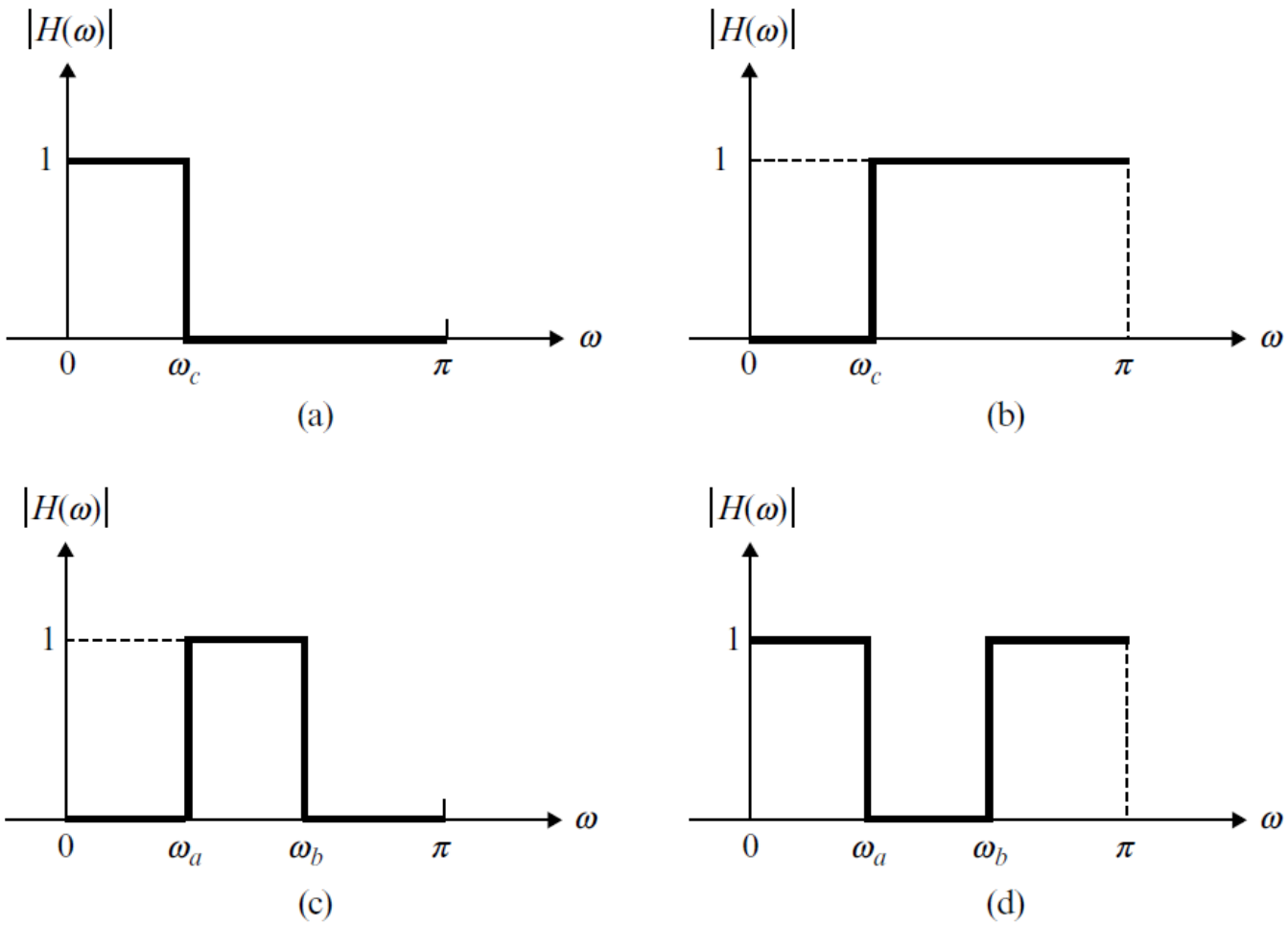
- amplitudska i fazna karaktersitika

$$|Y(\omega)| = |X(\omega)|\,|H(\omega)|$$

$$\phi_Y(\omega) = \phi_X(\omega) + \phi_H(\omega)$$

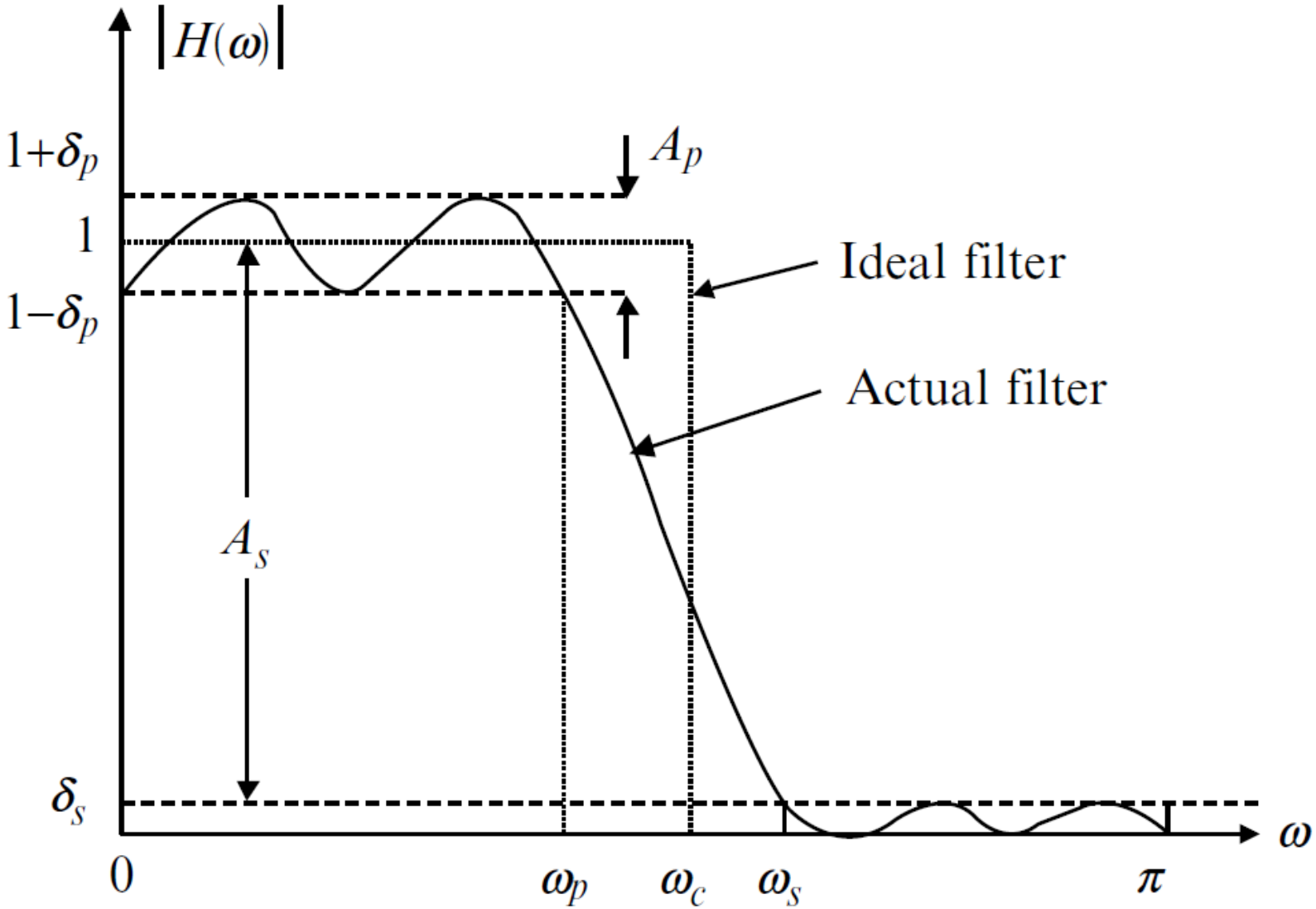
- stabilnost
- vreme uspona (*rising time*)
- *settling time*
- *overshot*

Vrste filtara



Slika: (a) *lowpass*, (b) *highpass*, (c) *bandpass*, (d) *bandstop* filtar.

Specifikacija filtra



δ_p - *passband peak deviation*
δ_s - *stopband peak deviation*
Ω_p - *passband edge frequency*
Ω_s - *stopband edge frequency*

$$1 - \delta_p \leq |H(\Omega)| \leq 1 + \delta_p \qquad 0 < \Omega < \Omega_p$$
$$|H(\Omega)| < \delta_s \qquad \Omega_s \leq \Omega \leq \pi$$
$$\alpha_p = 20 \log_{10} \left(\frac{1 + \delta_p}{1 - \delta_p} \right) \text{ [dB]}$$
$$\alpha_s = -20 \log_{10} \delta_s \text{ [dB]}$$

- prenosna funkcija ima L nenultih koeficijenata

$$H(z) = \sum_{i=0}^{L-1} b_i\, z^{-i} = \sum_{n=-\infty}^{+\infty} h[n]\, z^{-n} = \begin{cases} \forall n \in [0,\, L-1] & h[n] = b_n \\ \forall n \notin [0,\, L-1] & h[n] = 0 \end{cases}$$

- ima linearnu faznu karakteristniku i konstantno grupno kašnjenje

$$\phi(\Omega) = \begin{cases} -M\,\Omega & H(\Omega) \geq 0 \\ \pi - M\,\Omega & H(\Omega) < 0 \end{cases}$$

$$M = \text{floor}\left(\frac{L-1}{2}\right)$$

- linearnost faze važi ukoliko važi simetrija koeficijenata $b_l = b_{L-1-l}$ ili antisimetrija koeficijenata $b_l = -b_{L-1-l}$
- može da se pojednostavi funkcija prenosa

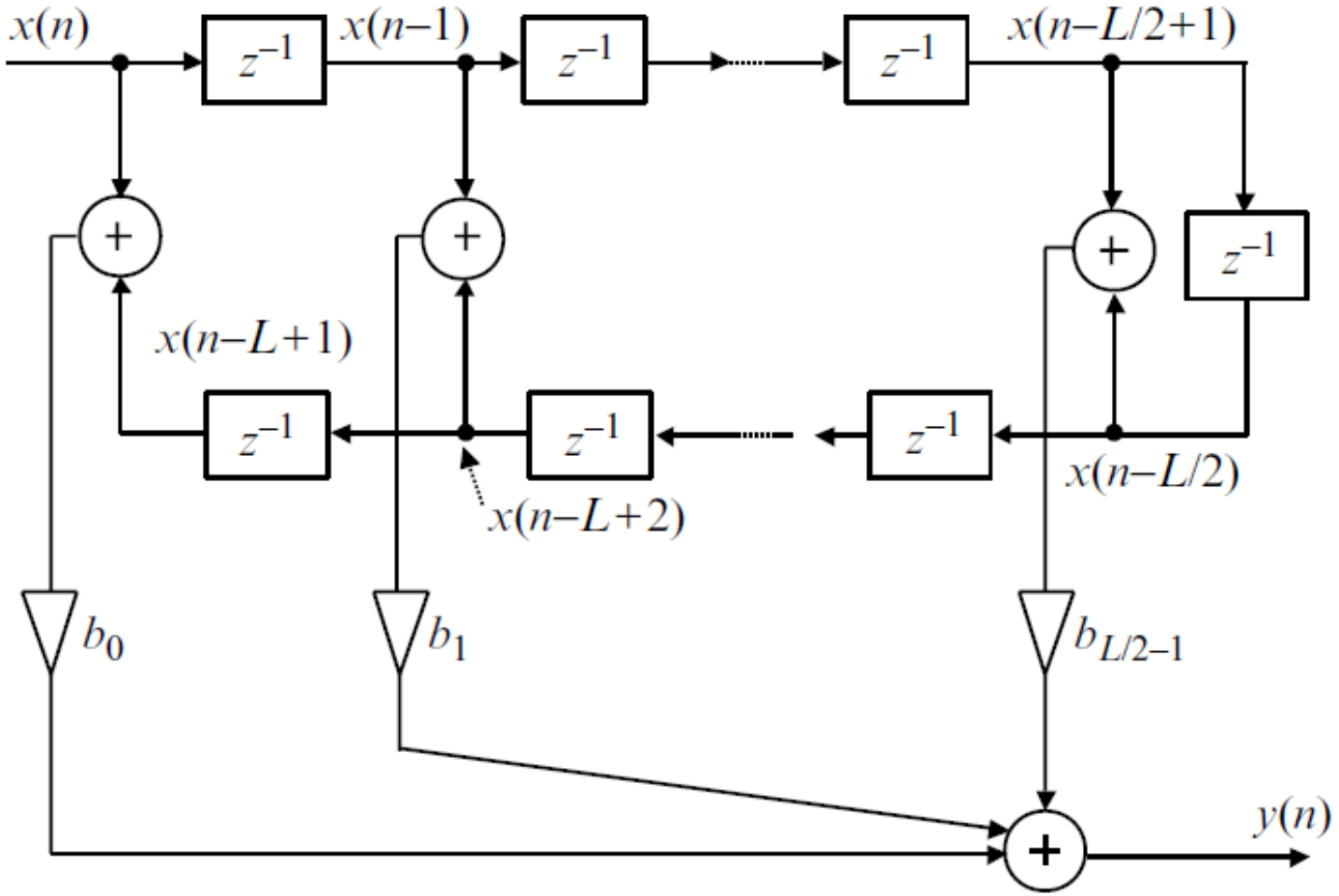
$$H(z) = b_0 \left(1 + z^{-L+1}\right) + \cdots + b_{L/2-1} \left(z^{-L/2+1} + z^{-L/2}\right)$$

- jednačina postaje

$$y[n] = \sum_{l=0}^{L/2-1} b_l \left(x[n-l] \pm x[n-L+1+l]\right)$$

- specijalne asemblerske funkcije `TMS320C55xx`: `FIRSADD` i `FIRSSUB`

Primer realizacije simetričnog FIR filtra:



Realizacija FIR filtra

- *sample-by-sample* ili *block-by-block* računanje
- linearna ili brza konvolucija

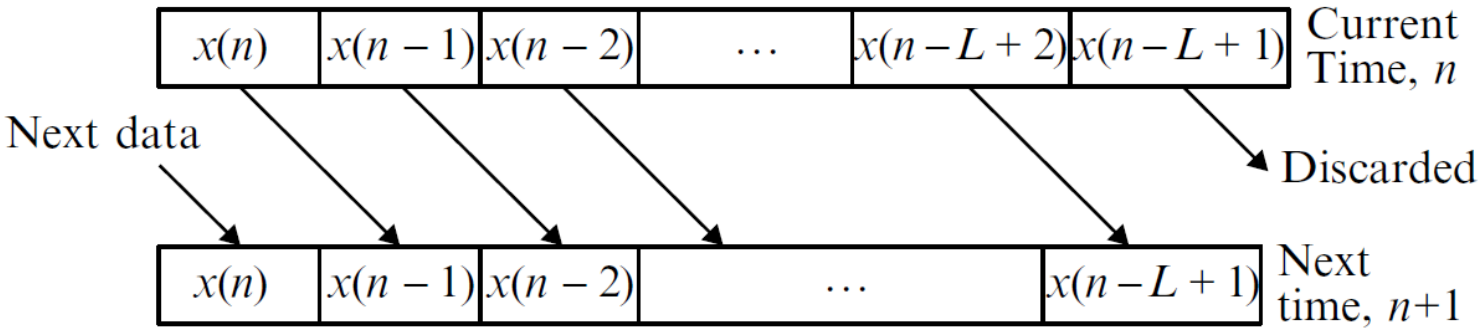
$$y[n] = \sum_{l=0}^{+\infty} h[l]\, x[n-l]$$

Linearna konvolucija obuhvata

- *"folding"* oko $l = 0$ - $x[l]$ se „presavija" da se dobije $x[-l]$
- šiftovanje - $x[-l]$ se pomera udesno za n odbiraka i daje $x[n-l]$
- množenje - $h[l]\,x[n-l]$
- sabiranje

Ako je veličina ulaznog signala M odbiraka, a impulsnog odziva FIR filtra L, izlazni signal ima $L + M - 1$ odbirak.

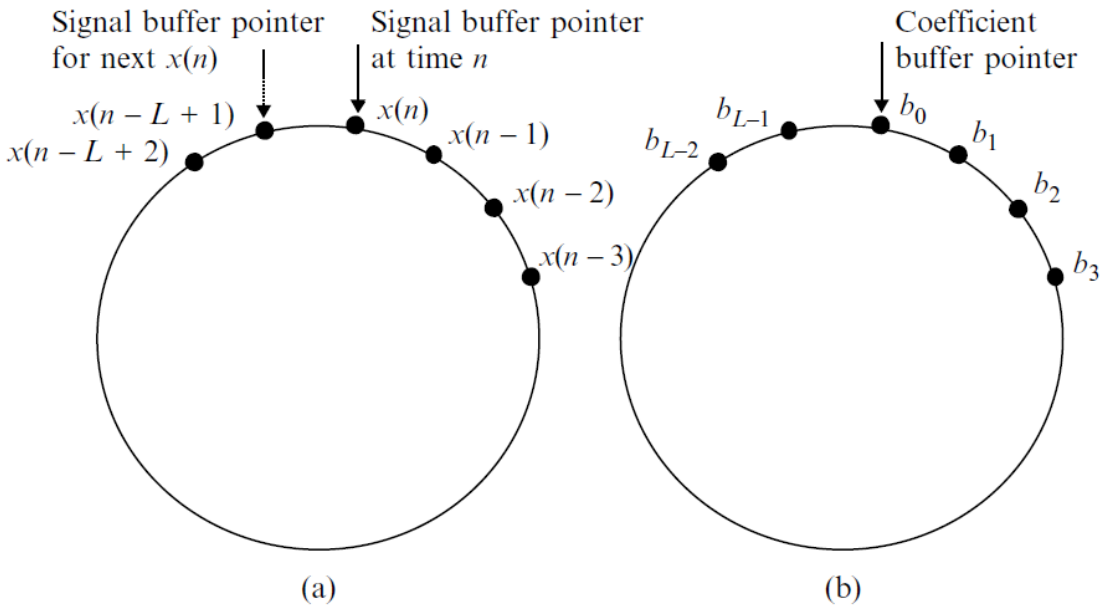
- koeficijenti FIR filtra su konstantni, ali se ulazni signal menja sa svakim periodom odabiranja



- signalni bufer se osvežava svake periode odabiranja i odbirci se pomeraju

Cirkularni bufer

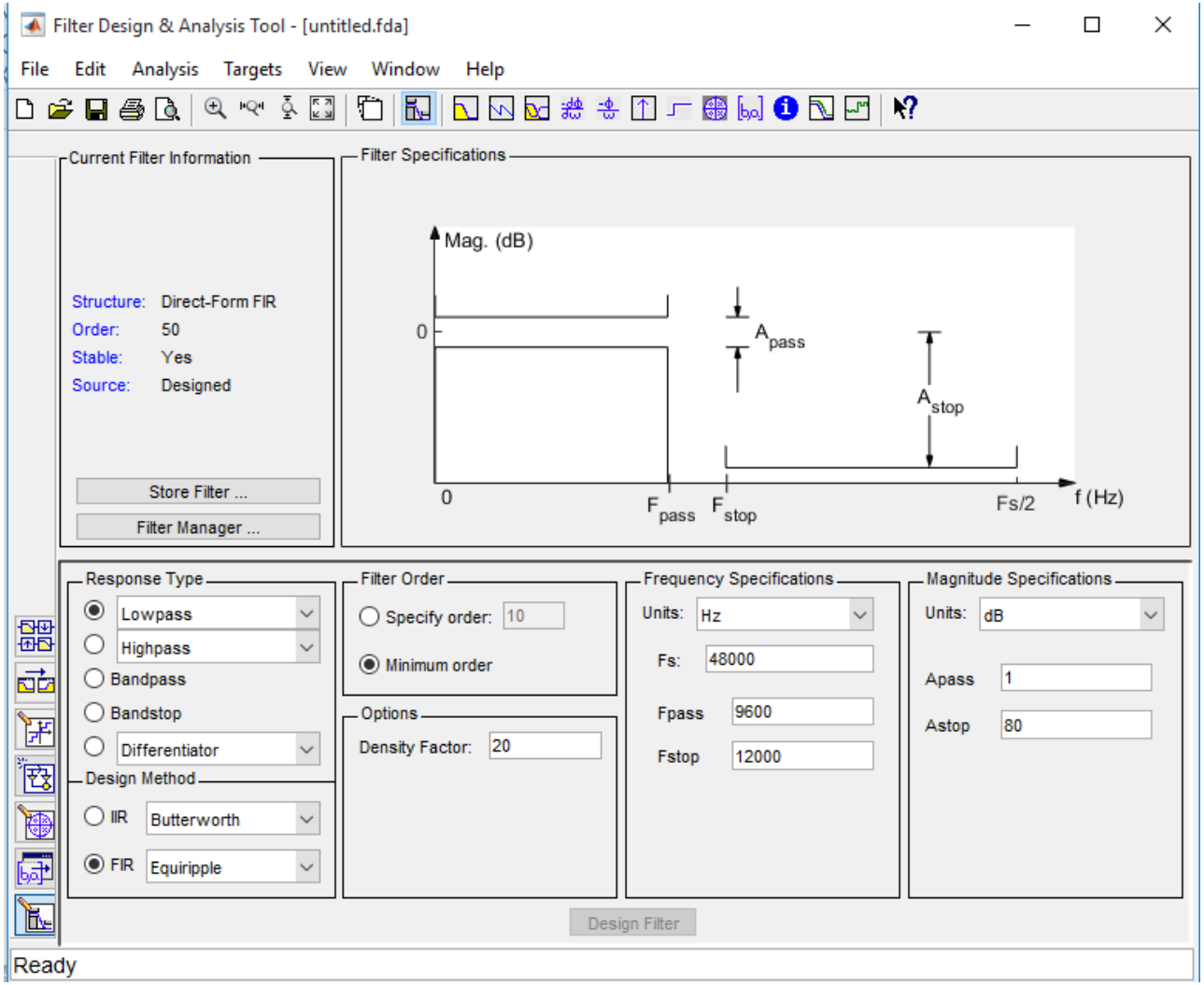
- podaci se u kružnom baferu ne šiftuju, već se početna adresa bafer pomera *counterclockwise* svake periode
- prethodni odbirci su poređani *clockwise* od početno $x[n]$
- nakon računanja $y[n]$, pomera se pokazivač bafera na $x[n-L+1]$ i tu se upisuje novi podatak ($x[n+1]$)



Slika: (a) kružni bafer za signal; (b) kružni bafer koeficijenata.

Projektovanje filtra u MATLAB-u korišćenjem paketa *fdatool*:

- pokretanje komandom *fdatool*
- otvara se prozor sa slike

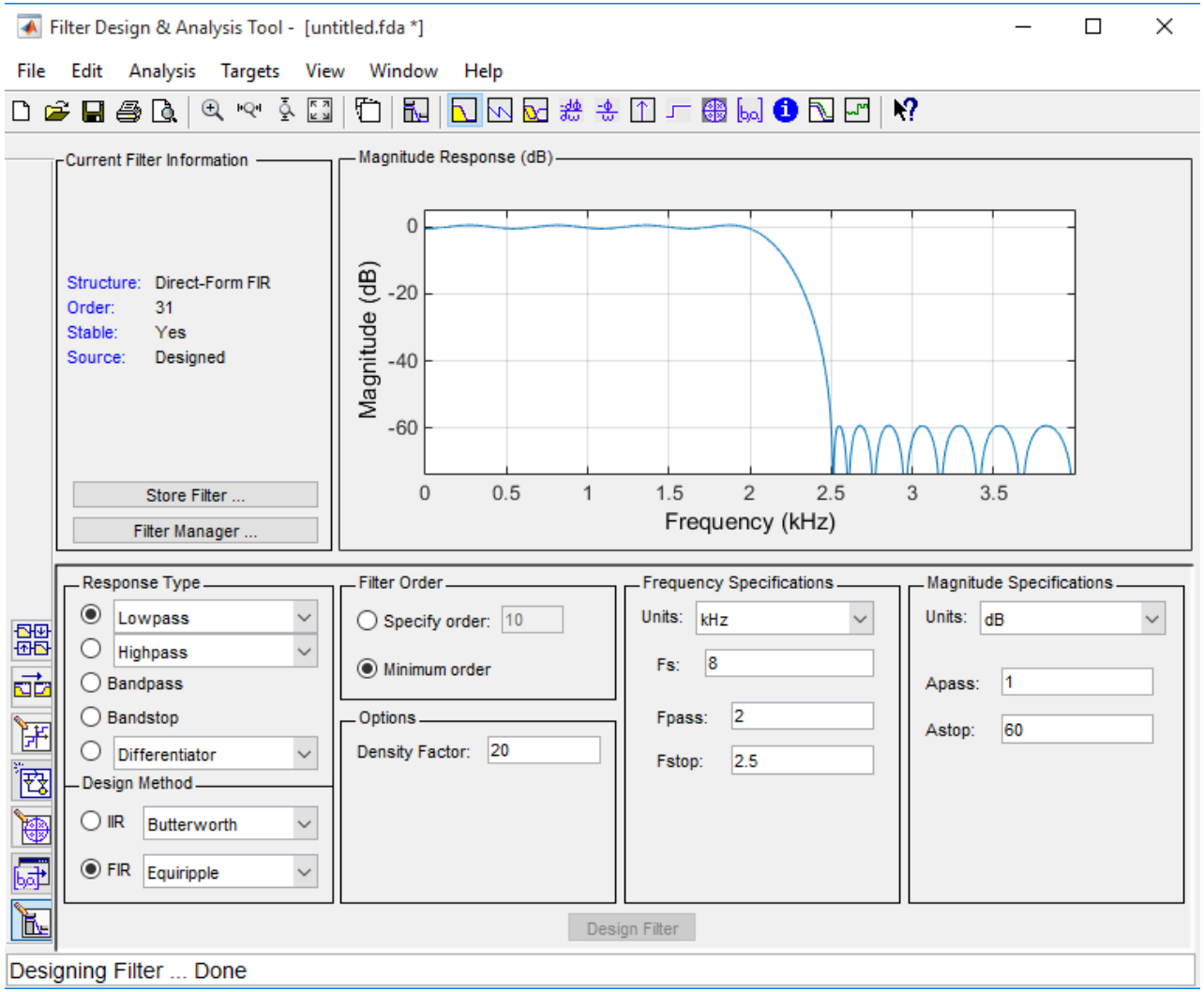


Primer

Dizajnirati NF FIR filtar

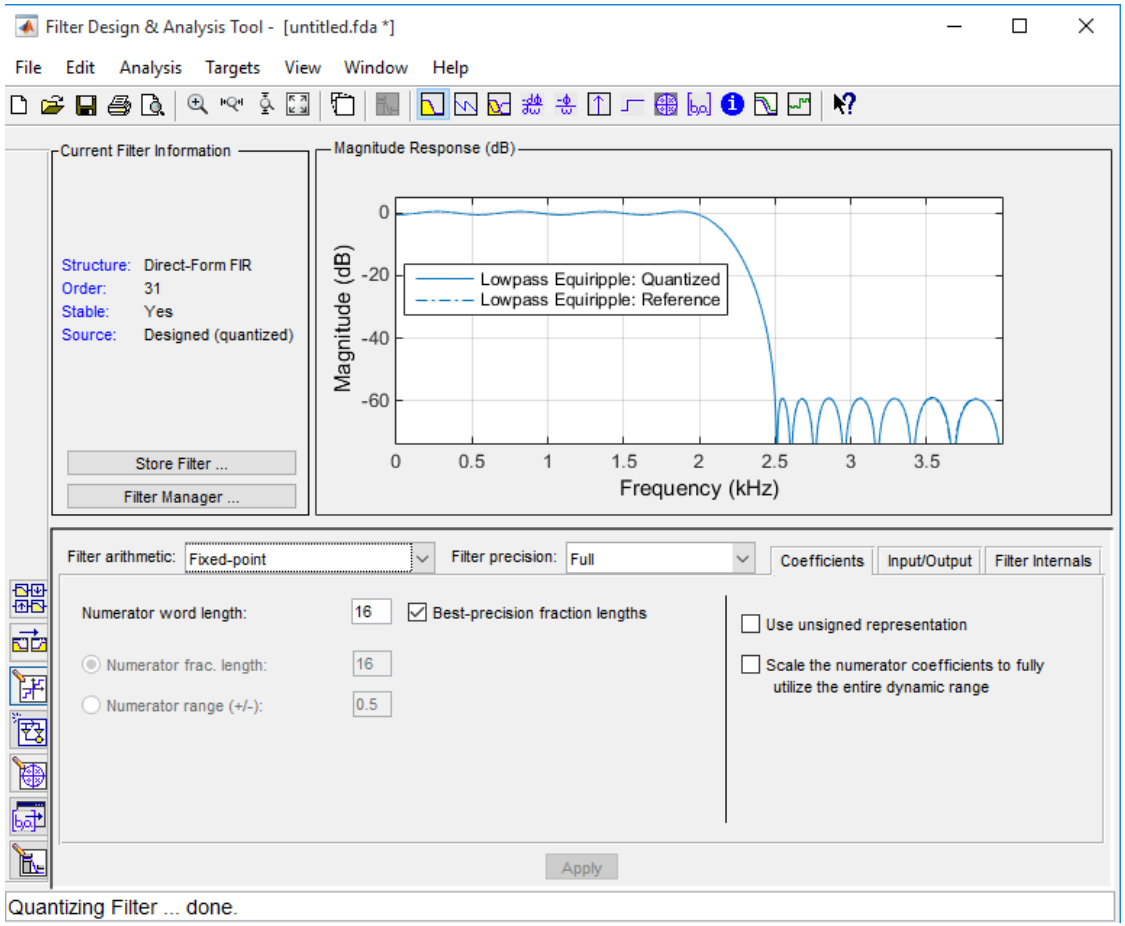
- učestanost odabiranja $F_S = 8$ kHz
- *passband cutoff* učestanost $F_{pass} = 2$ kHz
- *stopband cutoff* učestanost $F_{stop} = 2.5$ kHz
- *passband ripple* $A_{pass} = 1$ dB
- *stopband attenuation* $A_{stop} = 60$ dB

Dizajnirani filtar



Kvantizacija koeficijenata

- `Quantization Parameters` ikonica
- podešavanje preciznosti, dužine reči, *fixed-point*/*floating-point*...

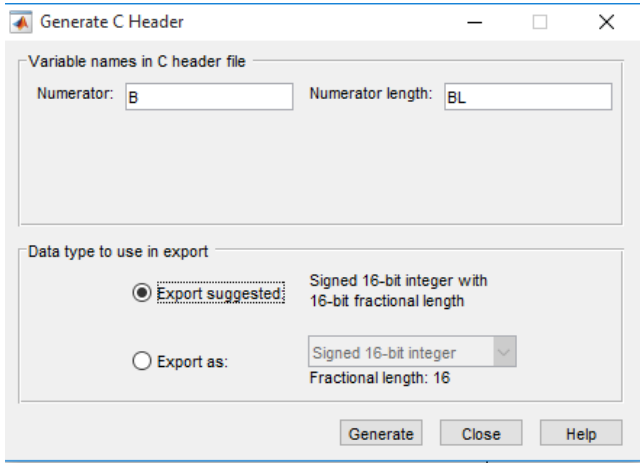


Export to ...

- File → Export → Coefficient File (ASCII) and Format: Decimal, Hexadecimal, Binary
- čuva se u .fcf fajlu

Generate C header

- Targets → Generate C header



Primeri realizacije FIR filtra

1. *FIR Filtering Using Fixed-Point C*
2. *FIR Filtering Using C55xx Assembly Program*
3. *Symmetric FIR Filtering Using C55xx Assembly Program*
4. *Optimization Using Dual-MAC Architecture*
5. *Real-Time FIR Filtering*

FIR Filtering Using Fixed-Point C

Pregled fajlova projekta

Fajl	Opis
`fixedPointBlockFirTest.c`	program za testiranje blok FIR filtra
`fixedPointBlockFir.c`	C funkcija za *fixed-point* blok FIR filtar
`fixedPointFir.h`	C header fajl
`firCoef.h`	fajl sa koeficijentima FIR filtra
`tistdtypes.h`	definicija standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni fajl

Zadatak

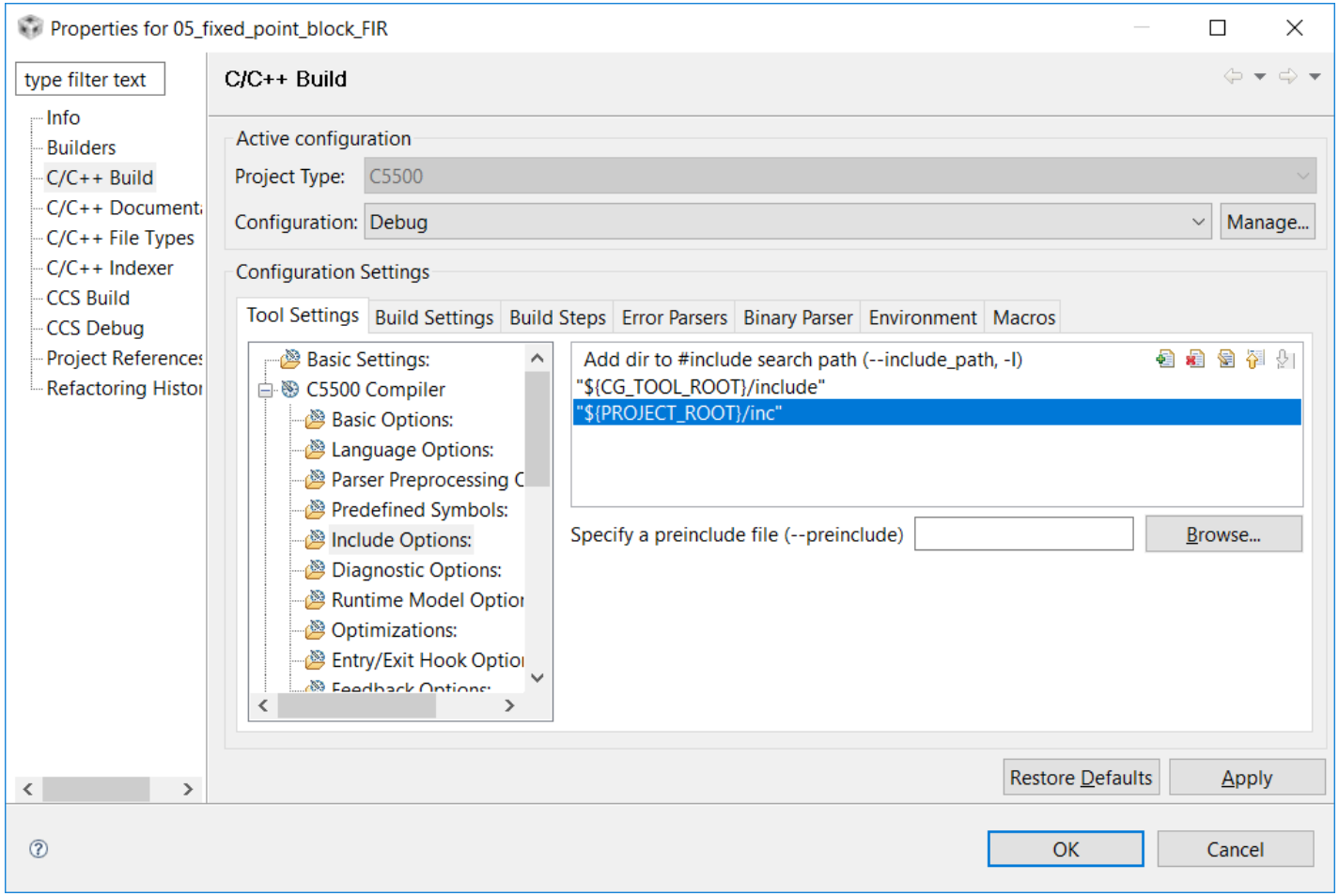
- Otvoriti CCS, prekopirati projekat `fixedPoint_BlockFIR` i importovati ga.
- `Load program` i pokrenuti program korišćenjem ulaznog fajla `input.pcm`.
- Poslušati ulazni i izlazni fajl.
- Proveriti efikasnost programa.
- Prikazati amplitudske karakteristike ulaznog i izlaznog signala.

- Implementira se *fixed-point* blok FIR filtar.
- Ulazni signal je semplovan učestanošću $F_S = 8$ kHz i sastoji se od tri sinusoide frekvencija 800 Hz, 1800 Hz i 3300 Hz.
- FIR filtar je propusnik opsega učestanosti od 1600 Hz do 2000 Hz.

Implementira se Q15 format za odbirke u opsegu -1 do $1 - 2^{-15}$.

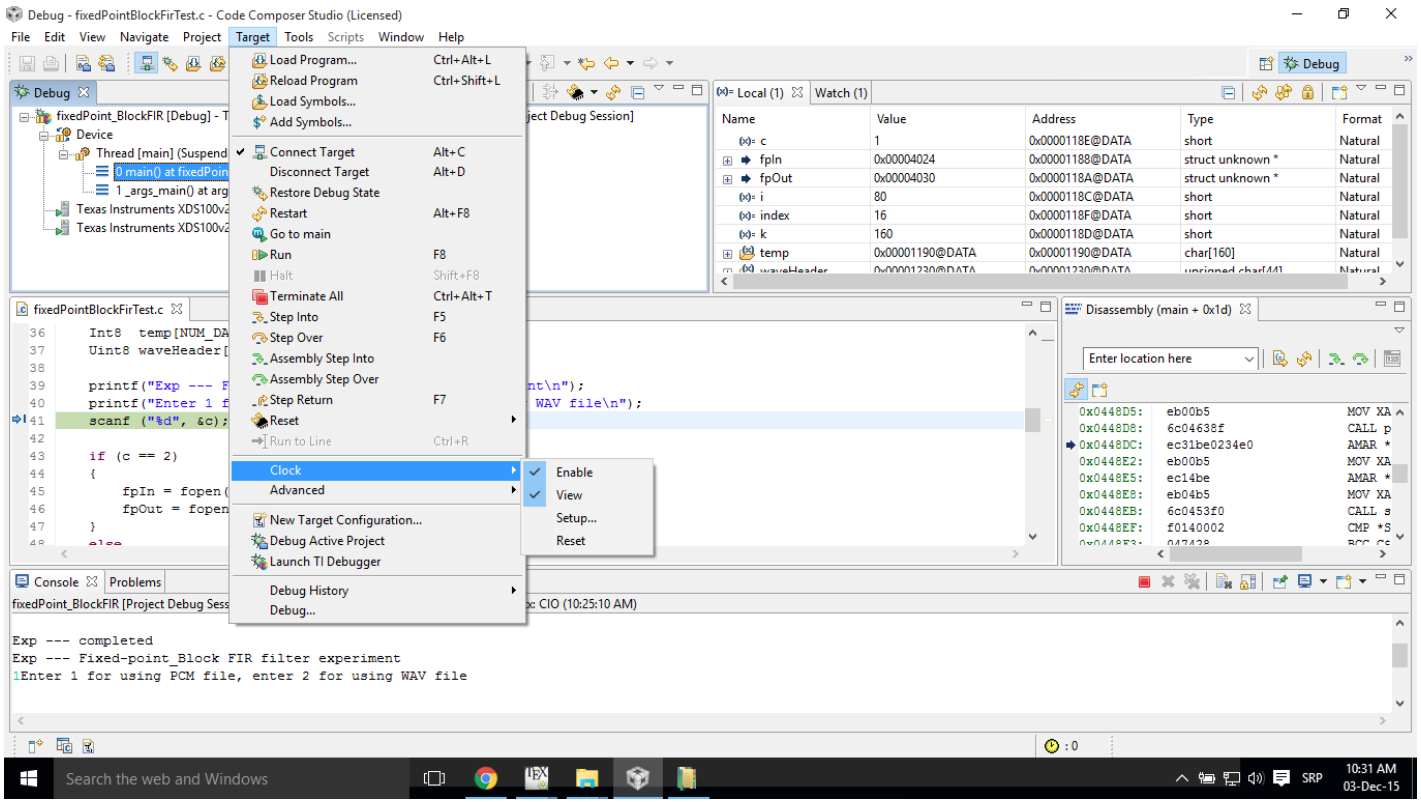
```
1  #include "tistdtypes.h"
2  #include "fixedPointFir.h"
3  /* Define DSP system memory map */
4  #pragma CODE_SECTION(fixedPointBlockFir, ".text:fir");
5
6  void fixedPointBlockFir(Int16 *x, Int16 blkSize, Int16 *h, Int16 order,
7                          Int16 *y, Int16 *w, Int16 *index)
8  {
9      Int16 i,j,k;
10     Int32 sum;
11     Int16 *c;
12
13     k = *index;
14     for (j=0; j<blkSize; j++)                    // Block processing
15     {
16         w[k] = *x++;                          // Get the current data to delay line
17         c = h;
18         for (sum=0, i=0; i<order; i++)        // FIR filter processing
19         {
20             sum += *c++ * (Int32)w[k++];
21             if (k == NUM_TAPS)                // Simulate circular buffer
22                 k = 0;
23         }
24         sum += 0x4000;                        // Rounding
25         *y++ = (Int16)(sum>>15);              // Save filter output
26
27       if (k-- <=0)                            // Update index for next time
28             k = NUM_TAPS-1;
29     }
30     *index = k;                               // Update circular buffer index
31 }
```

- Prekopirati u `workspace` projekat `fixedPoint_BlockFIR` i napraviti novi projekat u CCS-u sa istim imenom.
- Obratiti pažnju na `include` putanje kao na sledećoj slici.



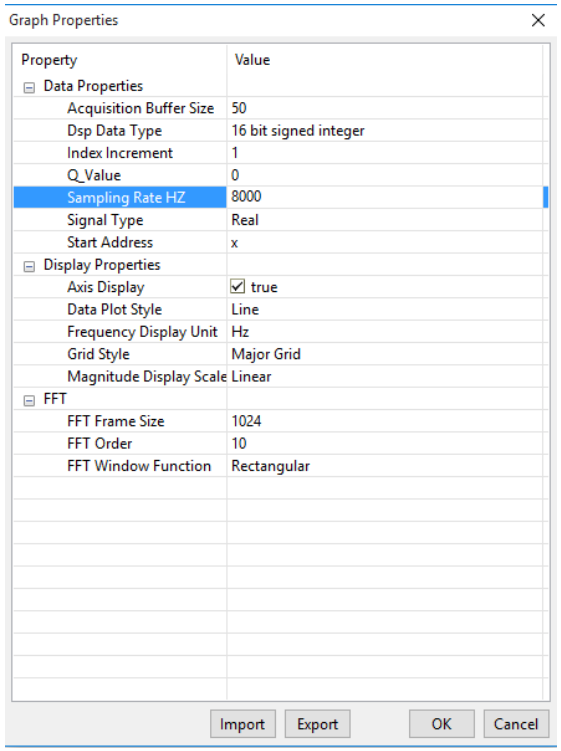
- `Build project`.

- Target → Debug Active Project
- Connect Target
- Target → Load program
- Podesiti clock prema slici.



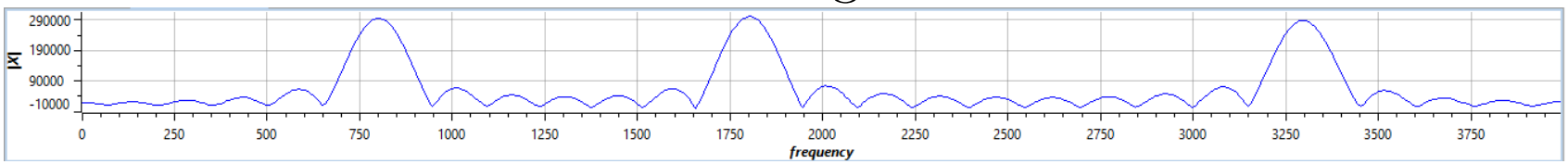
Rezultati

- Funkcija za računanje izlaznog signala iz FIR filtra `fixedPointBlockFir` se izvršava 147170 taktnih intervala!
- Grafici se crtaju `Tools` → `Graph` → `FFT Magnitude` i podešavanjem ulaznog signala prema slici.

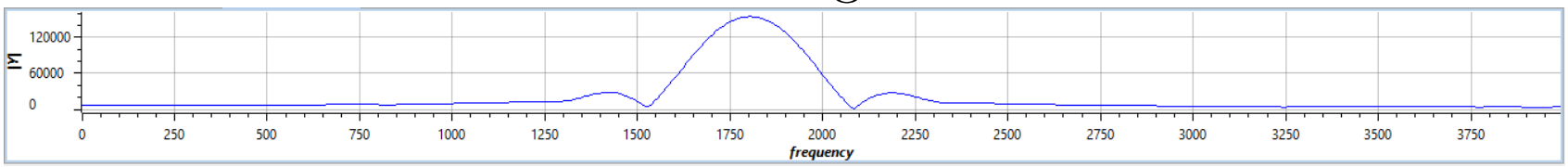


Rezultati

Ulazni signal:



Izlazni signal:



FIR Filtering Using C55xx Assembly Program

Pregled fajlova projekta

Fajl	Opis
`blockFirTest.c`	program za testiranje blok FIR filtra
`blockFir.asm`	asemblerski program za blok FIR filtriranje
`blockFir.h`	C header fajl
`blockFirCoef.h`	fajl sa koeficijentima FIR filtra
`tistdtypes.h`	definicija standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni fajl

- Implementira se asemblerska funkcija koja koristi kružne koeficijente i signalne bafere.
- Koriste se isti FIR filtar i ulazni signal kao u prethodnom primeru.
- Potrebno je red filtra + 4 ciklusa da se procesira jedan odbirak izlaznog signala. Filtar sa 48 odbiraka zahteva 52 ciklusa za generisanje jednog odbirka izlaznog signala.

Zadatak

- Prekopirati projekat `asm_BlockFIR` i importovati ga u CCS.
- `Build program` i `Load program` koristeći ulazne podatke `input.pcm`.
- Prikazati amplitudsku karakteristiku ulaznog i izlaznog signala radi provere rada programa.
- Izmeriti potreban broj taktnih intervala za izvršavanje asemblerske funkcije.

Asemblerske direktive

.bss

Rezerviše nealociranu memoriju za podatke u `.bss` sekciji. Često se koristi za alociranje memorije za ulazno/izlazne bafere.

```
.bss xn_buffer, size_in_words
```

.data

Daje direktivu asembleru da smešta kod u `.data` sekciju memorije. Primer su tabele podataka.

.sect

Definiše kod ili podatke i „govori” asembleru da ih smešta u posebno definisanu sekciju memorije. Zgodno za razdvajanje logičkih celina u kodu.

```
.sect "user_section"
```

Asemblerske direktive

.usect

Slično kao i `.bss` rezerviše neinicijalizovani memorijski prostor i deli ga u posebno definisane sekcije. Zgodno za razdvajanje velikih sekcija podataka u logičke celine.

```
symbol .usect ''section_name'', size_in_words
```

.text

Daje direktivu asembleru da smešta kod u `.text` sekciju memorije u koju se podrazumevano uvek smešta kod samog programa.

.int/.word

Smešta jednu ili više 16-bitnih `integer` vrednosti na uzastopne memorijske lokacije u trenutnoj memorijskoj sekciji.

```
data1 .word 0x1234
data2 .int 1010111b
```

Asemblerske direktive

.set/.equ

Dodeljuje vrednost simbolu. Za definisanje konstanti.

```
symbol .set value
```

.global/.def/.ref

Simbol ili funkcija se definiše kao globalna sa `.global`. Direktiva `.ref` definiše globalnim simbol definisan u tekućem fajlu, a `.ref` referencira kao globalan simbol definisan u nekom drugom fajlu.

```
.def symbol_name
```

.include/.copy

Čita *source file* nekog drugog fajla.

```
.include "file_name"
```

Mixed C and Assembly Programming

- Konvencija naziva - koristi se _ kao prefiks varijabli i rutina definisanih u asembleru koje će se koristiti u C fajlovima. U C-u se pozivaju kao `asm_var` ili `asm_func`, dok su u asembleru definisane sa `_asm_var` ili `_asm_func`.
- Varijable koje su globalne i za C i za asemblerski kod moraju biti globalne (`.global`/`.def`/`.ref`).
- C kompajler podešava bit `CPL = 1` za korišćenje `SP` pri ulasku u asemblersku rutinu. Poželjno je indirektno adresiranje. Za `DP` direktno se mora postaviti `CPL = 0`.
- Prosleđivanje argumenata striktno po pravilima C55xx kompajlera.

Mixed C and Assembly Programming

- Pozivom rutine se argumenti prosleđuju redosledom kojim su dati u funkciji.
- Povratni argumenti funkcije se vraćaju u registru `AC0` ako je podatak 32-bitni i `T0` za 16-bitni. `DP` se upisuje u `XAR0` i struktura se vraća na lokalni stek.

Argument type	Register assignment order
16-bit data pointer	AR0, AR1, AR2, AR3, AR4
23-bit data pointer	XAR0, XAR1, XAR2, XAR3, XAR4
16-bit data	T0, T1, AR0, AR1, AR2, AR3, AR4
32-bit data	AC0, AC1, AC2

Mixed C and Assembly Programming

- Upotreba registara koji se koriste za čuvanje podataka između tekuće i pozvane funkcije je strogo definisana.

Registers	Preserved by	Used for
AC0–AC2	Calling function Save-on-call	16, 32, or 40-bit data 24-bit code pointers
(X)AR0–(X)AR4	Calling function Save-on-call	16-bit data 16 or 23-bit pointers
T0 and T1	Calling function Save-on-call	16-bit data
AC3	Called function Save-on-entry	16, 32, or 40-bit data
(X)AR5–(X)AR7	Called function Save-on-entry	16-bit data 16 or 23-bit pointers
T2 and T3	Called function Save-on-entry	16-bit data

Asemblerske direktive fajla `blockFir.asm` u kome je definisana funkcija `blockFir`. Funkcija je prikazana na sledećem slajdu.

```
.mmregs

.sect ".text:fir"
.align 4

.def _blockFir

;------------------------------------------------------------------------
;   void blockFir(Int16 *x,         => AR0;   Int16 blkSize,       => T0
;                 Int16 *h,         => AR1;   Int16 order,         => T1
;                 Int16 *y,         => AR2;   Int16 *w,            => AR3
;                 Int16 *index)     => AR4
;------------------------------------------------------------------------
```

```
1  _blockFir:
2       pshm   ST1_55                  ; Save ST1, ST2, and ST3
3       pshm   ST2_55
4       pshm   ST3_55
5
6       or     #0x340,mmap(ST1_55)     ; Set FRCT,SXMD,SATD
7       bset   SMUL                    ; Set SMUL
8       mov    mmap(AR1),BSA01         ; AR1=base address for coeff
9       mov    mmap(T1),BK03           ; Set coefficient array size (order)
10      mov    mmap(AR3),BSA23         ; AR3=base address for signal buffer
11      or     #0xA,mmap(ST2_55)       ; AR1 & AR3 as circular pointers
12      mov    #0,AR1                  ; Coefficient start from h[0]
13      mov    *AR4,AR3                ; Signal buffer start from w[index]
14   ||  sub    #1,T0                  ; T0=blkSize-1
15      mov    T0,BRC0                 ; Initialize outer loop to blkSize-1
16      sub    #3,T1,T0                ; T0=order-3
17      mov    T0,CSR                  ; Initialize inner loop order-2 times
18   ||  rptblocal sample_loop-1       ; Start the outer loop
19      mov    *AR0+,*AR3              ; Put the new sample to signal buffer
20      mpym   *AR3+,*AR1+,AC0         ; Do the 1st operation
21   ||  rpt    CSR                    ; Start the inner loop
22      macm   *AR3+,*AR1+,AC0
23      macmr  *AR3,*AR1+,AC0          ; Do the last operation with rounding
24      mov    hi(AC0),*AR2+           ; Save Q15 filtered value
25 sample_loop
26
27      popm   ST3_55                  ; Restore ST1, ST2, and ST3
28      popm   ST2_55
29      popm   ST1_55
30      mov    AR3,*AR4                ; Update signal buffer index
31      ret
```

Rezultati

- Funkcija za računanje izlaznog signala iz FIR filtra `blockFir` se izvršava 4135 taktnih intervala što je mnogo manje od 147170 taktnih intervala!

Symmetric FIR Filtering Using C55xx Assembly Program

Pregled fajlova projekta

Fajl	Opis
`symFirTest.c`	program za testiranje simetričnog FIR filtra
`symFir.asm`	asemblerska rutina simetričnog FIR filtra
`symFir.h`	C header fajl
`symFirCoef.h`	fajl sa koeficijentima FIR filtra
`tistdtypes.h`	definicija standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni fajl

- Implementira se asemblerska funkcija koja koristi kružne koeficijente i kružne bafere.

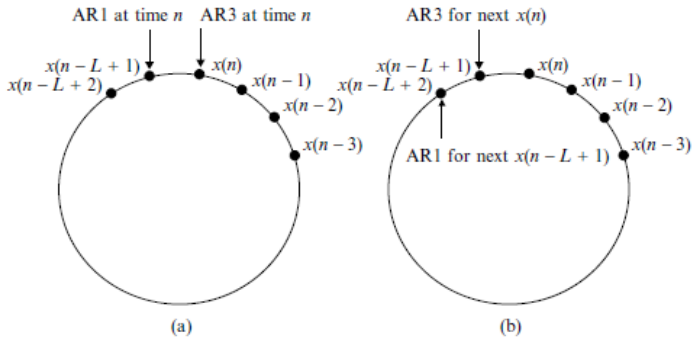


Figure 5.26 Circular buffer for accessing signals for a symmetric FIR filtering. The pointers to $x(n)$ and $x(n-L+1)$ are updated at the counter-clockwise direction: (a) circular buffer for a symmetric FIR filter at time n, and (b) circular buffer for a symmetric FIR filter at time $n+1$

- Koriste se isti FIR filtar i ulazni signal kao u prethodnom primeru.
- Za implementaciju filtra se koristi osobina simetrije/antisimetrije čime se smanjuje broj računanja proizvoda i zbira u linearnoj konvoluciji. Pri tome se koriste instrukcije `FIRSADD` i `FIRSSUB`.
- Potrebno je red filtra / 2 + 5 ciklusa da se procesira jedan odbirak izlaznog signala. Filtar sa 48 odbiraka zahteva 29 ciklusa za generisanje jednog odbirka izlaznog signala.

Zadatak

- Prekopirati projekat `asm_BlockFIR` i importovati ga u CCS.
- `Build program` i `Load program` koristeći ulazne podatke `input.pcm`.
- Prikazati amplitudsku karakteristiku ulaznog i izlaznog signala radi provere rada programa.
- Prikazati signal na ulazu i na izlazu u vremenskom domenu.
- Izmeriti potreban broj taktnih intervala za izvršavanje asemblerske funkcije. (2312)

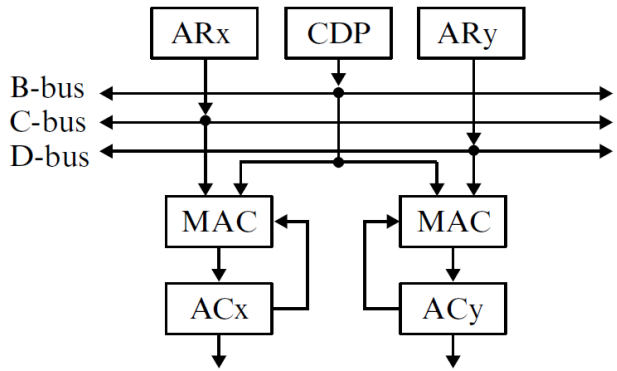
```
1       mov     mmap(T1),BK03               ; Set signal buffer size = order
2       or      #0x340,mmap(ST1_55)         ; Set FRCT,SXMD,SATD
3       bset    SMUL                        ; Set SMUL
4       mov     XAR1,XCDP                   ; CDP as coefficient pointer
5       mov     mmap(AR1),BSAC              ; Set up base address for CDP
6       sfts    T1,#-1                      ; T1 = order/2
7   ||  mov     #0,CDP                      ; Start from the 1st coefficient
8       mov     mmap(T1),BKC                ; Set the coefficient array size
9       mov     XAR3,XAR1                   ; AR1 & AR3 are signal buffer pointers
10      mov     mmap(AR3),BSA01             ; Set base address of AR1 for signal buffer
11      mov     mmap(AR3),BSA23             ; Set base address of AR3 for signal buffer
12      or      #0x10A,mmap(ST2_55)         ; CDP, AR1, AR3 are circular pointers
13      mov     *AR4,AR3                    ; AR3 is the Head of signal buffer
14      mov     *AR4,AR1                    ; AR1 is the Tail of signal buffer
15  ||  sub     #1,T0
16      amar    *AR1-                       ; Adjust tail starting point
17  ||  mov     T0,BRC0                     ; Outer loop counter blkSize-1
18      sub     #3,T1,T0                    ; Inner loop for (order/2-2) iteration
19      mov     T0,CSR
20      mov     T1,T0                       ; Set up update offset for AR1
21      sub     #2,T1                       ; Set up update offset for AR3
22      mov     *AR0+,AC1                   ; Get the first sample
23  ||  rptblocal sample_loop-1             ; To prevent overflow in addition
24      mov     #0,AC0                      ;   input is scaled to Q14 format
25  ||  mov     AC1<<#-1,*AR3               ; Put input to signal buffer in Q14
26      add     *AR3+,*AR1-,AC1             ; AC1=[x(n)+x(n-L+1)]<<16
27  ||  rpt     CSR                         ; Do order/2-2 iterations
28      firsadd *AR3+,*AR1-,*CDP+,AC1,AC0
29      firsadd *(AR3-T0),*(AR1+T1),*CDP+,AC1,AC0
30      macm    *CDP+,AC1,AC0               ; Finish the last macm instruction
31      mov     rnd(hi(AC0<<#1)),*AR2+      ; Store the rounded & scaled result
32  ||  mov     *AR0+,AC1                   ; Get next sample
33  sample_loop
```

Optimization Using Dual-MAC Architecture

Pregled fajlova projekta

Fajl	Opis
`dualMacFirTest.c`	program za testiranje dual-MAC FIR filtra
`dualMacFir.asm`	asemblerska rutina dual-MAC FIR filtra
`dualMacFir.h`	C header fajl
`dualMacFirCoef.h`	fajl sa koeficijentima FIR filtra
`tistdtypes.h`	definicija standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.pcm`	ulazni fajl

- Računaju se dva odbirka izlaznog signala paralelno u dve MAC jedinice.
- Potrebna su dva akumulatora za čuvanje dva izlazna signala paralaleno.
- U toku dual-MAC FIR implementacije čitaju se tri podatka istovremeno koristeći sve magistrale podataka: dva podatka i jedan koeficijent.
- Instrukcija dual-memory zahteva poravnanje podataka na parnu reč, što se podešava komandom `.align 4`.



Zadatak

- Prekopirati projekat `asm_BlockFIR` i importovati ga u CCS.
- `Build program` i `Load program` koristeći ulazne podatke `input.pcm`.
- Prikazati amplitudsku karakteristiku ulaznog i izlaznog signala radi provere rada programa.
- Prikazati signal na ulazu i na izlazu u vremenskom domenu.
- Izmeriti potreban broj taktnih intervala za izvršavanje asemblerske funkcije.

Real-Time FIR Filtering

Implementacija filtra korišćenjem dual-MAC arhitekture. Filtar filtrira podatke dobijene sa audio kodeka i vraća ih nazad na audio izlaz. Transfer podataka se vrši DMA transferima kako bi se omogućilo filtriranje u realnom vremenu.