

# Arhitekture i algoritmi DSP 2

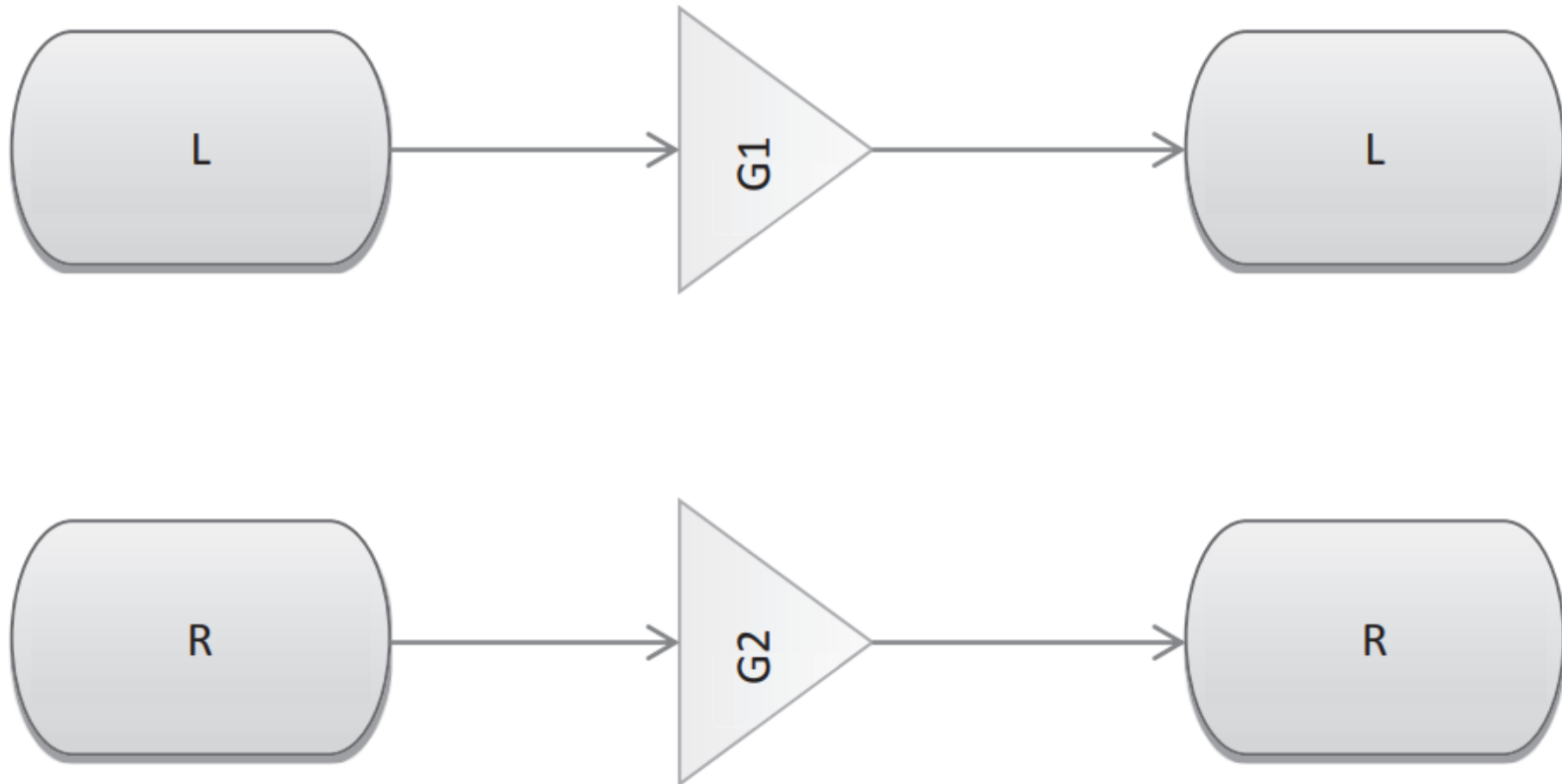


Auditorne vežbe AU-5[0]  
Metodologija razvoja DSP aplikacija

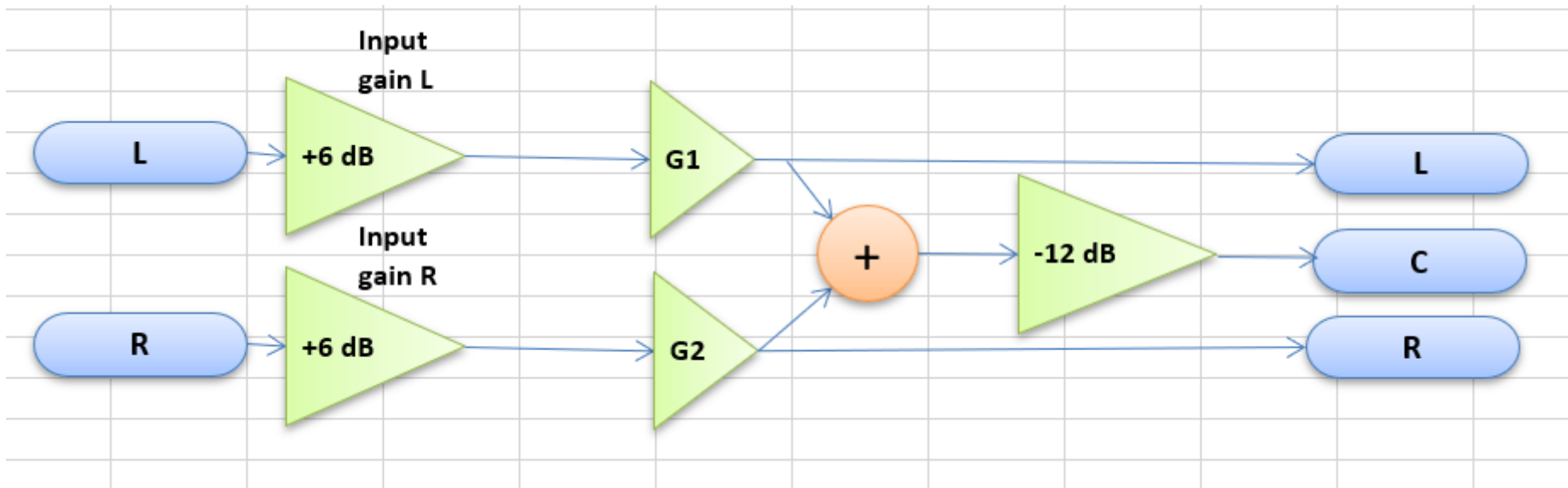


Odsek za računarsku tehniku i računarske komunikacije

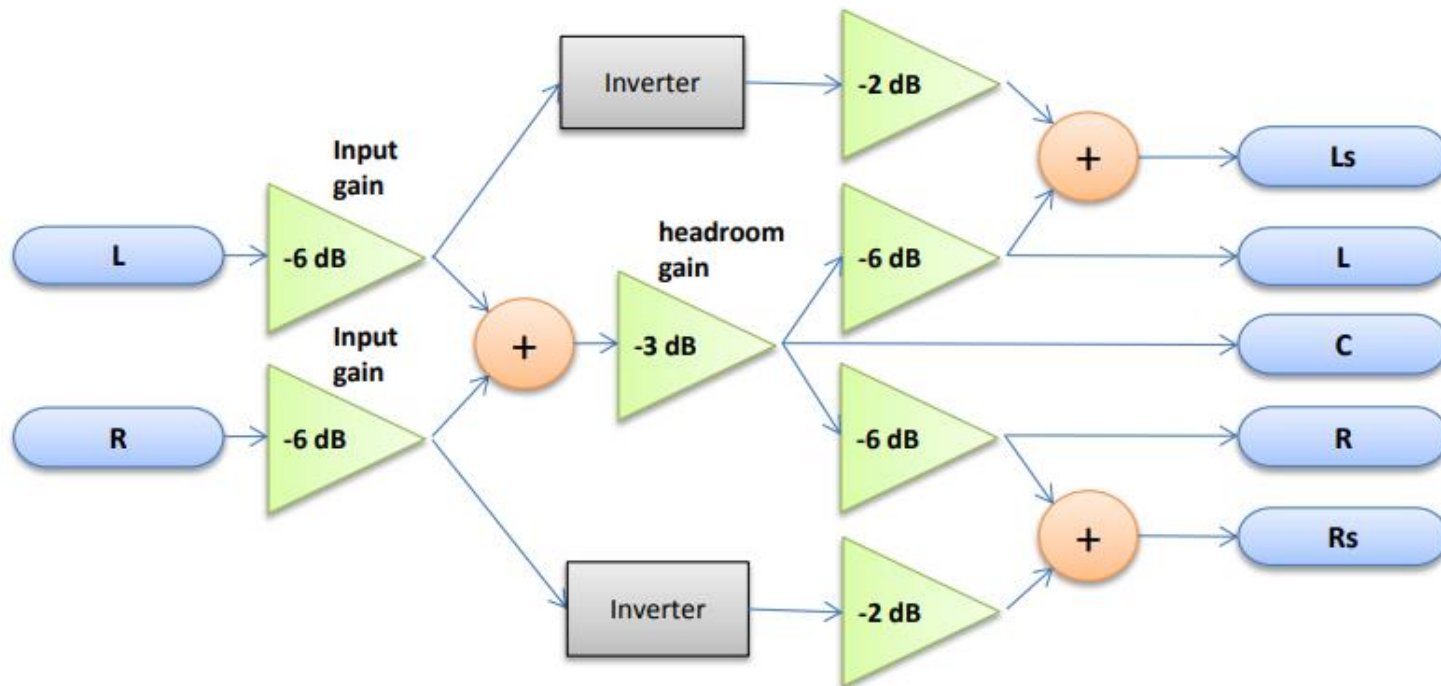
# Prvi audio sistem



# Drugi audio sistem



# Primer projektnog sistema



# Metodologija razvoja



<b>Referentni kod</b> (Model 0)	
<b>Model 1</b> Verzija koda	<i>Korak 1:</i> Optimizacije referentnog C koda
<b>Model 2</b> Verzija koda	<i>Korak 2:</i> Modifikacija algoritma i C koda za tipove podataka sa nepokretnim zarezom
<b>Model 3</b> Verzija koda	<i>Korak 3:</i> Izmene vezane za ciljnu arhitekturu
<b>Finalni kod</b> Kod izvršiv na ciljnoj platformi	<i>Korak 4:</i> Integracija u okruženje, dalje optimizacije, verifikacija

*Slika 3.1 - Tok implementacije softvera na digitalnim signal procesorima sa aritmetikom u nepokretnom zarezu*

# Metodologija razvoja kroz vežbe



Model	Radno okruženje	Aritmetika
0	Visual Studio	Floating-point native
1	Visual Studio	Floating-point native
2	Visual Studio	Fixed-point C++ emulation [fixed-point range]
3 - početni	CLIDE	Fixed-point native (in C)
3 - optimizacije	CLIDE	Fixed-point native (in ASM)
Integracija u okruženje // Finalni kod!	CLIDE	Fixed-point native

# Arhitekture i algoritmi DSP 2



Auditorne vežbe AU-5[1]  
Model 0 – Referentni kod



Odsek za računarsku tehniku i računarske komunikacije

# Model 0



- ❑ Referentni model
- ❑ C/C++ kod
- ❑ Razvija se isključivo kako bi procesing algoritam proradio (univerzalno rešenje bez obzira na konkretnu DSP platformu)
- ❑ Jednom završen i verifikovan model 0 se ne sme više menjati
- ❑ Izlazi iz svih ostalih modela se porede sa izlazima iz modela 0!
- ❑ Na vežbama se radi u Visual Studio-u (kompajler za x86)




# Potrebna (pred)znanja



- ❑ Osnove DSP obrade
- ❑ C/C++
- ❑ Rad sa Visual Studio

---

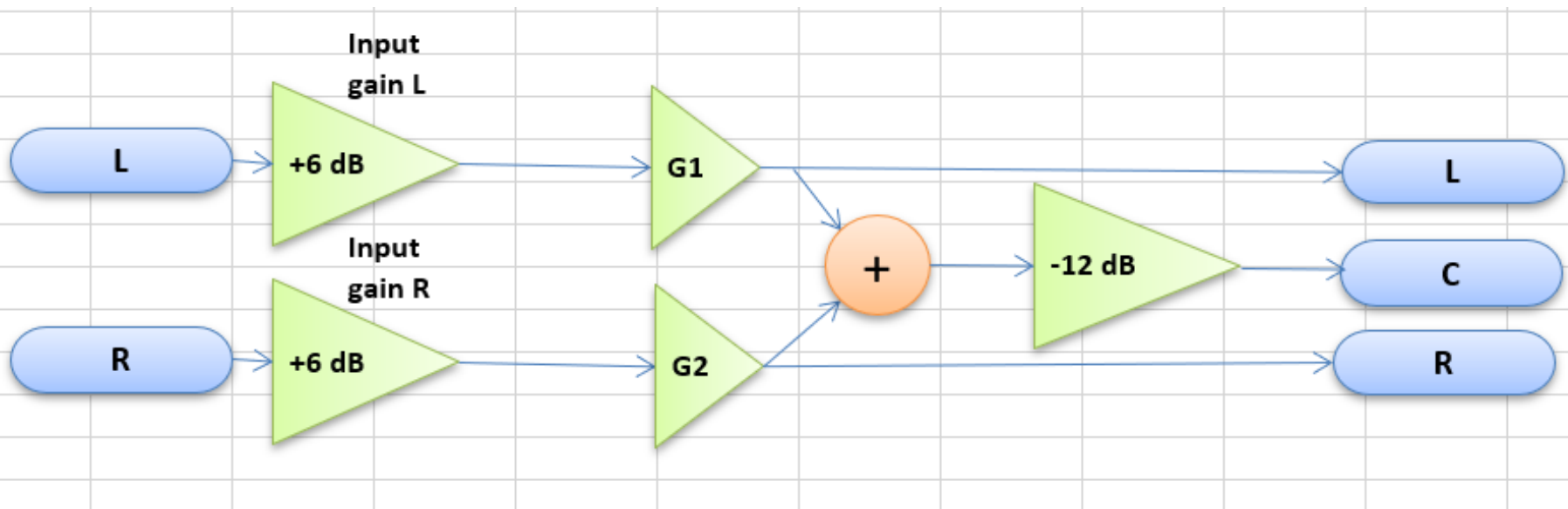
⋮  c\_podsetnik.pdf

---

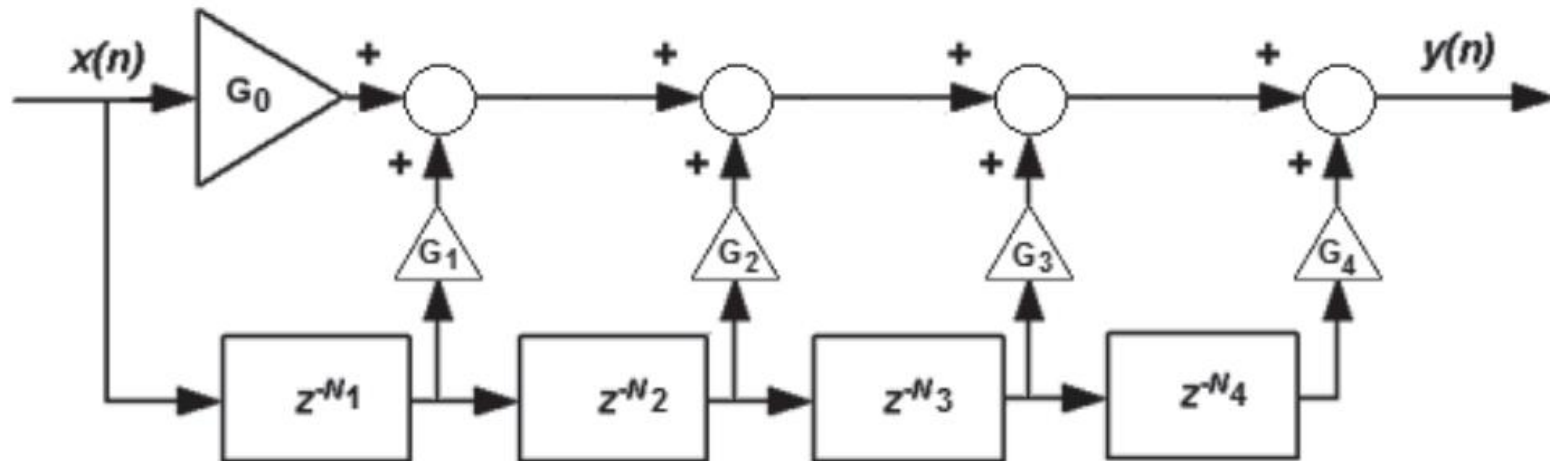
⋮  Osnovne\_komponente\_digitalne\_obrade\_signala.pdf

---

# Uvodni model za (auditorne) vežbe



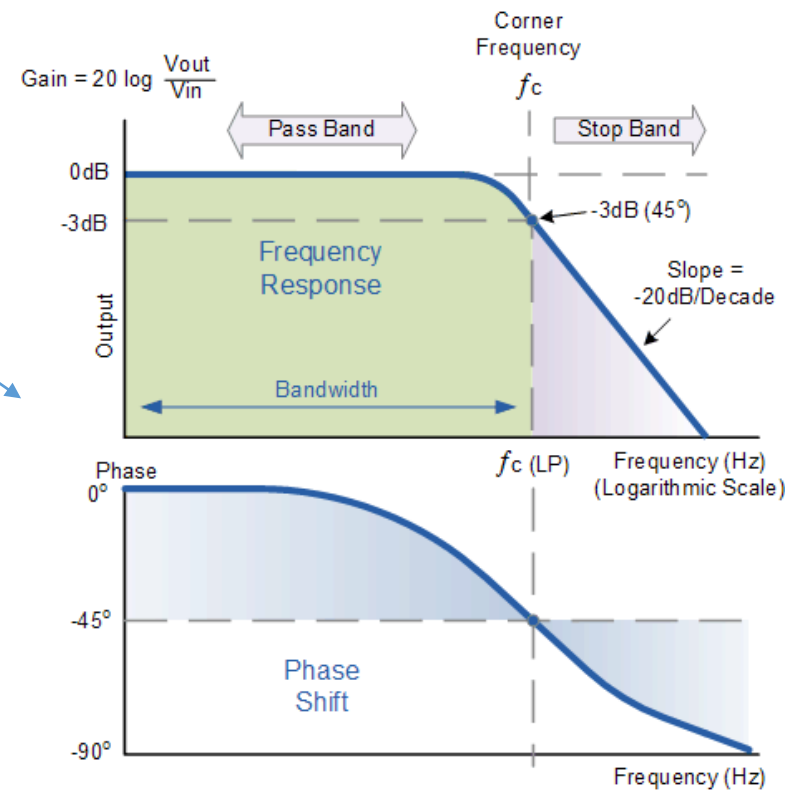
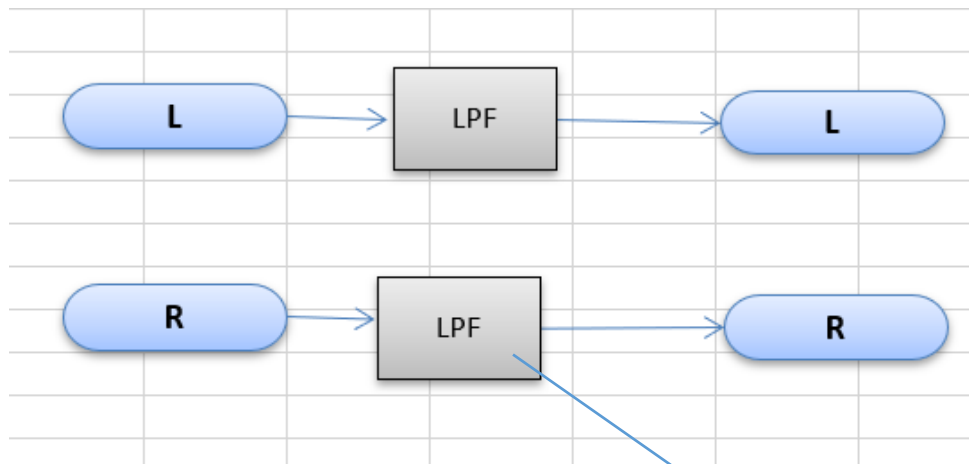
# Model za lab. vežbe: Višestruki eho efekat



Slika 3.8 – Blok dijagram sistema za dodavanje višestrukog eho efekta

Opis sistema pročitati obavezno u Vezba3.pdf, **3.4.2 Zadatak 2: Primena metodologije razvoja aplikacije zasnovane na C kompajleru na realizaciju bloka za dodavanje višestrukog eho efekta**

# Primer realizacije efekta – low pass filter



# Arhitekture i algoritmi DSP 2



Auditorne vežbe AU-5[2]  
Model 1 – Funkcionalna optimizacija C koda



Odsek za računarsku tehniku i računarske komunikacije

# Model 1



- ❑ Funkcionalne optimizacije C koda
- ❑ C/C++ kod
- ❑ Izmene koda prema hardverskim proširenjima CS497xx procesora
  - Broj registara
  - Veličina memorije
  - Veličina steka
  - Rad adresnog generatora
  - Hardverske petlje

# Funkcionalne optimizacije



## ❑ Organizacija podataka

- Smanjenje broja argumenata kod funkcija obrade
- Uklanjanje lokalnih struktura i promenljivih

## ❑ Pristupanje podacima

- Prilagoditi operacije pristupa podacima prema radu AGU jedinice
- Umesto C-ovskog indeksiranja koristiti pristup preko pokazivača na element

## ❑ Optimizacija programskih petlji

- Izbacivanje nezavisnih/invarijatnih delova koda iz petlji
- Skraćivanje tela petlji

# Pristupanje podacima



- ❑ Najosetljiviji korak u Modelu 1
- ❑ Rad sa pokazivačima i duplim pokazivačima
- ❑ Zašto ne indeksiranje?
- ❑ CS497xx ne podržava indeksiranje u C-ovskom kontekstu, na DSP-u indeksiramo memoriju (x ili y)
- ❑ C-ovsko indeksiranje na CS497xx zauzima puno instrukcija!

```
a1 = i1          # početna adresa niza
a2 = i0          # C indeks
a3 = a1 + a2     # izračunavanje krajnje adrese niza
i4 = a3          # smeštanje u i4
a0 = xmem[i4]    # pristup elementu
```



# C pokazivači (podsećanje)



❑ `my_array[i] = 10;`

`*(my_array + i) = 10`

`i++;`

❑ `p_my_array = my_array;`

`*p_my_array = 10;`

`p_my_array++;`