

SISTEMUL DE DEZVOLTARE TMS320VC5416 DSK

1. Prezentare generală

TMS320VC5416 DSK este un sistem de dezvoltare de sine stătător. Permite evaluatorilor să examineze diferite caracteristici ale procesoarelor de semnal C5416, pentru a observa dacă îndeplinesc cerințele aplicației lor. Modulul este o platformă excelentă pentru a dezvolta și a rula programe pentru procesoarele digitale din familia TMS320VC5416.

O schemă simplificată a plăcii VC5416 DSK este prezentată în figura 1:

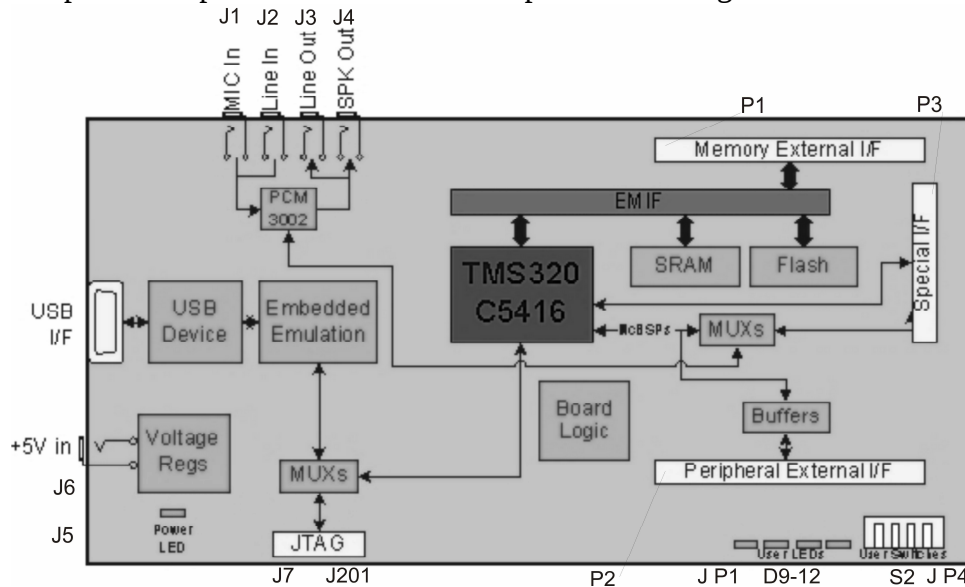


Figura 1. Placa TMS320VC5416 DSK

Sistemul are următoarele caracteristici:

- 64k cuvinte memorie SRAM;
- 256k cuvinte on-board flash ROM;
- 3 conectoare de extensie (interfață memorie, interfață pentru periferice, interfață pentru portul gazdă - HPI – Host Port Interface);
- controler USB JTAG cu drivere plug and play;
- codec stereo Burr Brown PCM 3002;
- operații la +5V;
- VC5416 lucrează la 16-160 MHz.

DSK 5416 este o placă multi-layer de 210x115mm, alimentată de o sursă de putere externă de 5V. Placa este echipată cu o interfață I/O externă care suportă porturi paralele I/O și porturi serial-sincrone (echipate cu buffere) pe mai multe canale. Interfețele importante ale DSK includ interfețele pentru RAM și ROM, FPGA, interfața pentru Codec și interfața de expansiune. Patru mufe jack stereo asigură intrări și ieșiri de la codec.

DSK permite verificarea codului VC5416 la viteză maximă. Cu 64k cuvinte memorie RAM, 256k cuvinte memorie flash ROM, și un codec stereo Burr Brown PCM 3002, placa reprezintă soluția pentru o mulțime de probleme.

Elementul central al plăcii este procesorul de semnal TMS320C5416 a cărei arhitectură multibus avansată este prezentată în continuare:

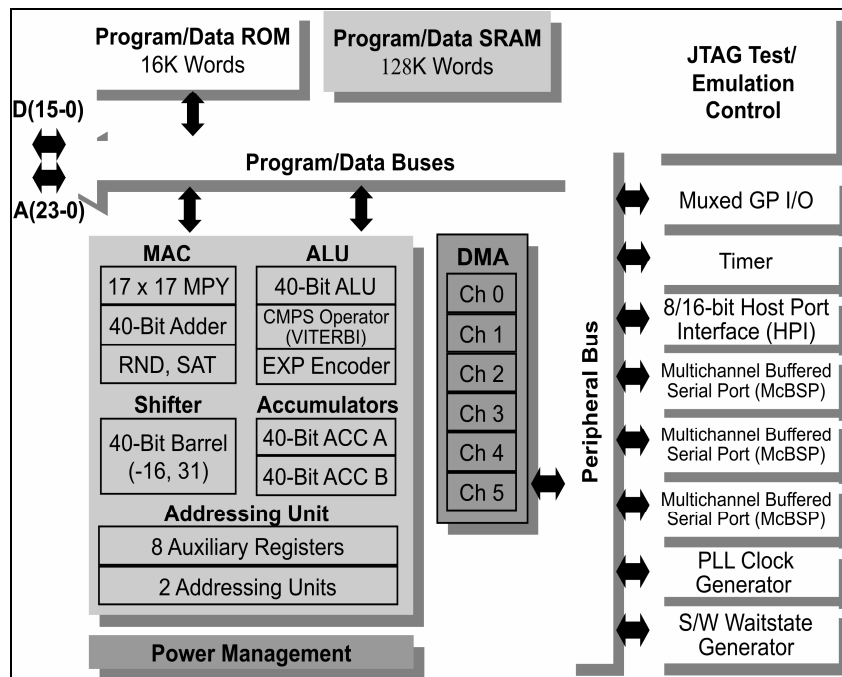


Figura 2. Schema bloc a procesorului de semnal TMS320C5416

- arhitectură avansată multibus (3 bus-uri separate de memorie date pe 16 biți și unul de memorie program);
- unitate aritmetică și logică pe 40 biți, incluzând 2 registre acumulatori pe 40 biți și unul de shiftare;
- multiplicator paralel 17x17 biți cuplat cu un sumator dedicat, pe 40 biți, pentru operații de multiplicare/acumulare într-un singur ciclu “non-pipelined”;
- unitate de comparare, selectare și stocare (CSSU) pentru selecție, adunare/comparare a operatorului Viterbi;
- codificator exponențial pentru a calcula o valoare exponențială a unei valori din acumulatorul de 40 biți (1 ciclu);
- două generatoare de adrese cu 8 registre auxiliare și 2 unități de registre aritmetice auxiliare;
- mod de adresare extins pentru 8M x 16biți – spațiu maxim de adresare pentru program extern;
- spațiu maxim de adresare memorie 192k x 16 biți (64k cuv. program, 64k cuv. date, 64k cuv. I/O);
- memorie ROM pe cip cu câteva configurații de memorie date/program;
- memorie RAM pe cip cu acces dual;
- repetarea unei instrucțiuni și operații de repetare pe bloc pentru codul de program;
- instrucțiuni de mutare blocuri de memorie - pentru o mai bună programare și manevrare a datelor;
- instrucțiuni cu operanzi pe cuvinte de 32 biți;
- instrucțiuni cu citiri de 2 sau 3 operanzi;
- instrucțiuni aritmetice cu stocare și încărcare paralelă și stocare condiționată;
- întoarcere rapidă din întreruperi;

1.2. Harta memoriei sistemului

DSK include 64k cuvinte memorie SRAM și 256k cuvinte flash ROM pentru încărcare. Decodarea memoriei externe este realizată printr-un circuit CPLD. Programul asociat acestui dispozitiv selectează memoria RAM, flash ROM sau perifericele de pe placă.

Memoria de date

Spațiul pentru memoria de date se întinde pe 64k cuvinte (2 pagini de câte 32k cuvinte). Pentru sistemul C5416 DSK memoria externă de date este mapată între adresele 0x8000h și 0xFFFFh. Memoria DARAM (Dual Access RAM) este mapată în spațiul de date (între adresele 8000h și FFFFh) dacă bitul DROM (bitul 3 din registrul PMST) este setat pe „1”. În caz contrar în acest spațiu este mapată memoria externă.

Hex	Data		
0x0000	Memory-Mapped Registers	Address ranges for on-chip DARAM in data memory are: DARAM0: 0x0080-0x1FFF DARAM1: 0x2000-0x3FFF DARAM2: 0x4000-0x5FFF DARAM3: 0x6000-0x7FFF DARAM4: 0x8000-0x9FFF DARAM5: 0xA000-0xBFFF DARAM6: 0xC000-0xDFFF DARAM7: 0xE000-0xFFFF	
0x005F			
0x0060	Scratch-Pad RAM		
0x007F			
0x0080	On-Chip DARAM0-3 (32K x 16 bit)		
0x7FFF			
0x8000	On-Chip DARAM4-7 (DROM=1) or External (DROM=0)		
0xFFFF			

Figura 3. Memoria de date a DSK TMS320VC5416

Memoria program

Configurarea memoriei program se poate realiza în două moduri folosind bitul 5, OVLY, al registrului PMST. Dacă OVLY=0, memoria RAM on-chip este adresabilă în memoria de date iar dacă OVLY=1, memoria RAM este mapată atât în memoria de date cât și în memoria program, conform următoarei figuri:

Hex		Page 0 Program		Hex		Page 0 Program	
0x0000		Reserved (OVLY=1) External (OVLY=0)		0x0000		Reserved (OVLY=1) External (OVLY=0)	
0x007F				0x007F			
0x0080		On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)		0x0080		On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)	
0x7FFF				0x7FFF			
0x8000		External		0x8000		External	
				0xBFFF			
				0xC000		On-Chip ROM (4K x 16 bit)	
0xFF7F				0xFEFF			
0xFF80		Interrupts (On-Chip)		0xFF00		Reserved	
				0xFF7F			
0xFFFF				0xFF80		Interrupts (On-Chip)	
				0xFFFF			
MP/MC=1 (Microprocessor Mode)				MP/MC=0 (Microcomputer Mode)			

Hex	Program	Hex	Program	Hex	Program	Hex	Program	Hex	Program
010000	On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)	020000	On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)	030000	On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)	040000	On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)	7F0000	On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)
017FFF		027FFF		037FFF		047FFF		7FFF00	
018000	On-Chip (DARAM4-7 (MP/MC=0) External (MP/MC=1)	028000	On-Chip (SARAM0-3 (MP/MC=0) External (MP/MC=1)	038000	On-Chip (SARAM4-7 (MP/MC=0) External (MP/MC=1)	048000	External	7F8000	External
01FFFF		02FFFF		03FFFF		04FFFF			
Page 1 XPC=1		Page 2 XPC=2		Page 3 XPC=3		Page 4 XPC=4		Page 127 XPC=7FH	

După trimiterea cuvântului de control, CPLD va transmite automat datele, serial, interfeței de control a CODEC-ului, prin semnalele de control master. Semnalele de control serial master ale CODEC-ului sunt: CODEC_MC (master clock), CODEC_MD (master data) și CODEC_ML (master load).

În plus, CPLD generează toate semnalele de tact pentru PCM3002 printr-un oscilator de clock și prin registrul de control CODEC_CLK. Valoarea implicită a ceasului sistem este de 12.288 MHz. CPLD folosește ceasul sistem al CODEC-ului pentru a genera un semnal de tact de 3.0122 MHz și un semnal de sincronizare cadre de 48 kHz. Registrul CODEC_CLK permite programatorului să modifice frecvența de eșantionare prin controlul ratei de divizare a frecvenței CODEC-ului PCM 3002. Datele sunt transmise codecului prin interfața VC5416, McBSP2.

Un bit din registrul MISC al CPLD va permite registrului McBSP2 să fie rutat la conectorul de expansiune HPI; calea de bază este asigurată prin Codecul PCM3002.

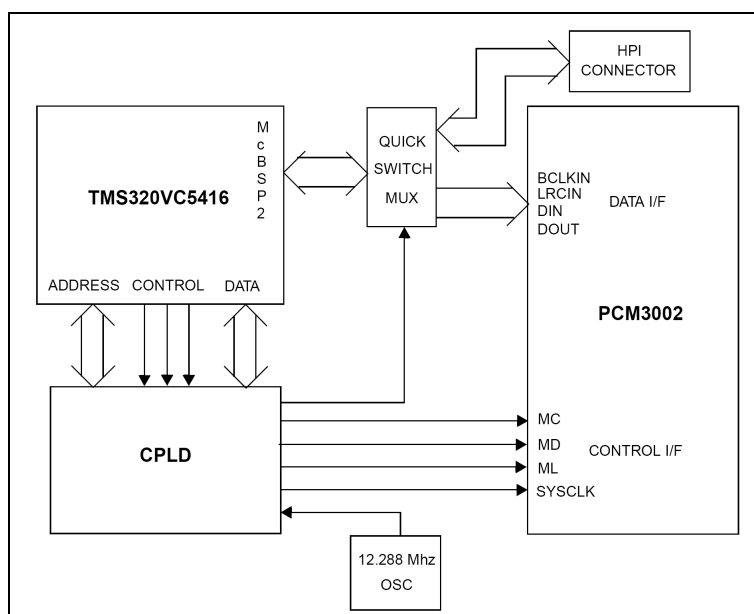


Figura 5. Interconexiunile dintre PCM3002, CPLD și DSP

PCM3002- Interfața de control are 4 regiștrii interni pe 16 biți care sunt controlați software printr-o interfață serială.

Programarea interfeței de Date

Interfața de Date a PCM3002 este conectată la pinii McBSP2 ai VC5416.

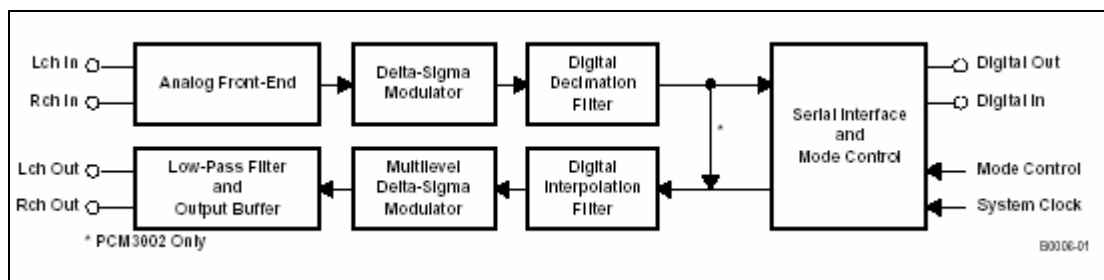


Figura 6.Codecul PCM3002

PCM3002 este un codec audio cu un singur cip (convertoare analog-digitale și digital-analogice). CAD și CDA implică modulație delta-sigma cu rata de supraeșantionare egală cu 64. CAD include un filtru de decimare digital, iar CDA implică un filtru de interpolare digital pentru supraeșantionarea cu 8. CDA include de asemenea atenuare digitală, detecție zero infinită, și funcția soft mute, pentru a forma un subsistem complet. PCM3002 oferă un mod de putere care funcționează cu ambele convertoare independente.

PCM3002 este fabricat folosind un proces avansat CMOS, și sunt disponibile într-un pachet SSOP pe 24 de pini. Sunt potrivite pentru o gamă largă de aplicații, pentru care costurile sunt importante, aplicații pentru care performanța este o prioritate.

Pentru PCM3002, funcțiile programabile sunt controlate prin software.

Funcții:

- CAD si CDA pe 20 biți, monolitice
- intrare/ieșire 16/20 biți
- Control prin software: PCM3002
- Funcții speciale (PCM3002): atenuare digitală (256 pași)
 - soft mute
 - digital loopback
 - patru formate de date audio digitale alternative
- Rata de eșantionare: 4 kHz până la 48 kHz
- Alimentare la 3 V
- Aplicații audio portabile/mobile

CAD stereo	CDA stereo
- intrare pentru alimentare la un singur capăt - filtru antialias - supraeșantionare cu 64 - performanțe ridicate: SNR: 90 dB	- ieșire pentru alimentare cu un singur capăt - filtru trece jos analogic - supraeșantionare cu 64 - performanțe ridicate: SNR: 94 dB

Funcțiile speciale ale codecului PCM3002 sunt controlate utilizând 4 regiștrii fiecare de câte 16 biți, vezi Anexa 1.

1.4. Conectorii sistemului:

Placa DSK are 16 conectori prin care se realizează accesul utilizatorilor la placă. Conectorii, mărimea acestora, funcția fiecăruia, sunt prezentate în tabelul de mai jos, iar descrierea lor în Anexa 2.

Connector	# Pins	Function
P1	80	Memory
P2	80	Peripheral
P3	80	HPI
J1	2	Microphone
J2	2	Line In
J3	2	Line Out
J4	2	Speaker
J5 *	4	Optional Power Connector
J6	2	+5 Volt
J7	14	External JTAG
J201	5	USB JTAG
JP1	10	CPLD Programming
JP4	8	DSP Configuration Jumper

Tabelul 1. Conectorii plăcii TMS320VC5416 DSK

2. Aplicații

Aplicațiile prezentate în continuare se vor realiza folosind mediul Code Composer Studio- C5416 Simulator. După descrierea pașilor necesari pentru a crea un nou proiect de simulare, pentru fiecare aplicație în parte se vor descrie conceptele teoretice de bază folosite. De asemenea vom prezenta formulele utilizate, schemele bloc ale aplicațiilor, sursele programelor și reprezentările grafice ale semnalelor, atât înainte cât și după procesare.

2.1. Etapele necesare pentru a crea un nou proiect de simulare

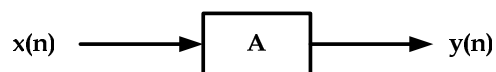
Procedul următor a fost scris pentru a instala aplicația *template*, care este stocată în fișierul *template.zip*. Pentru alte proiecte, schimbați numele *template* în cel al fișierului zip folosit (de exemplu *delay*). Execuția acestei proceduri necesită câteva minute. Este necesar programul WinZip.

1.	Start Code Composer Studio pentru TMS320C5416 DSK.
2.	Select <i>Project</i> -> <i>New</i> . Pentru Project Name, tastați cuvântul <i>template</i> . Click pe butonul Finish. Un nou proiect a fost creat. Rețineți directorul în care a fost salvat proiectul, apoi minimizați Code Composer Studio.
3.	Folosind Windows Explorer, mergeți în directorul unde a fost salvat noul proiect, de exemplu <i>C:\ti\myprojects\template</i>
4.	Copiați fișierul <i>template.zip</i> în directorul <i>C:\ti\myprojects\template</i>
5.	Extrageți fișierele din <i>template.zip</i> folosind WinZip la directorul curent cu opțiunea Extract Here.
6.	Intoarceti-vă la Code Composer Studio. Select <i>Project</i> -> <i>Add Files to Project</i> . Selectați calea <i>C:\ti\myprojects\template</i> . Folosiți Control+click mouse stânga pentru a evidenția toate fișierele .c. Click pe butonul Open pentru a le adăuga la proiect.
7.	Selectați <i>Project</i> -> <i>Add Files to Project</i> . Folosind săgeata jos în Files la Type box, selectați Linker Command File (*.cmd). Folosiți click mouse stânga pentru a evidenția fișierul <i>template.cmd</i> . Click pe butonul Open pentru a adăuga fișierul <i>template.cmd</i> la proiect.
8.	Selectați <i>Project</i> -> <i>Add Files to Project</i> . Folosind săgeata jos în Files la Type box selectați Asm Source Files (*.a*, *.s). Folosiți Control + click mouse stânga pentru a evidenția fișierele .asm (dacă există). Dacă există fișiere .asm în proiect, click pe butonul Open pentru a adăuga fișierele .asm la proiect. Dacă nu, click pe Cancel.
9.	Nu este nevoie să adăgați fișierele .h la proiect. Aceasta se face în mod automat.
10.	Selectați <i>Project</i> -> <i>Add Files to Project</i> . Mergeți la <i>C5400/cgtools/lib</i> și încărcați fișierul <i>rts</i> .
11.	Selectați <i>Project</i> -> <i>Rebuild All</i> . De data aceasta proiectul ar trebui să se construiască cu succes și fișierul <i>template.out</i> poate fi încărcat folosind <i>File</i> -> <i>Load Program</i> .
12.	În continuare se urmează pașii descriși în lucrarea anterioară.

2.2. Controlul câștigului

Controlul câștigului este implementat în proiectul numit VOLUME. Avem creată o aplicație care folosește opțiunile oferite de CCS [adaugă fișiere watch sau GEL] pentru a regla un câștig variabil aplicat semnalului de intrare.

Formula folosită în implementare este: $y(n) = A * x(n)$;



Observație: în cazul în care câștigul este egal cu 1, aplicația repetă semnalul de la intrare la ieșire.

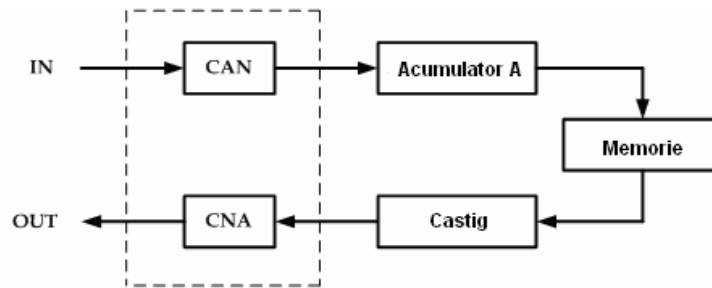


Fig. 7 Schema bloc a aplicației

În continuare se prezintă programul sursă:

```

/*  VOLUME . C  */

#include <stdio.h>
#include "volume.h"

/* Declarații globale */
int inp_buffer[BUFSIZE];
int out_buffer[BUFSIZE];
int gain = MINGAIN;
unsigned int processingLoad = BASELOAD;
/* Funcții */

extern void load(unsigned int loadValue);
static int processing(int *input, int *output);
static void dataIO(void);

/*programul principal */
void main()
{
    int *input = &inp_buffer[0];
    int *output = &out_buffer[0];
    puts("volume example started\n");

    /* buclă infinită */
    while(TRUE)
    {
        dataIO();
        /* Citește datele de intrare folosind un probe-point conectat la un fișier gazdă.*/
        #ifdef FILEIO
        puts("begin processing")
        #endif
        processing(input, output);
    }
}

/*Funcția processing: aplica o transformare (înmulțire cu un factor de castig) unui semnal primit la intrare;
PARAMETRII: adresele buferelor de la intrare și ieșire ; VALOAREA RETURNATA: TRUE.*/
static int processing(int *input, int *output)
{
    int size = BUFSIZE;
    while(size--)
    {
        *output++ = *input++ * gain; /* se calculează ieșirea ca fiind intrarea înmulțită cu un coeficient,
        și se incrementează pointerul buferelor */
    }
    load(processingLoad);
    return(TRUE);
}

/*Funcția dataIO: citește semnalul de la intrare și scrie semnalul procesat la ieșire.*/
static void dataIO()
{
    return;
}

/*****

```

Aplicând un câștig de 10 vom obține următorul rezultat:

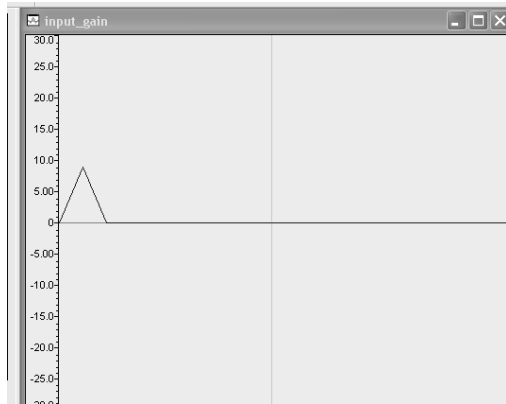


Fig. 8 Semnal intrare



Fig. 9. Semnal ieșire

2.3. Intârzierea

Un alt fenomen des întâlnit în procesarea semnalelor este cel al întârzierii semnalului. Aceasta poate să apară datorită condițiilor nefavorabile, imperfecțiunilor canalului sau datorită diferitelor tehnici aplicate semnalului.

Formula de calcul, schema bloc și sursa aplicației sunt prezentate în continuare:

$$y(n) = x(n-d);$$

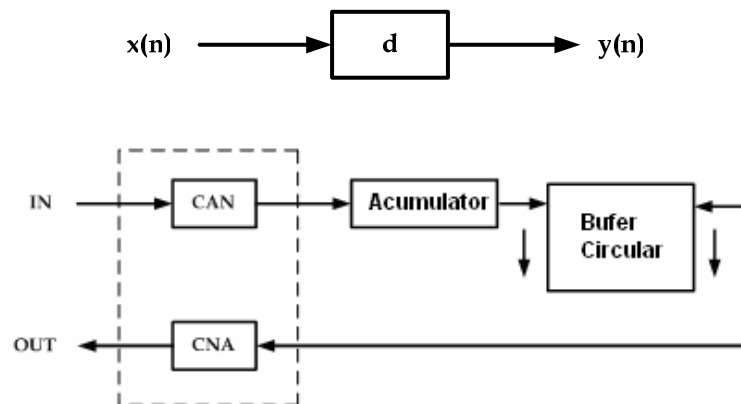


Fig. 10 Schema bloc a aplicației

```

/* DELAY.C */
#include <stdio.h>
#include "delay.h"

/* Declaratii gobale */
int inp_buffer[BUFSIZE];
int out_buffer[BUFSIZE];
int delay = MINGAIN;

/* procesarea buferelor de date */
/* variabila de control a volumului */

/* Functii */
extern void load(unsigned int loadValue);
static int processing(int *input, int *output);
static void dataIO(void);

/* programul principal */
void main()
{
    int *input = &inp_buffer[0];
    int *output = &out_buffer[0];

```

```

puts("Delay example started\n");          /* afisare mesaj */

/* bucla infinita */
while(TRUE)
{
    dataIO();    /*Citeste datele de intrare folosind un probe-point conectat la un fisier
                  gazda. Scrie datele de iesire intr-un graf conectat printr-un probe-point.*/

    #ifdef FILEIO
    puts("begin processing")              /* afisare mesaj la inceputul procesarii */
    #endif

    /* se aplica intarzierea folosind functia processing */
    processing(input, output);
}
}

/* Functia processing:aplica o transformare (atenuare si intarziere) unui semnal primit la intrare. PARAMETRII:
adresele buferelor de la intrare si iesire. VALOAREA RETURNATA: TRUE.*/
static int processing(int *input, int *output)
{
    int size = BUFSIZE;                  /* marimea memoriei tampon */

    while(size-->0)                      /* pentru toate esantioanele bufferului*/
    { if (size<(BUFSIZE-delay))          /* daca elementele cerute se afla in memorie*/

        *output = *(input-delay);        /* intrarea intarziata se reproduce la iesire */
        else
        *output = 0;                     /* altfel se transmite 0 */

        output++;                        /* se incrementeaza pointerul bufferului */
        input++;
    }
    return(TRUE);
}

/*Functia dataIO :citeste semnalul de la intrare si scrie semnalul procesat la iesire. PARAMETRII: nu are; VALOAREA
RETURNATA: nu are.*/

static void dataIO()
{
    return;
}

/*****

```

Programul folosit ne permite să fixăm variabila de întârziere. Pentru o valoare echivalentă cu 2ms obținem următoarele grafice:

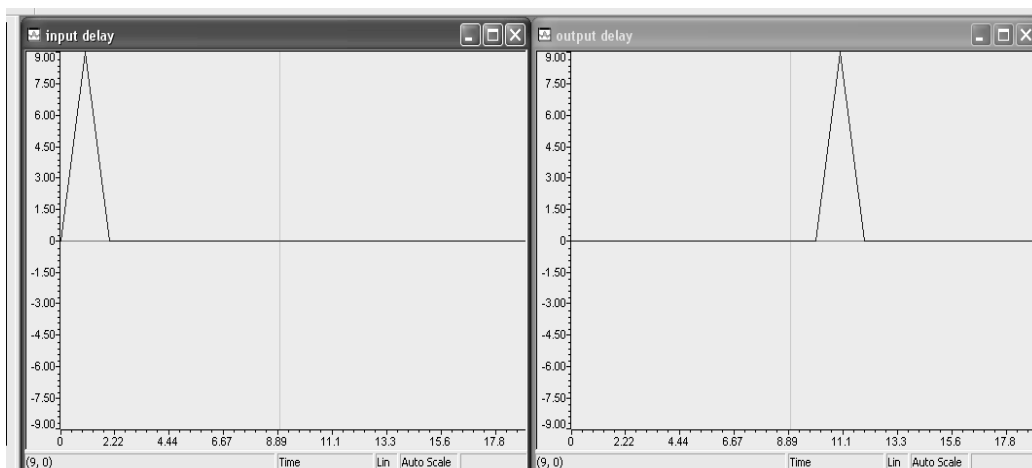


Fig. 11. Semnal intrare

Fig. 12. Semnal ieșire

2. 4. Reverberația

Reverberatia este persistentă sunetului într-un anumit spațiu după ce sunetul original este înlăturat. Când sunetul este produs într-un spațiu, un număr mare de ecouri se formează și apoi scad încet, în timp ce sunetul este absorbit de pereți și de aer, creând astfel reverberații. Acest lucru este observat în special când sursa sunetului se oprește iar reflecțiile continuă, micșorându-se în amplitudine, până când nu mai pot fi auzite.

Formula folosită în implementarea software a reverberației este:

$$y(n) = x(n-d) + 1/2 * x(n-2d) + 1/4 * x(n-3d) + 1/8 * x(n-4d) ;$$

Exercițiu: Plecând de la programele anterioare să se realizeze un proiect care să implementeze efectul de reverberație respectând următoarea schema bloc:

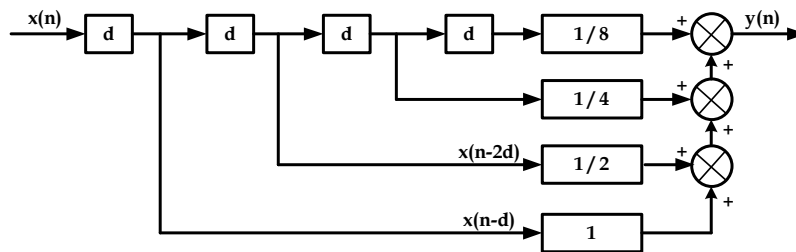


Fig. 13 Schema bloc a aplicației

Diagrama semnalului:

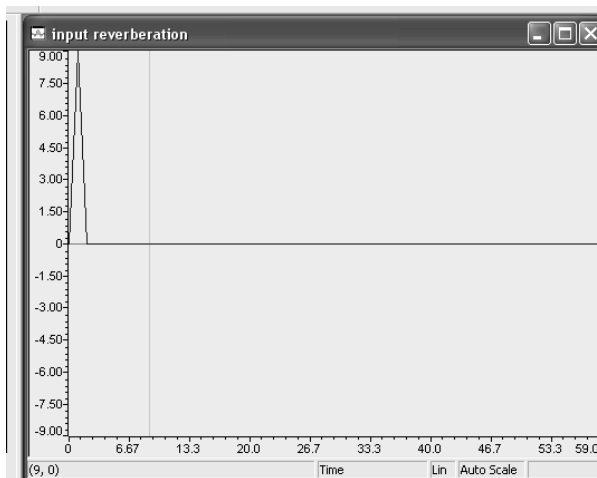


Fig. 14. Semnal intrare

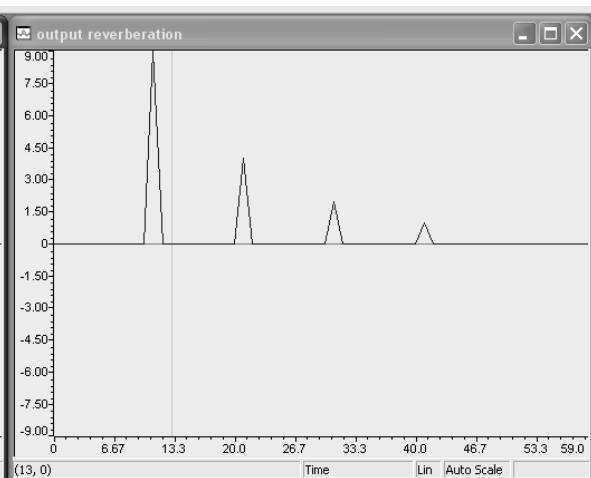


Fig. 15. Semnal ieșire

ANEXA 1

PCM3002 – descrierea regiștrilor

Există 4 regiștrii, iar biții 9 și 10 determină care registru este utilizat:

A1	A0	Registru
0	0	Registru 0
0	1	Registru 1
1	0	Registru 2
1	1	Registru 3

Registru 0 -- DAC Attenuation Data Left Channel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res					A1	A0	LDL	AL0-AL7							

Biți	Denumire	Valoare implicită	Descriere
B15-B11	res	00000	Reservați
B10	A1	0	Adresa Registrului
B9	A0	0	Adresa Registrului
B8	LDL		DAC Attenuation Data Load Control for Left Channel
B7-B0	AL0-AL7	1111 1111	DAC Attenuation Data for Left Channel

LDL (DAC Attenuation Data Load Control for Left Channel) - bit utilizat pentru a seta simultan ieșirile analogice a canalului din stânga, respectiv dreapta.

- 1 nivelul ieșirii este controlat de datele de atenuare AL (7:0)

- 0 noile date de atenuare vor fi ignorate și nivelul ieșirii va rămâne la nivelul anterior de atenuare.

Observație: Bitul LDR din Registrul 1 are funcții echivalente cu ale LDL. Când LDL sau LDR este setat pe "1", nivelele de ieșire a canalelor dreapta și stânga sunt controlate simultan.

AL0-AL7 (DAC Attenuation Data for Left Channel) - nivelul de atenuare (ATT) este dat de:

$$ATT = 20 \cdot \log_{10} \left(\frac{ATT \text{ Data}}{255} \right) \text{ (dB)}$$

AT (7:0)	00h	01h	FEh	FFh
Nivel atenuare	− ∞ dB (Mute)	−48.16 dB	−0.03 dB	0 dB (default)

Registru 1 - DAC Attenuation Data Right Channel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res					A1	A0	LDR	AR0-AR7							

Biți	Denumire	Valoare implicită	Descriere
B15-B11	res	00000	Reservați
B10	A1	0	Adresa Registrului
B9	A0	1	Adresa Registrului
B8	LDR		DAC Attenuation Data Load Control for Right Channel
B7-B0	AR0-AR7	1111 1111	DAC Attenuation Data for Right Channel

LDR (DAC Attenuation Data Load Control for Right Channel):- are aceeași funcție ca și LDL

AR0-AR7 - DAC Attenuation Data for Right Channel

Nivelul de atenuare (ATT) este dat de aceeași formulă ca și pentru Registrul 0.

Registrul 2 - Power Down and DAC Control (valoare implicită 0802h)

1	1	10	9	8	7	6	5	4	3	2	1	0
5	1											
res	A	A	PDA	BYP	PDD	AT	IZ	OU	DEM	DEM	MU	
	1	0	D	S	A	C	D	T	1	0	T	

PDAD (ADC Power-Down Control): 0 - modul Power Down dezactivat

1 - modul Power Down activat

Acest bit setează CAD în modul de consum redus de putere. Operațiile CAD sunt întrerupte prin oprirea alimentării cu curent a secțiunii CAD, DOUT fiind setat la 0, în timpul activării acestui mod.

BYPS (ADC High-Pass Filter Bypass Control): 0 - activat

1 - dezactivat (Bypassed)

PDDA (ADC POWER DOWN): 0 - modul Power-Down dezactivat

1 - modul Power-Down activat

Setează CDA în modul de consum redus de putere. Operațiile CDA sunt întrerupte prin oprirea alimentării cu curent a secțiunii CDA, VOUT fiind legat la GND în timpul activării acestui mod.

ATC (DAC Attenuation Channel Control): 0 - Individual Channel Attenuation Data Control

1 - Common Channel Attenuation Data Control

când este setat pe "1", datele de atenuare ale Registrului 0 pot fi folosite pentru ambele canale ale CDA. În acest caz, datele de atenuare ale Registrului 1 sunt ignorate.

IZD (DAC Infinite Zero Detection Circuit Control): 0 - detecție zero infinit dezactivată

1 - detecție zero infinit activată

Acest bit activează circuitul de detecție zero infinit al PCM3002: când este activ, acest circuit va deconecta ieșirea analogică a amplificatorului de la CAD delta-sigma dacă intrarea este zero continuu pentru 65.536 de cicluri consecutive ale BCKIN.

OUT (DAC Output Enable Control): 0 - ieșirile CDA active (operație normală)

1 - ieșirile CDA dezactivate

Când sunt setate pe "1", ieșirile sunt forțate la $V_{CC}/2$ (zero bipolar). În acest caz, toți regiștrii din PCM3002 rețin datele curente. Când sunt setate pe "0", ieșirile se întorc la stările programate anterior.

DEM1/DEM0 (DAC De-emphasis Control)

DEM1	DEM0	De-emphasis
0	0	44.1kHz ON
0	1	OFF (default)
1	0	48kHz ON
1	1	32kHz ON 3

MUT (DAC Soft Mute Control): 0 - mute dezactivat (implicit)

1 - mute activ

Când este setat pe "1", atât ieșirea de pe canalul de stânga, cât și ieșirea de pe canalul de dreapta ale CDA sunt puse pe mute în același timp. Acest procedeu are loc prin atenuarea datelor din filtrul digital.

Registrul 3 - Audio Data Control (valoare implicită 0600h)

15	11	10	9	8	6	5	4	3	2	1	0
res	A1	A0	res	LOP	res	FMT1	FMT0	LRP	res		

LOP (ADC to DAC Loop-Back Control): 0 - loop-back dezactivat (default)

1 - loop-back activ

Când acest bit este setat pe "1", datele audio ale CAD sunt trimise direct către CDA. Formatul datelor va fi în standardul I²S.

FMT1/FMT0 (Audio Data Format Select)

FMT1	FMT0	Format
0	0	16-bit, MSB first (default), DAC R-justified, ADC L-justified
0	1	20-bit, MSB first, DAC R-justified, ADC L-justified
1	0	20-bit, MSB first, Left-justified
1	1	20-bit, MSB first, I ² S

LRP (Polaritatea LRCIN): - se aplică doar formatelor 0, 1 și 2.

0 - canalul stânga este "H", canalul-dreapta este "L".

1 - canalul stânga este "L", Canalul-dreapta este "H".

ANEXA 2

Conectori de expansiune ai sistemului DSK TMS320C5416

DSK TMS320VC5416 conține trei conectori de expansiune care urmează standardele Texas Instruments: P1

- Conector de expansiune a memoriei

P2 - Conector de expansiune pentru periferice

P3 - Conector de expansiune HPI

J1 - Conector pentru microfon: microfonul interacționează cu placa PCM3002E printr-un circuit simplu op-amp. Intrarea o reprezintă o mufă jack stereo de 3.5 mm.

J2 - Conector Audio de intrare: este o intrare stereo. Intrarea comunică cu circuitul PCM3002E printr-un circuit bias op-amp. Conectorul este reprezentat de mufa jack stereo de 3.5 mm.

J3 - Conector Audio de ieșire: este o ieșire stereo, iar comunicarea cu PCM3002E se realizează ca și la conectorul J2. Este reprezentat de mufa jack de 3.5 mm.

J4 - Conector pentru căști/boxe: este condus de un mic amplificator TPA302 conectat la codecul plăcii, PCM3002E.

J5 - Conector opțional de putere: va opera cu sursa de putere standard a calculatorului.

J6 - Conector la +5V: puterea (5V) este adusă pe TMS320VC5416 DSK prin acest conector. Conectorul are un diametru exterior de 5.5 mm și unul interior de 2.5 mm.

J7 - Conector extern JTAG: o interfață cu 14 pini a TMS320VC5416 DSK și reprezintă interfața standard folosită de emulatoarele JTAG pentru a interfața cu DSP-urile Texas Instruments.

JP1 - Conector PLD de programare: acest conector interacționează cu ALTERA CPLD, U18. Este folosit pentru programarea CPLD.

J201 - Conector Universal Serial Bus (USB) încastrat JTAG: Conectorul J201 asigură o interfață USB emulatorului JTAG încastrat (situat) pe DSK. Aceasta permite dezvoltarea softului și debugger-ului fără a utiliza un emulator extern.

LED-urile: TMS 320VC5416 are patru LED-uri care pot fi definite de utilizator. Aceste LED-uri sunt utilizate de către POST (Power On Self Test), dar sunt disponibile pentru programele realizate de utilizator. Pot fi accesate prin adresa I/O 0X0000.

LED-uri de sistem: placa are 4 LED-uri de sistem ce indică diferitele stări în care se află placa.

Switch-uri: TMS320VC5416 are 2 switch-uri, unul Reset, și un switch DIP pe 4 poziții pentru utilizator.

Reset Switch/Reset Logic: există 3 Reset-uri pe placă. Primul reset este „power on reset”. Acest circuit așteaptă până când alimentarea este între anumite limite specificate, înainte de a alimenta pinul reset al plăcii TMS320VC5416. Surse externe care controlează reset-ul sunt butonul S1, și emulatorul USB JTAG.

Switch-ul DIP pe 4 poziții: DSK TMS320VC5416 are un switch DIP pe 4 poziții, pentru utilizator, S2. Este accesibil prin Registrul 0 CPLD la locația I/O 0X0000.