Operatii elementare in prelucrarea digitala a semnalelor.

Notiuni de aritmetica binara.

Reprezentarea numerelor binare include un bit de semn, un sir de biti reprezentind modulul si o virgula (imaginara) care separa partea intreaga de partea fractionara.

Uzual bitul de semn este 0 pentru numere pozitive si 1 pentru numere negative. Formatele de reprezentare cele mai utilizate sint: modul-semn, complement fata de 1 si complement fata de 2. ADSP 2101 foloseste formatul complement fata de 2 care prezinta avantajul ca numarul 0 are o unica reprezentare. In formatul complement fata de 2 numarul 0 este considerat pozitiv; prin urmare modulul amplitudinii celui mai mare numar negativ este mai mare cu 1LSB decit modulul amplitudinii celui mai mare numar pozitiv. De exemplu reprezentarea in complement fata de 2 pe k+1 biti (1bit de semn si k biti de amplitudine) are o gama dinamica intre 2^k-1 si -2^k.

Un numar pozitiv are aceeasi reprezentare in complement de 2 ca si reprezentarea binara naturala. Calculul unui numar negativ in complement fata de 2 se poate face in doua moduri:

1) Se inverseaza toti bitii modulului numarului negativ dupa care se aduna 1 pe pozitia LSB, sau

2) Se inverseaza toti bitii din stinga celui semnificativ bit 1 din modulul numarului negativ.

De exemplu: sa se calculeze reprezentarea binara a numarului -72. (72 in zecimal = 48 in hexazecimal).

a) 72 -> 01001000
Inversare biti -> 10110111
Adunare 1 LSB --> 00000001

-72 -> 10111000

b) 72 **0100**1000
Inversarea |
tuturor bitilor |
la stinga celui \/
mai putin sgn. 1 10111000

Daca virgula (imaginara) are pozitie fixa formatul se numeste format in virgula fixa (sau aritmetica se numeste aritmetica in virgula fixa).

Daca un numar este reprezentat prin sirul de biti:

S A A A . B B B B
/ |___/ \ ____|
semn / \ \
parte intreaga \ parte fractionara
virgula (imaginara)

, aritmetica utilizata este in virgula fixa si formatul se mai numeste **format N.M** unde N reprezinta numarul de biti ai partii intregi inclusiv bitul de semn, iar M reprezinta numarul de biti ai partii fractionare. De exemplu **formatul 1.15** indica o reprezentare de numere fractionare (intre -1 si aproximativ +1) iar **formatul 16.0** indica o reprezentare de numere intregi (intre -65536 si +65535).

Un numar in format 1.15 este calculat astfel: X=(-1)S +

$$\sum_{i=1}^{15} X_i 2^{-i} \text{ <--> } X = S.x_1x_2...x_{15}$$

/
virgula imaginara

Procesorul ADSP 2101 este optimizat pentru a efectua calcule aritmetice pe 16 biti in format 1.15, in simpla precizie. De asemenea se pot efectua operatii aritmetice in dubla precizie (32 biti) sau in tripla precizie (48 biti) utilizind rutine scrise de utilizator. Operatiile in simpla precizie sint efectuate direct, prin instructiuni, de catre procesor.

In continuare sint prezentate rutinele de efectuare a calculelor aritmetice (+,-,*,/) in dubla si tripla precizie. Aceste rutine pot fi usor extinse pentru a efectua calcule pe un numar mai mare de biti (multiplu de 16).

Operatii elementare în prelucrarea semnalelor

În aceasta sectiune se vor studia cele 3 operatii elementare în prelucrarea semnalelor : întîrzierea (implementarea unei linii de întîrziere digitale) , scalarea (înmultirea cu o constanta) si adunarea. Cu aceste operatii elementare se pot implementa ,practic,toate sistemele digitale.

Operatia de întîrziere

Structura de date de baza , pentru realizarea operatiei de întîrziere este bufferul circular.

Un buffer circular reprezinta o zona de memorie ,de lungime L , pentru care adresarea se face modulo L . Daca se doreste o întîrziere de N pasi atunci se defineste un buffer circular de lungime L=N. Acest buffer (**circ_buff**) este adresat ,de exemplu , de registrele I0,L0 si M0 (adica I0 = ^ **circ_buff** , L0 = 5 (lungimea bufferului) , M0 = 1).

Executînd secventa urmatoare:

```
ax0 = dm(port);    {citeste port de intrare}
dm(I0,M0) = ax0;           {stocheaza continutul portului în memorie}
```

de un numar de ori mai mare decît L se obtine situatia ilustrata în figura 1.

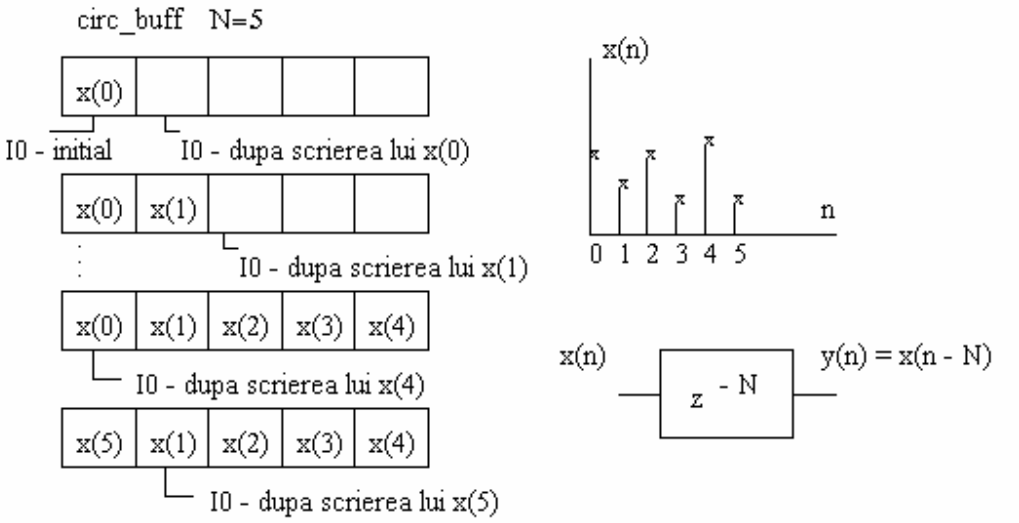

Figura 1

Generarea semnalului întîrziat se face cu secventa urmatoare:

```
M0 = 0; M1 = 1; I0 = ^ circ_buff ;  {initializari}
ax0 = dm(I0,M0);           { citeste x(n - N)}
dm(port_out) = ax0;        {scrie în portul de iesire}
ax0 = dm(port_in);         {citeste portul de intrare}
dm(I0,M1) = ax0 ;          {scrie x(n) în buffer}
```

Un exemplu de program complet este prezentat în continuare:

```
.MODULE/RAM/ABS=0 DELAY_SIM;                    {Beginning of DELAY Program}
.PORT   port_in;                                {Memory mapped Input Port}
.PORT   port_out;                               {Memory mapped Output Port}
.CONST          delay_size = 10;

.VAR/DM/CIRC circ_buff[delay_size];     {Circular buffer, length delay_size}

{----------Interrupt Vectors----------}
          JUMP start; NOP; NOP; NOP;            {Start Interrupt}
     JUMP delay; NOP; NOP; NOP;                 {External Pin Interrupt IRQ2}
          RTI; NOP; NOP; NOP;                   {SPORT0 Transmit Interrupt}
          RTI; NOP; NOP; NOP;                   {SPORT0 Receive Interrupt}
          RTI; NOP; NOP; NOP;                   {SPORT1 Transmit Interrupt}
          RTI; NOP; NOP; NOP;                   {SPORT0 Receive Interrupt}
          RTI; NOP; NOP; NOP;                   {TIMER Interrupt}
```

```
{-----Initialization: Circular Buffer---}
start:  I0=^cir_buff;                   {Index to top of cir_buf}
        L0=%cir_buff;                   {Enable circular buffer}
        M0=0;                           {No post-increment}
        M1=1;                           {Post-increment by 1}
{-----Clear circular buffer----------}
        CNTR=delay_size;                {Set up counter}
        DO clr_buf UNTIL CE;            {Initialize all cir_buff values
clr_buf:DM(I0,M1)=0                      {to 0}
        ICNTL=0x07;                     {Enable edge sensitive}
                                        {interrupt}
        IMASK=0x20;                     {Enable IRQ2 interrupt}

{------Wait for Sample----------------}
wait:   IDLE;                           {Wait until next sample}
        JUMP wait;

{----Process sample and generate delay--}
delay:  AX0=DM(I0,M0);                  {Get delayed sample from }
                                        {cir_buf}
        DM(port_out)=AX0;               {send delayed sample out}
        AX0=DM(port_in);                {Get input sample in AX0}
        DM(I0,M1)=AX0;                  {Move current sample to cir_buf}
        RTI;                            {Return from Interrupt}

{-----End of Program--------------------}
.ENDMOD;                                {End of DELAY Program}
```

Acest program utilizeaza cîteva facilitati ale simulatorului :

- simularea porturilor paralele
- simularea generarii întreruperilor (pe IRQ2)

O aplicatie simpla a operatiei de întîrziere este simularea fenomenului de ecou. Daca semnalul de intrare este x(n) atunci semnalul ecou ,y(n), este

$$y(n) = a.x(n) + b.x(n - N) , a+b <= 1.$$

Exercitiu: Pentru a=b=0,5 sa se scrie un program care sa implementeze ecuatia de mai sus.

Solutie:

```
.MODULE/RAM/ABS=0 ECHO_SIM;             {Beginning of ECHO Program}
.PORT   port_in;                        {Memory mapped Input Port}
.PORT   port_out;                       {Memory mapped Output Port}
.CONST          delay_size = 10;
.VAR/DM/CIRC circ_buff[delay_size];     {Circular buffer, length delay_size}
{----------Interrupt Vectors-----------}
        JUMP start; NOP; NOP; NOP;      {Start Interrupt}
    JUMP delay; NOP; NOP; NOP;          {External Pin Interrupt IRQ2}
        RTI; NOP; NOP; NOP;            {SPORT0 Transmit Interrupt}
        RTI; NOP; NOP; NOP;             {SPORT0 Receive Interrupt}
        RTI; NOP; NOP; NOP;             {SPORT1 Transmit Interrupt}
        RTI; NOP; NOP; NOP;             {SPORT0 Receive Interrupt}
        RTI; NOP; NOP; NOP;             {TIMER Interrupt}

{-----Initialization: Circular Buffer---}
start:  I0=^cir_buff;                   {Index to top of cir_buf}
        L0=%cir_buff;                   {Enable circular buffer}
        M0=0;                           {No post-increment}
        M1=1;                           {Post-increment by 1}
```

```
{-----Clear circular buffer----------}
            CNTR=delay_size;                {Set up counter}
            DO clr_buf UNTIL CE;            {Initialize all cir_buff values}
clr_buf:  DM(I0,M1)=0                       {to 0}
          IMASK=0x20;                       {Enable IRQ2 interrupt}
          ICNTL=7;
{------Wait for Sample------------------}
wait:   IDLE;                               {Wait until next sample}
            JUMP wait;

{----Process sample and generate delay--}
delay:  AY0=DM(I0,M0);                      {Get delayed sample from }
                                            {cir_buf}
            SI=DM(port_in);                 {Get input sample in SI}
            SR=ASHIFT SI by -1 (HI);        {Scale down current sample by}
                                            {50%}
            AR=SR1+AY0;                     {Sum current and delayed}
                                            {samples}
            DM(port_out)=AR;                {Send delayed sample out}
            DM(I0,M1)=SR1;                  {Move current sample to cir_buf}
            RTI;                            {Return from Interupt}
{-----End of Program----------------}
.ENDMOD;                                                {End of ECHO Program}
```

Generarea functiei sinus

Functia sinus poate fi aproximata prin urmatoarea formula :

$$\sin(x) = 3{,}140625 \cdot x + 0{,}02026367 \cdot x^2 - 5{,}325196 \cdot x^3 + 0{,}5446778 \cdot x^4 + 1{,}800293 \cdot x^5$$

Formula este valabila pentru x între 0 si 90 grade (primul cadran). Avînd în vedere ca $\sin(-x) = -\sin(x)$ si $\sin(x) = \sin(180 - x)$ putem obtine sin(x) pentru întreg cercul trigonometric.

Coeficientii care apar în formula trebuie reprezentati în format 4.12 ($C = S.(-1) + c_0 \cdot 2^2 + c_1 \cdot 2^1 + c_2 \cdot 2^0 + c_3 \cdot 2^{-1} + ... + c_{14} \cdot 2^{-12}$).Argumentul x va fi în format 1.15.

Se considera ca x este cuprins între - 180 ° si 180 ° , deci valoarea de - 180 ° corespunde valorii minime (- 1 în format 1.15) si valoarea de 180 ° corespunde valorii maxime (+ 1 în format 1.15) ca în figura 2 :

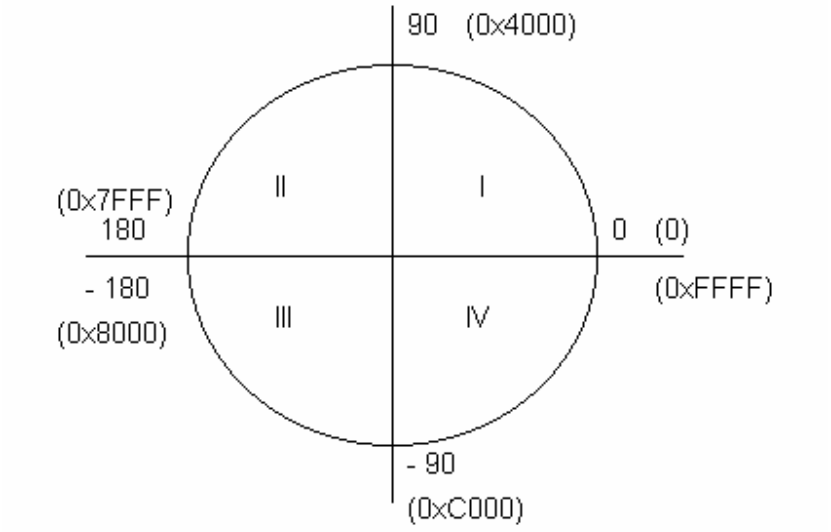

Figura 2

Rutina pentru generarea functiei sinus este prezentata în continuare :

```
        .MODULE Sin_Approximation;

{
    Sine Approximation
        Y = Sin(x)

    Calling Parameters
        AX0 = x in scaled 1.15 format
        M3 = 1
        L3 = 0

    Return Values
        AR = y in 1.15 format

    Altered Registers
        AY0,AF,AR,MY1,MX1,MF,MR,SR,I3
}

.VAR/DM sin_coeff[5];

.INIT   sin_coeff : H#3240, H#0053, H#AACC, H#08B7, H#1CCE;

.ENTRY  sin;

sin:    I3=^sin_coeff;                  {Pointer to coeff. buffer}
    AY0=H#4000;
    AR=AX0, AF=AX0 AND AY0;         {Check 2nd or 4th quad.}
    IF NE AR=-AX0;                  {If yes, negate input}
    AY0=H#7FFF;
    AR=AR AND AY0;                  {Remove sign bit}
    MY1=AR;
    MF=AR*MY1 (RND), MX1=DM(I3,M3);     {MF = x**2}
    MR=MX1*MY1 (SS), MX1=DM(I3,M3);     {MR = C1*x}
    CNTR=3;
    DO approx UNTIL CE;
        MR=MR+MX1*MF (SS);
approx:     MF=AR*MF (RND), MX1=DM(I3,M3);
    MR=MR+MX1*MF (SS);
    SR=ASHIFT MR1 BY 3 (HI);
    SR=SR OR LSHIFT MR0 BY 3 (LO);          {Convert to 1.15 format}
    AR=PASS SR1;
    IF LT AR=PASS AY0;                  {Saturate if needed}
    AF=PASS AX0;
    IF LT AR=-AR;                       {Negate output if needed}
    RTS;
.ENDMOD;
```

Initial se face un test asupra valorii argumentului .Daca argumentul este în cadranul I atunci acesta nu se modifica . Daca argumentul este în cadranul II , negarea sa (în aritmetica complement fata de 2) îl aduce în cadranul III , iar o schimbare de semn îl aduce în cadranul I. Daca argumentul este în cadranul III schimbarea semnului îl aduce în cadranul I. Daca argumentul este în cadranul IV atunci negarea sa îl aduce în cadranul I.Aceste prelucrari sînt exemplificate în figura 3.

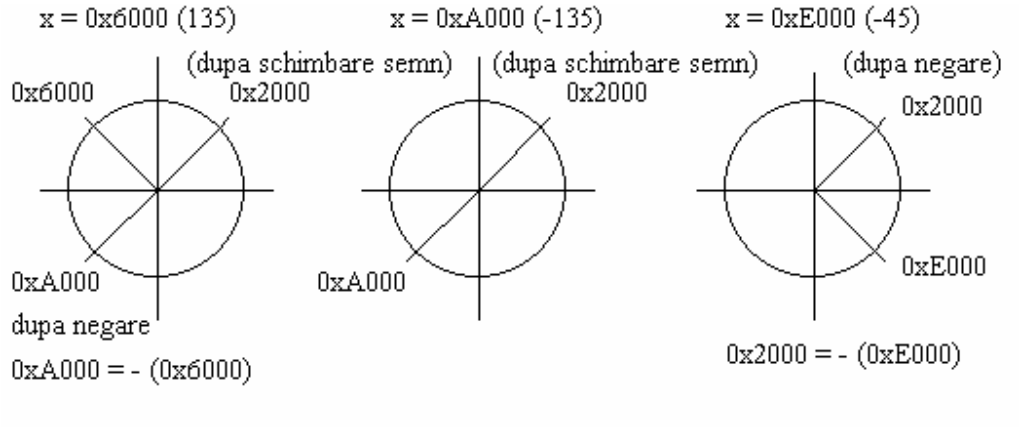

Figura 3

Este necesara o corectie ulterioara de semn.În exemplul din figura 3.9 se calculeaza , în toate cazurile sin(45 °) dar sin(135 °) = sin(45 °) , sin(- 135 °) = - sin(45 °) si sin(- 45 °) = - sin (45 °). Deci daca argumentul functiei se afla în cadarnele III si IV se schimba semnul rezultatului (functiei sinus calculate pentru un argument redus în cadranul I) .

Dupa reducerea argumentului la primul cadran se efectueaza înmultirile necesare; initial se calculeaza x^2 (în MF) apoi $c_2 . x$, dupa care urmeaza o bucla cu 3 pasi în care se calculeaza succesiv ($c_1 .x + c_2 . x^2$, x^3) , ($c_1 .x + c_2 . x^2 + c_3 . x^3$, x^4) , ($c_1 .x + c_2 . x^2 + c_3 . x^3 + c_4 . x^4$, x^5) . Apoi se calculeaza $c_1 .x + c_2 . x^2 + c_3 . x^3 + c_4 . x^4 + c_5 . x^5$ si se satureaza rezultatatul (la 0x 7fff) daca este cazul (MV = 1). Se face corectia de semn a rezultatului.

Exercitiu: Utilizînd programul prezentat anterior sa se genereze semnalul sin(2.π.1000.t) daca se cunoaste frecventa de esantionare de 8 kHz.