Circuit de filtrare a semnalelor digitale

Data:

Student: Zaharia Robert

Cuprins

1.Introducere

1.1) Context

Acest proiect are ca obiectiv implementarea unui circuit de filtrare a semnalelor digitale. Acesta transforma o secventa de valori de intrare X(i) intr-o secventa de valori de iesire pe baza unei formule de forma :

$$Y_{(k)} = X_{(k)} * a_1 + X_{(k-1)} * a_2 + X_{(k-2)} * a_3$$

Acest circuit poate fi folosit pentru a separa si restaura semnalele digitale folosite în tehnică, având la bază două valori logice, 0 și 1 (biți), care au fiecare câte o reprezentare în funcție de modul în care sunt transmise. Separarea semnalului este necesara atunci când un semnal a fost contaminat cu interferențe pentru a putea fi analizat, in timp ce restaurarea semnalului este utilizată atunci când un semnal a fost distorsionat într-un fel.

1.2) Specificatie

Implementarea va fi realizata in cadrul unui proiect Xilinx in limbaj VHDL. Pentru semnalele de intrare se vor folosi valori de tipul intreg intr-un BUS, prin urmare si semnalele de iesire vor fi tot de tipul celor de intrare, adica de tip intreg.

1.3) Obiective

Ca si obiective se doreste utilizarea a 15-20 de semnale de intrare in cadrul testelor pentru a putea realiza o analiza corecta a semnalului digital. Iar pentru a testa corectitudinea programului vom efectua doua teste pe aceleasi semnale de intrare dar cu doua functii diferite. In urma acestor teste vom obtine multiple semnale de iesire de tip intreg pe care le vom analiza. Semnalele digitizate sunt trecute printr-un convertor digital-analogic si un alt filtru trece-jos setat la frecventa Nyquist. Acest filtru de iesire este denumit filtru de reconstructie si poate include cresterea frecventei descrisă anterior.

2.Studiu bibliografic.

2.1) Operatiile folosite

In cadrul functiei alese de noi de tipul $Y_{(k)} = X_{(k)}*a_1+X_{(k-1)}*a_2+X_{(k-2)}*a_3$ va trebui sa folosim operatia de adunare si de inmultire pe intregi. Din fericire acestea sunt deja implementate pentru numere intregi.

Sintaxa de adunare pentru 2 numere intregi este:

Rezultat <= Intreg1 + Intreg2

Sintaxa de inmultire pentru 2 numere intregi este:

Rezultat <= Intreg1 * Intreg2

2.2) Obtinerea valorilor de intrare

Pentru cele 10-20 valori de intrare vom avea nevoie de o serie de numere intregi pentru a realiza testele. Pentru a genera numere la intamplare avem 2 variante :

1)Putem realiza o functie de rand_int care va genera un numar intreg in intervalul (0 , 1028).

```
function rand_int(min_val, max_val : integer) return integer is

variable r : real;

begin

uniform(seed1, seed2, r);

return integer(round(r * real(max_val - min_val + 1) + real(min_val) -0.5));

end function;
```

2). Vom folosi pachetul Random din libraria OSVVM.

```
Library OSVVM;

Use osvvm.RandomPKG.all;
```

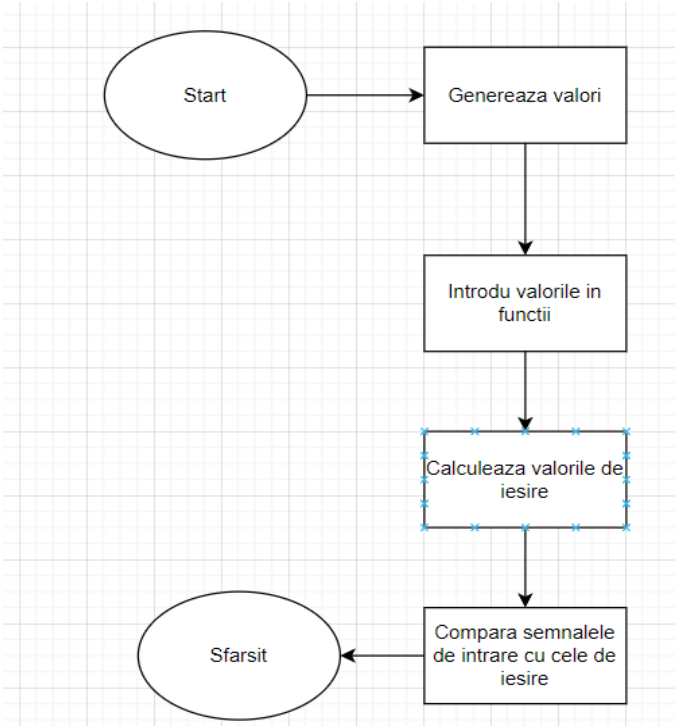
2.3) Afisarea rezultatelor

Afisarea proiectului nu va fi realizata pe placuta. In schimb rezultate calculelor vor fi ilustate in cadrul unei simulari pe Vivado pintr-un Waveform pentru fiecare din cele 2 teste. Vom incerca sa incarcam rezultatele in matlab pentru a vedea schimbarea semnalelor mai clara. Acestea fiind incarcate ulterior in documentatie.

3.Design

Pasii urmariti de programul realizat in Vivado vor fi:

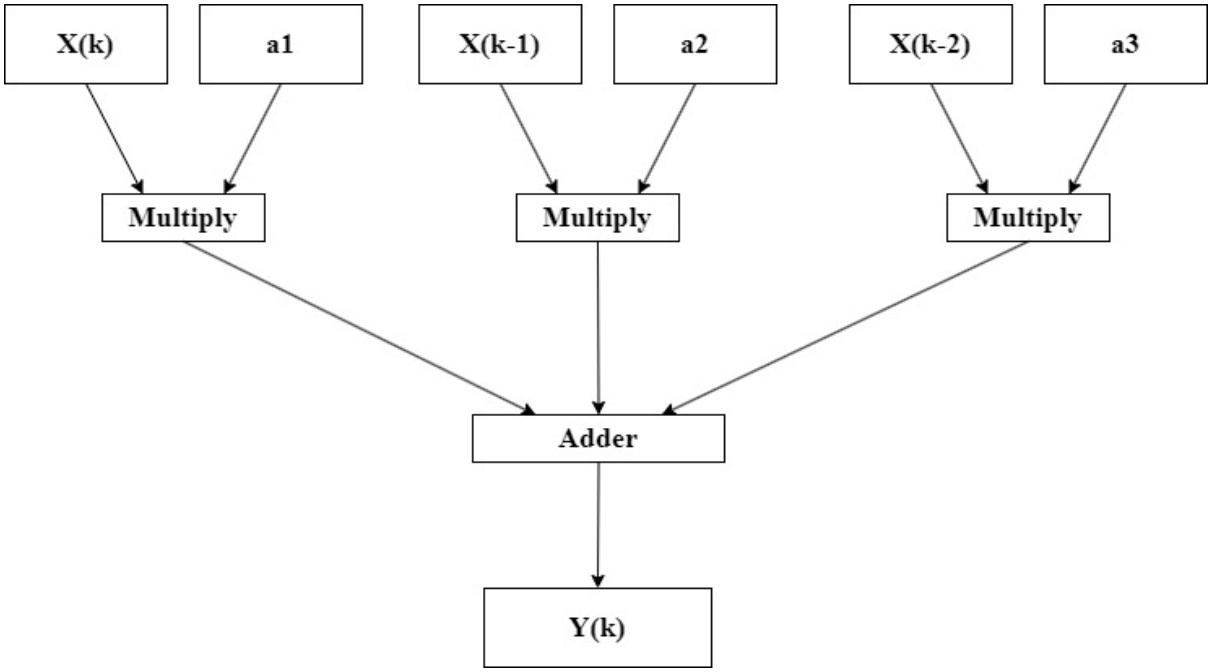
- Va primii 15 valori de intrare (le vom genera la intamplare)
- Vom introduce valorile in functiile de conversie
- Vom calcula valorile pentru semnalele de iesire
- Vom compara modificarile realizate intre semnalele originale si cele noi calculate



Principala parte a programului este functia de conversie a valorilor de intrare.Algorimul functiei de conversie este format din doi pasi pentru a ajunge la rezultatul final:

- Primul pas al functie de conversie este inmultirea functiei de intrare X(K) cu una dintre constante.
- Al doilea pas este adunarea rezultatelor de la pasul anterior.

Pentru partea de adunare vom folosi functia de adunare a valorilor intregi in VHDL iar pentru inmultire vom folosi functie de inmultire a valorilor intregi.



4.Implementare

In cadrul implementarii programului am declarat un type pentru cele 15 valori de intrare si unul pentru cele 15 valori de iesire.

```
type Arr is array (0 to 15) of integer;
signal intrari : Arr:=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
signal iesiri : Arr:=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
```

Pentru a initializa semnalele de intrare vom apela functia rand_int care returneaza un numar la intamplare intre cei doi parametri dati functiei.

```
impure function rand_int(min_val, max_val : integer) return integer is
  variable r : real;
  variable seed1 : positive;
  variable seed2 : positive;
begin
  seed1:=min_val;
  seed2:=max_val;
  uniform(seed1, seed2, r);
  return integer(
    round(r * real(max_val - min_val + 1) + real(min_val) - 0.5));
end function;
```

In cadrul unui proces dupa ce am initializat valorile de intrare am folosit un for de la 2 la 14 in cadrul caruia am calculat valorile de iesire utilizand functia de

conversie. Pentru primele 2 valori de iesire le vom calcula separat fiind diferite de celelate deoarece nu exista $X_{(-1)}$ si $X_{(-2)}$.

```
begin
process(intrari)
begin
inceput: for k in 0 to 14 loop
      intrari(k)<=rand_int(k*20,1024-(k*20));
end loop inceput;

sfarsit: for k in 2 to 14 loop
        iesiri(k)<=intrari(k)/2 + intrari(k-1)*10/7 + intrari(k-2)*3/2;

end loop sfarsit;
iesiri(1)<=intrari(1)*10/7 + intrari(0)*3/2;
iesiri(0)<=intrari(0)*3/2;

end process;
```

Pentru a verifica corectitudinea programului vom realiza inca un test cu alte valori de intrare si o alta functie de conversie.

```
process(intrari2)
begin
inceput: for k in 0 to 14 loop
      intrari2(k)<=rand_int(k*12,1004-(k*53));
end loop inceput;

sfarsit: for k in 2 to 14 loop
        iesiri2(k)<=intrari2(k)/3 - intrari2(k-1)*3/2 + intrari2(k-2)*2;

end loop sfarsit;
iesiri2(1)<=-intrari2(1)*3/2 + intrari2(0)*2;
iesiri2(0)<=intrari2(0)*2;

end process;
```

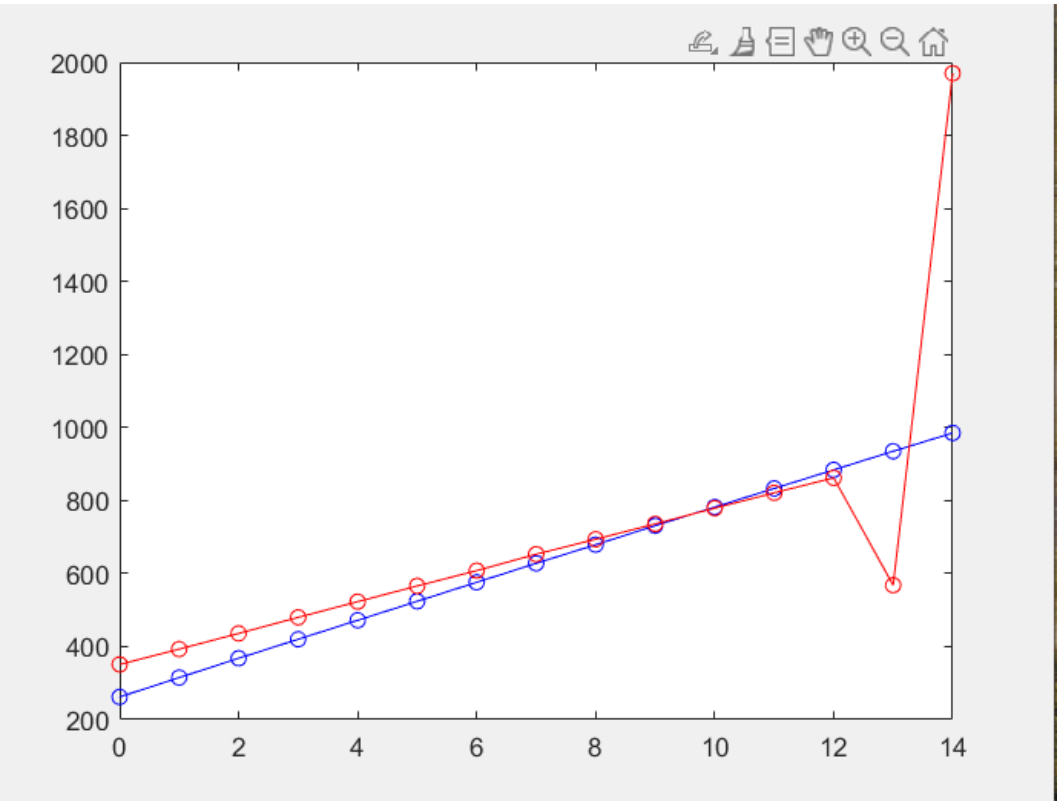
Rezultatele proiectului nu vor fi afisate pe placuta , in schimb ele vor fi afisate in cadrul unui grafic creat in matlab. Din pacate rezultate trebuie preluate manual din simularea din Vivado si inlocuite in vectorii din matlab.

```
Y=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
Intrari1=[1005,966,927,888,849,810,771,731,691,652,612,571,531,491,450]
Iesiri1=[1507,2887,3350,3217,3082,2949,2815,2681,2545,2409,2273,2137,1998,1859,1722]

Intrari2=[985,935,884,833,782,731,679,628,576,524,472,420,368,315,262]
Iesiri2=[1970,568,862,821,779,736,694,653,608,566,523,480,436,393,351]
```

Primul grafic va folosi rezultatele primului test. Vor fi evidentiate punctele prin cercuri.

```
figure
plot(Y,flip(Intrari1),'bo-')
hold on
plot(Y,flip(Iesiri1),'ro-')
```



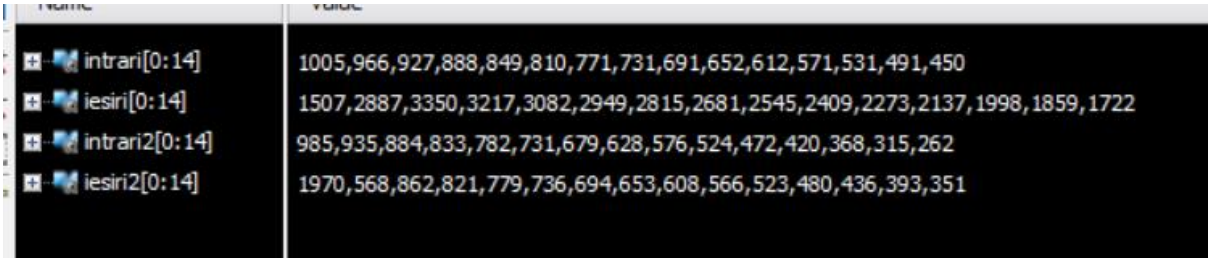
5.Testare

Pentru testarea programului am folosit 2 formule pentru a calcula semnalele de iesire

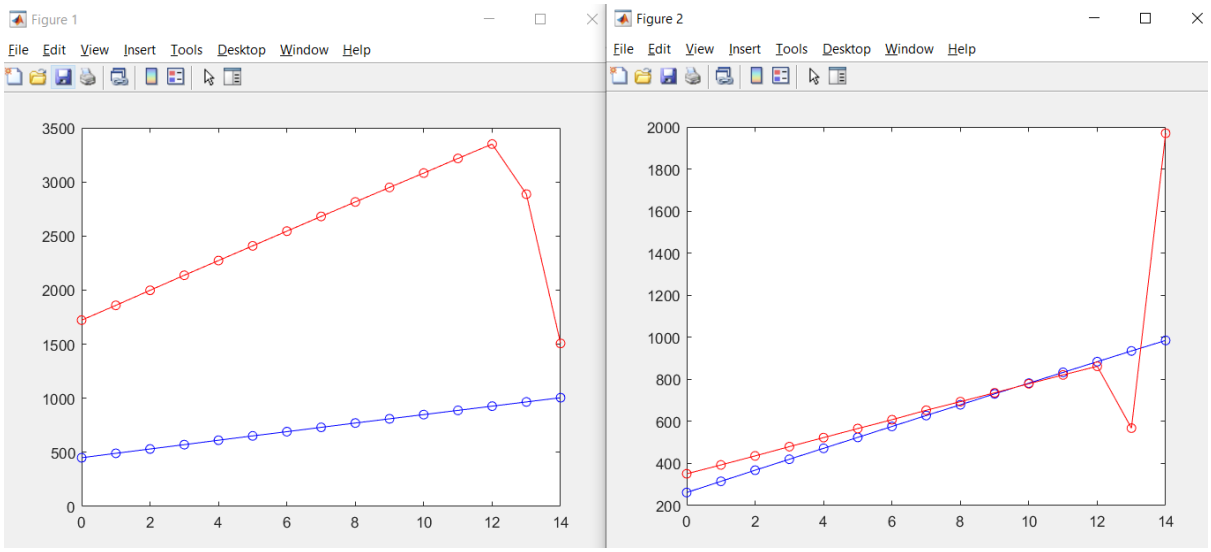
1) $Y_{(k)} = X_{(k)}*0.5+X_{(k-1)}*1.42+X_{(k-2)}*1.5$

2) $Y_{(k)} = X_{(k)}*0.33-X_{(k-1)}*1.5+X_{(k-2)}*2$

Rezultatele obtinute in urma simularii au fost urmatoare:



Iar in matlab graficele create cu aceste valori sunt:



Din pacate pentru realizarea graficelor semnalele de iesire calculate in vivado trebuie preluate manual si introduse in matlab. Acest lucru poate intarzia afisarea rezultatelor complete in cadrul proiectelor.

6.Concluzie

Obiectivul proiectului a fost implementarea unui circuit de filtrare a semnalelor digitale print transformarea lor pe baza unei formule.

Realizarea proiectului a fost o activitate placuta. Cea mai dificila parte a proiectului a fost gasirea unei modalitati de a genera o serie de numere la intamplare in Vivado deoarece nu exista o functie prestabilita. Ulterioare modificari ar fi realizarea unei extensii matlab pentru a prelua rezultatele direct simulare VHDL