Rîcea Alexandra
Grupa: CR2.3A

Laborator 4 -Structura și Organizarea Calculatoarelor-

~Raport tehnic~

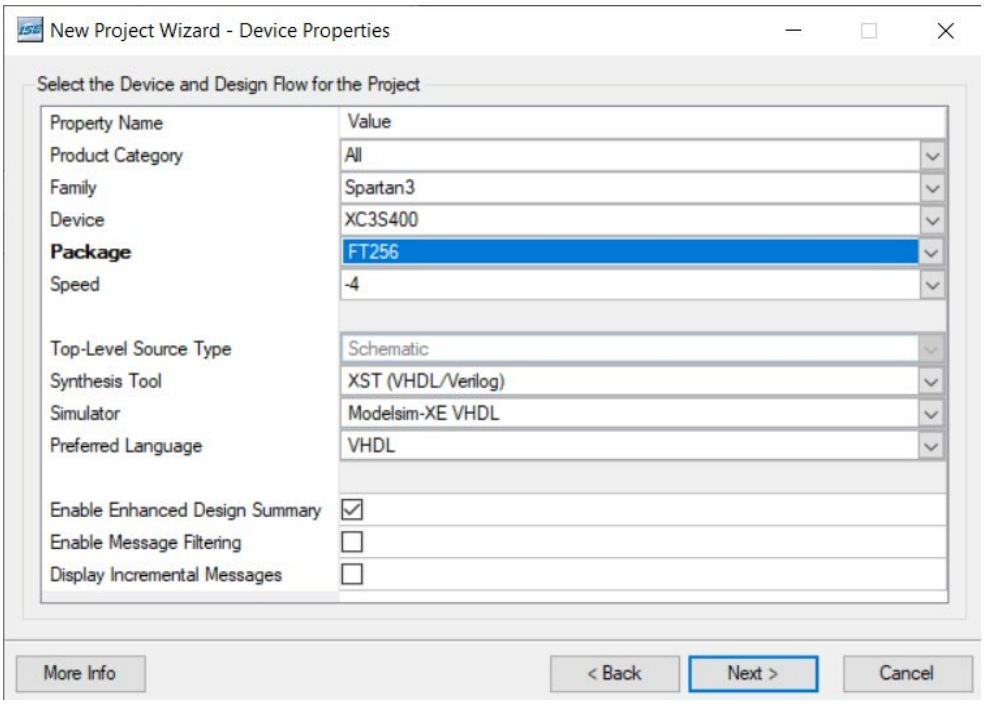
Implementarea și simularea unui sumator/scăzător pe 4 biți în VHDL

Aspecte teoretice:

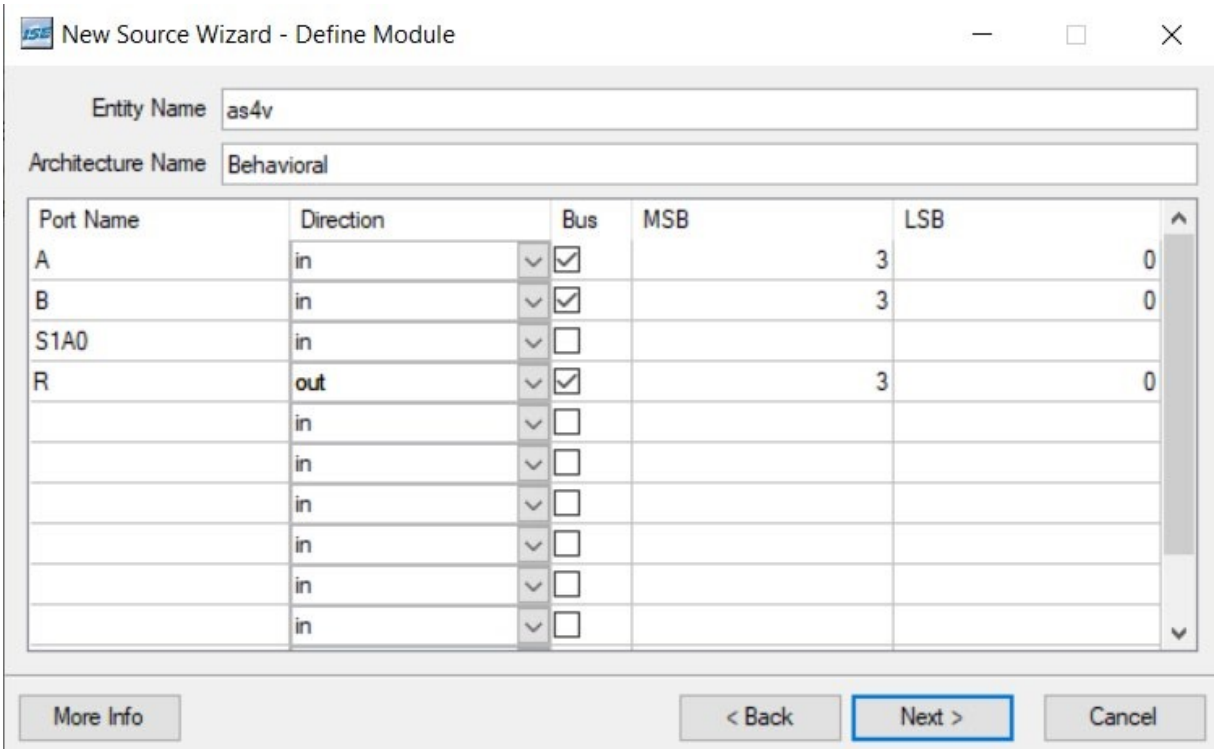
Un scăzător/sumator pe 4 biți este un circuit logic care poate efectua atât operații de adunare, cât și de scădere pe doua numere reprezentate pe 4 biți si returnează rezultatul și un bit de transport(carry-out). Acesta este un element important în arhitectura computerelor și poate fi utilizat pentru o varietate de aplicații. În acest raport tehnic, o sa prezint o implementare a unui astfel de circuit, implementarea a fost realizata folosind Xilinx, iar corectitudinea proiectului a fost verificata folosind Modelsim.

Pașii proiectului:

Pentru început am particularizat pentru a putea fi implementata pe placa Spartan 3.



Apoi am introdus porturile de intrare: A(3:0), B(3:0) fiecare pe 4 biți și S1A0 și portul de ieșire.



Am scris codul in VHDL:

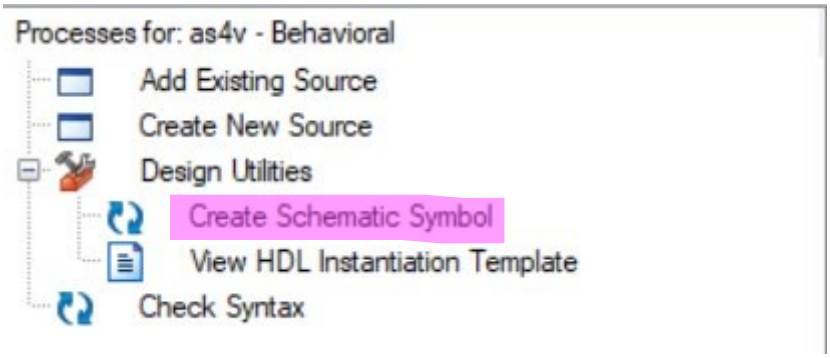
```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity as4v is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : in  STD_LOGIC_VECTOR (3 downto 0);
           S1A0 : in  STD_LOGIC;
           R : out  STD_LOGIC_VECTOR (3 downto 0)); --directie out, de tipul std_logic_vector cu 4 elemente, gama descrescatoare
end as4v;

architecture Behavioral of as4v is
   signal NB : std_logic_vector(3 downto 0);
   signal BNB : std_logic_vector(3 downto 0);

begin
   NB <= not B; --inversor for(k=0; k < 4 ; k++) NB(k) = !B(k)
   BNB <= B when S1A0='0' else NB; --multiplexor
   R <= A + BNB + S1A0;
end Behavioral;
```

În secțiunea "entity", sunt specificate porturile: A și B, care sunt intrări de tip std_logic_vector cu 4 elemente, S1A0, care este o intrare de tip std_logic, și R, care este o ieșire de tip std_logic_vector cu 4 elemente.

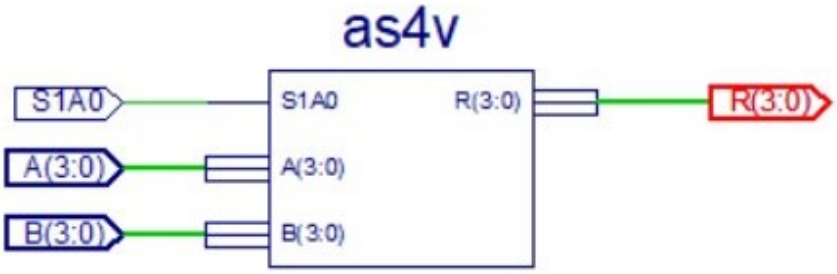
În secțiunea "architecture", se declară două semnale: NB și BNB. Semnalul NB este un inversor pentru intrarea B și este un std_logic_vector cu 4 elemente. Semnalul BNB este o ieșire de tip std_logic_vector cu 4 elemente și este calculat utilizând un multiplexor, care selectează între B și NB, în funcție de valoarea lui S1A0.

În final, ieșirea R este calculată ca suma dintre A, BNB și S1A0.

Apoi am creat o schema bloc a codului in VHDL.



Am adăugat marcher I/O pentru intrări și ieșire.



Corectitudinea sumatorului/scăzătorului a fost verificată cu ajutorul Modelsimului, in Test Bench WaveForm. Testbench-ul wave generează două numere binare de 4 biți (A și B) și le aplică ca intrare în circuit. Am adăugat câteva combinații dintre cele 512(deoarece avem 9 intrări => 2^9) și le-am verificat. Am descoperit că circuitul funcționează corect și generează rezultatul așteptat.

Am ales varianta Decimal(signed)(*Figura 1*), pentru a putea folosi numerele cu minus, cele din interiorul cercului[-8;7] (*Figura 2*), nu cele din mijloc [0,15]. Din acest motiv sunt șanse să ne lovim pe parcursul testelor de rezultate eronate(overflow).

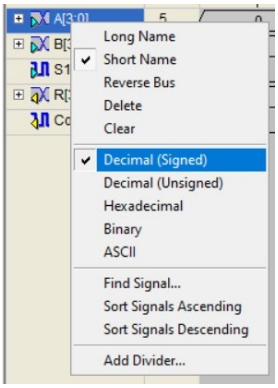
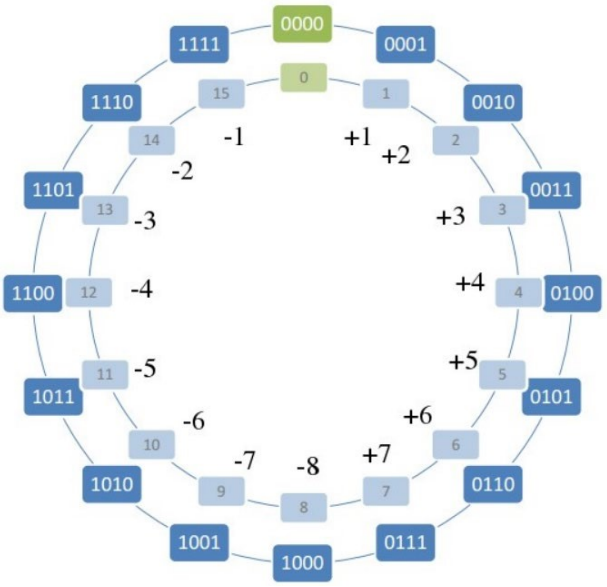


Figura 1



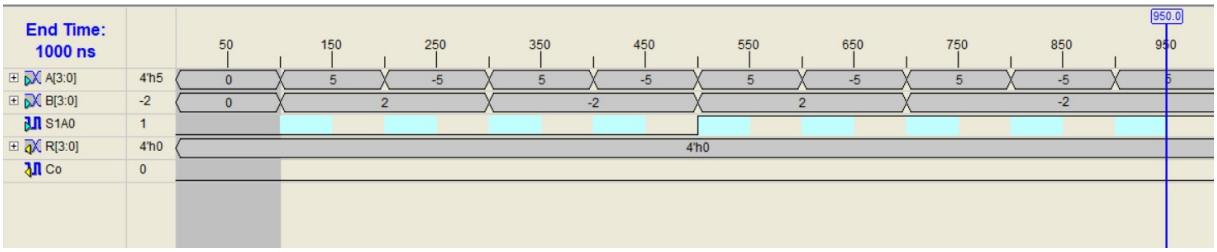
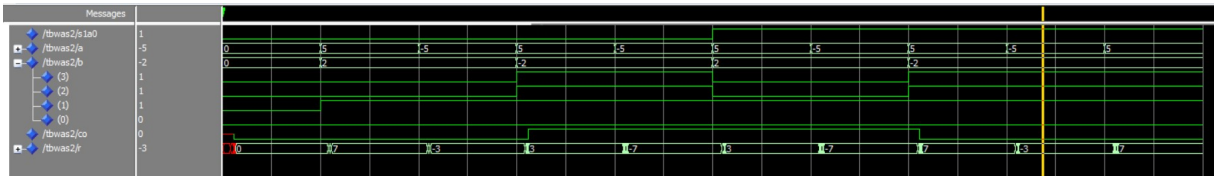


Figura 2



Circuitul scăzător/sumator pe 4 biți este un circuit logic util in industria calculatoarelor pentru a efectua operații de adunare si scădere pe numere reprezentate pe 4 biți, utilizând semnalul de selecție pentru a indica tipul de operație aritmetica care trebuie efectuata.

Doua întrebări

Sunt cele două scheme echivalente din punct de vedere funcțional?

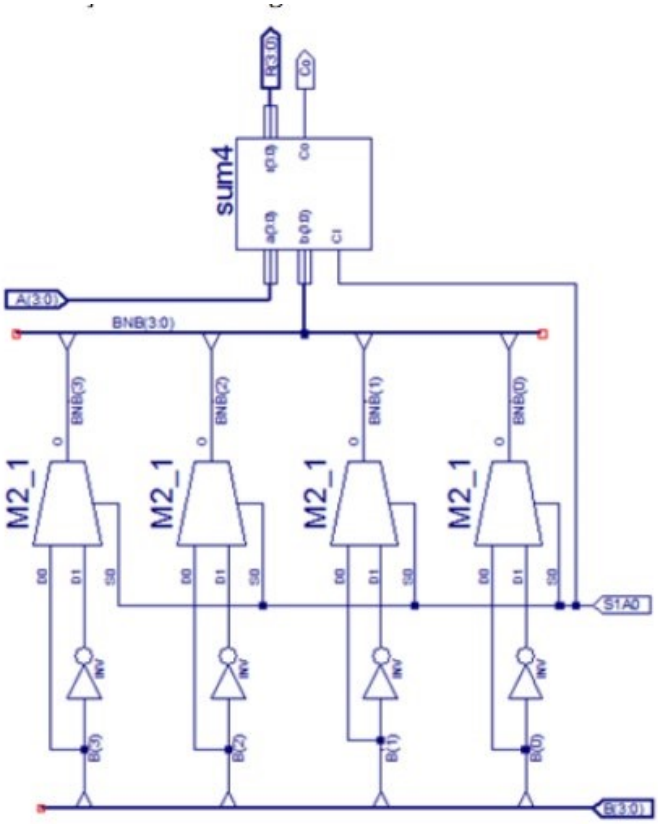
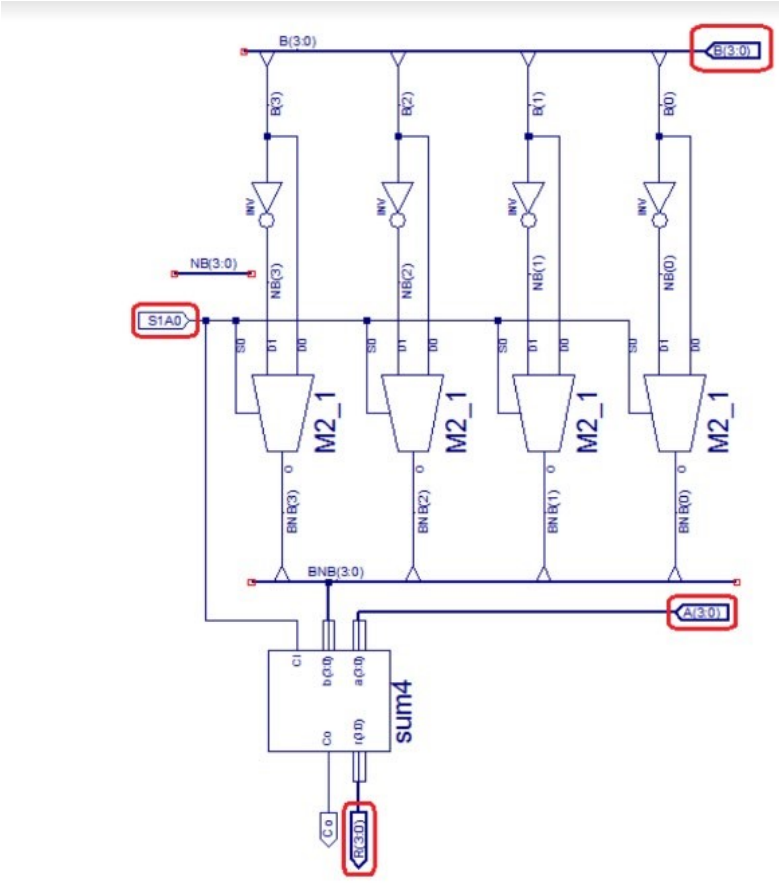


figura 10



În esență, cele două scheme descriu un sumator/scăzător cu două intrări A și B și o intrare de selecție S1A0, care determină operația de adunare sau scădere între cele două intrări. Operația de adunare se realizează prin simpla adunare a celor două intrări, în timp ce operația

de scădere se realizează prin inversarea valorii intrării B și apoi adunarea acesteia cu intrarea A și intrarea S1A0.

Dacă analizăm schema din figura 1, putem observa că aceasta este formată din patru inversoare, patru multiplexoare și un sumator. Inversoarele sunt folosite pentru a calcula inversul valorii intrării B, astfel încât să putem realiza operația de scădere. Multiplexoarele sunt utilizate pentru a selecta valoarea B sau inversul acestuia, în funcție de valoarea intrării de selecție S1A0. Sumatorul realizează operația de adunare sau scădere a celor două intrări, în funcție de valorile selectate de multiplexoare.

În figura 10, putem observa că aceasta este formată din aceleași elemente ca și schema din figura 1, dar ordinea elementelor este inversată. Sumatorul este descris prima dată, urmat de multiplexoare și apoi inversoare. Acest lucru nu afectează funcționarea schemei, deoarece aceeași operație de adunare sau scădere sunt realizate în ambele cazuri. În concluzie, cele două scheme sunt echivalente din punct de vedere funcțional și pot fi implementate în același mod, folosind aceleași elemente și aceeași ordine.

Dacă inversăm ordinea declarațiilor va mai funcționa corect sumator/scăzătorul?

Ordinea inițială a declaraților:

```
NB <= not B;

BNB <= B when S1A0 = '0' else NB;

R <= A + BNB + S1A0;
```

Ordinea inversată a declaraților:

```
R <= A + BNB + S1A0;

BNB <= B when S1A0 = '0' else NB;

NB <= not B;
```

Da, sumatorul va funcționa corect indiferent de ordinea declarațiilor, deoarece semnalul BNB este calculat in funcție de descrierea hardware si nu depinde de ordinea declarațiilor. In plus, limbajul VHDL este independent de ordinea declarațiilor, deci inversarea ordinii declarațiilor nu va afecta funcționalitatea circuitului generat.

Doar două declarații VHDL

BNB <= B when S1A0 = '0' else not B; - declarația de atribuire a valorii lui B la BNB atunci când S1A0 este 0 și a valorii complementului lui B atunci când S1A0 este 1.

R <= A + BNB + S1A0; - declarația de atribuire a sumei dintre A, BNB și S1A0 la R.