Rîcea Alexandra
Grupa: CR2.3A

Laborator 3 -Structura și Organizarea Calculatoarelor-

~Raport tehnic~

Implementarea și simularea unui sumator/scăzător pe 4 biți

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.

Componente circuitului:

Pentru a construi un scăzător/sumator pe 4 biți, am avut nevoie de: 4 multiplexoare 2:1, 4 inversoare si 4 sumatoare pe un bit(care a fost utilizat ca schema bloc din laboratorul precedent). Intrările A și B reprezintă cele două numere pe care dorim să le adunăm sau să le scădem, iar semnalul de selecție indică operația dorită (adunare sau scădere).

Explicația fiecărei componente din circuit:

- Un multiplexor este un circuit logic cu mai multe intrări și o singură ieșire. Acesta selectează una dintre intrările sale pe baza unui semnal de selecție și trimite valoarea respectivă la ieșire;
- Inversorul este un alt circuit logic care inversează valoarea sa de intrare, adică dacă intrarea este 1, atunci ieșirea va fi 0 și viceversa;
- Un sumator pe un bit este un circuit logic care poate aduna doi biți și poate produce un rezultat pe un 1 bit, împreună cu un bit carry-out.

Rolul fiecărei componente in circuit:

- Intrare de selecție pentru a selecta între adunare și scădere, in cazul in care semnalul este 0, se va realiza operația de adunare, iar in caz contrar se va realiza operația de scădere;
- Cea mai importanta componenta este sumatorul pe un singur bit deoarece aceasta este componenta de baza a întregului circuit. În general, un sumator pe un bit efectuează adunarea a doua numere binare de un singur bit si generează un rezultat si o valoare de transport. Pentru a construi sumatorul pe 4 biți, putem conecta în cascada 4 sumatoare pe un bit astfel încât bitul de transport generat de un sumator pe un bit să fie conectat la bitul de intrare de transport al sumatorului pe un bit următor. În acest fel, putem aduna 4 biți individuali și să generăm un rezultat de 4 biți și un bit de transport final.(creând o schema bloc si formând un sumator pe 4 biți);
- Cele patru multiplexoare sunt folosite pentru a selecta între valorile normale ale lui B sau complementul la doi al lui B în funcție de semnalul de selecție A/S. Dacă

semnalul de selecție este 0, atunci valoarea normală a lui B este introdusă în circuit prin intermediul multiplexoarelor. Dacă semnalul de selecție este 1, atunci complementul la doi al lui B este introdus în circuit prin intermediul multiplexoarelor și inversoarelor, cu ajutorul intrării de selecție.

- Cele patru inversoare sunt folosite pentru a obține complementul la doi al lui B, dacă semnalul de selecție este setat la 1. Complementul la doi al lui B se obține prin inversarea tuturor biților din B și adăugarea la acesta a valorii 1. Acest complement la doi este apoi introdus în intrările multiplexoarelor.

Funcționarea circuitului

În continuare, voi descrie modul de funcționare pentru fiecare operație.

a. Funcționarea circuitului in cazul adunării

Semnalul de selecție este setat la 0, pentru a indica adunarea. Intrările A si B reprezintă numerele pe 4 biți pe care dorim sa le adunam, iar bitul de transport de intrare(Ci) va lua valoarea semnalului de selecție(0), care trebuie de asemenea si el adunat deoarece aceasta este funcționalitatea unui sumator. Intrarea a(3:0) a sumatorului este conectata la magistrala A, iar intrarea b(3:0) a sumatorului este conectata la magistrala de ieșire a multiplexoarelor. Cum semnalul de selecție este setat la 0 => multiplexoarele selectează valorile normale ale lui B. Sumatorul pe 4 biți adună numerele A și B și adaugă bitul de transport(Ci-inițial). Rezultatul sumei pe 4 biți este obținut la ieșirea r(3:0).

b. Funcționarea circuitului in cazul scăderii

Semnalul de selecție este setat la 1, pentru a indica scăderea. Intrările A si B reprezintă numerele pe 4 biți pe care dorim sa le scădem, iar bitul de transport de intrare(Ci) va lua valoarea semnalului de selecție(1), care trebuie de asemenea si el adunat deoarece aceasta este funcționalitatea unui sumator. Intrarea a(3:0) a sumatorului este conectata la magistrala A, iar intrarea b(3:0) a sumatorului este conectata la magistrala de ieșire a multiplexoarelor. Cum semnalul de selecție este setat la 1 => multiplexoarele selectează complementul fata de 2 a lui B. Complementul fata de 2 a lui B se calculează cu ajutorul formulei (!B+1), pentru început negam cu ajutorul inversoarelor fiecare bit a lui B, iar la final când adăugam bitul de transport(Ci) => complementul fata de 2 al numărului B. Sumatorul pe 4 biți adună numerele A și B și adaugă bitul de transport(Ci). Rezultatul sumei pe 4 biți este obținut la ieșirea r(3:0). După cum se poate observa, scăderea pe care am vrut să o realizăm s-a transformat de fapt tot într-o adunare.

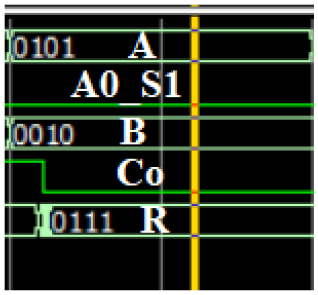
Un exemplu de intrare a sumatorului/scăzătorului pe 4 biți:

a. Adunare

A=0101 B=0010

1. Setăm semnalul de selecție la 0 pentru a indica operația de adunare;
2. Intrările A și B sunt conectate direct la intrările sumatorului pe 4 biți. A este 0101, iar B este 0010;
3. Cele patru multiplexoare sunt folosite pentru a selecta valorile normale ale lui B, deoarece semnalul de selecție este setat la 0. Valorile normale ale lui B sunt 0010;
4. Cele patru inversoare nu sunt utilizate în această operație, deoarece semnalul de selecție este 0;
5. Începem operația de adunare, iar sumatorul pe 4 biți adună numerele A și B, adăugând bitul de transport inițial, care este 0;
6. Sumatorul pe 4 biți obține rezultatul adunării, care este 0111;
7. Rezultatul sumei pe 4 biți este obținut la ieșirea r(3:0), iar Co=0.

Rezultat Modelsim:

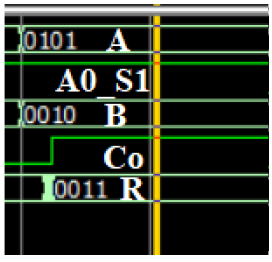


b. Scădere

A=0101 B=0010

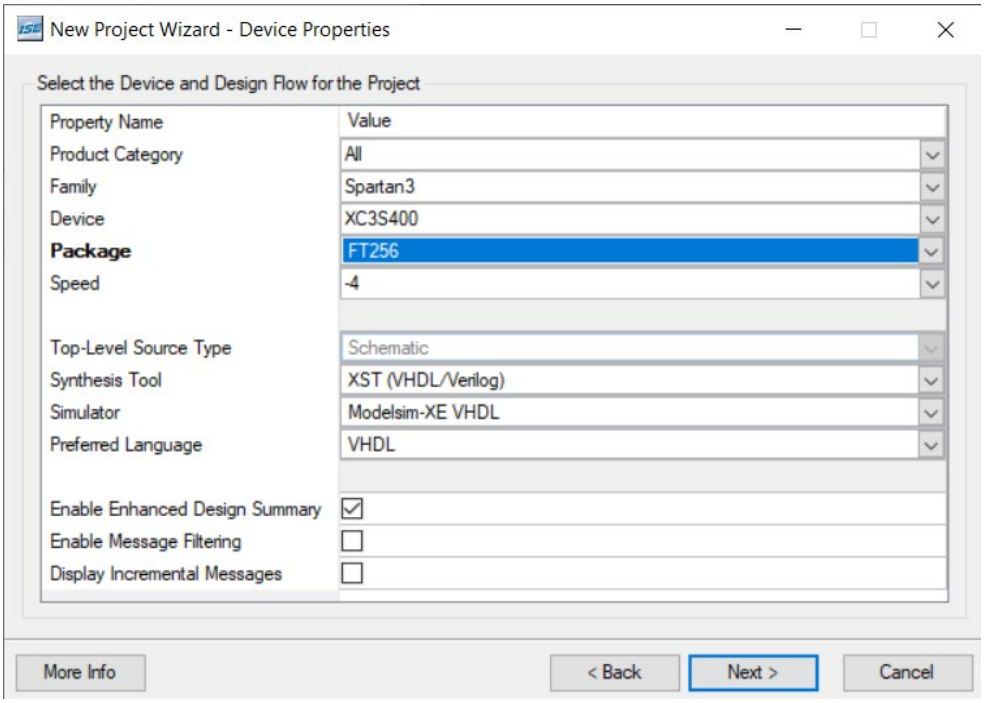
1. Setăm semnalul de selecție la 1 pentru a indica operația de scădere;
2. Intrările A și B sunt conectate direct la intrările sumatorului pe 4 biți. A este 0101, iar B este 0010;
3. Cele patru multiplexoare sunt folosite pentru a selecta complementul la doi al lui B, deoarece semnalul de selecție este setat la 1;
4. Introducem în multiplexoare fiecare bit a lui B negat. Cele patru inversoare sunt utilizate pentru a obține inversul numărului. Inversăm fiecare bit din B(1101);
5. Începem operația de adunare, iar sumatorul pe 4 biți adună A cu !B si cu carry-in-ul inițial, care este 1 (de la semnalul de selecție);
6. Sumatorul pe 4 biți obține rezultatul corect, care este 0011;
7. Rezultatul scăderii numerelor binare 0101 și 0010 este obținut la ieșirea r(3:0), iar Co=1.

Rezultat Modelsim:

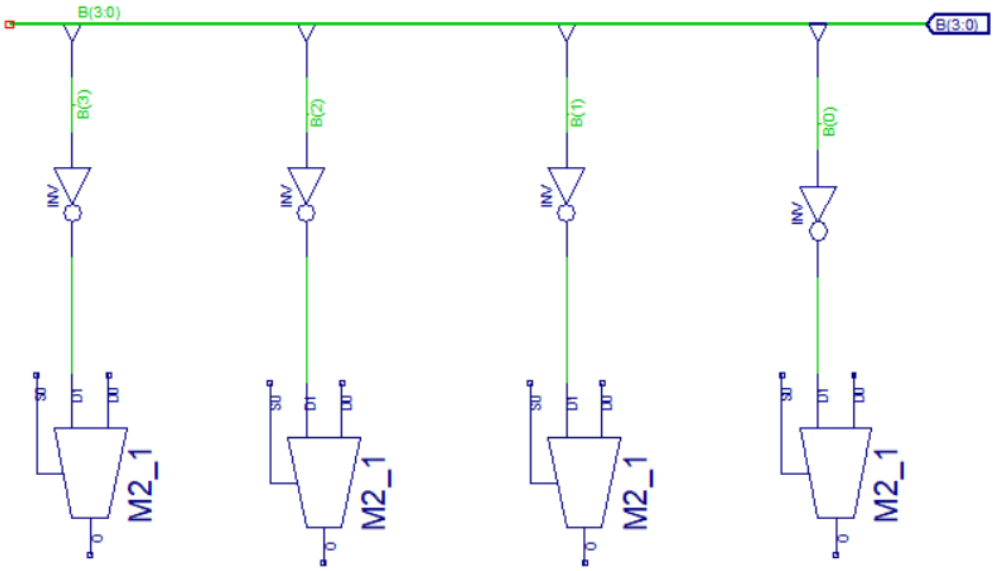


Pașii proiectului:

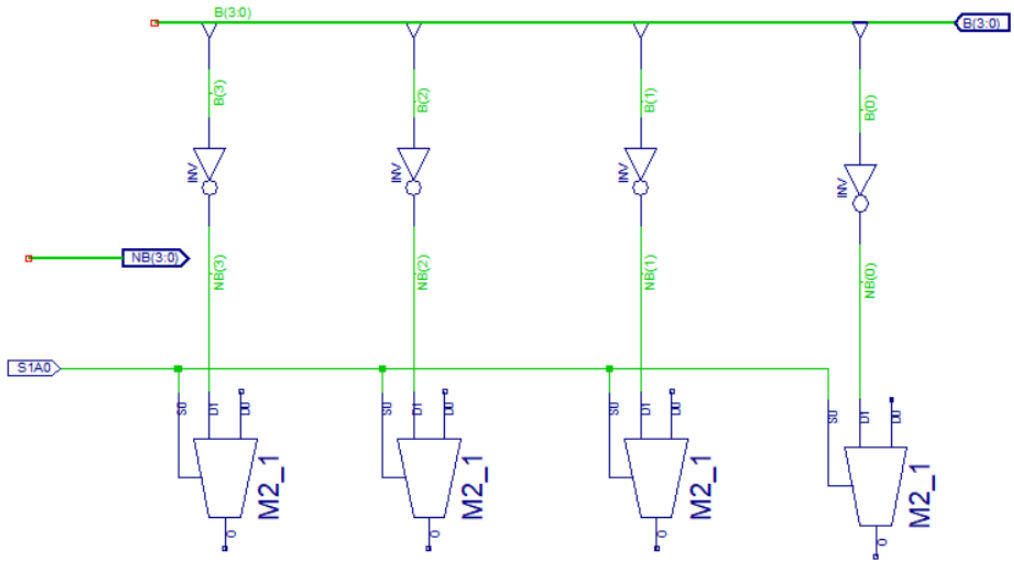
Pentru început am particularizat pentru a putea fi implementata pe placa Spartan 3.



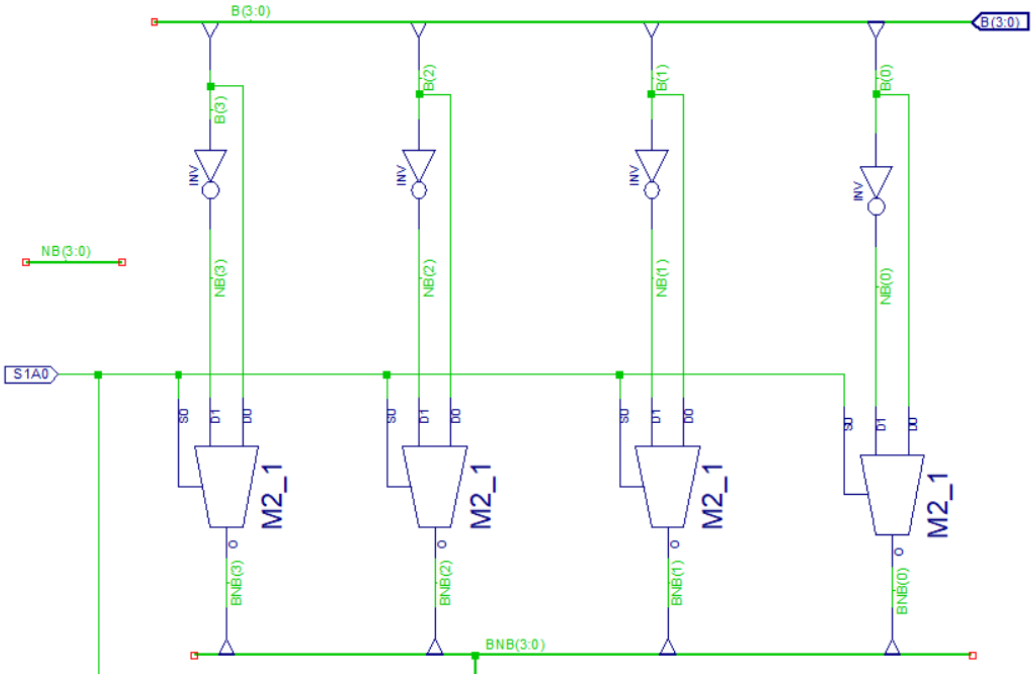
Am adăugat prima magistrală de intrare B(3:0), 4 multiplexoare și 4 inversoare. Multiplexoarele mă vor ajuta pentru a selecta operația de intrare si ieșire, iar inversoarele pentru a calcula complementul numărului B fata de 2.



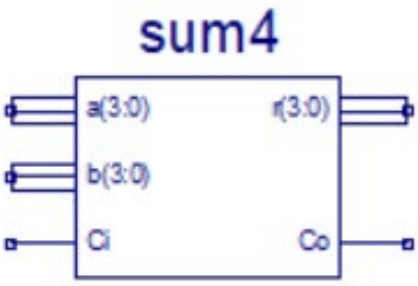
Iar apoi, am adăugat intrarea de selecție care a fost conectata la intrarea de selecție a multiplexoarelor. Și am selectat numele bitului pe care dorim să îl luam de pe magistrala.



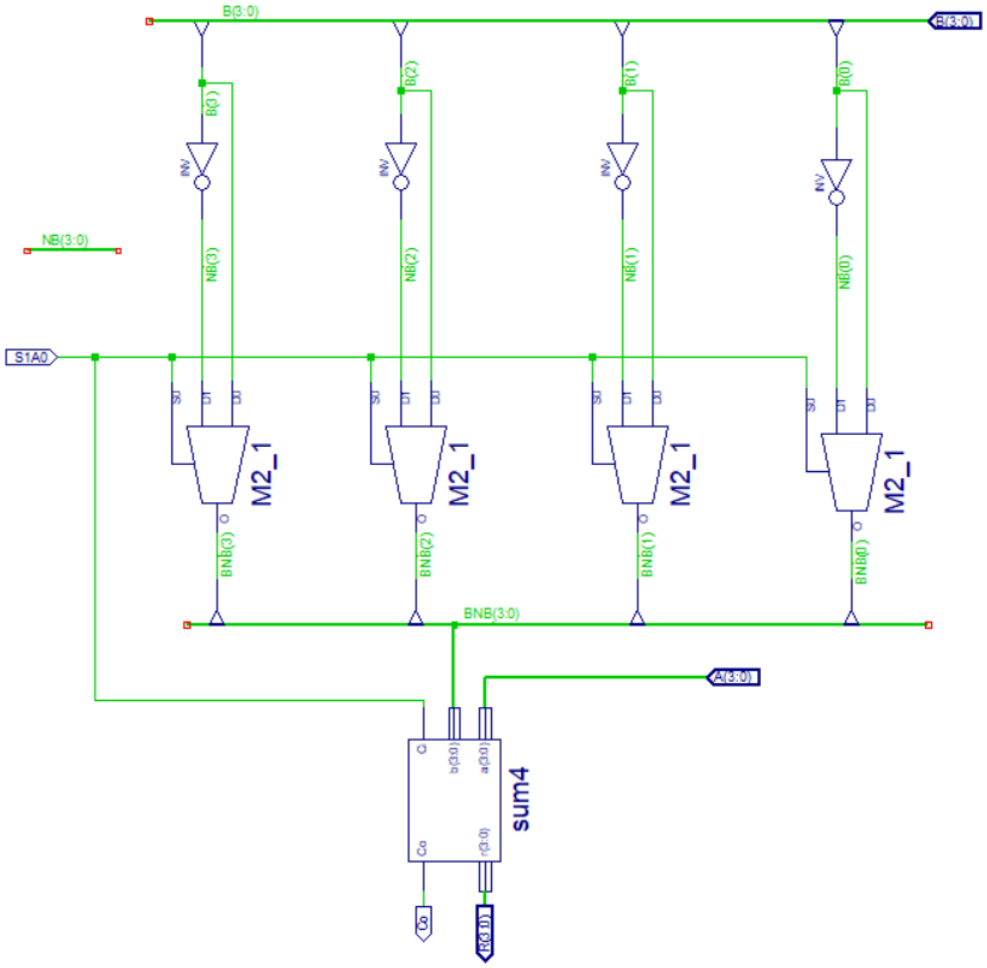
Ieșirile fiecărui multiplexor au fost adăugate in magistrala BNB(care primește valoarea B sau !B, în funcție de intrarea de selecție) și am selectat numele bitului pe care dorim să îl luăm pe magistrala.



Am adăugat o copie a sursei din laboratorul precedent(sumatorul pe 4 biți) si am creat o schema bloc a acesteia, pentru a putea realiza operația de adunare/scădere pe 4 biți.



La final, am conectat intrarea sumatorului a(3:0) la magistrala de intrare A(3:0) și intrarea sumatorului b(3:0) la magistrala rezultata din multiplexoare BNB(3:0), iar la Ci am conectat intrarea de selecție. Rezultatele sunt evidențiate de marcherele de ieșire Co și R(3:0).


B(3:0)
B[3:0]
B(3)
B(2)
B(1)
B(0)
INV
NB(3:0)
NB(3)
NB(2)
NB(1)
NB(0)
S1A0
S0
D1
D0
M2_1
BNB(3)
BNB(2)
BNB(1)
BNB(0)
BNB(3:0)
A[3:0]
b(3:0)
a(3:0)
Co
f(3:0)
sum4
So
F[3:0]

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).

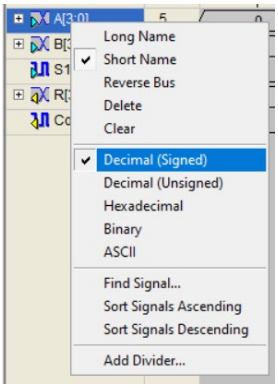


Figura 1

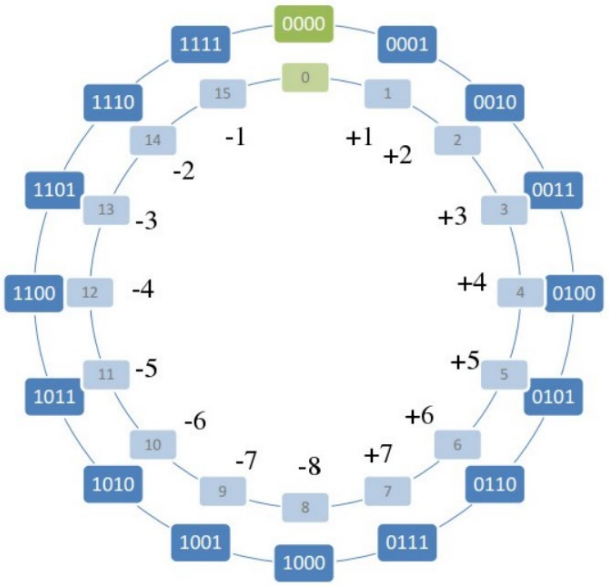
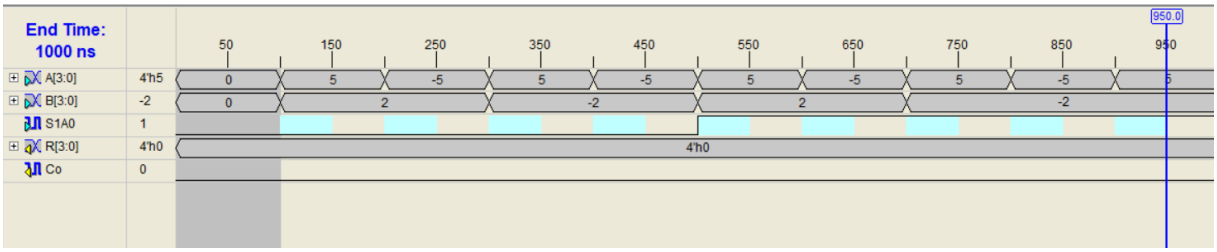
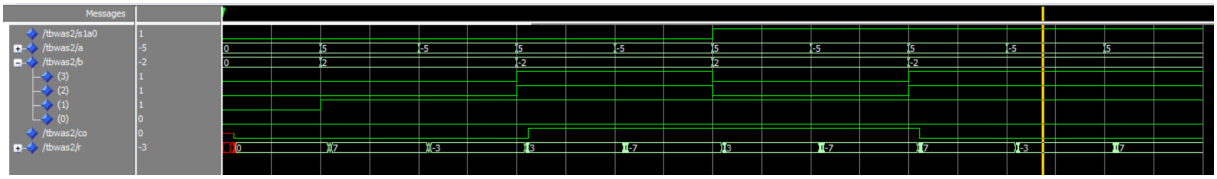


Figura 2





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.

1. Să se scrie tabelul de adevăr pentru funcția depășire, funcție ce depinde de variabilele enumerate mai sus.

0->+, 1->-

Semnul lui A(A)	Operația(S)	Semnul lui B(B)	Semnul rezultatului(R)	Overflow
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Pentru a determina dacă există overflow pe 4 biți, trebuie să luăm în considerare următoarele reguli:

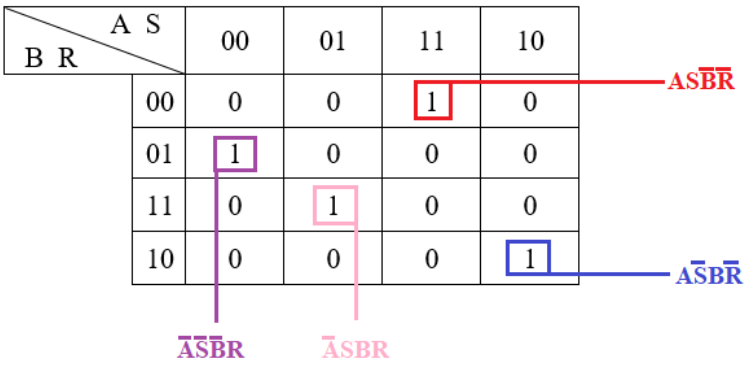
- Dacă se adună două numere pozitive și rezultatul este mai mare decât 7, sau se scad două numere negative și rezultatul este mai mic decât -8, atunci există overflow pe 4 biți.
- Dacă se adună un număr pozitiv și unul negativ, sau se scad un număr pozitiv și unul negativ, atunci nu poate exista overflow pe 4 biți.

2. Să se implementeze această funcție (Overflow) și să se minimizeze.

Operație (0->+,1->-)	A(3:0)	B(3:0)
+	+5	+4
+	-5	+4
+	+5	-2
+	-5	-4
-	+5	+2
-	-5	+4
-	+5	-4
-	-5	-2

A = Semnul lui A; B = Semnul lui B; S = Semnul operației; R = Semnul rezultatului

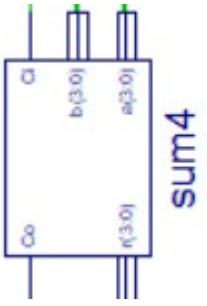
B R \ A S	00	01	11	10
00	0	0	1	0
01	1	0	0	0
11	0	1	0	0
10	0	0	0	1



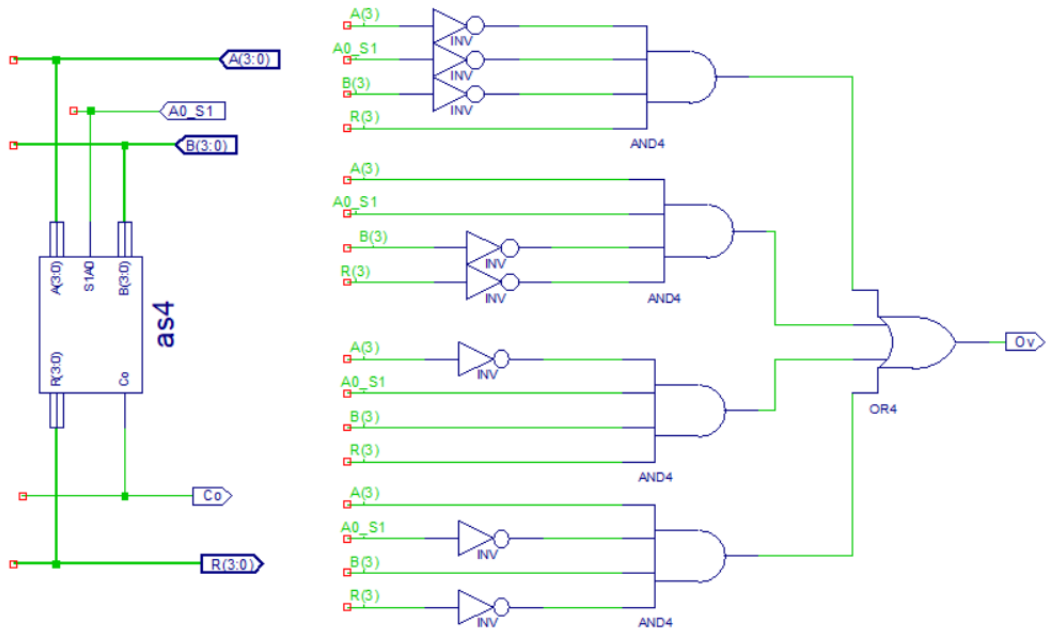
$$F(A,S,B,R) = \bar{A}\bar{S}\bar{B}R + AS\bar{B}\bar{R} + \bar{A}SBR + A\bar{S}B\bar{R}$$

Aceasta funcție am implementat-o in Xilinx, iar valorile din tabelul precedent le-am verificat cu ajutorul Modelsimului.

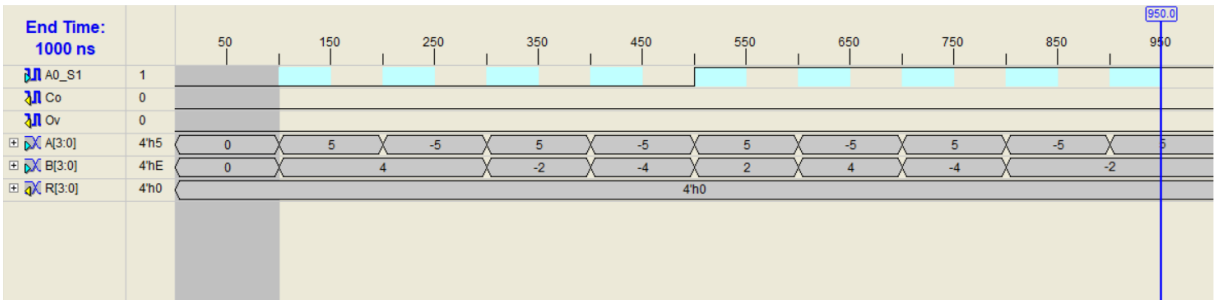
Pentru început, am adăugat o copie a sursei a sumatorului/scăzătorului pe 4 biți și am creat o schema bloc a acesteia pentru a simplifica schema finala.



Am adăugat cele 2 magistrale de intrare: A(3:0) si B(3:0), numerele care trebuie adunate si intrarea de selecție(A0_S1), care stabilește ce operație se va realiza(Adunare = 0, Scădere = 1) si doua ieșiri pentru sumatorul/scăzătorul pe 4 biți: Co(carry-out) si R(3:0), care reprezintă suma/diferența pe 4 biți. Iar apoi am implementat funcția de mai sus cu 13 porți logice: 4 AND4, 8 INV si OR4. A(3), B(3), R(3) reprezintă biții cei mai semnificativi ai numerelor A, B si R(suma), care în decimal cu semn reprezintă biții semnului deoarece pentru a afla dacă s-a produs depășire ne interesează doar semnul numerelor: A, B, R si operația care s-a realizat între ele (A0_S1, intrarea de selecție). Variabila care ne indica daca a apărut overflow se numește Ov.



In TBW am introdus valorile indicate de tabelul de la exercițiul 2.



Rezultatele din Modelsim au fost corecte, conform tabelului.

Operație (0->+,1->-)	A(3:0)	B(3:0)	Overflow
+	+5	+4	1
+	-5	+4	0
+	+5	-2	0
+	-5	-4	1
-	+5	+2	0
-	-5	+4	1
-	+5	-4	1
-	-5	-2	0

Se poate observa ca ov are valoarea 1 doar atunci când am testat valorile: (5,4); (-5,-4), (-5,4) si (5,-4).