 UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE



CALCULATOR DE BUZUNAR

Îndrumător: Ing. Rusu Vlad-Andrei, student: Toader Sidonia

Grupa 30218, anul I

1. Specificația proiectului

Cerința:

*Să se proiecteze un calculator de buzunar cu operatii aritmetice fundamentale (adunare, scadere, inmultire, impartire). Operatiile de inmultire si impartire se vor implementa folosind algoritmi specifici, nu operatorii limbajului. Operanzii sunt reprezentati pe 8 biti cu semn. Operanzii si operatorii vor fi introdusi secvential in forma zecimala. Se vor folosi afisajele cu 7 segmente de pe placutele cu FPGA.*

Așadar, în cadrul proiectului se va realiza un calculator de buzunar capabil sa realizeze operațiile aritmetice fundamentale. Capacitatea acestuia este limitată la 8 biți cu semn, așadar operanzii introdusi trebuie sa fie intre -128 si 127.

Proiectul este scris in limbajul VHDL, iar sintetizarea și implementarea codului este realizata cu ajutorul programului Vivado.

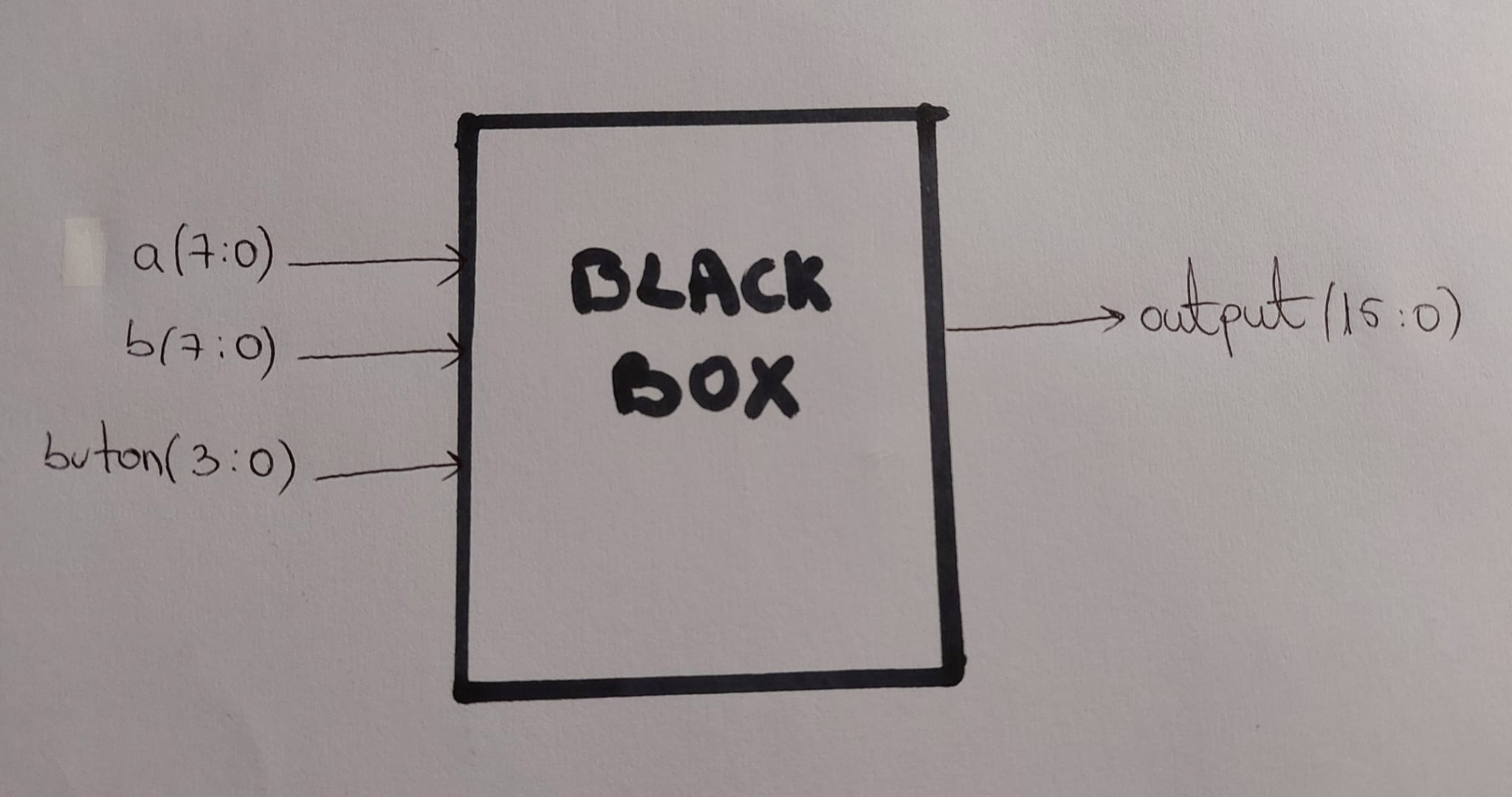
2. Proiectarea

În continuare, am început efectiv proiectarea calculatorului de buzunar, în mai multe etape.

2.1 Schema bloc

*Cutia neagra*

Puțin mai jos găsim cutia neagră, adică o privire de ansambul asupra proiectului. Aici am scos în evidență toate intrările și ieșirile necesare funcționării calculatorului nostru de buzunar.



Intrările sunt reprezentate de două numere pe 8 biți, dintre care 1 bit pentru semn. De asemenea, intrarea “buton”, pe 4 biți stabilește operația pe care urmează să o efectuăm, in functie de butonul pe care apasam.

*0001 – adunare , 0010 – scădere, 0100 – înmulțire, 1000 – împărțire*

**

IMPARTIRE

ADUNARE

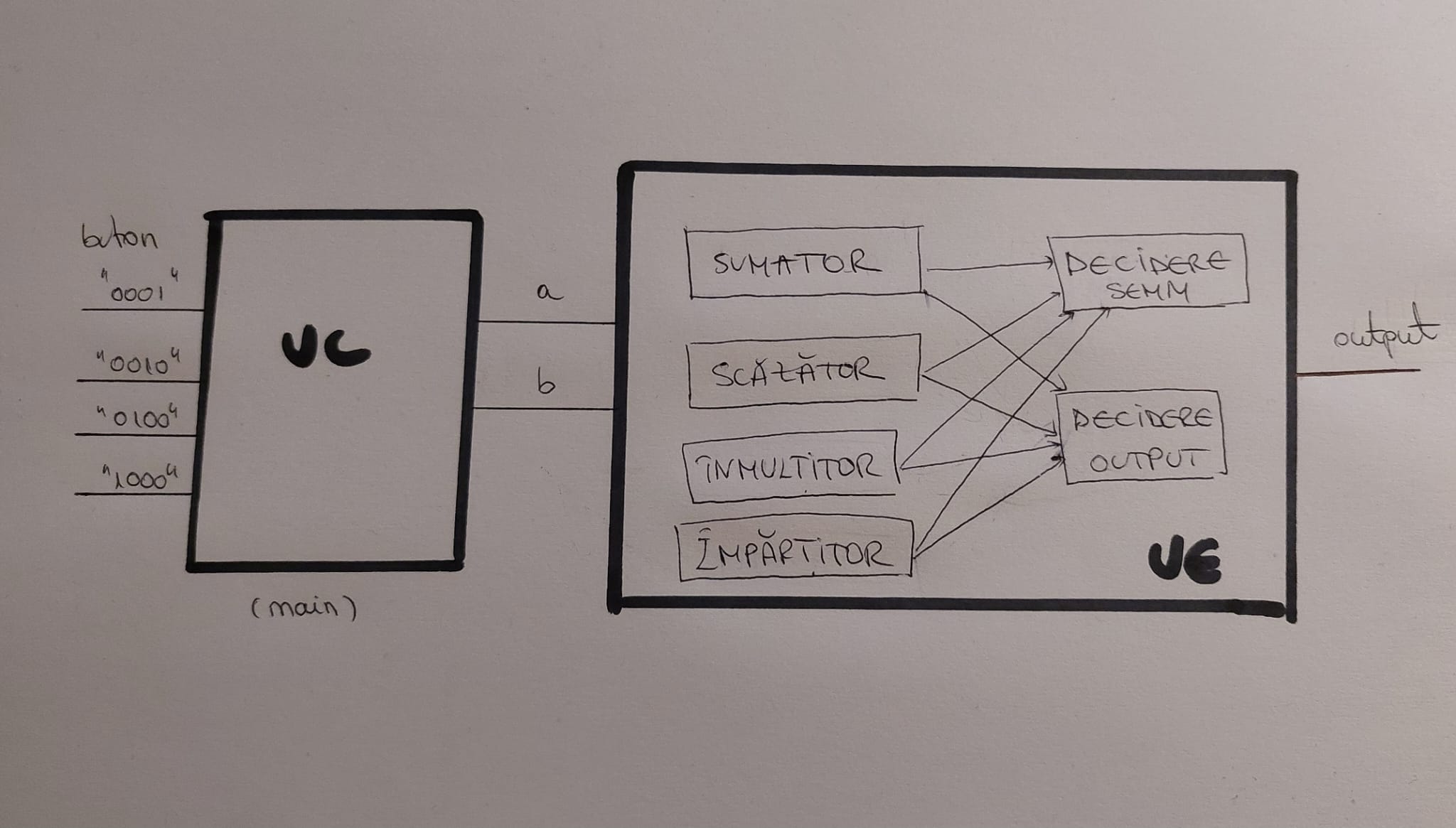
SCADERE

INMULTIRE

Ca ieșiri vom avea 1 output final pe 16 biti: se va alege intre rezultatul adunarii, diferentei, împărțirii (semnale pe 8 biti la care s-au mapat restul de 0) si inmultirii, rezultat care se va extinde la 16 biti (15 biti ai bitul de semn).

2.2 Unitatea de comandă și unitatea de execuție

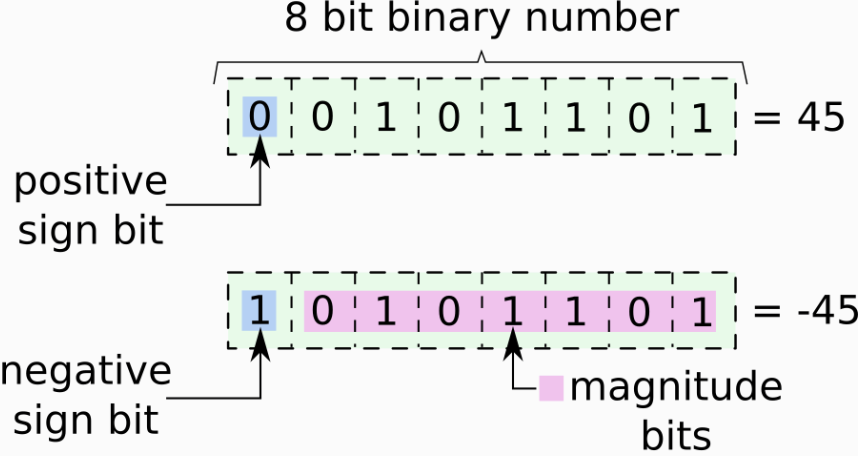
După ce am stabilit toate intrările și ieșirile, următorul pas este împărțirea acestora unității potrivite. Întrucât unitatea de comandă are “responsabilitatea” de a stabili operația dorită de utlizator si de a transmite mai departe unității de execuție care realizează efectiv operația respectivă, schema arată în felul următor:



2.3 Componentele unității de execuție

După cum am spus și mai sus, unitatea de execuție este “creierul” întregului ansamblu, aceasta se ocupă de realizarea tuturor opearațiilor.

Așadar, conține toate componentele principale, enumerate mai jos.

De mentionat este faptul ca toate componentele opereaza cu modulul numerelor introduse, ulterior stabilindu-se semnul rezultatului in functie de operatia selectata.

1. SUMATOR

Realizeaza suma, respectiv diferenta numerelor in functie de semnul fiecarui input. Ambelor numere le este mapat un bit de 0 in locul celui mai semnificativ bit, urmand ca ulterior sa se stabileasca semnul rezultatului comparand modulul acestora.

1. SCAZATOR

Scazatorul se comporta asemanator ca si sumatorul prezentat anterior.

1. INMULTITOR PE 8 BITI

Înmulțitorul este o alta component importantă, aceasta realizează înmultirea prin adunări repetate. Întrucât rezultatul obținut poate depăși 7 biți (8 cu tot cu semn), cum ar fi în cazul înmulțirii (-128) \* (-128), rezultatul se extinde la 16 biti inclusiv cu semn. Așadar, am realizat, de asemenea, un sumator pe 16 biți pentru a realiza corect înmultirea si a aduna produsele partiale.

d) IMPARTITOR PE 8 BITI

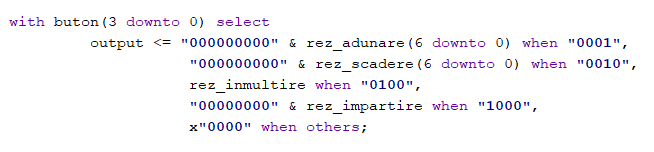
Această componentă realizează împărțirea a doua numere pe 8 biți (7 biti si un 0 mapat in locul semnului), rezultatul fiind tot pe 8 biți. De asemenea pe lângă rezultat vom avea și un rest, numerele fiind întregi.

1. DECIDERE SEMN

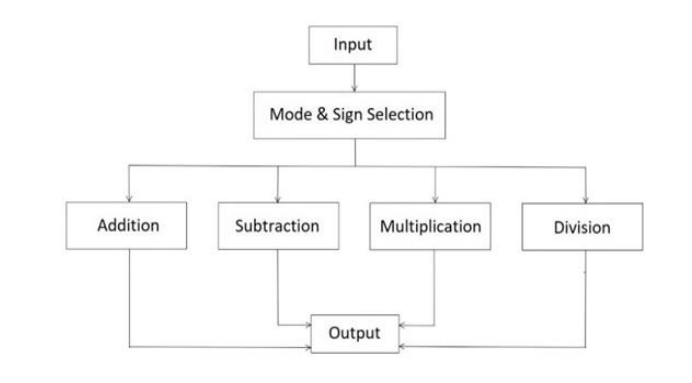
Prin intermediul acestei componente se decide semnul fiecarei operatii. In functie de butonul apasat, se trimite ca input afisorului semnul corespunzator operatiei asociate butonului respectiv.

1. DECIDERE OUTPUT

Aceasta componenta atribuie output-ului, printr-o simpla instructiune concurenta, rezultatul operatiei selectate de catre utilizatorul calculatorului de buzunar.



2.4 Diagrama bloc a functionarii calculatorului



2.5 Lista detaliata a principalelor componente folosite

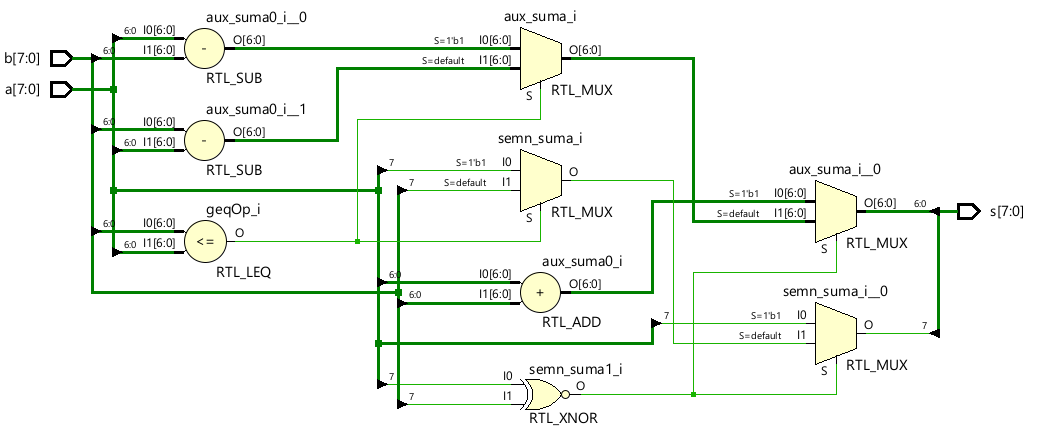
1. SUMATOR

Sumatorul, precum a fost prezentat anterior, poate realiza atat operatia de adunare, cat si de scadere a numerelor in functie de semnul acestora. Astfel, apar 2 cazuri:

1. Cand semnul celor 2 numere coincide, atunci se realizeaza suma modulelor lor, semnul rezultatului fiind preluat de la unul dintre ele.
2. Cand semnul celor 2 numere este opus, atunci se realizeaza diferenta dintre modulul numarului mai mare si modului numarului mai mic, urmand ca rezulatul sa preia semnul numarului cu magnitudinea mai mare.

Se poate observa comportamentul din spatele al sumatorului in imaginea urmatoare:

*Schema circuitului:*



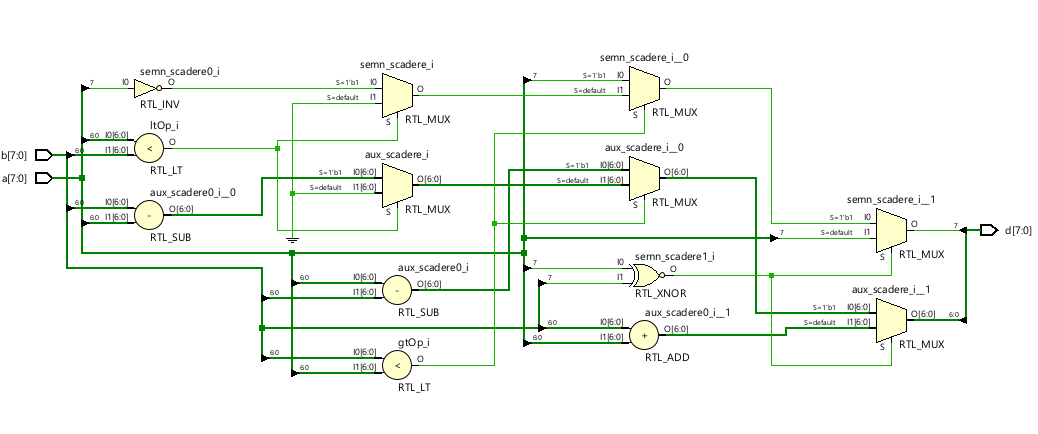
1. SCAZATOR

Asemenea sumatorului, scazatorul poate realiza atat operatia de scadere, cat si de adunare a numerelor. Si aici putem intalni 2 cazuri:

1. Cand semnul celor 2 numere este opus, atunci se realizeaza suma modulelor, urmand ca rezulatul sa preia semnul unuia dintre ele.
2. Cand semnul celor 2 numere coincide, atunci vor exista inca alte 2 subcazuri, in functie de magnitudinea numerelor: daca modulul primului numar este mai mare decat al celui de-al doilea, se va face diferenta dintre ele, iar rezultatul pastreaza semnul numerelor; in caz contrar, se schimba operanzii intre ei, iar semnul rezultatului va fi opus semnului numerelor.

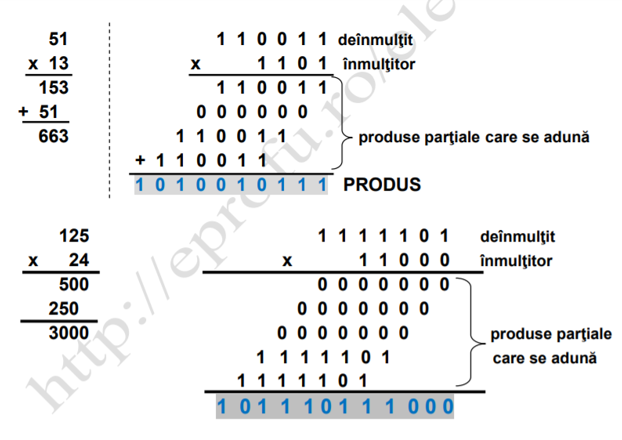
Se poate observa comportamentul din spate al scazatorului in imaginea urmatoare:

*Schema circuitului:*



1. INMULTITOR PE 8 BITI

Pentru a realiza înmulțitorul, am urmărit tehnica de înmulțire a numerelor zecimale, după care am înlocuit numerele în numere binare. Poza următoare prezintă un exemplu de înmulțire, din care reiese și algoritmul folosit.



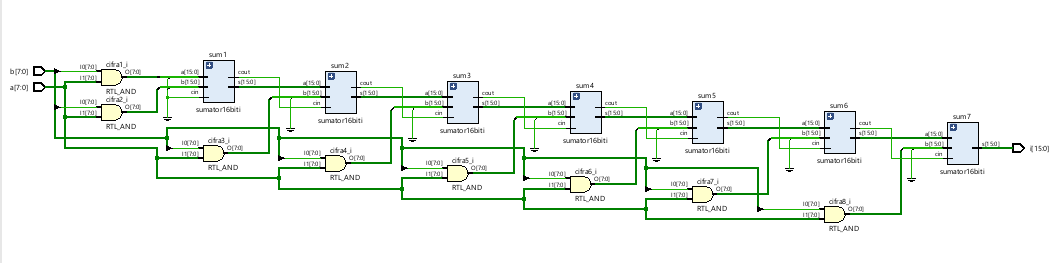
În primul rând, trebuie să înmulțim fiecare cifră a deînmulțitorului cu înmulțitoru. Pentru a realiza acest lucru fără a folosi operatorii limbajului, am concatenat fiecare bit al deînmuțitului cu el însuși pana am obținut un numar de 8 biți, după care am realizat un ȘI (and) între acesta și înmulțitor. Această operație a dus la același rezultat ca în exemplu. Am reținut rezultatul obținut, si am repetat operația cu următorii biți pana la finalizarea acestora.

În acest moment, avem toate produsele parțiale, pe care mai trebuie doar să le adunăm. Însă aceasă adunare nu este una clasică, întrucât, după cum arată și exemplul, pentru a obține rezultatul correct, fiecare produs partial se mută cu o poziție la dreapta. Am rezolvat această problemă și am obținut rezultatul corect concatenând zero-uri la începutul și finalul produsului partial, astfel încât pozițiile să ramână cele corecte, după care am realizat adunarea, cu ajutorul a 7 sumatoare pe 16 biți. Am folosit sumatoare pe 16 biți deoarece adăugarea zero-urilor a mărit numărul de biți.

Menționez că am adăugat zero-urile la începutul și sfârșitul numerelor fără nicio grijă, fiindcă acestea nu modifică în niciun fel valoarea corectă a numărului.

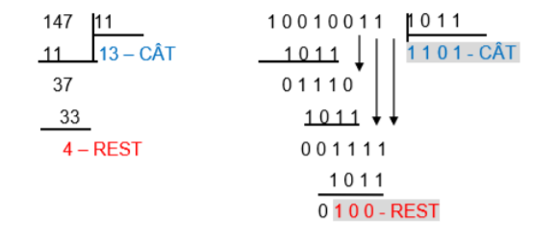
Se poate observa comportamentul din spate al inmultitorului in imaginea urmatoare:

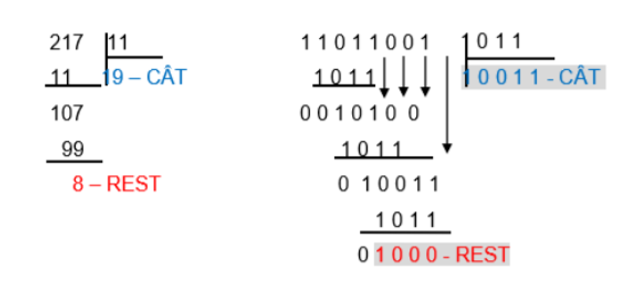
*Schema circuitului:*



1. IMPARTITOR PE 8 BITI

La împărțitor am procedat la fel ca la înmulțitor, urmărind îndeaproape algoritmul de împărțire a numerelor zecimale, după care a celor binare. Mai jos atașez două exemple sugestive.





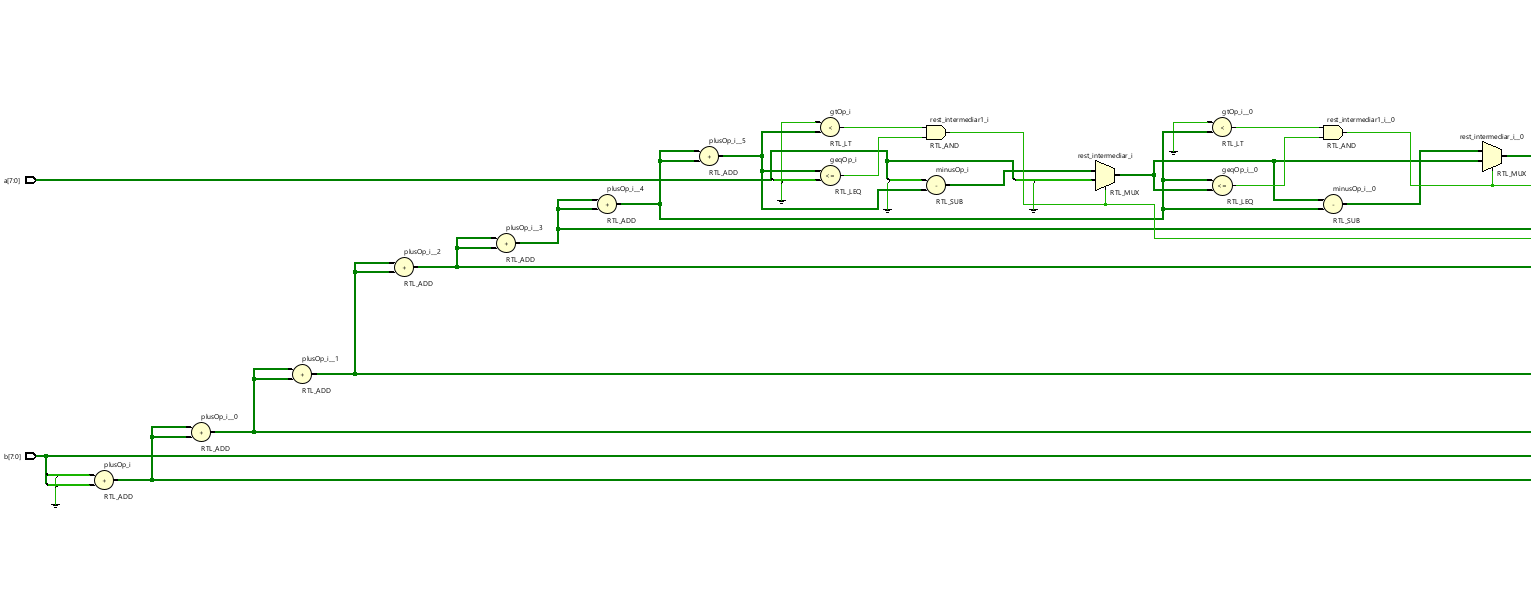
În urma exemplelor de mai sus, am observat că în algoritmul împărțirii intervine un factor extrem de important, și anume dacă valoarea împărțitorului este mai mica sau mai mare decât valoarea deîmpărțitului ( nr de biți egali cu ai împărțitorului de la stânga la dreapta). Adică, în cazul în care împărţitorul 1011 este mai mare decât primii 4 biţi ai deîmpărţitului 1001 împărţitorul se va împărţii la primi 5 biţi ai deîmpărţitului 10010, (poza 1). Iar în caz contrar, când numărul format din primi 4 biţi ai deîmpărţitului 1101 este mai mare decât 1011, împărțirea se face normal, iar câtul devine 1.

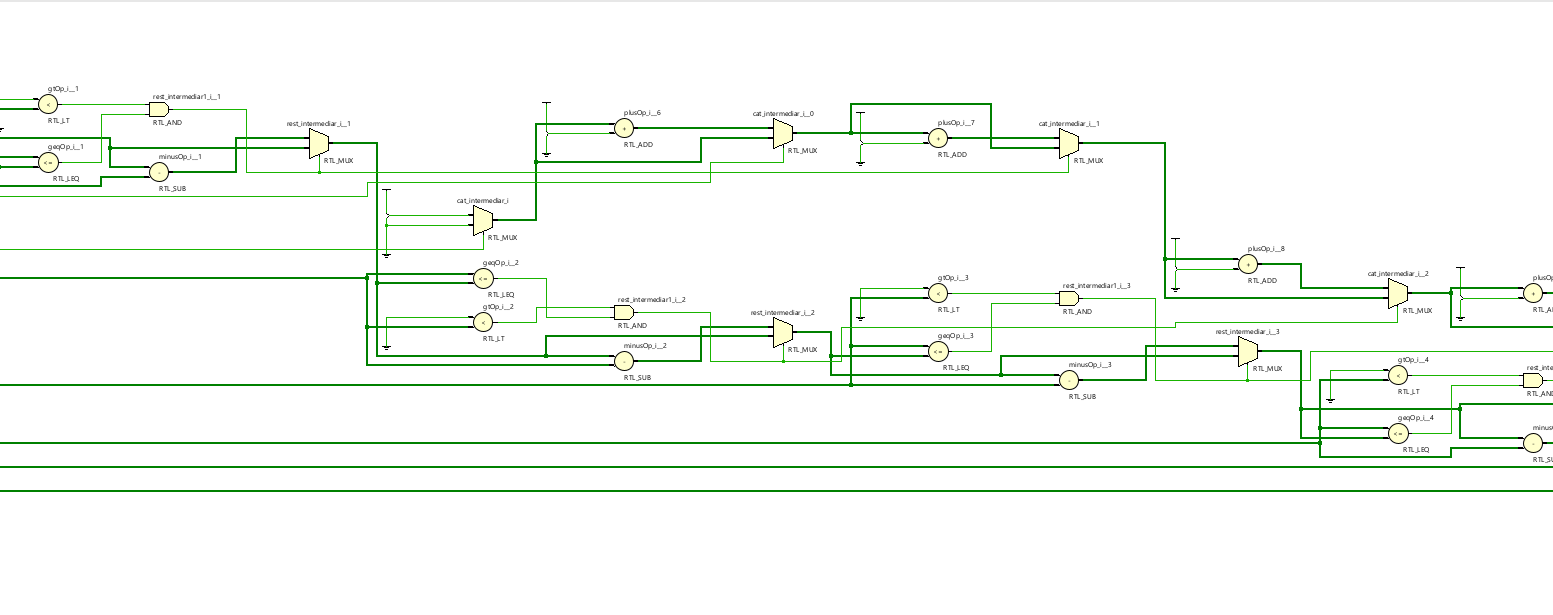
Pentru a evita primul caz care complică împărțirea, si pentru a mă asigura ca voi avea de-a face de fiecare dată cu cel de-al doilea caz din a doua poză, am hotărât sa realizez shiftarea împărțitorului de nr. max de ori, adica de 8 ori, și să rețin rezultatele. Împărțirea se va realiza doar în momentul în care deîmpărțitul devine mai mic decât împărțitorul. Lucru care poate deveni posibil dupa prima shiftare, a doua shiftare, etc.. motiv pentru care am decis să realizez toate shiftarile posibile, iar apoi să verific în care caz se încadrează.

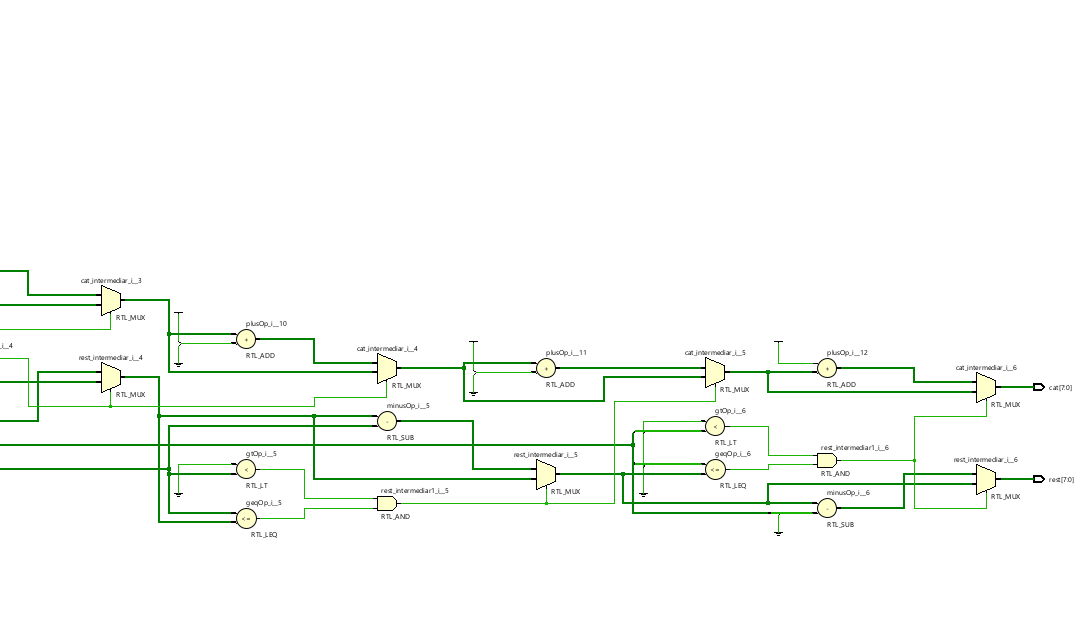
De asemenea, am avut grijă să nu pierd valoarea împărțitorului prin acele shiftări la stânga. Pentru aceasta, în momentul în care am gasit poziția corectă, am realizat scăderea dintre deîmpărțit și împărțitor, si am pus în cât valoarea corespunzătoare numărului de shiftare. Dacă am ajuns doar la prima shiftare, adaug 2, dacă am ajuns la a doua shiftare, adaug 2^2, și tot așa până la numarul maxim de shiftări, când adaug 2^7=128. Este posibil și să nu fie nevoie de nicio shiftare, moment când adăugăm 1.

Se poate observa comportamentul din spate al impartitorului in imaginea urmatoare:

*Schema circuitului:*

**

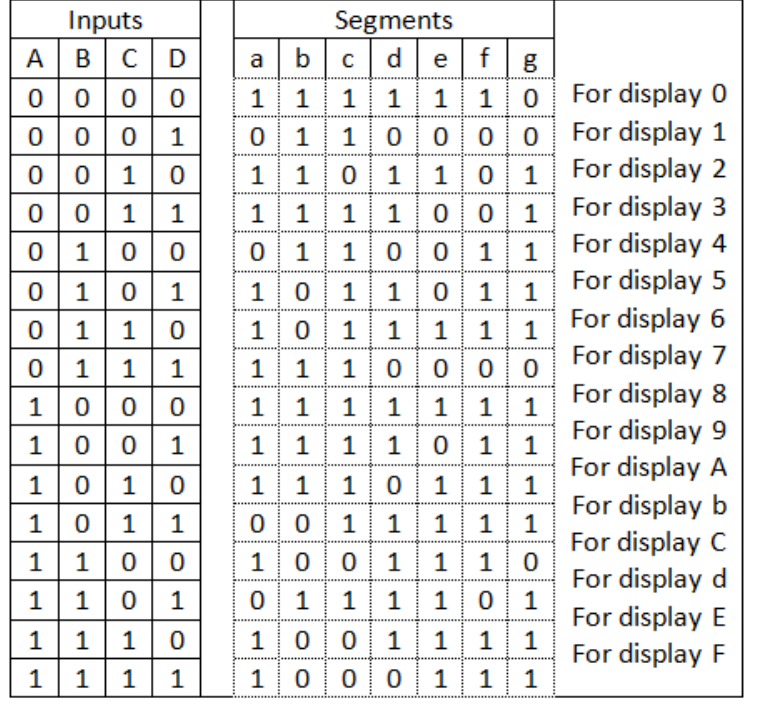
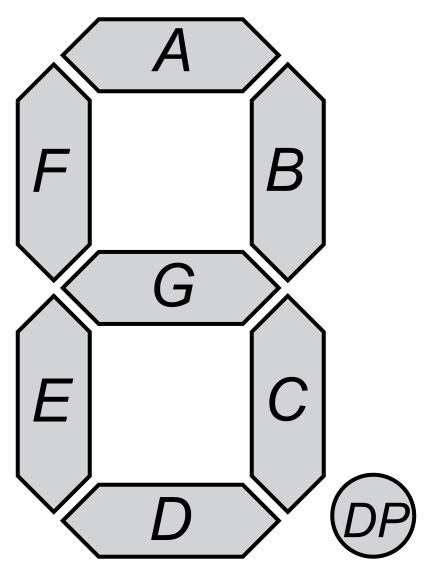
**

**

*Afisorul 7-segmente*

Pasul final in implementarea calculatorului de buzunar este afisarea input-urilor si output-ului pe afisorul 7 segmente al placutei Nexys. Primii 4 catozi sunt rezervati operanzilor, iar urmatorii 4 rezultatului, alaturi de semnul corespunzator, toate numerele fiind reprezentate in hexazecimal.

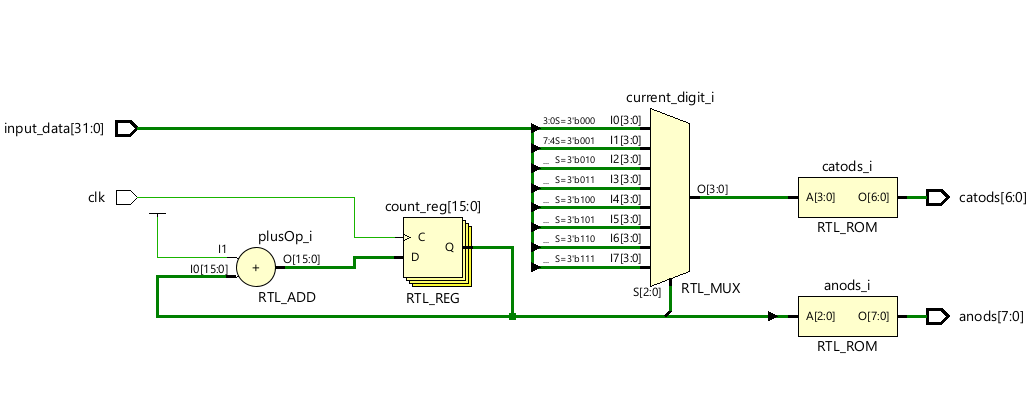
Am realizat afisorul 7 segmente pe baza urmatorului tabel de adevar:

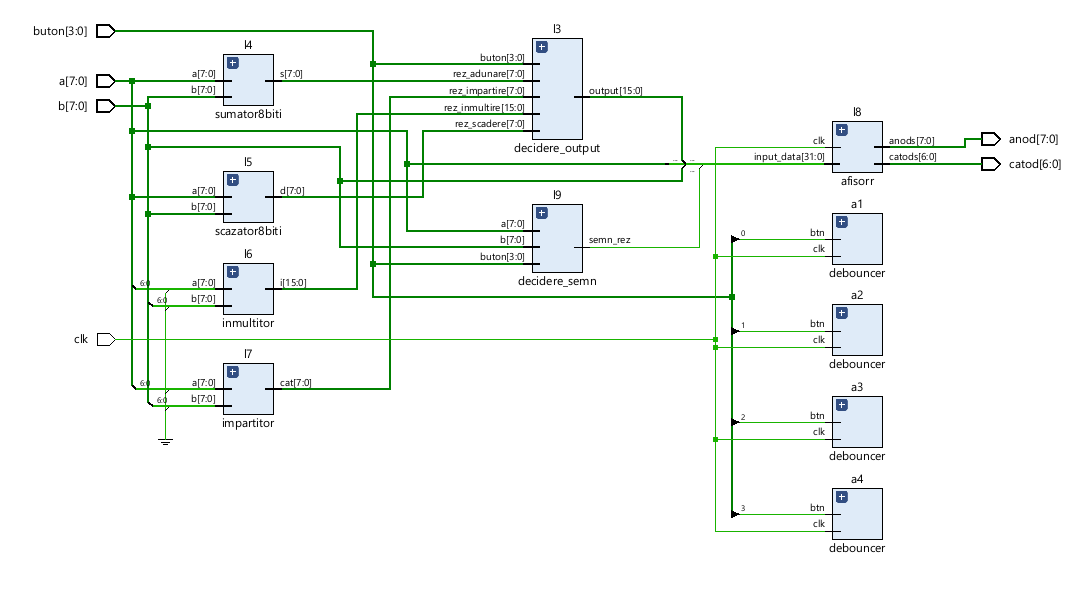
Am folosit cei mai semnificativi biti de la iesirea unui numarator binar drept selectie pentru un multiplexor 8:1, prin intermediul caruia activez pe rand cate un catod al anozilor. Datorita vitezei foarte mari, se creeaza utilizatorului impresia ca vede cifrele output-ului in aceasi timp.

Se poate observa comportamentul din spate al afisorului in imaginea urmatoare:

*Schema circuitului:*



*Schema finală a proiectului:*



3.Justificarea soluției alese

Scopul acestui proiect a fost crearea unui calcualtor simplu de buzunar, care poate efectua operatii matematice de baza, precum adunarea, scaderea, inmultirea si impartirea a 2 numere cu semn pe 8 biti. Am ales aceasta abordare pentru a imbina si a face conexiuni intre informatiile invatate pe parcursul laboratoarelor de Proiectarea Sistemelor Numerice: crearea unui sumator, scazator, comparator, afisor 7 segmente, debouncer, unitate aritmetico-logica si multe altele.

De asemenea, acest proiect poate avea mare aplicabilitate si in viata de zi cu zi cu siguranta.

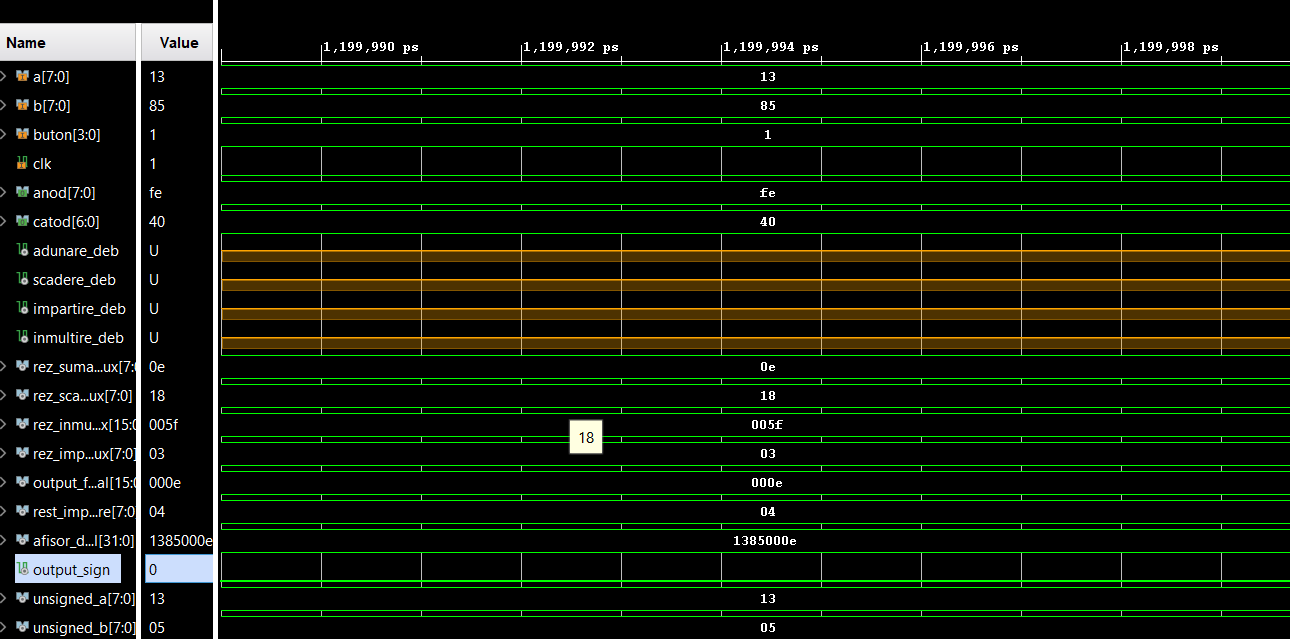
4. Manual de utilizare și întreținere

Pentru vizualizarea funcționării calculatorului de buzunar, tot ceea ce trebuie să facem este să deschidem proiectul în programul Vivado, și să rulăm simularea, moment în care vor apărea informațiile așteptate.

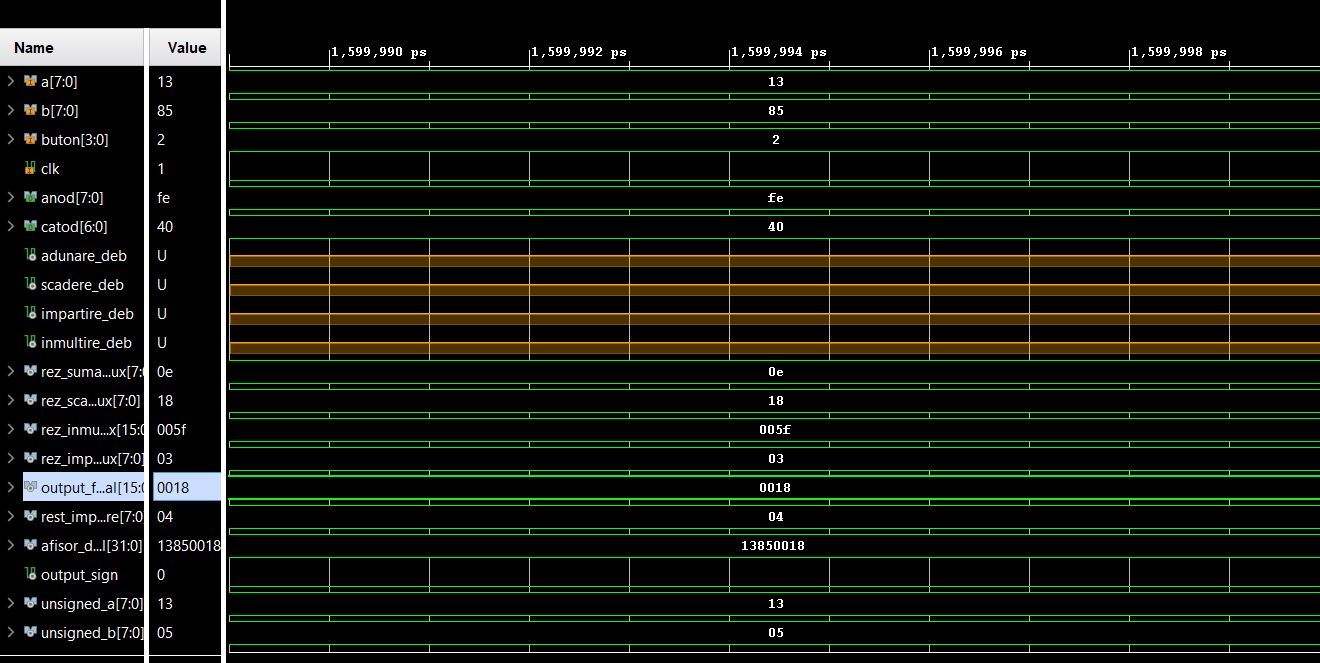
Dupa cum se poate observa, valoarea output-ului final se schimba in functie de butonul ales, odata cu semnul rezultat in urma operatiei efectuate.

In exemplul de mai jos, am introdus numerele 13 (19 in zecimal), respectiv -5 (tot -5 in zecimal) in baza hexazecimala, si am ales primul buton, asociat operatiei de adunare.

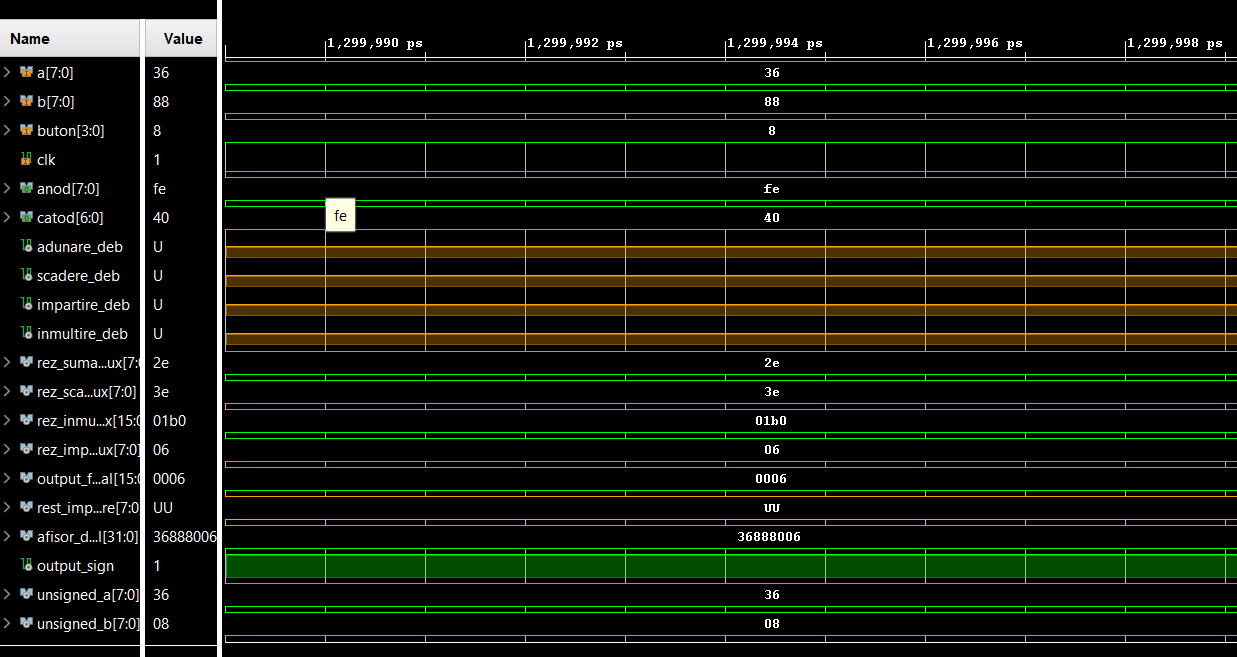
Dupa efectuarea operatiei, putem vedea faptul ca output-ul are valoarea ’e’, respectiv 14 in zecimal, iar bitul de semn al output-ului este ‘0’, indicand faptul ca rezultatul este pozitiv.



In acest caz, am selectat o alta operatie, cea de scadere, valoarea output-ului final schimbandu-se in „18” (24 in zecimal), semnul output-ului fiind de asemenea +.

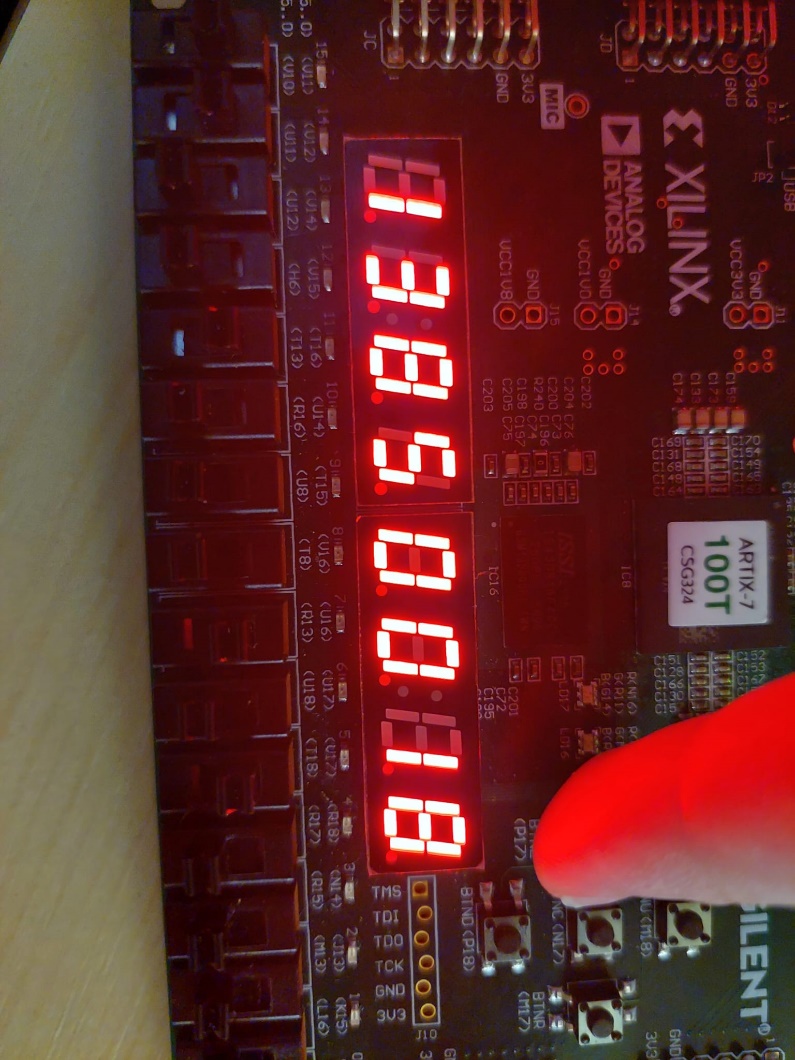


Alt exemplu care sa demonstreze functionalitatea calculatorului de buzunar: am introdus numerele 36 (54 in zecimal) si -8 (-8 in zecimal) si am efectuat operatia de impartire, output-ul primind valoarea 6 (la fel si in zecimal), iar bitul de semn ’1’.



De asemenea, putem observa functionalitatea proiectului prin implementarea pe o placuta FPGA.

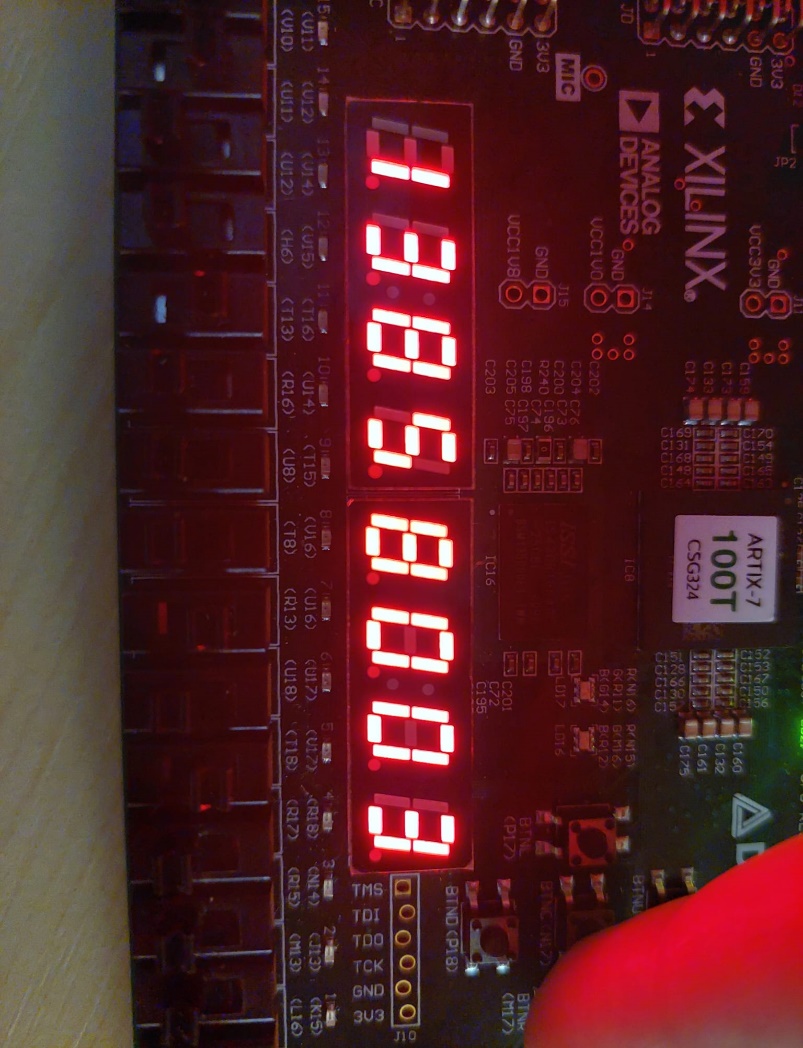
*Operatia de scadere:*



*Operatia de inmultire:*



*Operatia de impartire:*



5. Posibilități de dezvoltare ulterioară

* extinderea capacității operanzilor pe 16 sau 32 biți;
* posibilitatea de a realiza mai multe operații, pe lângă cele deja implementate;
* găsirea unei soluții pentru afișarea numerelor in format zecimal
* găsirea unei soluții pentru afișarea semnului numerelor foarte mari

6. Bibliografie

* <https://en.wikipedia.org>
* <https://www.xilinx.com>
* <https://maryxp.files.wordpress.com/2010/06/adunarea-80a0a0bai-sc80a0a0e3derea-numerelor-binare-cu-semn.pdf>
* <https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>