Calculator de Buzunar

Structura Sistemelor de Calcul

Student: Voicu Laura Luisa
Grupa: 30236

Cuprins

1. Rezumat

Proiectul realizat pentru Structura Sistemelor de Calcul consta in implementarea unui calculator de buzunar folosind limbajul VHDL si ale carui rezultate au fost testate cu ajutorul simulatorului din aplicatia Vivado Design Suite si placii de dezvoltare FPGA Basys3.

Utilizatorii pot fectua operatii matematice de baza prin intermediul butoanelor, rezultatele fiind afisate pe Afisorul cu Sapte Segmente. Implementarea include debouncing pentru o interactiune lina si un modul de gestionare a timpului pentru controlul frecventei de actualizare a ecranului.

2. Introducere

Obiectivul proiectului este acela de implementarea a unui calculator pe o platforma FPGA in limbajul VHDL. Calculatorul este destinat sa ofere functionalitati de baza pentru operatiile aritmetice de baza si sa serveasca drept instrument educational pentru invatarea si aplicarea cunostintelor practice in proiectarea sistemelor digitale.

Principalele obiective sunt:

- Implementarea Functionalitatilor de Baza:

Adunarea, scaderea, inmultirea si impartirea: Calculatorul trebuie sa ofere suport pentru operatiile aritmetice fundamentale.

- Instrument Educational:

Implementarea proiectului constituie o oportunitate de aprofundareare a intelegerii si implementarii unor algoritmi fundamentali de aritmetica combinationala si secventiala, esentiali in proiectarea sistemelor digitale. De asemenea, prin intermediul realizarii proiectului s-au imbunatatit abilitatile de programare in limbajul VHDL, obtinand o profunda integere a acestuia si a functionalitatii placii FPGA, precum si a testari si rezolvari erorilor prin intermediul celor doua mentionate.

- Extensibilitate & Imbunatatire

Presupune crearea unei structuri modulare si extensibile, premitand adaugarea ulterioara de noi functionalitati si imbunatatiri, precum si deschiderea unor posibilitati de inovatie in hardware si extinderea capacitatilor calculatorului de buzunar.

Prin atingerea acestor obiective, proiectul urmareste se ofere o solutie practica si utila, contribuind la dezvoltarea cunostintelor si abilitatilor in proiectarea sistemelor digitale .

3. Fundamentare teoretica

In cadrul proiectului s-a optat pentru utilizarea unor algoritmi si metode consacrate pentru realizarea inmultirii, impartirii, adunarii si scaredii, adaptate specificatilor proiectului.

3.1 Metode folosite pentru efectuarea operatiilor

3.1.1 Adunarea prin medota Ripple Carry Adder

Ripple Carry Adder este un tip de adunare in aritmetica binara. Acesta aduna bitii corespunzatori de la intrarile sale si genereaza un rezultat binar si un bit de transmitere catre nivelul urmator al adunarii. Principiul de baza consta in adunarea treptata a bitilor, incepand de la cel mai purin semnificativ, si propagarea transmiterii catre bitii mai seminificativi.

3.1.1.1 Structura de baza

Un Ripple Carry Adder este compus dintr-o serie de blocuri de adunare completa(full adders). Fiecare celula sumatoare primeste trei intrari: bitii corespunzatori ai operanzilor si un bit de transmitere de la celula sumatoare precedenta.

3.1.1.2 Full Adder

Operatia de adunare folosind un Full Adder este descrisa de relatiile:

- ***Sum = A xor B xor Cin***
- ***Cout = (A and B) or ((A xor B) and Cin)***

3.1.1.3 Performanta si limitari

Performanta acestui tip de sumator este direct afecata de numarul total de biti si de nivelul de propagare al transmiterii Carry-ului. O limitare a acestui tip de sumator consta in **timpul de propagare liniar** , ceea ce poate duce la timpi de calcul mai lungi in cazul adunatilor pe biti seminificativi.

3.1.2 Scaderea – RCA & Complement fata de 2

Algoritmul implementat pentru scaderea a doua numere binare in VHDL utilizeaza doua etape principale: calcularea complementului fata de doi al numarului ce va fi scazut si adunarea acestuia cu numarul initial.

3.1.2.1 Calcularea Complementului fata de Doi

Calcularea acestuia se realizeaza prin inversarea fiecarui bit al numarului si adunarea cu 1 la rezultat. In cadrul implementarii, acest lucru se realizeaza printr-o bucla :

```
process(X,Y)
begin
  for i in 15 downto 0 loop
    oneComplement(i) <= not Y(i);
  end loop;
end process;
```

3.1.2.2 Adunarea valori initiale cu C2

Dupa obtinerea C2, acesta se va aduna cu primul dintre numere.

Prin implementarea acestui algoritm, utilizatorul are posibilitatea de a efectua operatii de scadere a numerelor pe 16 biti (numere fara semn).

3.1.3 Inmultirea prin Tehnica Booth

Metoda Booth este o tehnica eficienta pentru inmultirea numerelor binare cu semn. Aceasta tehnica se baseaza pe observatia ca operatia de inmultire poate fi simplificata prin reducerea numarului de adunari necesare pentru obtinerea produsului. Principiul de baza consta in a identifica secventele de biti consecutivi cu aceleasi valori si a le inlocui cu valori intermediare, reducand astfel numarul de adunari.

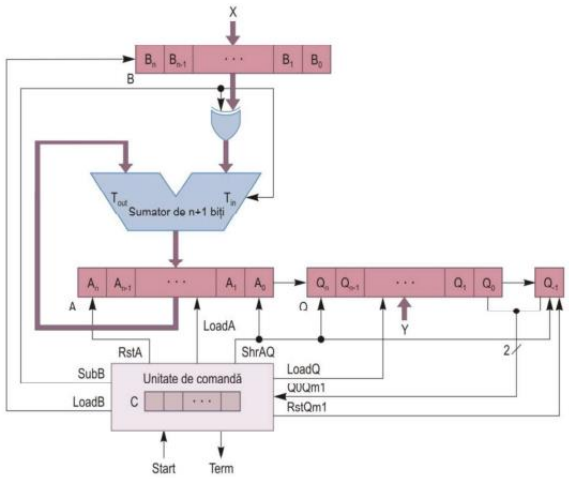


Figura 5.3. Schema bloc a unui circuit de înmulțire prin metoda Booth.

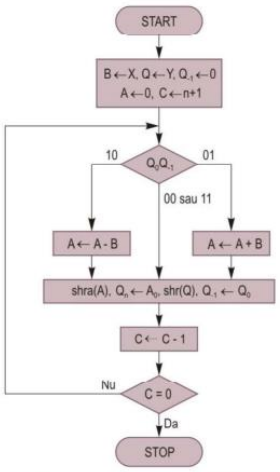


Figura 5.4. Organigrama operației de înmulțire prin metoda Booth.

Imagini preluate din indrumatorul de laborator

3.1.3.1 Procesarea operatiilor – Adunarea produselor partiale

Multiplicatorul e analizat in blocuri de 2 biti consecutivi (Q0, Q-1) . Daca o secventa de 2 biti contine “01”, atunci in rezultatul ce e procesat iterativ se va scadea valoarea celul de-al doilea termen al adunarii. In schimb, daca secventa de biti este “10”, atunci la rezultatul din acumulator se va adauga cea de-a doua valoare. Pentru cazurile in care cei doi biti sunt identici nu au loc operatii asupra acumulatorului, ci se va trece direct la shiftarea la dreapta a acestuia.

3.1.3.2 Finalizarea operatiei

Procesul de adunare/scadere si deplasare este repetat pana cand toti bitii din primul termen al inmultirii sunt procesati. Produsul final este reprezentat prin concatenarea continutului registrelor de produs partial si acumulator (A&Q)

Metoda Booth contribuie la eficientizarea operatiei de inmultire din punct de vedere al resurselor si a timpului de calcul. Implementarea sa in cadrul proiectului aduce beneficii semnificative in ceea ce priveste performanta operatiei pe placa Basys3.

3.1.4 Impartirea prin metoda refacerii restului partial pentru numerele fara semn

Metoda refacerii restului partial a numerelor fara semn reprezinta un algoritm eficient pentru impartirea in sistem binar. Fiecare etapa a operatiei de impartire incepe cu o deplasare a restului partial la stanca cu o pozitie. Se efectueaza apoi scaderea impartitorului din restul partial, obtinandu-se noul rest partial. Daca se obtine un numar pozitiv, cifra corespunzatoare catului este 1, altfel aceasta e 0.

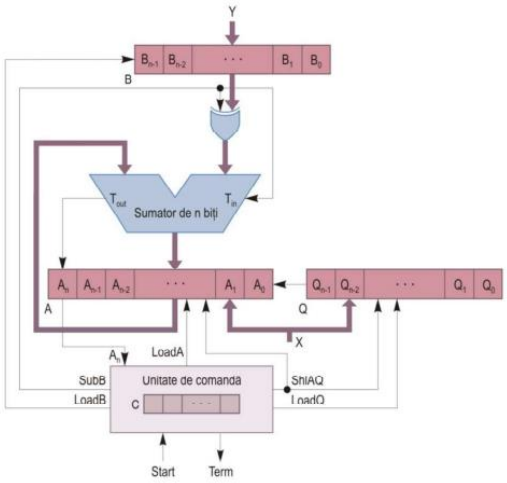


Figura 5.5. Schema bloc a unui circuit de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

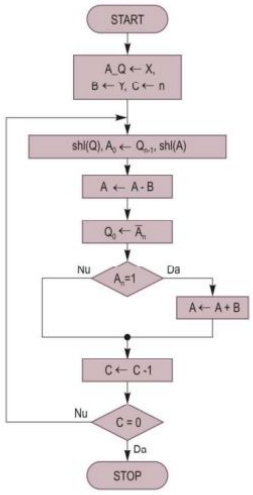


Figura 5.6. Organigrama operației de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

3.1.4.1 Initializare

Algoritmul incepe prin initializarea numarului ce va fi impartit si a divizorului.

3.1.4.2 Iterare

Algoritmul functioneaza prin iteratii consecutive. In fiecare itaratie se va realiza o schiftare la stanga a numarului ce va fi impartit si acumulatorului, urmate de scaderea acumulatorului cu impartitorul. Daca cel mai din stanga bit al acumulatorului a fost 1 inainte de shiftare, atunci acumuatorul se va aduna cu impartitorul. Acest proces se va repeda te n ori, unde n reprezinta numarul de biti al impartitorului si deimpartitului.

3.1.4.3 Refacerea restului partial

In fiecare iteratie, restul este refacut pentru a imbunatati precizia rezultatului. Procesul de refacere a restului se repeta in fiecare iteratie pentru a obtine o aproximare cat mai precisa a rezultatului real.

3.1.4.4 Generarea rezultatului

La finalul iteratiilor, rezultatul se va obtine din Q (acesta fiind catul operatiei) si din A (reprezentand restul).

Prin implementarea metodei refacerii restului partial se va obtine o solutie eficienta si rombusta pentru impartirea numerelor fara semn in cadrul Calculatorului de Buzunar.

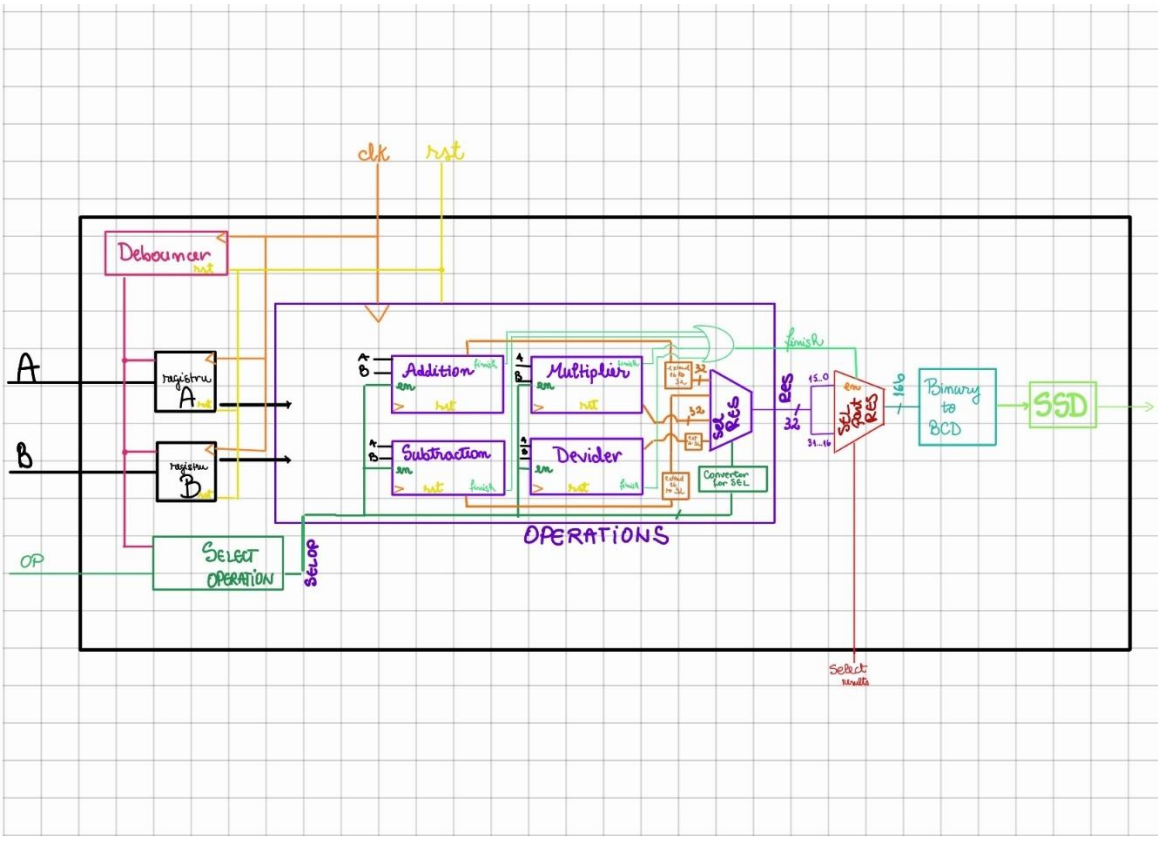
4. Proiectare si implementare

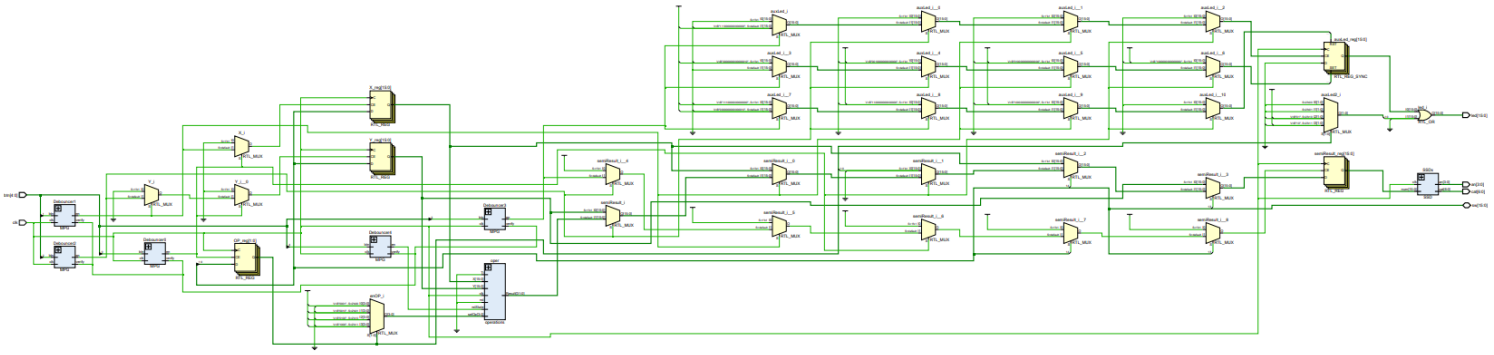
4.1 Arhitectura generală a sistemului

Arhitectura generala a sistemului este formata din 7 module principale:

- pocket_calculator – reprezinta modulul principal al programului
- operations – modului ce stabileste ce operatie trebuie sa fie executata
- booth_multiplier – modul in care are loc efectuarea operatiei de inmultire
- divider – modul in care are loc efectuarea operatei de impartire
- addition – modul in care are loc efectuarea operatiei de adunare
- subtraction – modul in care are loc efectuarea operatiei de scadere
- debouncer

4.1.1 Schema arhitecturii

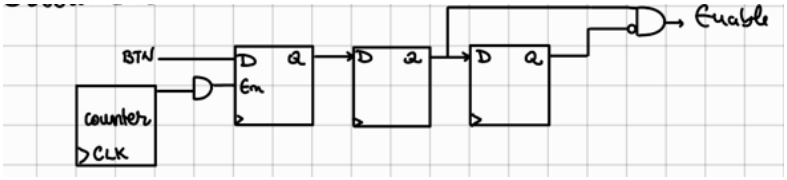




4.1.2 Proiectare - Componente

4.1.2.1 Debouncer

Debouncerul este folosit pentru a elimina instabilitatea adusa de apasarea butoanelor de pe placuta, folosite la incarcarea operantiilor si a operatorilor.

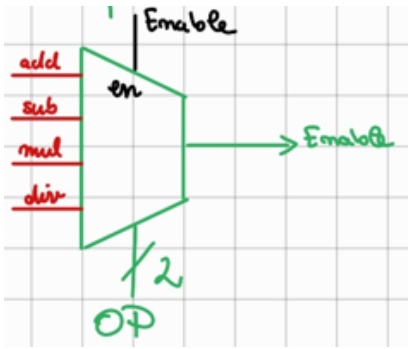


4.1.2.2 Select Operation

Multiplexor ce are ca scop alegerea operatiei ce urmeaza sa fie executata, astfel ca intr-un timp se va executa o singura operatie, nu toate in paralel, eficientizand din punct de vedere al vitezei functionalitatea calculatorului.

Multiplexorul are ca semnale de intrare datele de intrare, selectia data de OP si un buton de Enable. Datele de intrare au urmatoarea semnificatie :

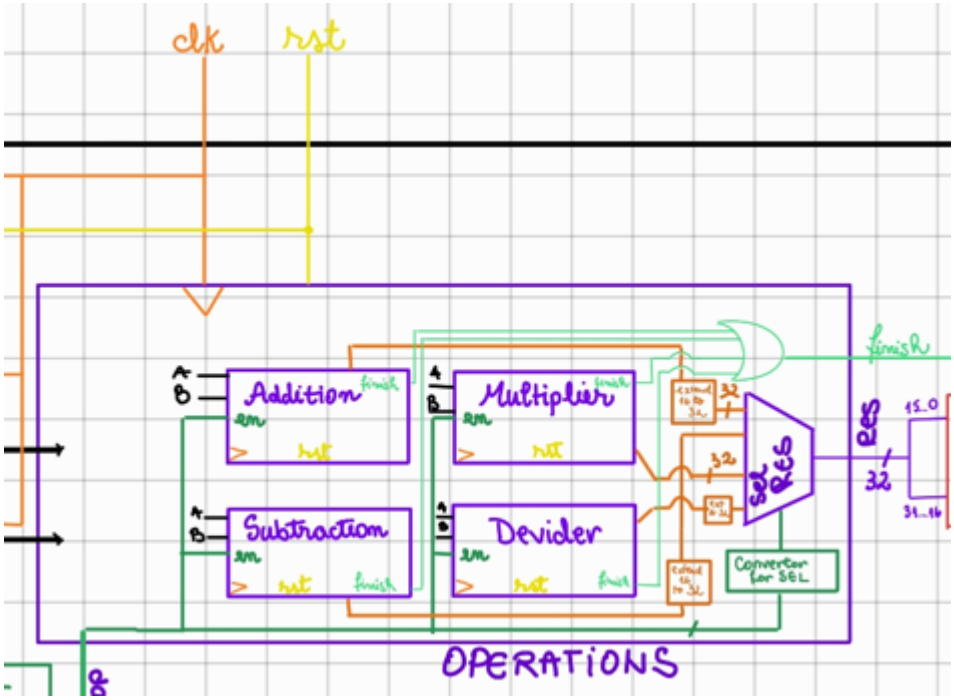
- 0001 = addition
- 0010 = subtraction
- 0100 = multiplication
- 1000 = division



4.1.2.3 Reg A, Reg B, Reg OP

Sunt necesari registi pentru retinerea operatiei alese si a operantilor dupa introducerea acestora in binar si apasarea butoanelor corespunzatoare pentru salvarea lor.

4.1.2.4 Operations

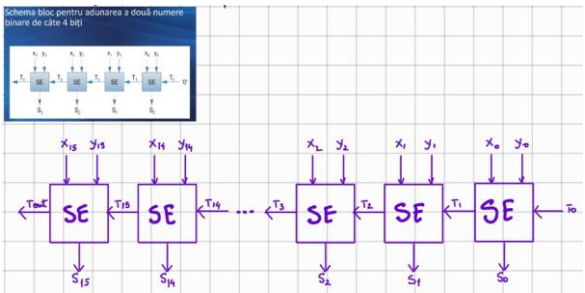


- Intrari: clk, rst, regA, regB, selOPReg
- Iesiri: Res (32b) , finish (afisare rezultat pe SSD doar daca operatia s-a incheiat)

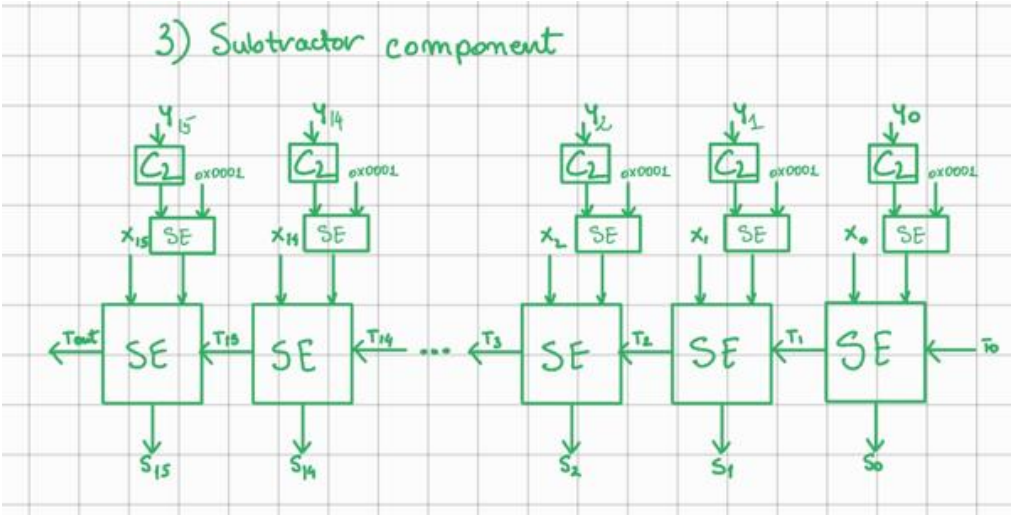
4.1.2.4.1 Extend 16b to 32b

Componenta e folosita pentru a alinia bitii la 32b pentru ca datorita inmultirii, resultatul poate sa depaseasca 16b => pentru usurarea afisarii, si pentru restul operatiilor se vor afisa de fapt numere pe 32b (cate 2 perechi de 4 elemente).

4.1.2.4.2 Addition : Ripple Carry Adder



4.1.2.4.3 Subtraction : RCA with Two complement



4.1.2.4.4 Booth Multiplier

Pentru realizarea operatiei de inmultire cu tehnica Booth se vor folosi

- Unitate de Comanda : stabileste valorile semnalelor RstA, SubB, LoadB etc.. corespunzatoare
- Sumator de 16 biti: se opteaza tot pentru un RCA
- Registre de Shiftare la stanga pentru acumulator in inmultitor
- Registru pentru salvarea celui de-al toilea termen al inmultirii
- Registru pentru salvarea ultimului bit al lui Q

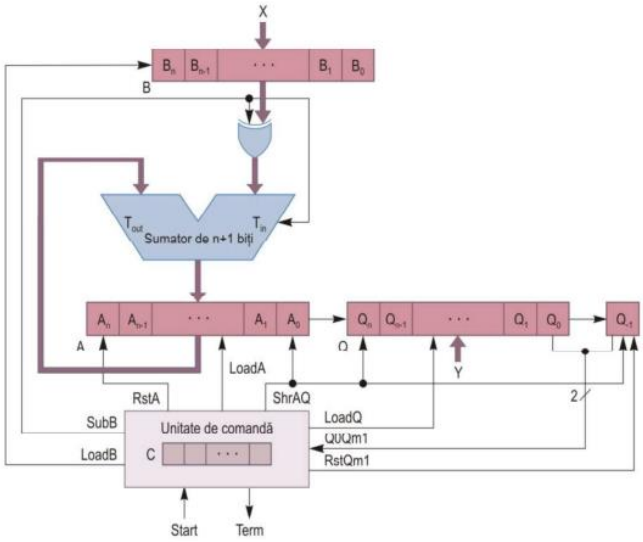
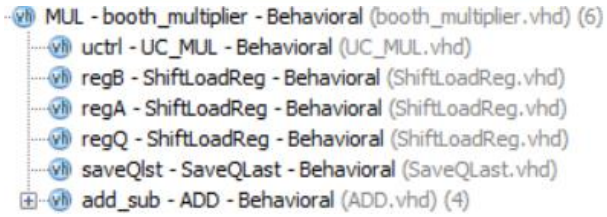


Figura 5.3. Schema bloc a unui circuit de înmulțire prin metoda Booth.



4.1.2.4.5 Divider

Similar inmultirii Booth, impartirea foloseste Unitate de Comanda, Registre de shiftare la dreapta, registre de salvare a celui de-al doilea operand.

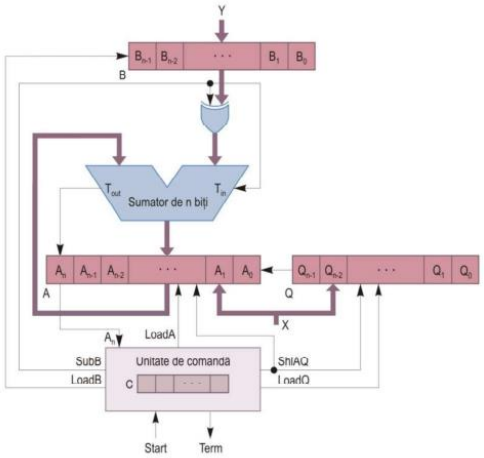
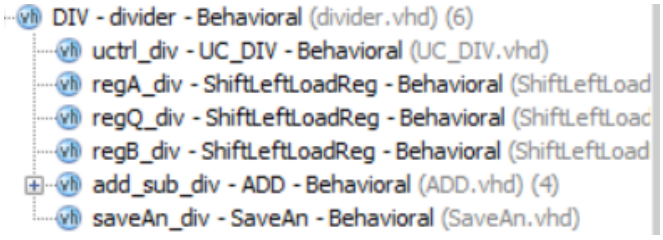


Figura 5.5. Schema bloc a unui circuit de împărțire prin metoda refacerii restului parțial pentru numere fără semn.



4.1.2.4.6 Finish Gate

Reprezinta o poarta logica cu 5 intrari pe 1 bit ce e folosita pentru a activa semanul de finish atunci cand operatia va termina executia

Acest semnal este necesar deoarece timpul de executie pentru inmultitor si impartitor e semnificativ mai mare, deci drept urmare, daca dorim sa afisam un rezultat corect pe afisor e necesar sa „asteptam" terminarea efectuarii operatiei.

4.1.2.5 Convertor For Sel

Componenta transforma semnalul primit de la switch-uri in semnale valide pentru activarea operatiei din modului **operations.** Conversia se realizeaza astfel pentru semnalul SEL ca si semnal de intrare:

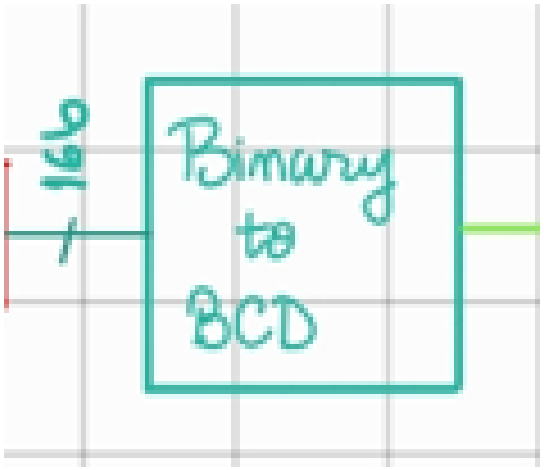
- 00 = 0001 => adunare
- 01 = 0010 => scadere
- 10 = 0010 => inmultire
- 11 = 0011 => impartire

4.1.2.6 Selection Port Result

Aceasta componenta are rolul de a alege care din cifre vor fi afisare pe SSD: ultimele 4 sau urmatoarele 4. Selectia se realizeaza prin switch-ul ... .

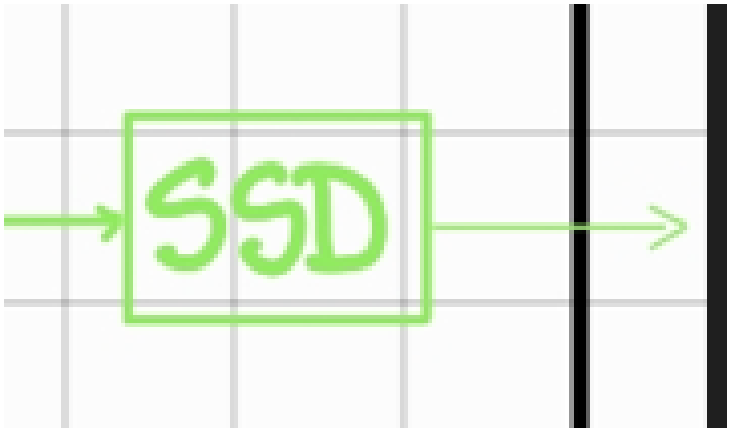
4.1.2.7 Binary to BCD

Componenta are rolul de a converti numerele binare in numere zecimale si transmiterera rezultatului la Seven Segment Display.



4.1.2.8 Seven Segment Display

Componenta ce transfera datele in zecimal la display.



4.2 Detalii de implementare

Implementarea proiectului este constituita din 7 module principale.

- Din pocket-calculator se vor prelua datele de intrare (operanzii) X si Y de la switch-urile placutei pe 16b, precum si operatia selectata
- In operations se vor trimite mai departe datele mentionate mai sus unde, in functie de operatia aleasa, X si Y se vor trimite mai departe carte modulele ce efectueaza operatia.
- Dupa efectuarea operatiei, rezultatul este trimis inapoi in pocket-calculator unde va fi afisat pe SSD.

4.3 Manual de instructiuni

- Se va alege operandul X de la switch-urle 0-15, dupa care se va apasa butonul T18 (sus) pentru a-l incarca in registru
    - Vizualizarea continutului registrului se poate realiza cu switch-ul SW14

- Se va alege operandul Y de la switch-urle 0-15, dupa care se va apasa butonul W19 (stanga) pentru a-l incarca in registru
    - Vizualizarea continutului registrului se poate realiza cu switch-ul SW13

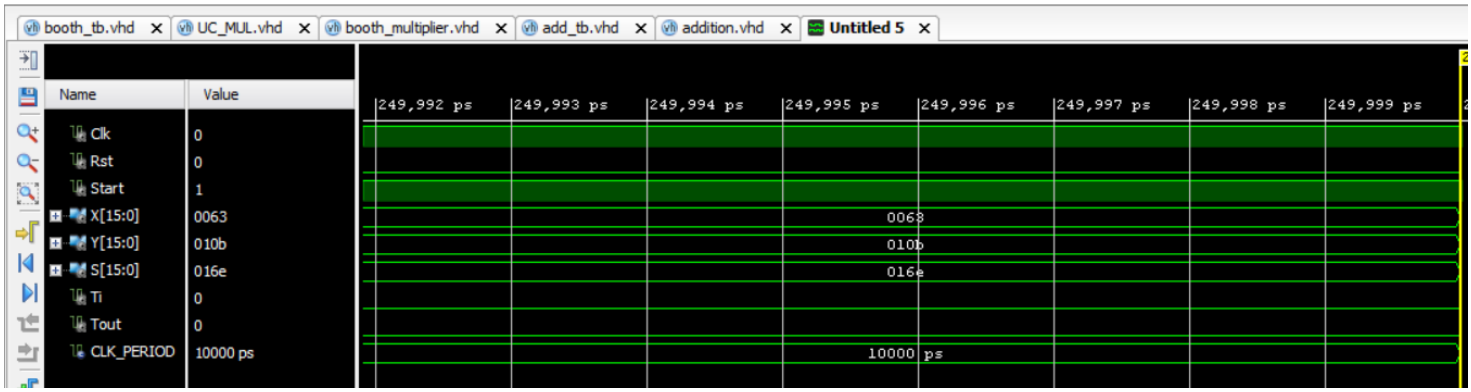
- Se va alege operatorul din sw1-sw01 (00 – adunare, 01 – scadere, 10 – inmultire, 11 – impartire) si se va apasa butonul V18 (centru)
    - Daca se alege inmultire, se va aprinde ledul 1
    - Daca se alege impartire, se va aprinde ledul 0

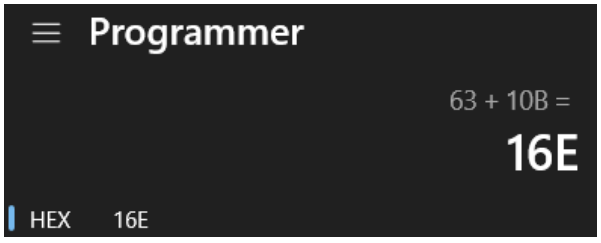
- Pentru vizualizarea rezultatului se va apasa butonul T17 (dreapta)

5. Rezultate experimentale

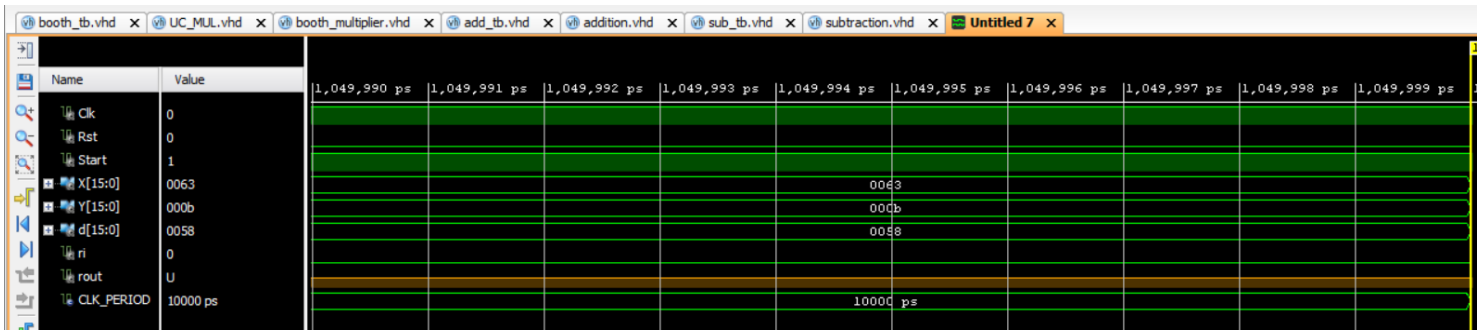
5.1Rezultate obtinute in mediul Vivado – Simulare

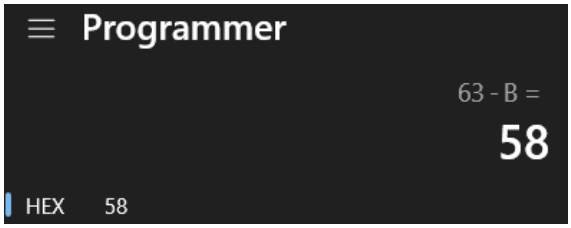
5.1.1 Simulare – Adunare



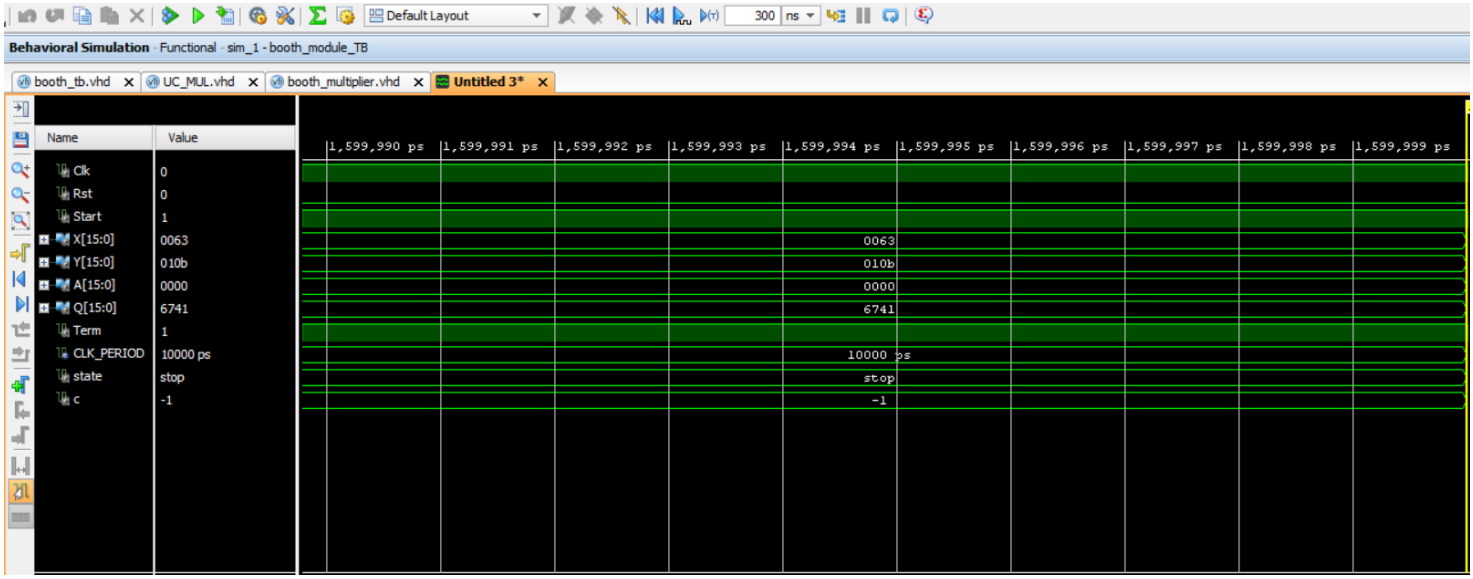


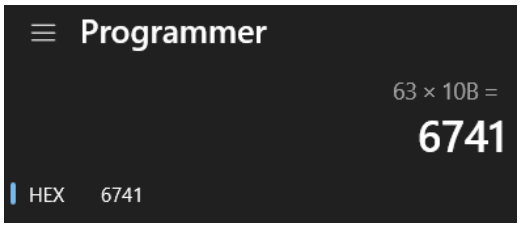
5.1.2 Simulare – Scadere

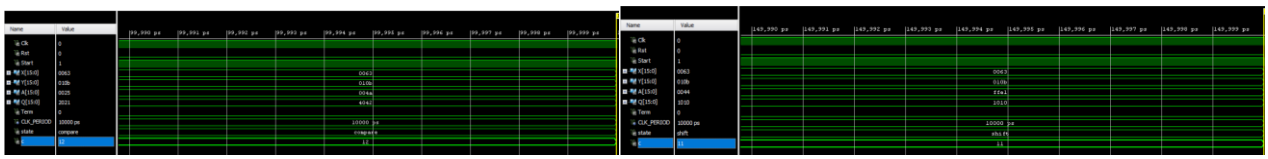




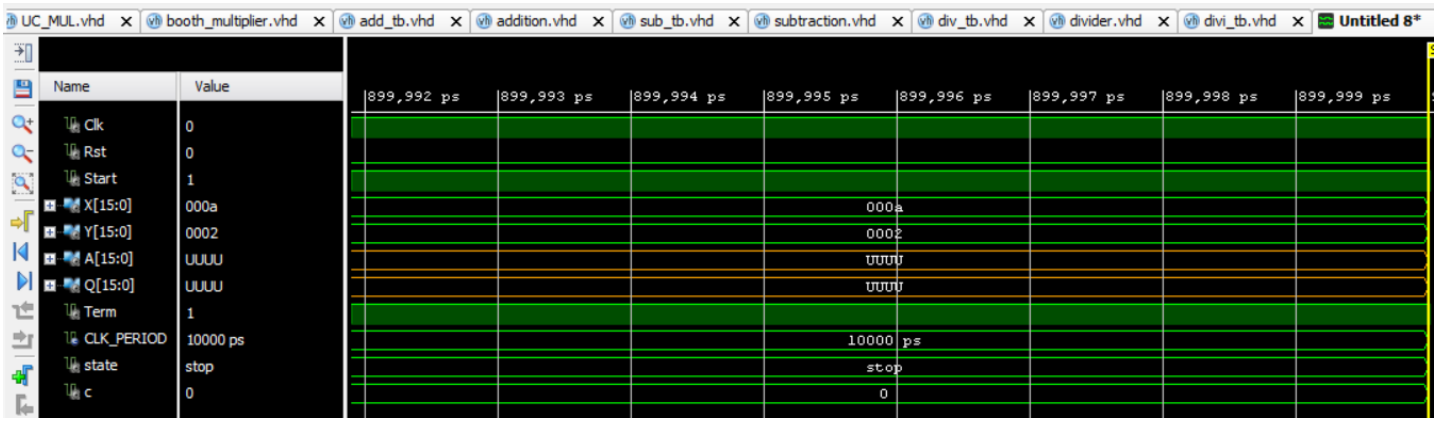
5.1.3 Simulare – Inmultire





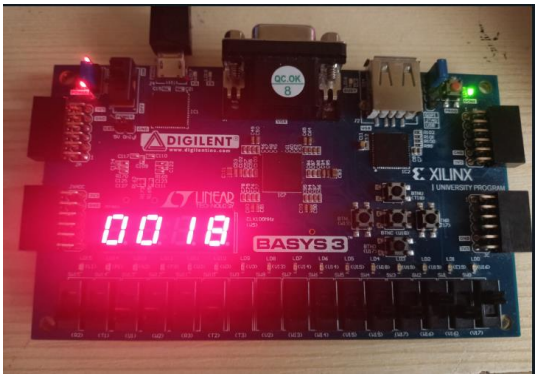


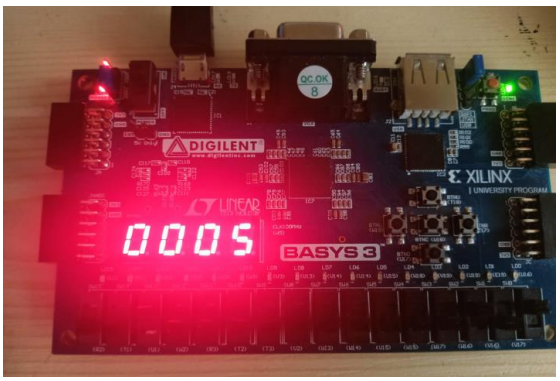
5.1.3 Simulare – Impartire

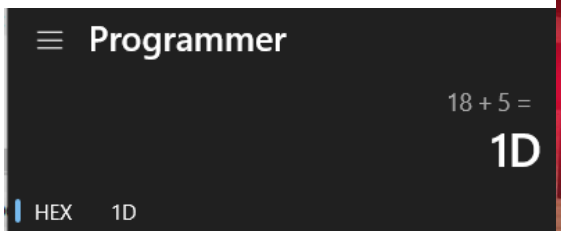


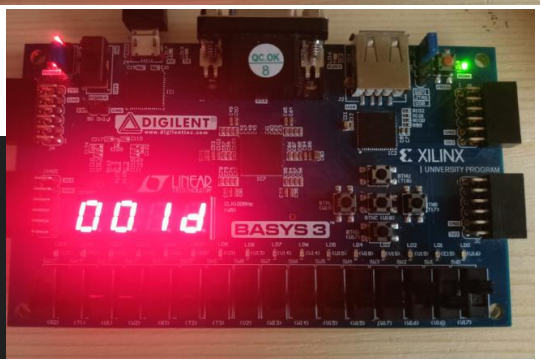
5.2 Rezultate obtinute in mediul Vivado – Simulare

5.2.1 Adunare

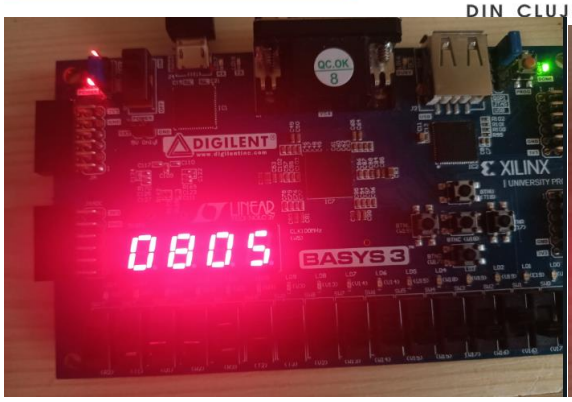


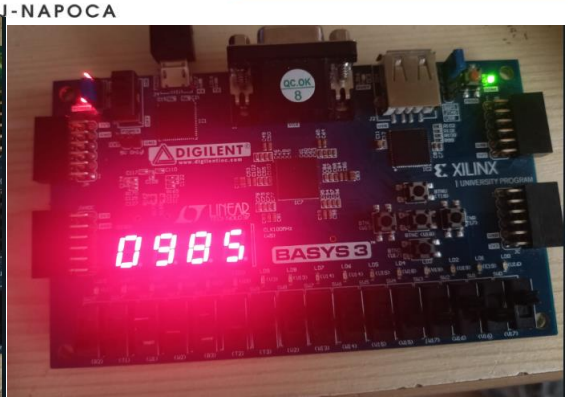


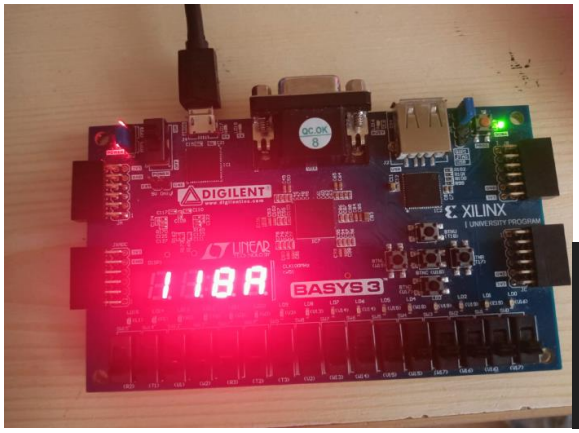


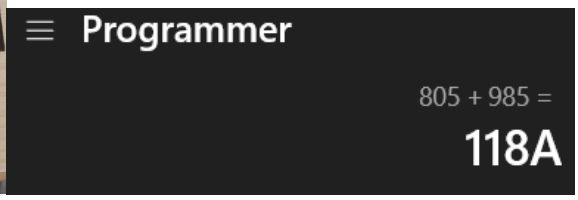


=

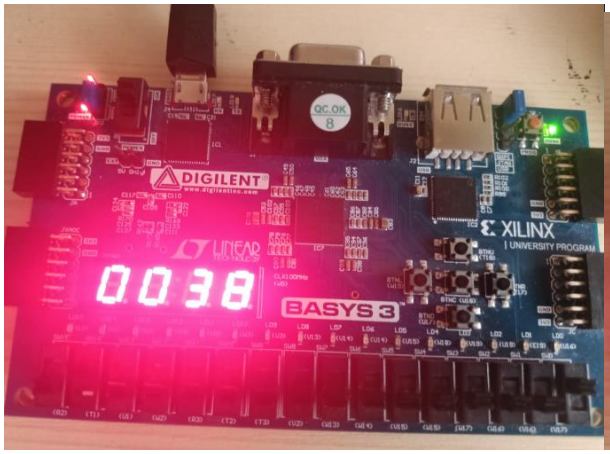

0805
BASYS 3
DIGILENT
XILINX

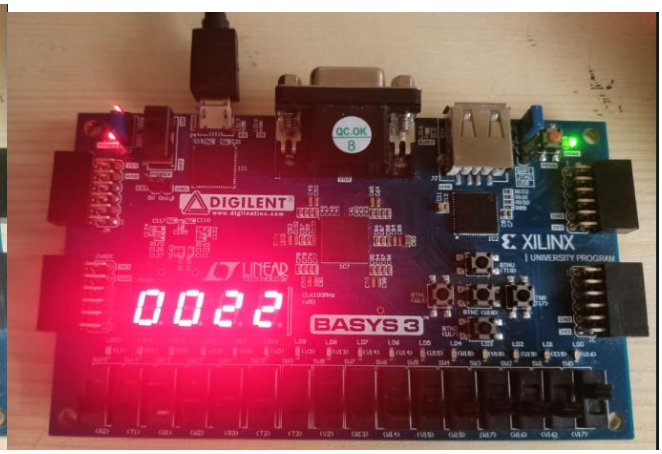

0985
BASYS 3
DIGILENT
XILINX

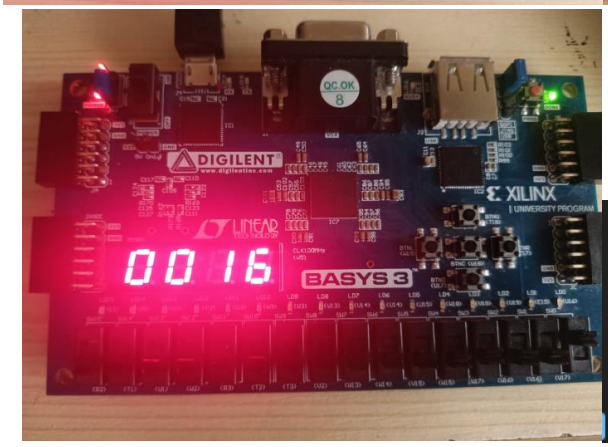

118A
BASYS 3
DIGILENT
XILINX

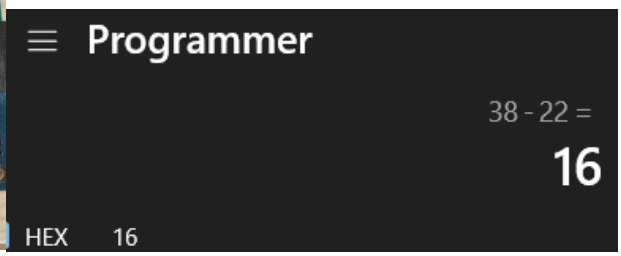

Programmer
805 + 985 =
118A

5.2.3 Scadere

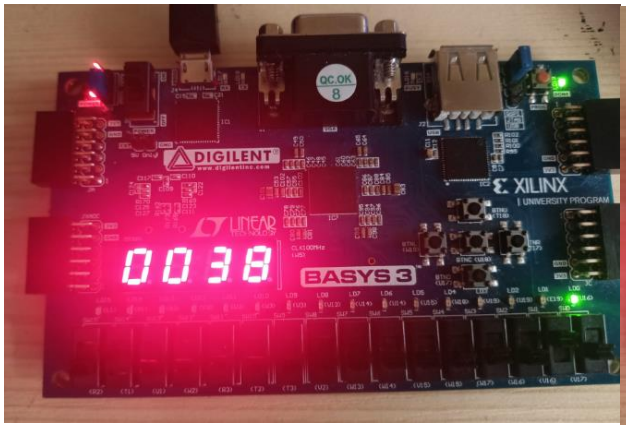

DIGILENT
XILINX
0038
BASYS 3

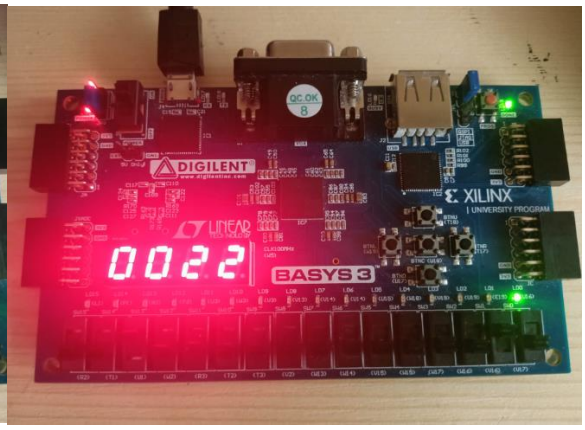

DIGILENT
XILINX
0022
BASYS 3

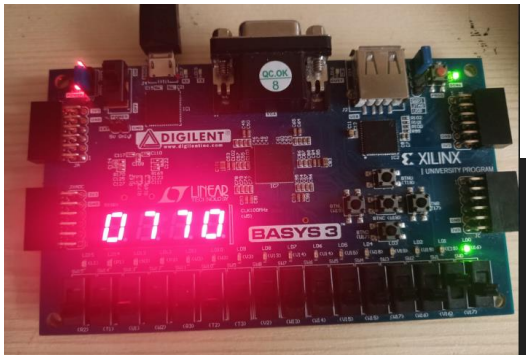

DIGILENT
XILINX
0016
BASYS 3

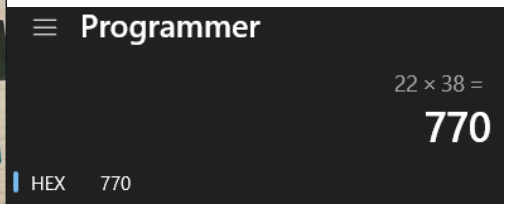

Programmer
38 - 22 =
16
HEX 16

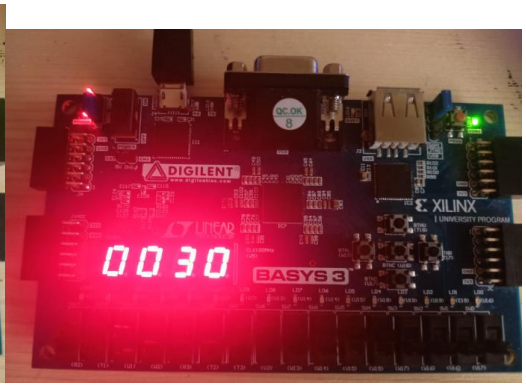
5.2.1 Inmultire

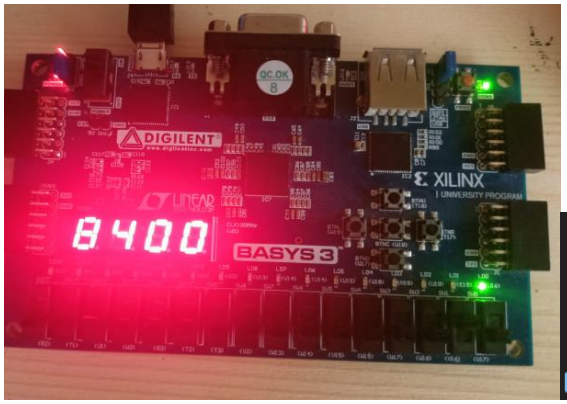


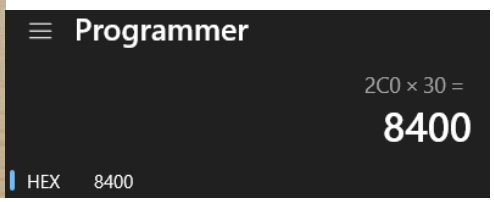












6. Concluzii

In incheierea acestui proiect, este evident ca implementarea unui calculator de buzunar in limbajul VHDL pe platforma Basys 3 aduce multiple beneficii. Prin realizarea acestui proiect, am avut oportunitatea de a explora si intelege in profunzime algoritmi esentiali de aritmetica combinationala si secventiala. Procesul de proiectare si implementare a modulelor pentru operatii precum inmultire, impartire, adunare si scadere a contribuit semnificativ la dezvoltarea competentelor in programarea FPGA si in inyelegerea functionalitatilor placii Basys 3.

1. Bibliografie

1. Booth multiplier: http://vlabs.iitkgp.ernet.in/coa/exp7/index.html
2. Booth multiplier: https://www.javatpoint.com/booths-multiplication-algorithm-in-coa
3. SSD: https://ro.farnell.com/c/optoelectronics-displays/displays/led-displays/7-segment-led-displays
4. Debouncer: https://forum.digikey.com/t/debounce-logic-circuit-vhdl/12573
5. Indrumatorul de laborator si cursurile de la materia SSC