TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

FACULTATEA: Automatică și Calculatoare

SPECIALIZAREA: Calculatoare și Tehnologia Informației

DISCIPLINA: Proiectarea sistemelor numerice

PROIECT: Calculator de buzunar

Student:

Loga Dara

Cuprins

Specificatii si informatii generale.....3

Descrierea proiectului 3

Componentele principale 5

Pasii pentru implementare 6

De ce am ales acest proiect 13

Rezultate experimentale 13

Imbunatatiri ulterioare 16

Concluzie 16

Bibliografie..... 16

1. Specificatii si informatii generale:

Scopul acestui proiect este sa calculeze operatii simple precum adunarea, scaderea, inmultirea si impartirea. Operanzii sunt in intervalul [-128, 127], unde 7 biti reprezinta valoarea absoluta, iar un bit suplimentar este folosit pentru semn. Proiectul este implementat in limbajul de descriere hardware VHDL.

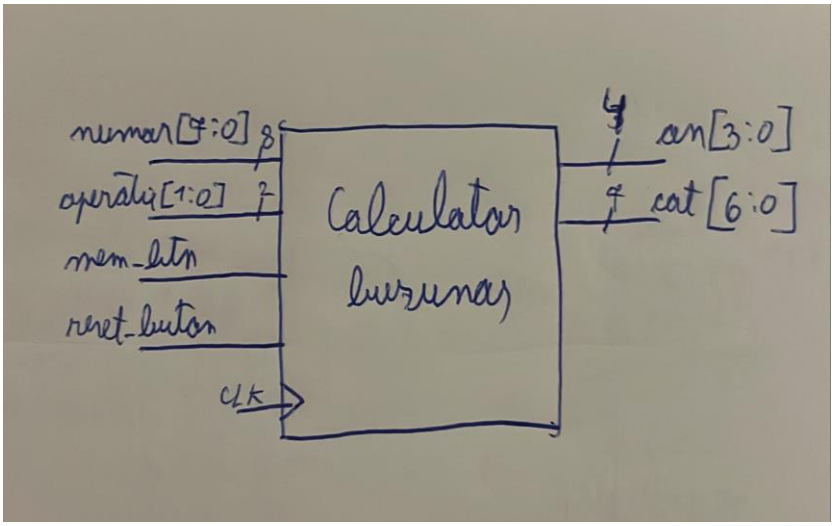
Cerinta:

Sa 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.

2. Descrierea proiectului

Pentru a putea implementa proiectul, mai intai a fost nevoie sa desenez cateva scheme ale celor mai importante componente. Asadar, avem:

A) Schema bloc:



Mai jos sunt prezentate intrarile si iesirile:

Intrarile sunt:

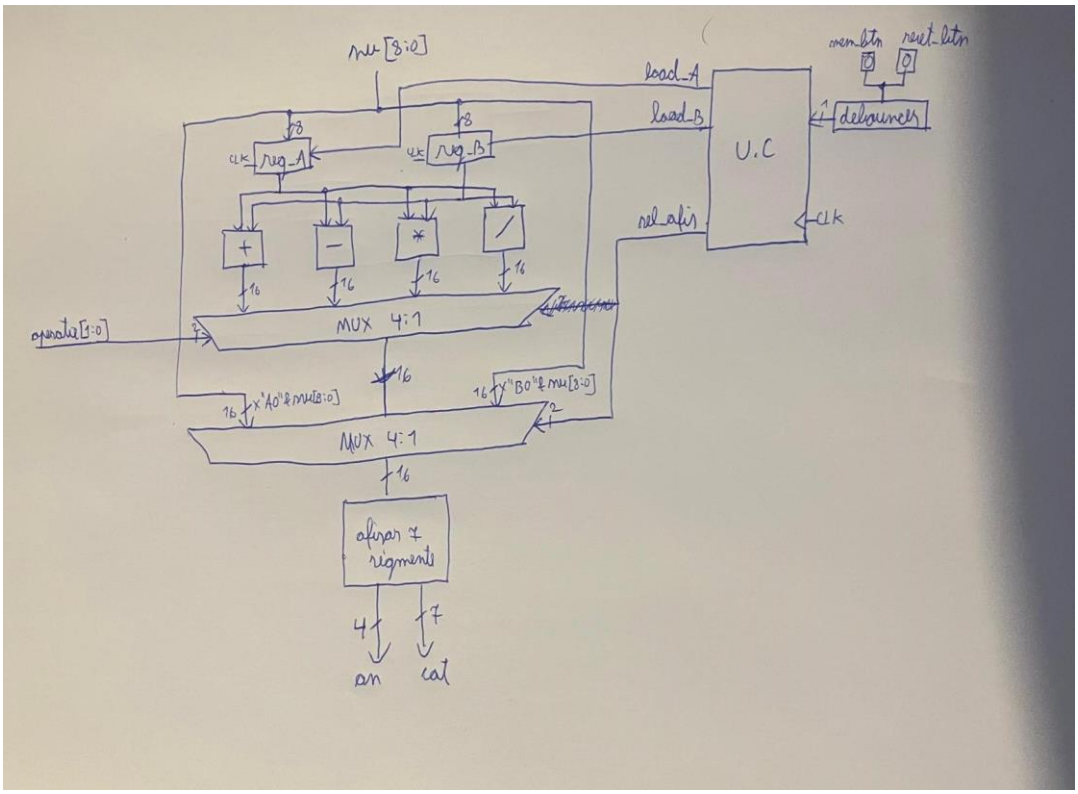
- Clk: Semnalul fizic de tact care ajuta la crearea componentelor secventiale
- Numar: Reprezinta un numar pe 8 biti cu semn
- Operatie: Reprezinta selectia rezultatului pe afisor a operatiei matematice
- Mem_btn: Semnal pe 1 bit care fizic este reprezentat de un buton care la apasare va memora, in ordine, doua numere, dupa care la a 3-a apasare va afisa rezultatul operatiei
- Reset_btn: Semnal pe 1 bit care reseteaza comportamentul automatului

Iesirile sunt:

- An
- Cat

Iesirile sunt reprezentate de 4 anozi și 7 catozi care vor clipi unul cate unul la frecvența inalta, astfel incat ochiul uman să poata observa diferenta.

B) Unitatea de control si unitatea de executie:



3. Componente principale

a) Calculator buzunar(main):

Entitatea in care sunt declarate si legate intre ele toate componentele din unitatea de executie cu unitatea de control care care fac posibila aplicarea operatiilor matematice si afisarea lor

b) Unitate de control:

Reprezinta procesul in care sunt introduse si calculate datele de intrare si afisate pe afisor in functie de apasarea butonului. Aceasta entitate este reprezentata de organigrama descrisa in urmatorul topic.

c) Sumator:

Aceasta este componenta principala si este realizata complet structural. Ea aduna doua numere pe 8 biti formate din cifrele introduse. Rezultatul va fi pe 16 biti pentru a putea fi afisate pe toate cele 4 afisoare.

d) Scazator:

Aceasta este o alta componenta principala realizata tot structural. Ea realizeaza scaderea intre doua numere pe 8 biti formate din cifrele introduse. Rezultatul va fi pe 16 biti. biti+semn) . Deci de exempli pentru numerele(hexazecimale) FF – 02 = FFFD.

e) Multiplicator:

Componenta care inmulteste doua numere pe 8 biti prin metoda de “shift and add”. Rezultatul va fi reprezentat tot pe 16 biti.

f) Impartitor

Aceasta componenta imparte doua numere pe 8 biti prin shiftari si scaderi succesive. Primii 8 biti(cei mai semnificativi) vor reprezenta catul si urmatori 8 biti vor reprezenta restul.

g) Registru memorare

Aceasta componenta memoreaza numerele introduse de catre utilizator si memorarea acestora(load = ‘1’) este data de unitatea de control.

h) Afisor pe 7 segmente

Componenta care ajuta vizualizarea datelor introduse si rezultatul operatiilor matematice

k) Multiplexoare

Componente care conditioneaza afisarea rezultatului final dupa switch-uri si selectia operatiei matematice

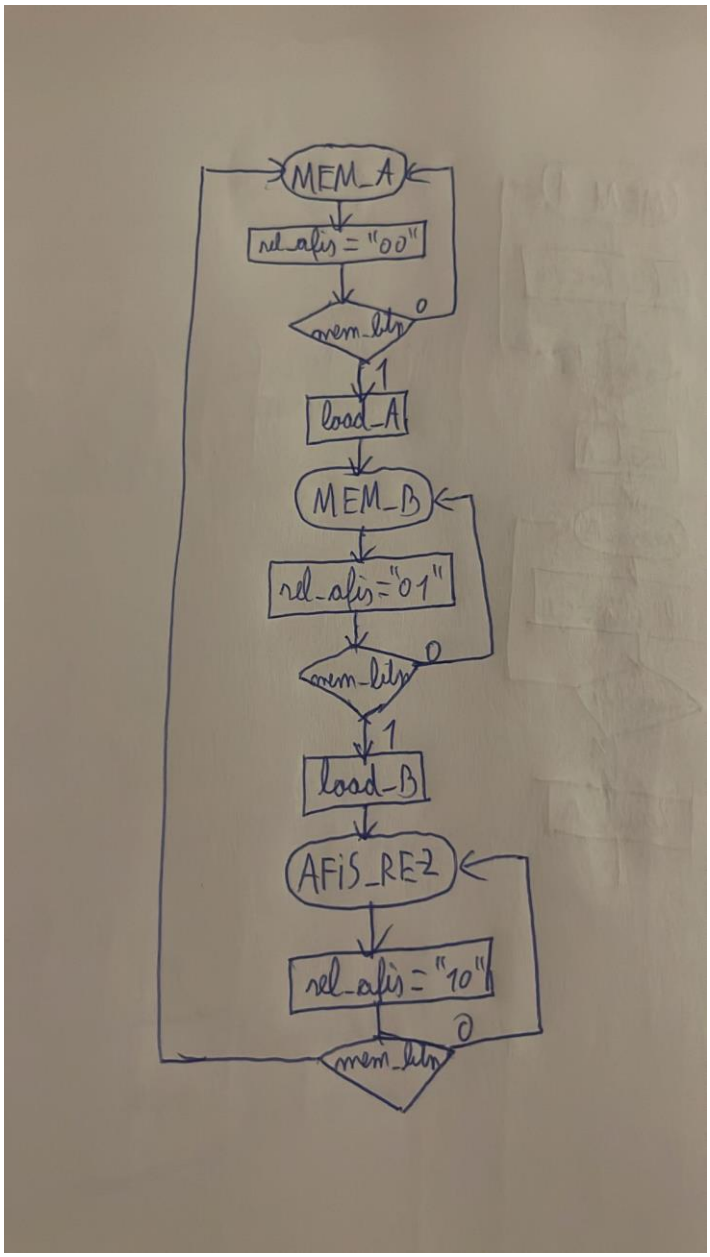
4. Pasii pentru implementare

Automatul are urmatorul comportament:

Introducerea numarului A(switch-uri) -> Apasare buton de memorare a numarului A -> introducerea numarului B(switch-uri) -> Apasare buton pentru vizualizarea rezultatului final -> Selectia afisari +,-,*,/ (switch-uri)

Asadar organigrama este implementata in unitatea de control.

a) Unitate de control



Rezulta 3 stari: MEM_A, MEM_B, AFIS_REZ

MEM_A: Atunci cand automatul este in starea MEM_A, automatul are ca output sel_afis = “00” pentru afisorul pe 7 segmente care va afisa numarul A care este reprezentat pe 8 biti(numarul este afisat pe primele 2 afisoare). Automatul va astepta apasarea butonului mem_btn pentru a incarca numarul in registru pentru numarul A, va afisa urmatoare stare pe afisor(adica numarul B) si va trece in starea MEM_B

MEM_B : Atunci cand automatul este in starea MEM_B, automatul are ca output sel_afis = “01” pentru afisorul pe 7 segmente care va afisa numarul B. Automatul va astepta apasarea butonului mem_btn pentru a incarca numarul in registru pentru numarul B, va afisa urmatoare stare pe afisor(adica rezultatul final) si va trece in starea AFIS_REZ.

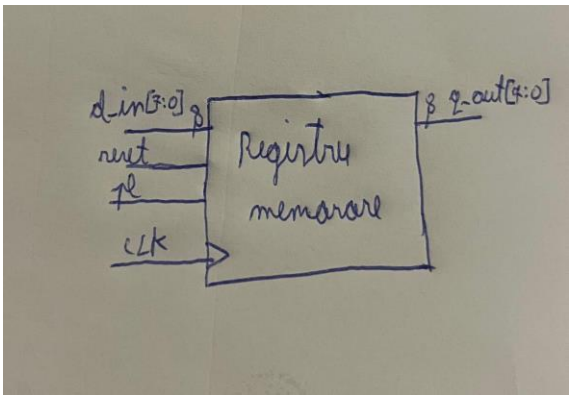
AFIS_REZ : Aceasta stare va afisa pe cele 4 afisoare, rezultatul final al operatiilor (sel_afis = “10” si va astepta pana cand utilizatorul va apasa din nou butonul mem_btn pentru a reia ciclul de calcul(trece in starea MEM_A). In aceasta stare, utilizatorul poate folosi doua switch-uri pentru a alege ce rezultat sa fie afisat.

Atunci cand automatul se afla in starea AFIS_REZ, utilizatorul poate selecta ce operatie poate fi vizualizata folosind 2 switch-uri. Selectia pentru afisare este:

- “00” : Afisare rezultat ADUNARE
- “01” : Afisare rezultat SCADERE
- “10” : Afisare rezultat INMULTIRE
- “11” : Afisare rezultat IMPARTIRE

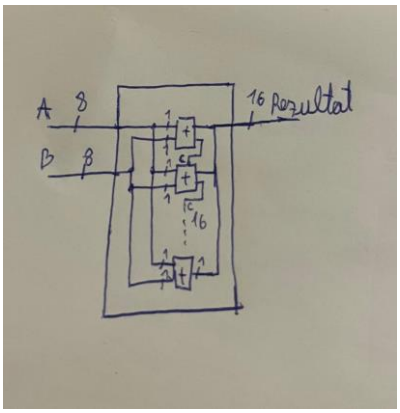
b) Registru memorare

Automatula va avea doua registre care memoreaza numerele A si B. Aceste numere vor fi trimise la fiecare componenta care vor efectua operatiile matematice. Registrul memoreaza numere reprezentate pe 8 biti.



c) Sumator

Sumatorul pe 8 biti estie implement structural cascadand 16 sumatoare pe 1 bit. In implementarea in VHDL, s-a folosit functia “signed” pentru conversia numerelor care sunt reprezentate in std_logic_vector.

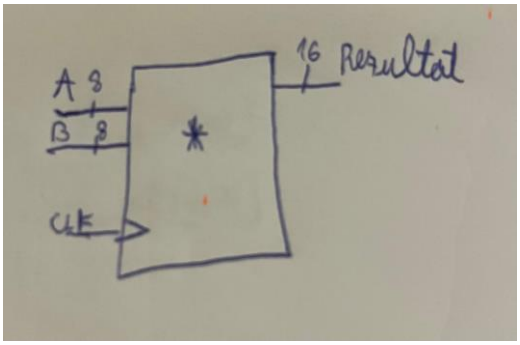


d) Scazator

Scazatorul foloseste aceeasi logica structurala a sumatorului. Folosind functia de conversie din VHDL “not signed(B)” numarul B va fi negat si se ajunge la o operatie : A + (-B).

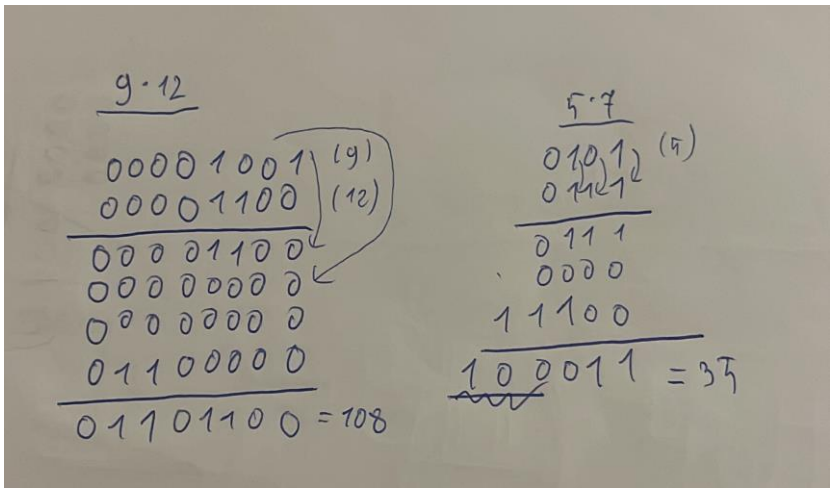
e) Inmultitor

Intrarile sunt doua numere pe 2 biti. Iesirea este produsul lor.



Inmultirea este realizata prin metoda adunarilor repetate.
Pentru A * B vom aduna B de A ori, in functie de bitii lui A. Cifra cea mai din dreapta a lui A este inmultita cu B, iar rezultatul este pregatit pentru adunare. Trecem la urmatorul bit si il inmultim si pe acesta cu B, dar de aceasta data facem o deplasare la stanga cu un bit. Aceasta procedura se repeta de 8 ori (deoarece folosim 8 biti).

Mai jos este prezentat un exemplu:



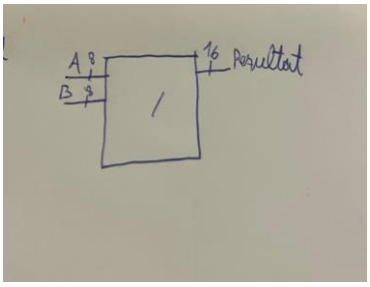
Initial, in cod, fiecare bit al lui A a fost verificat pentru a decide cate numere pozitive vor fi adunate. Daca bitul este „1”, atunci numarul va fi B, altfel va fi „0”.

Apoi facem deplasari la stanga in functie de fiecare bit al lui A. Atribuirea din dreapta va fi a(7) si va ramane neschimbata. Pentru a(6), se va face o deplasare la stanga cu o unitate. Pentru a realiza deplasarea, numarul va fi adunat cu el insusi. In cazul lui a(5), B-ul curent va fi adaugat sau se vor aduna de doua ori 0 cu valoarea precedenta. Exemplu: 3 devine 6 apoi 12: 0011 -> 0110 -> 1100.

In final, toate rezultatele obtinute din deplasari sunt adunate pentru a calcula rezultatul final.

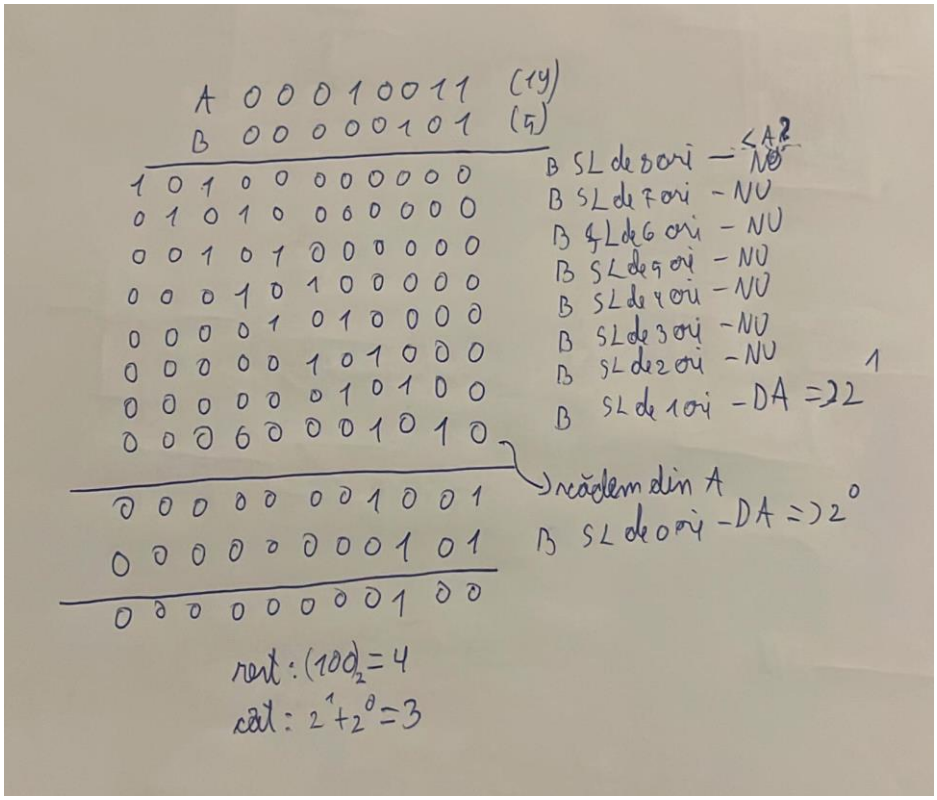
Acest algoritm foloseste adunatoare pe 8 biti construite prin concatenarea unor adunatoare pe 2 biti.

f) Impartitor



Daca dorim sa impartim A la B, B va fi plasat sub A prin deplasarea la stanga, astfel incat B sa fie cat mai mare posibil, dar totusi mai mic decat A. Valoarea catului, care initial este ‘0’, va fi crescuta cu 2 la puterea rangului deplasarii. Apoi, numarul deplasat va fi scazut din A, iar procedura se repeta pana cand toate cazurile sunt verificate. Numarul care ramane dupa scaderi reprezinta restul, iar catul este calculat prin adunarea puterilor lui 2 care au fost valide.

Iata un exemplu:



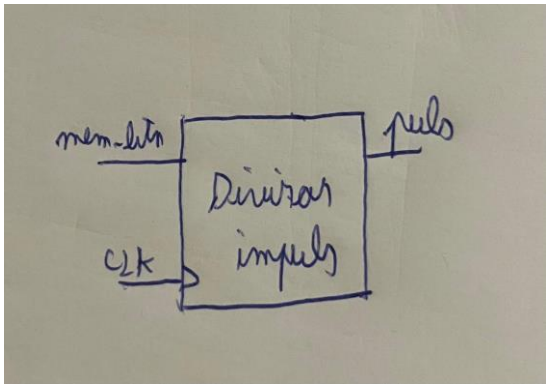
Cand am scris codul, am folosit 16 biti pentru ca niciun bit sa nu se piarda in timpul celor opt deplasari. Deplasarea la stanga se face in acelasi mod ca la inmultire, prin adunarea numarului curent de doua ori, dar in acest caz fara adunatoare.

Dupa ce sunt efectuate deplasarile, verificam (folosind adunarea) daca cea mai mare deplasare, prima fiind pe 8 biti, este valida. Daca se intampla acest lucru, din r, care preia valoarea lui A, scadem acea deplasare si adaugam la cat valoarea 2^8, adica 128. Daca nu, trecem mai departe. Aceasta procedura se aplica la fiecare deplasare efectuata.

Catul va fi egal cu suma puterilor lui 2 (acolo unde conditia este indeplinita), iar restul va fi ceea ce ramane dupa scaderi.

g) Debouncer

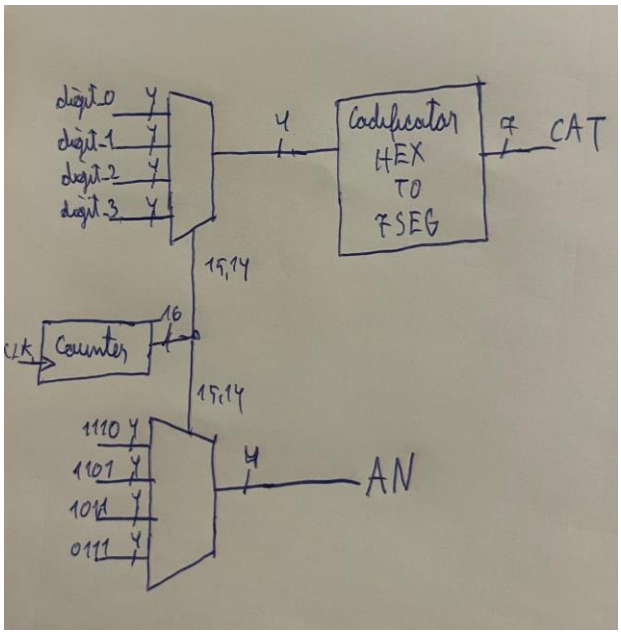
Divizorul de impuls este componenta care face apasarea butonului sa functioneze corespunzator. La apasarea butonului fizic de pe placa, se vor crea multiple oscilatii de 0 si 1 din cauza naturii mecanice a butonului. Asadar, acele semnale de ‘0’ trebuie transformate in 1 cat timp se tine apasat butonul.



h) Afisor pe 7 segmente

Afisorul pe 7 segmente foloseste un numarator care va face selectia afisarii fizice deoarece numaratorul se incrementeaza la frecventa impulsului de tact. Acest lucru realizeaza o afisare ciclica a numerelor (deoarece ochiul uman nu percepe asta), in functie de selectia reprezentata de numarator.

Exemplu de ciclu: Daca selectia multiplexoarelor au valorile “01” atunci pe anodul 1101(adica al 2-lea afisor), va fi reprezentat digit_1 care este codificat pentru display-ul pe 7 segmente

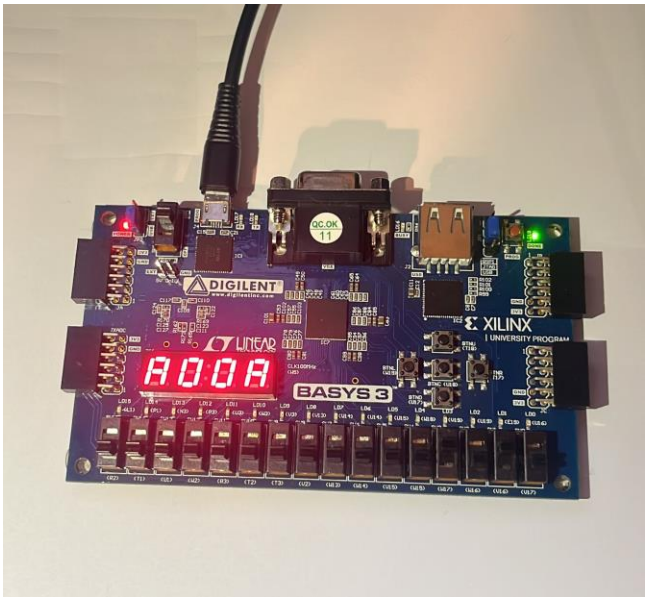


5. De ce am ales acest proiect

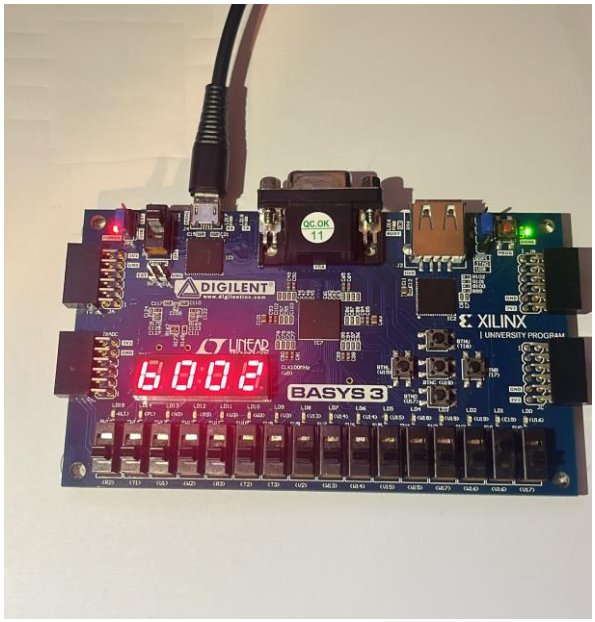
Am ales acest tip de implementare pentru ca parea cea mai usoara si, in acelasi timp, cea mai practica. Mai intai am construit componentele principale si secundare, iar la final le-am legat prin ultima componenta. Toate operatiile sunt efectuate in mod permanent. Rezultatul ramane neschimbat pana la apasarea butonului mem_btn sau reset_btn. Nu a fost nevoie sa implementez stari, deoarece semnalele si rezultatele se modifica doar atunci cand sunt aplicati stimuli externi.

6. Rezultate experimentale

Initial, pe placuta se va afisa numarul A, acesta se poate seta utilizant cele 8 switch-uri reprezentate in imaginea de mai jos:

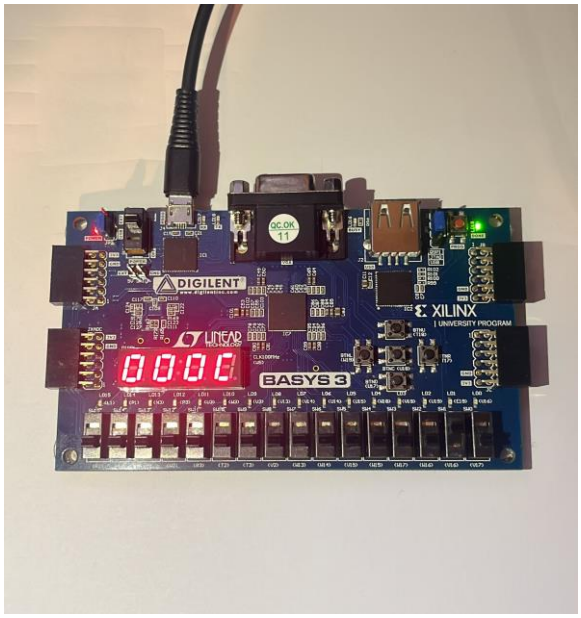


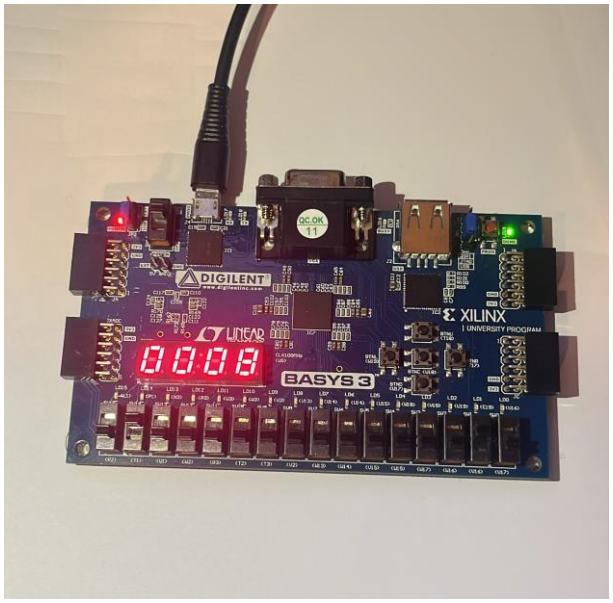
Dupa setarea numarului A, se va apasa butonul din stanga(mem_btn) si pe afisor va aparea numarul B reprezentat pe aceleasi 8 switch-uri.

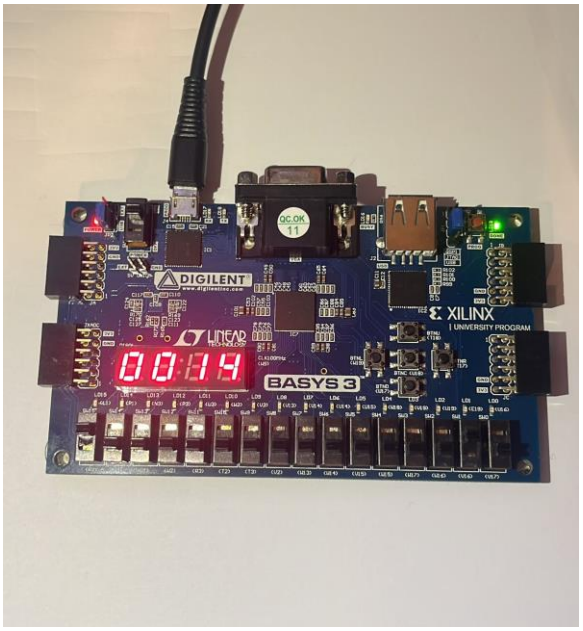


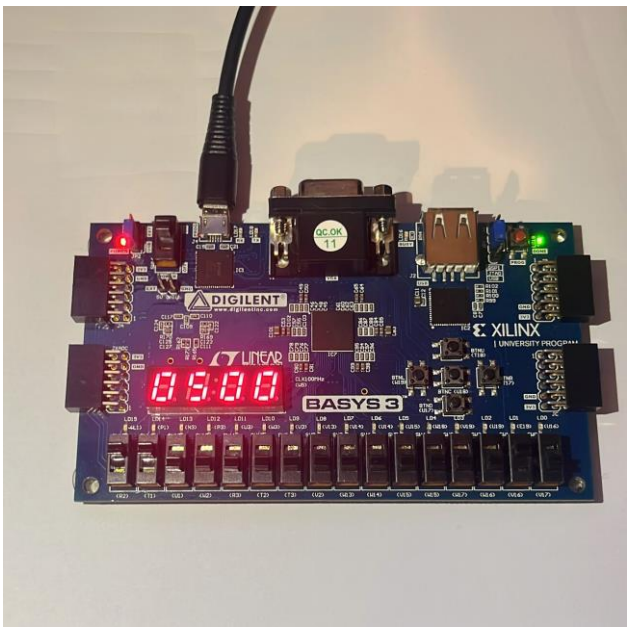
Numarul B se va seta in acelasi mod ca si numarul A si dupa pasarea aceluiasi buton, se va afisa rezultatul operatiilor pe afisor.

Ultimele doua switch uri vor seta ce operatie sa se afiseze. Rezultatele sunt exemplificate mai jos.









In cazul in care utilizatorul doreste resetarea calculatorului(starea in care se reintroduc numerele) acesta poate apasa butonul de sus pentru resetarea automatului.

7. Imbunatatiri ulterioare

O imbunatatire semnificativa a automatului poate fi un codificator care reprezinta datele sub forma zecimala, singura limitare fiind afisoarele pe 7 segmente de pe placuta care sunt putine la numar. Singura solutie este decrementarea numarului de biti pentru date de intrare si iesire.

8. Concluzie

Acest proiect mi-a pus cu siguranta abilitatile la incercare. Cred ca m-a ajutat sa imi imbunatatesc programarea si sa inteleg mult mai bine VHDL-ul. Atunci cand uiti ca este un limbaj descriptiv pentru hardware si il tratezi ca pe un limbaj de programare, problemele

pe care le vei intampina in cod se vor intensifica – si am invatat asta pe propria piele. Per total, a fost o experienta buna, iar satisfactia de la final a meritat efortul.

9. Bibliografie

1. http://www.secs.oakland.edu/~llamocca/

2. https://www.aldec.com/en/products/fpga_simulation/active_hdl_student Active-HDL Student Edition - FPGA Simulation - Products - Aldec – Active HDL for simulation

3. https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html Vivado for implementing the project on Basys 3