UNITATEA ARITMETICA SI LOGICA (UAL)

Pacurar Cristian
Grupa 30233
AC UTCN

CUPRINS

REZUMAT

Unitatea aritmetico-logica este un circuit care este capabil de efectuarea unor operatii pe biti in functie de anumite intrari. In cele ce urmeaza va fi prezent modul de lucru care a fost abordat in realizarea proiectului, algoritmii folositi si utilizarea proiectului prorpiu-zis. Rezultatele experimentale au aratat corectitudinea functionalitatilor implementate dar si complexitatea modulelor descrise. Concluzile au fost unele benefice in ceea ce priveste dezvoltarea personala in domeniul hardware si invatarea de lucruri noi.

INTRODUCERE

Proiectul are ca scop dezvoltarea unei aplicatii software si testarea hardware a acesteia pe o placuta de dezvoltare pentru realizarea unei unitati aritmetico-logice. Unitatea, asa cum ii spune numele, are doua parti principale: cea pentru efectuarea operatiilor aritmetice (+, -, *, /) si cea pentru operatiile logice (and, or, xor, shift), toate acestea fiind coordonate de o unitate de comanda.

Ca si tehnologii utilizate, pe partea software se va folosi limbajul VHDL, codul fiind scris in IDE-ul numit Vivado de la Xilinx. Tot de la Xilinx, pe partea hardware, pentru implementarea fizica se va folosi o placuta de dezvoltare Basys 3. Aceasta placuta este dedicata proiectelor mai mici, fiind destinata programatorilor incepatori pentru introducerea lor in luma programarii hardware, fiind o optiune perfecta pentru proiectul realizat, avand o complexitate relativ redusa.

Proiectul final trebuie sa poata efectua toate operatiile descire mai sus, interfata fiind usor de folosit si testat. Solutia propusa in urmatoarele capitole va folosi algoritmi si modele de circuite documentate si testate, avand avantaje de viteza fata de alte optiuni existente. Algoritmi precum inmultirea Booth, impartirea cu refacerea restului partial si adunarea cu anticiparea transportului au fost folositi pentru realizarea proiectului.

Pentru realizarea proiectului fiecare operatie trebuie descrisa prin unul sau mai multe circuite care comunica simultan, codul respectand standardul VHDL permis, fiind recomandata modulaizarea si generalizarea circuitelor descrise, in scopul usurarii extinderii ulterioare a acestora.

In cele ce urmeaza vor fi prezentate fundamentele teoretice din spatele algoritmilor si a implementarii cu toate detalii necesare intelegerii acestora, descrierea proiectarii si implementarii propriu-zise a proiectului atat software si cat si hardware, rezultate experminetale in urma implemetarii si imagini cu datele simulate in Vivado, precum si concluzii in ceea ce priveste informatiile asimilate in urma dezvoltarii unei astfel de aplicatii impreuna cu bibliografia necesara.

FUNDAMENTARE TEORETICA

In aceasta sectiune, vor fi descrise fundamentele teoretice legate de proiect, cum sunt modele, metode si tehnologii care pot fi utilizate. Este foarte importanta intelegerea circuitelor folosite si a functionarii acestora, asadar in cele ce urmeaza se vor prezenta pe scurt circuitele mai importante folosite in proiect si de asemenea functionalitatea lor.

Display 7 segmente (SSD) – este un circuit care face posibila afisarea numerelor in hexa zecimal pe afisorarele placutei de dezoltare, controlul afisarii facandu-se cu ajutorul anozilor si catozilor.

Debouncer (DB) – atunci cand un buton este apasat, semnalul nu este transmit intotdeauna corect, din cauza unor imperfectiuni fizice, iar acest circuit poate rezolva aceasta problema cu ajutorul unor bistabile D.

Unitate de comanda (UC) – dupa cum ii spune si numele, acest circuit se ocupa de comanda tuturor semnalelor modulului pentru care este alocata.

Sumator (ADDN) – circuit care se ocupa de suma a doua numere pe n biti.

SRRN – registru de deplasare la dreapta a n biti.
SLRN – registru de deplasare la stanga a n biti.
FDN – registru de n biti cu resetare sincrona.

Documentele teoretice necesare realizarii proiectului se pot gasi in resursele laboratorului de SSC la lucrarile Circuite aritmetice combinationale si secventiale, dar si in resursele laboratorului de AC de anul trecut.

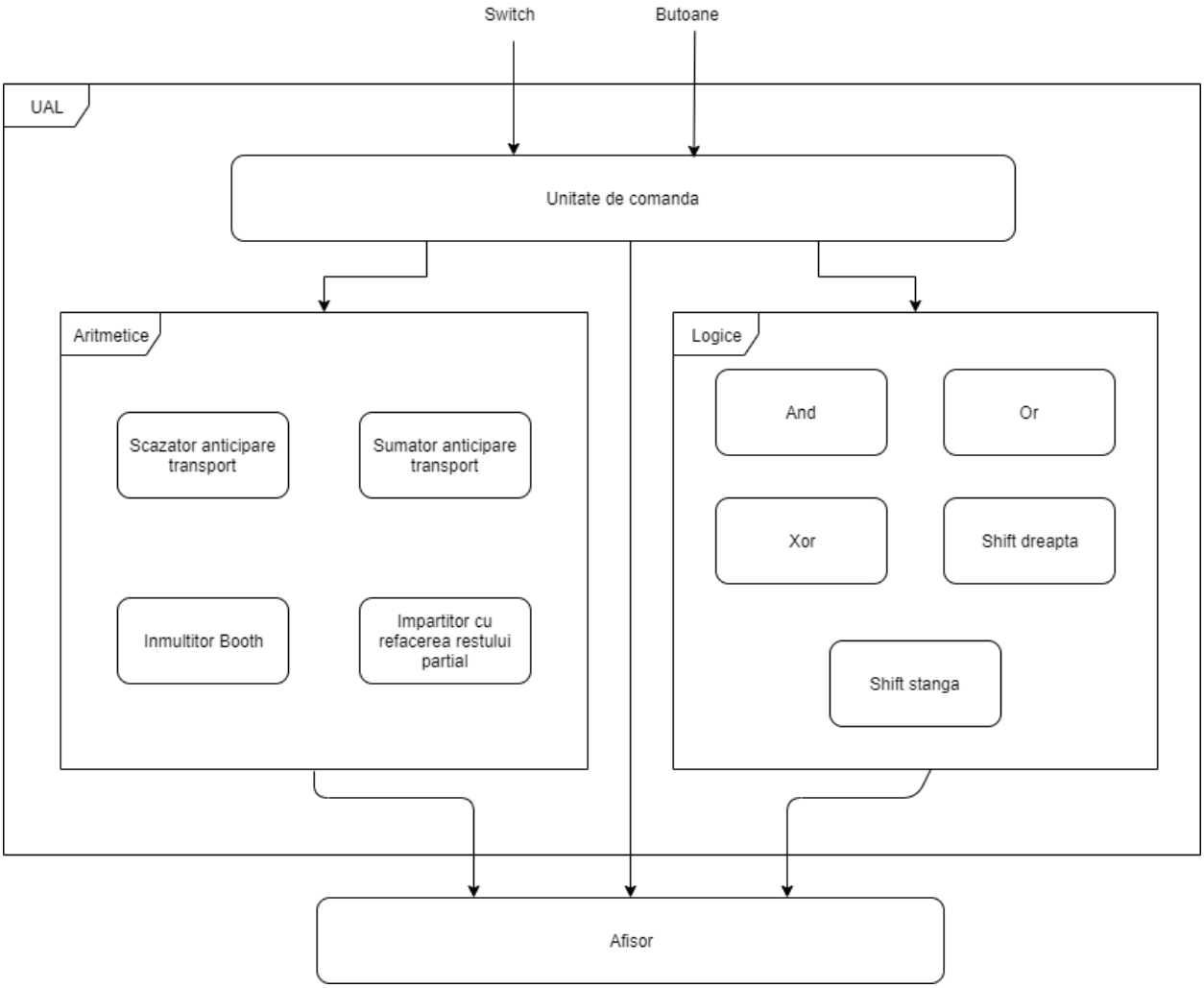
PROIECTARE SI IMPLEMENTARE

Pentru proiectarea initiala este foarte importanta determinarea butoanelor de comanda de pe placuta de dezvoltare Xilinx. Placuta Basys 3 are 16 switch-uri, 5 butoane si 4 afisoare 7 segmente. S-a incercat minimalizarea folosirii acestora cu ajutorul logicii pe biti pentru a fi mai usoara utilizarea placutei. Fiecare operatie asignata switch-urilor si butoanelor este necesara functionarii corecte a unitatii artimetico logice.

- **SW:**
  - 3-0: bit nr
  - 5-4: selectie parte nr
  - 6: enable load x si y
  - 8-7: selectie x, y, parte superioara/inferioara rez
  - 12-9: selectie op
  - 13: 0 sau 1 shift

- **BTN:**
  - btn mijloc: start inmultire/impartire
  - btn sus: reset
  - btn stanga: shift

- **LED:**
  - 15: transport/semn



Sumator cu anticiparea transportului:

Este primul circuit implementat din schema finala. Acesta se asigura de adunarea a doua numere pe n biti. In cazul proiectului, n-ul s-a ales sa fie 16, insa poate fi generalizat pe mai multi biti, singurul inconvenient fiind modificarea switch-urilor pentru comenzi.

Pentru realizarea unui sumator de 16 biti, s-au folosit 8 sumatoare pe 2 biti cu anticiparea transportului, in fiecare dintre aceste sumatoare folosindu-se formulele pentru Pi si Gi corespunzatoare:

Pi = xi or yi
Gi = xi and yi, xi si yi fiind bitul i a numarului x si y.

P si G final pentru fiecare sumator pe 2 biti au urmatoarele formule:
P(n-1)(n) = Pn or Pn-1

G(n-1)(n) = Gn or (Pn and Gn-1)

Pentru calcularea transporturilor fiecarui sumator in sumatorul pe 16 biti se foloseste urmatoare formula:
Tn = G(n-2)(n-1) or (P(n-2)(n-1) and T(n-2))

Suma este calculata pe cate 2 biti de odata, avand transportul T16.

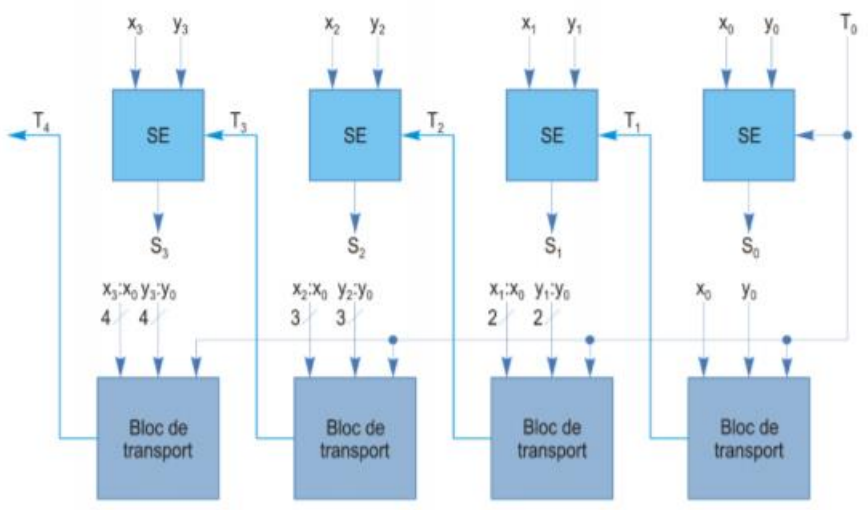


Figura 1. Schema bloc a unui sumator de 4 biți cu anticiparea transportului.

Scazatorul implemetat este defapt un sumator, cu diferenta ca numarul care se scade va fi scris in complement fata de 2 si apoi adunat normal cu celalalt numar in sumatorul prezentat mai sus. Complement fata de doi -> negarea bitilor si abunarea cu 1.

Inmultitor Booth:

In cazul in care se lucreaza cu o reprezentarea a numerelor in C2, se poate utiliza algoritmul lui Booth pentru inmultire. Avantajul unei asemenea metode constă în eliminarea operației de complementare a operanzilor și a rezultatului, dacă aceștia sunt negativi, ceea ce este important în cazul conversiei din C2 în mărime și semn, care este mai dificilă decât cea din C1 în mărime și semn.

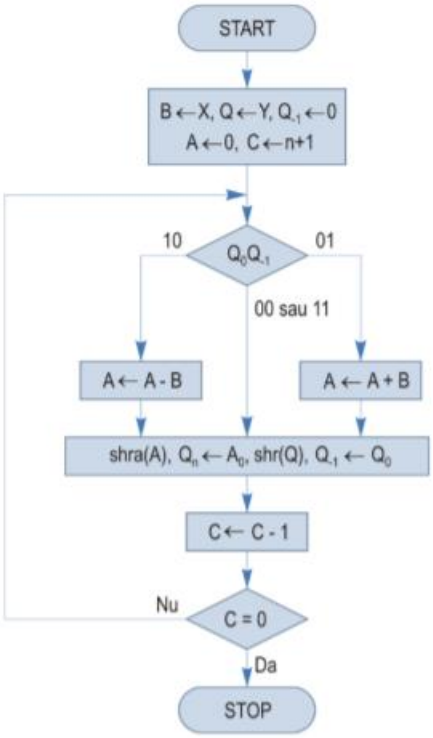


Figura 4. Organigrama algoritmului de înmulțire prin metoda Booth.

La inceputul algoritmului, registrul A este initializat cu 0, B cu x, Q cu y, bistabilul Q-1 cu 0 iar numarul C cu n+1 (n fiind numarul de biti cu care se lucreaza). Q0Q-1 reprezinta ultimii 2 biti a registrului Q.

Tabelul 3. Operațiile efectuate la înmulțirea prin metoda Booth, în funcție de valoarea biților testați.

$y_i\ y_{i-1}$	Operații
0 0	Deplasare produs parțial la dreapta
0 1	Adunare deînmulțit, deplasare produs parțial la dreapta
1 0	Scădere deînmulțit, deplasare produs parțial la dreapta
1 1	Deplasare produs parțial la dreapta

Apoi se verifica cei doi biti si in functie de tabelul de mai sus se va lua decizia necesara continuarii algoritmului. In functie de decizie se poate scadea sau aduna din registrul a registrul b urmand o serie de operatii sau se poate sarii peste adunare sau scadere trecandu-se direct la operatiile urmatoare. Aceste operatii constau in shiftarea la dreapta a registrului A si Q, bitul cel mai semnificativ a registrului Q ia valoarea bitului cel mai putin seminificativ al registrului A iar Q-1 ia valoarea lui Q0, apoi se scade numarul de pasi C cu 1 si se verifica daca C a ajuns la zero. In caz contrar se reia algoritmul de le verificarea celor doi biti pana cand C este zero.

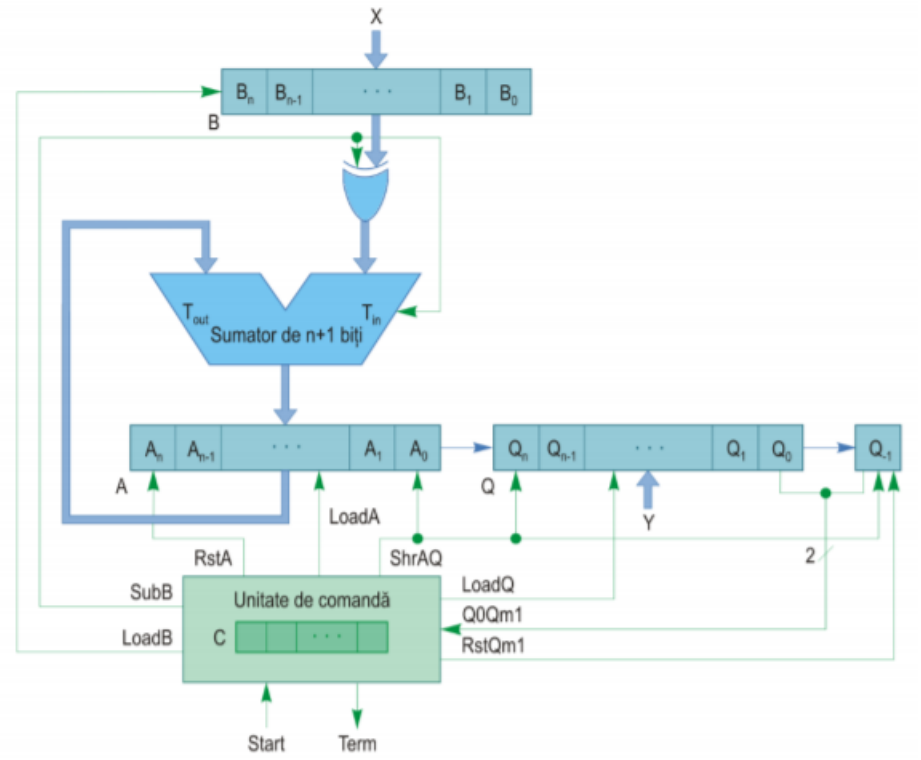


Figura 3. Schema bloc a unui circuit de înmulțire prin metoda Booth.

Pentru realizarea schemei a fost necesara implementarea unei unitati de comanda care a fost structurata pe ogranigrama prezentata a algoritmului. Unitatea de comadna este un automat sincron de stari. Starile folosite sunt: idle, initialize, test, minus, plus, shift, count, stop.

Automatul sta in stare idle pana primeste un semnal de start, dupa aceea trece in faza de initializare, iar apoi in cea de test. In functie de test, automatul poate sa treaca in starea plus sau minus si apoi in starea shift sau direct in shift. Dupa aceea starea count este incarcata in care se testeaza daca algoritmu s-a sfarsit sau nu. In functie de acest lucru urmatoarea stare poate fi test sau stop. Din stop, automatul va reveni inapoi in idle.

Stare in care se afla automatul va determina si stare semnalelor de iesire a unitatii de comanda pentru celelalte componente precum resetare registrului A si Q, incarcare registrului B, semnalul de scadere sau adunare pentru sumator, semnalul de shift pentru registre etc. La terminarea algoritmului semnalul Term va avea valoarea 1.

Rezultatul inmultirii numarului x si y va fi registrul A concatenat cu Q.

Impartitor cu refacerea restului partial:

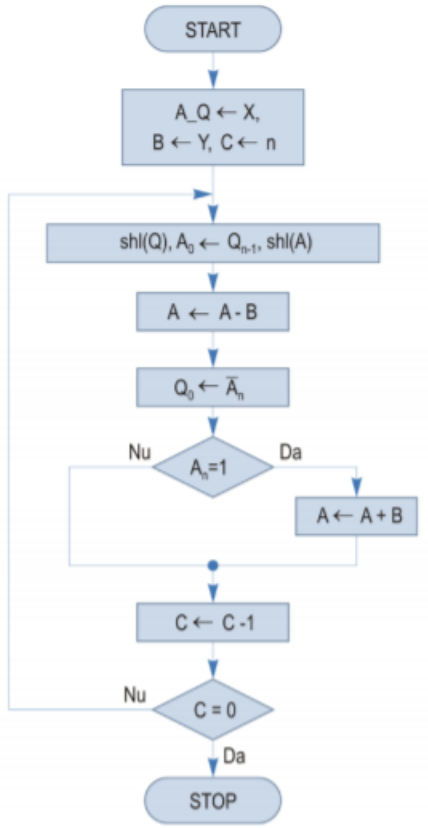


Figura 6. Organigrama operației de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

Algoritmul folosit este foarte asemanator cu in multirea Booth cu mici diferente. A concatenat cu Q se initializeaza cu x, iar C este egal cu n. In loc de shiftare la dreapta acum se face shiftare la stanga. Operatie de scadere din registrul A a registrului B se face intotdeauna, Q0 primind valoarea lui An negat. Testul se face cum in functie de An = 1. In caz afirmativ se va face o adunare si se va trece direct la starea de testare a lui C. Algorimtul se reia de la shiftarea la stanga a registrilor oprinduse in stare stop catunci cand C = 0.

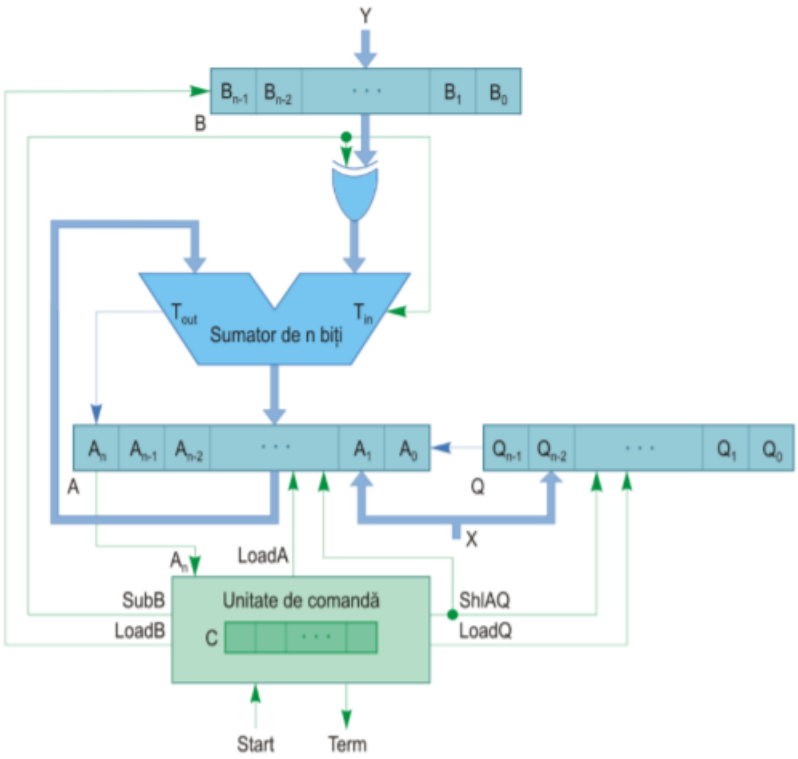


Figura 5. Schema bloc a unui circuit de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

Si in acest caz este nevoie de o unitate de comanda separata care functioneaza la fel ca si cea de la inmultire, insa cu semnalele specifice impartirii.
Catul impartrii lui x la y este se afla in registrul Q, iar restul partial in registrul A.

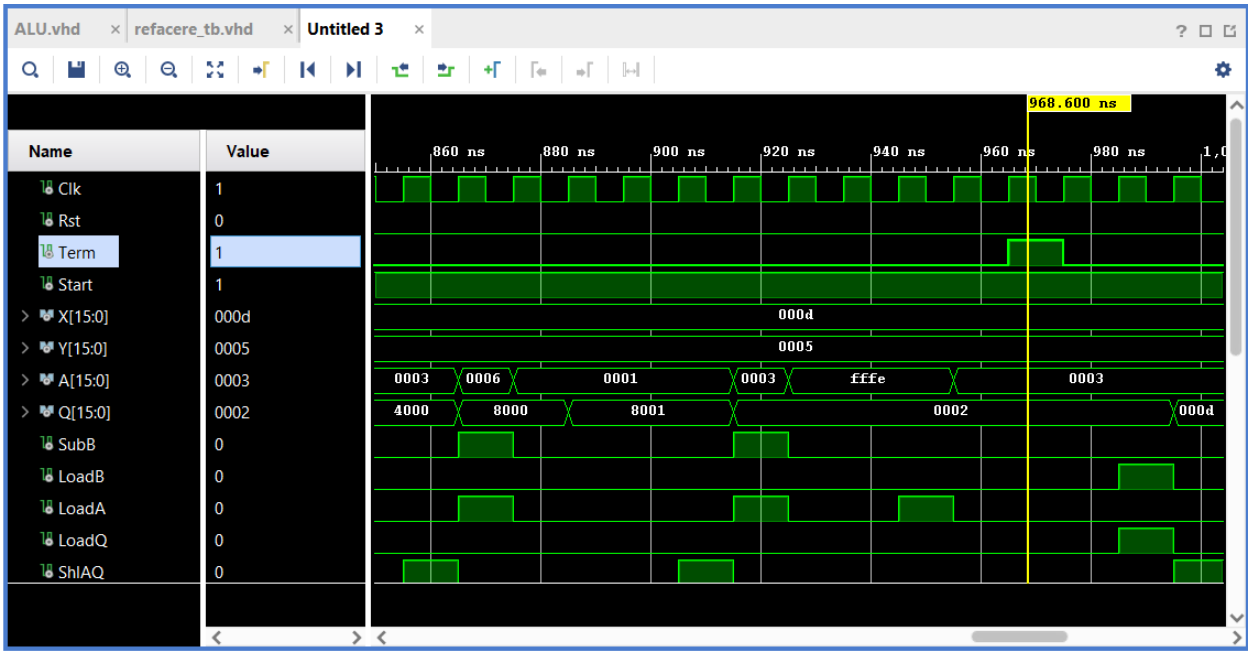
In ceea ce priveste operatiile logice au fost folosite porti logice simple precum SI, SAU, SAU-EXCLUSIV, iar pentru shiftare s-au folosit registrii SRRN si SLRN.

Toate operatii aritmetice si logice sunt comandate de unitatea de comanda a UAL. Aceasta in functie de combinatia switch-urilor si a butoanelor transmite x-ul si y-ul catre module, alege operatia efectuata, da startul atuomatelor definite si alege rezultat afisata pe cele 4 afisoare 7 segmente.

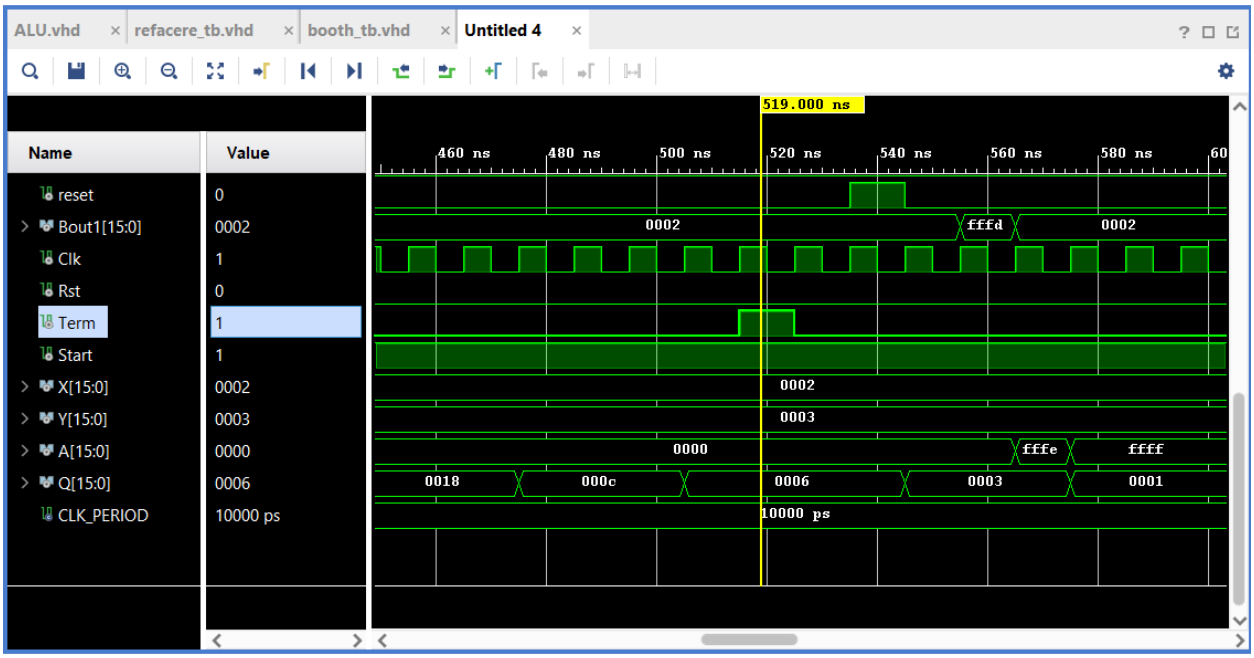
REZULTATE EXPERIMENTALE

Deoarece proiectul este unul destul de complex luand in calcul toate circuitele necesare realizarii acestuia, a trebuit testat fiecare modul in parte in simulatorul oferit de Vivado inainte implementarii si testarii fizice pe placuta. Asadar dupa testarea fiecarui modul principal, acestea au fost conectate si sintetizate.

Pentru tstarea impartitorului am luat x = D (13) si y = 5. Rezultatul catului se poate vedea in registrul Q (= 2), iar restul partial in A (= 3).



Pentru inmultire am luat x = 2 si y = 3. Rezultatul se poate vedea prin concatenarea registrului A (= 0) si Q (= 6).



Pentru simularile efectuate au fost folosite bancuri de test, adica o entitate goala, arhitectura identica cu cea originala si semnale declarate in arhitectura pentru testarea corectitudinii codului

scris. De asemenea s-au folosit instructiunile wait si wait for pentru implementarea clock-ului si a altor semnale de test.

CONCLUZII

Proiectul realizat a fost foarte util din punct de vedere al intelegerii materiei din acest semestru deoarece a folosit o parte din algoritmii predati la curs si laborator. Totodata am invatat lucuri noi despre interactiune dintre circuite din punct de vedere hardware si cum se pot descriere ele din punct de vedere software. Am invatat cum se poate face o simulare, fara a fi nevoit testarea fizica pe placa, ceea ce a scurtat timpul de implementare a proiectului.

Avantajul acestui proiect este ca cuprinde o varietate larga de operatii utile, folosind diferiti algoritmi clasici, fiind usor de utilizat fizic fara a necesita diferite module hardware in plus.
Alt avantaj este acela de extindere a proiectului atat cand vine vorba de functionalitate (se pot adauga operatii noi) cat si cand vine vorba de memorie (se pot extinde numerele pe mai multi biti folosindu-se tipul generic).

Un dezavantaj este folosirea switch-urilor si a afisoarelor care sunt limitate ca si numar. Aceasta problema s-ar putea rezolva cu folosirea altui tip de placa de dezvoltare, insa petru usurarea muncii s-a folosit Basys 3, fiind strict folosita pentru ceea ce era nevoie in implementare.

BIBLIOGRAFIE

1) http://users.utcluj.ro/~baruch/ssc/labor/Aritm-Combinationala.pdf

2) http://users.utcluj.ro/~baruch/ssc/labor/Aritm-Secventiala.pdf

3) http://users.utcluj.ro/~onigaf/files/teaching/AC/AC_indrumator_laborator.pdf