Universitatea Tehnică Cluj-Napoca
Departamentul de Calculatoare

Circuite de inmultire zecimala

Bicioi Constantina Luiza
Grupa 30238

Profesor îndrumător: Lisman Dragos Florin

Data: 08.01.2024

Cuprins

1. Rezumat

În contextul dispozitivelor hardware, importanța aritmeticii zecimale a crescut în ultima perioadă, oferind o alternativă la metodele tradiționale bazate pe sistemul binar pentru efectuarea operațiilor de calcul. Proiectul curent s-a concentrat pe proiectarea și implementarea a **două metode** de înmulțire zecimală pentru numere de 4 cifre, integrându-le într-un sistem digital. Am comparat aceste două metode pentru a evidenția utilitatea lor în contextul sistemelor actuale. Am folosit limbajul de descriere hardware VHDL în Vivado Design Suite pentru a realiza o descriere structurală a circuitelor. Fiecare circuit specific unei metode a fost implementat, iar apoi aceste circuite au fost integrate într-un sistem funcțional și ușor de utilizat. Rezultatele au fost testate cu simulatorul din Vivado Design Suite și apoi pe placa de dezvoltare FPGA Nexys 4 DDR. Acest proiect a necesitat cunoștințe solide în descrierea hardware-ului, algoritmi și a contribuit la dezvoltarea abilităților în proiectarea și implementarea unui circuit hardware complex, cu potențial de utilizare în domenii financiare, economice și tehnologice.

2. Introducere

Sistemul zecimal de numerotare reprezintă cea mai obișnuită și utilizată metodă de lucru cu datele numerice în relația dintre oameni sau om și calculator. Aritmetica zecimală este preferată în mediile care prelucrează informații numerice, cum ar fi domeniile științifice, financiare, economice și aplicațiile online.

Calculatoarele generale din prezent realizează operații zecimale folosind aritmetica binară. Informațiile binare sunt ușor de stocat și manipulat. Cu toate acestea, există diverse motive pentru a trece la aritmetica zecimală, începând de la familiaritatea umană cu aceste calcule și până la diferențele dintre reprezentările binare și cele zecimale ale acelorași valori.

Valorile binare în format zecimal pot doar să aproximeze anumite valori zecimale din cauza reprezentării lor interne. De exemplu, valoarea zecimală 0.1 necesită un șir infinit de valori binare 0 și 1 pentru reprezentare în baza 2. În multe situații reale, această discrepanță este inacceptabilă, mai ales în instituțiile financiare unde rezultatele calculatorului trebuie să fie identice cu calculele făcute manual. Pentru a asigura corespondența, calculele trebuie să fie procesate în sistemul zecimal.

Dispozitivele hardware care susțin aritmetica zecimală au devenit importante datorită creșterii necesității de putere de procesare. Astăzi, unitățile hardware pentru aritmetică zecimală sunt componente esențiale ale procesoarelor generale, înlocuind algoritmii hardware lenti și iterativi folosiți anterior pentru operațiile zecimale complexe, precum înmulțirea.

Tema acestui proiect implică proiectarea, implementarea și testarea unui circuit logic capabil să efectueze operații de înmulțire zecimală pentru numerele de 4 cifre.

Operaţia de înmulţire zecimală este mai complexă decât cea de înmulţire binară, deoarece cifrele înmulţitorului pot lua valori între 0 şi 9. Metodele de înmulţire zecimală depind de modul de reprezentare a numerelor cu semn.
Pentru numere zecimale reprezentate în MS, principalele metode de înmulţire sunt următoarele:
• Metoda adunării repetate;
• Metoda celor nouă multipli ai deînmulţitului;
• Metoda de înmulţire binară a numerelor zecimale;
• Metoda dublării şi înjumătăţirii.
Pentru numere reprezentate în complement faţă de 10, se pot utiliza metodele de înmulţire binară
a numerelor reprezentate în complement faţă de 2. O altă posibilitate este conversia numerelor din complement faţă de 10 în MS şi utilizarea unei metode pentru numere în această reprezentare.

Vor fi prezentate diversele metode de înmulțire existente, proiectând **un circuit care să integreze 2 dintre aceste metode**, oferind utilizatorului posibilitatea de a alege metoda pentru calcul. Proiectarea sistemului va folosi limbajul de descriere hardware VHDL, iar implementarea se va face cu ajutorul unei plăci de dezvoltare FPGA.

Cu ajutorul unei plăci de dezvoltare FPGA, circuitul proiectat va fi implementat, permițând utilizatorului să introducă numere de la switch-urile plăcii, să selecteze metoda dorită și să observe rezultatele operațiilor.

Soluția propusă pentru implementarea pe placa FPGA va include sistemul care înglobează tehnicile de înmulțire zecimală "The Repeated-Addition" și "The Nine-Multiplies-of-Multiplicand". Desi este adevarat ca metoda , "The Right-and-Left-Hand Components", este mult mai eficienta in în timp în comparație cu cele doua metode implementate de mine "The Repeated-Addition" și "The Nine-Multiplies-of-Multiplicand". Deși celelalte două metode sunt simple, ele oferă o eficiență scăzută - metoda "Adunărilor Repetate" este cea mai lentă, iar metoda celor " Metoda celor nouă multipli ai deînmulţitului " presupune un cost și un consum de timp ridicat din cauza încărcării regiștrilor acestia fiind noua la numar (doar cei care tine multiplii posibili)

În capitolele următoare, accentul se va muta asupra prezentării soluției alese pentru rezolvarea cerințelor proiectului, mecanismele utilizate, etapele de dezvoltare, rezultatele, concluziile și ideile de dezvoltare ulterioară a sistemului construit. În capitolul 3, intitulat "Fundamentare teoretică", va fi prezentat un rezumat al teoriei care stă la baza dezvoltării proiectului, în contextul progresului realizat până în acel moment. Capitolul 4, "Proiectare și implementare", reprezintă partea principală a lucrării și argumentează fiecare etapă din procesul de realizare a proiectului. În capitolul 6, "Rezultate experimentale", vor fi analizate și validate rezultatele obținute în urma utilizării sistemului proiectat, iar în ultimul capitol, "Concluzii", se va face un sumar al lucrării, se vor trage concluziile și observațiile aferente acestui proiect."

3. Fundamentare teoretică

3.1. Solutii posibile ale proiectului

Asa cum am mentionat mai sus, pentru numere zecimale reprezentate în MS, principalele metode de înmulţire sunt următoarele:

• Metoda adunării repetate;
• Metoda celor nouă multipli ai deînmulţitului;
• Metoda de înmulţire binară a numerelor zecimale;
• Metoda dublării şi înjumătăţirii.

Cele implementate in acest proiect sunt primele doua: Metoda adunării repetate si Metoda celor nouă multipli ai deînmulţitului. În cadrul utilizării limbajului de descirere hardware VHDL și al programului software Vivado, se vor realiza implementările celor 2 metode prezentate anterior, conform [3]: metoda „The Repeated-Addition” si metoda „The-Nine-Multiples-of-Multiplicand” Cu ajutorul acestor unelte software, se vor simula toate aceste metode și, conform rezultatelor, se vor efectua analize pentru eficiență și îmbunătățiri. Se va proiecta un circuit digital care va îngloba circuitele pentru cele 2 metode, precum și o logică suplimentară pentru a permite utilizatorului să aleagă o metodă din cele trei existente, introducerea unor valori proprii și vizualizarea rezultatelor.

3.1.1. Metoda adunării repetate

Cea mai simpla , dar si cea mai lenta metoda implementata , asa cum ni s-a spus si la laborator această metodă constă în examinarea cifrelor înmulţitorului, începând cu cifra c.m.p.s., şi adunarea repetată a deînmulţitului la produsul parţial. Numărul necesar de adunări este egal cu valoarea cifrei înmulţitorului, valoare determinată printr-o comparaţie. În locul comparaţiei, se poate testa dacă cifra înmulţitorului este 0; în caz contrar, se efectuează o adunare a deînmulţitului la produsul parţial, iar apoi o decrementare a cifrei înmulţitorului.

Operaţia se repetă până când cifra înmulţitorului devine 0.

După efectuarea adunărilor specifice unei anumite cifre a înmulţitorului, grupul de registre care păstrează produsul parţial se deplasează la dreapta cu o poziţie. Operaţia continuă până la examinarea tuturor cifrelor înmulţitorului

3.1.2. Metoda celor nouă multipli ai deînmulţitului

Această metodă constă în generarea la începutul operaţiei de înmulţire a celor nouă multipli ai deînmulţitului (X) şi memorarea acestora în grupuri de registre speciale. Prin examinarea fiecărei cifre a înmulţitorului (Y), se determină conţinutul cărui grup de 4 registre trebuie adunat la produsul parţial.

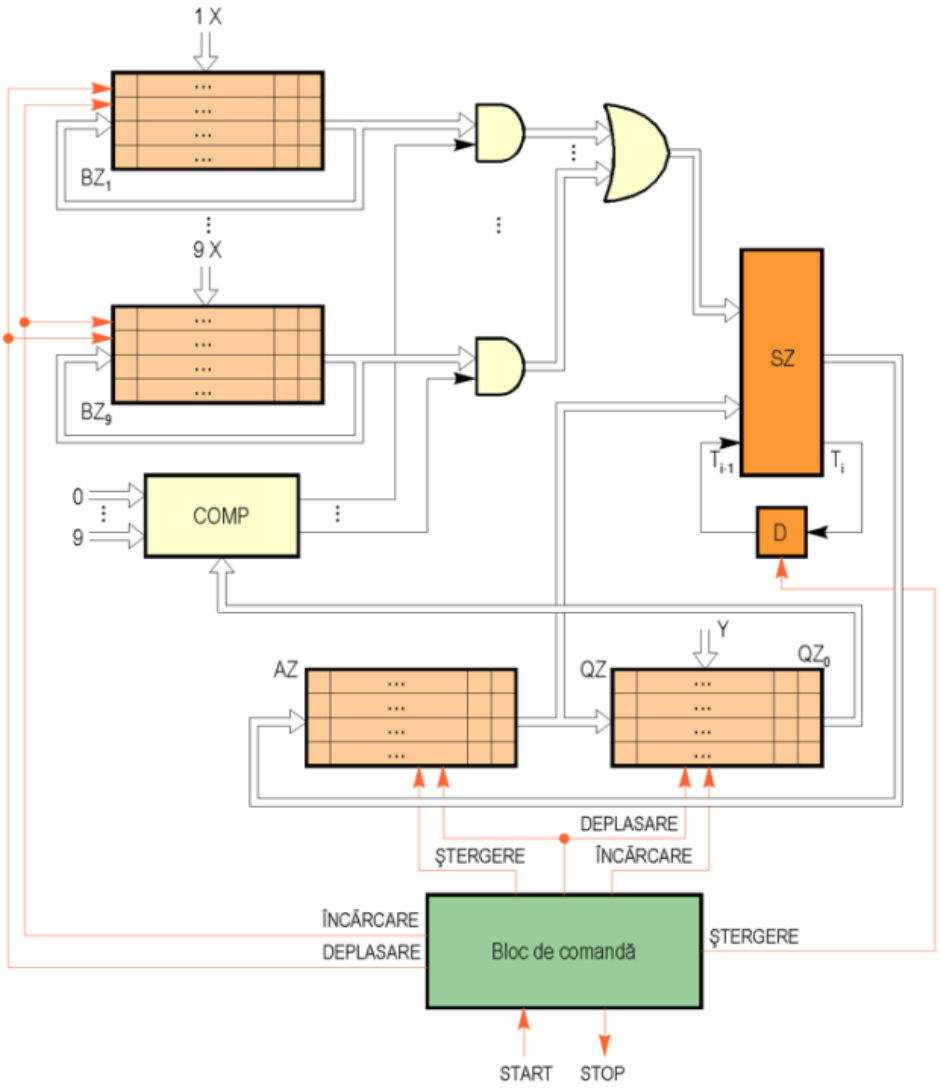


Fig. 1: Diagramă bloc pentru circuitul pentru metoda celor nouă multipli ai deînmulţitului

În faza de inițializare, valorile deînmulțitului și înmulțitorului sunt încărcate în registrele BZ1 și, respectiv, QZ. De asemenea, se pornește acumulatorul AZ și bistabilul D cu valoarea 0, iar se formează multiplii deînmulțitului. Aceasta poate fi realizată utilizând circuite combinaționale sau sumatorul zecimal. În scenariul sumatorului zecimal, după ce deînmulțitul este încărcat în registrul BZ1, acesta este adunat cu conținutul acumulatorului AZ (care a fost inițializat cu 0), rezultatul fiind transferat în AZ. În continuare, se adună conținutul registrelor BZ1 și AZ, iar rezultatul (2X) este transferat în BZ2. Același proces este repetat pentru a obține și ceilalți multipli ai deînmulțitului

3.2. Tehnologii utilizate

3.2.1. FPGA(Nexys4 DDR)

Un **FPGA** (**Field Programmable Gate Array**) este un circuit integrat digital configurabil, de către utilizator, după ce a fost fabricat (spre deosebire de dispozivele a căror funcție este implementată in procesul de fabricație). Configurarea FPGA se face, în general, cu ajutorul unui limbaj de descriere hardware HDL, similar cu cel folosit pentru dispozivele ASIC, dezvoltându-se recent și compilatoare care traduc instrucțiuni din limbajul C în limbaje HDL. Un astfel de compilator este Impulse C.

FPGA-urile sunt alcătuite din blocuri logice configurabile (programabile) legate între ele de o serie de conexiuni configurabile la rândul lor.

Placa Nexys4 DDR este o platformă completă, gata de utilizare pentru dezvoltarea de circuite digitale bazate pe cele mai recente FPGA-uri Artix-7 de la Xilinx®. Cu o mare capacitate FPGA, memorii externe generoase, precum și o colecție de USB, Ethernet și alte porturi, Nexys4 DDR poate găzdui modele variind de la circuite combinaționale introductive la procesoare puternice încorporate. Mai multe tipuri periferice încorporate, inclusiv un accelerometru, senzor de temperatură, MEMS microfon digital, un amplificator, și mai multe dispozitive I / O permit utilizarea plăcii Nexys4 DDR pentru o gamă largă de modele, fără a avea nevoie de alte componente[4].

Unitatea de calcul in virgula mobila pentru adunare si scadere ce urmeaza a fi implementata, utilizând mediul de proiectare Vivado, va fi încărcata pe placa FPGA Nexys4 DDR. Astfel functionarea corecta a acestui circuit poate fi evaluata fizic, si nu doar cu ajutorul simulatorului pus la dispozitie de mediul de proiectare Vivado.

3.2.2. Limbajul utilizat

Pentru dezvoltarea acestui proiect, se va utiliza limbajul de descriere hardware VHDL (acronim de la Very High Speed Integrated Circuit Hardware Description Language). Acesta este destinat descrierii comportamentului și arhitecturii unui sistem logic de calcul, iar cu ajutorul acestuia vom realiza o descriere structurală a circuitului de înmulțire zecimală proiectat.

4. Proiectare şi implementare

4.1. Arhitectura sistemului

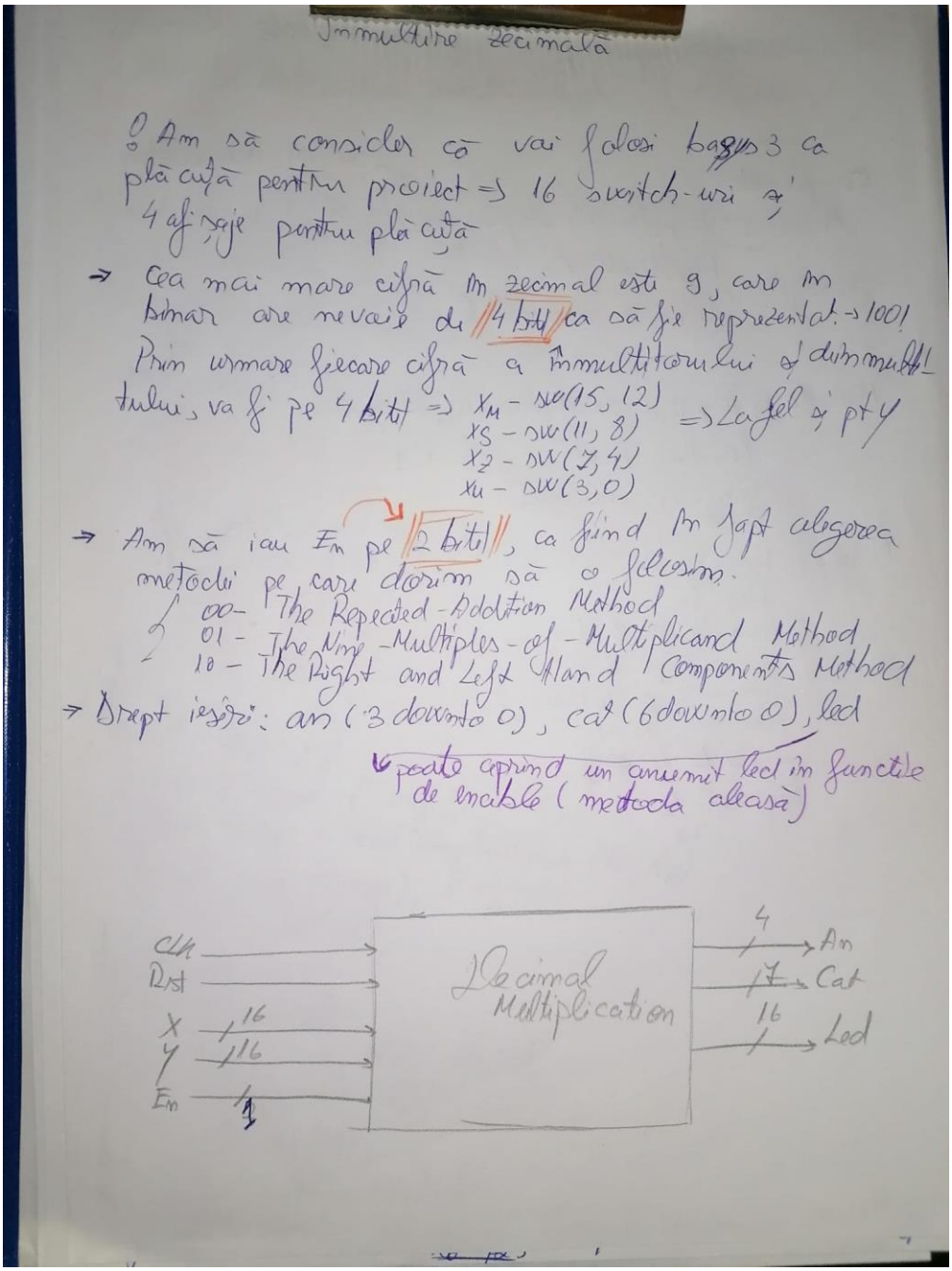


Fig. 2:Black box

Diagrama mai detaliata a proiectului

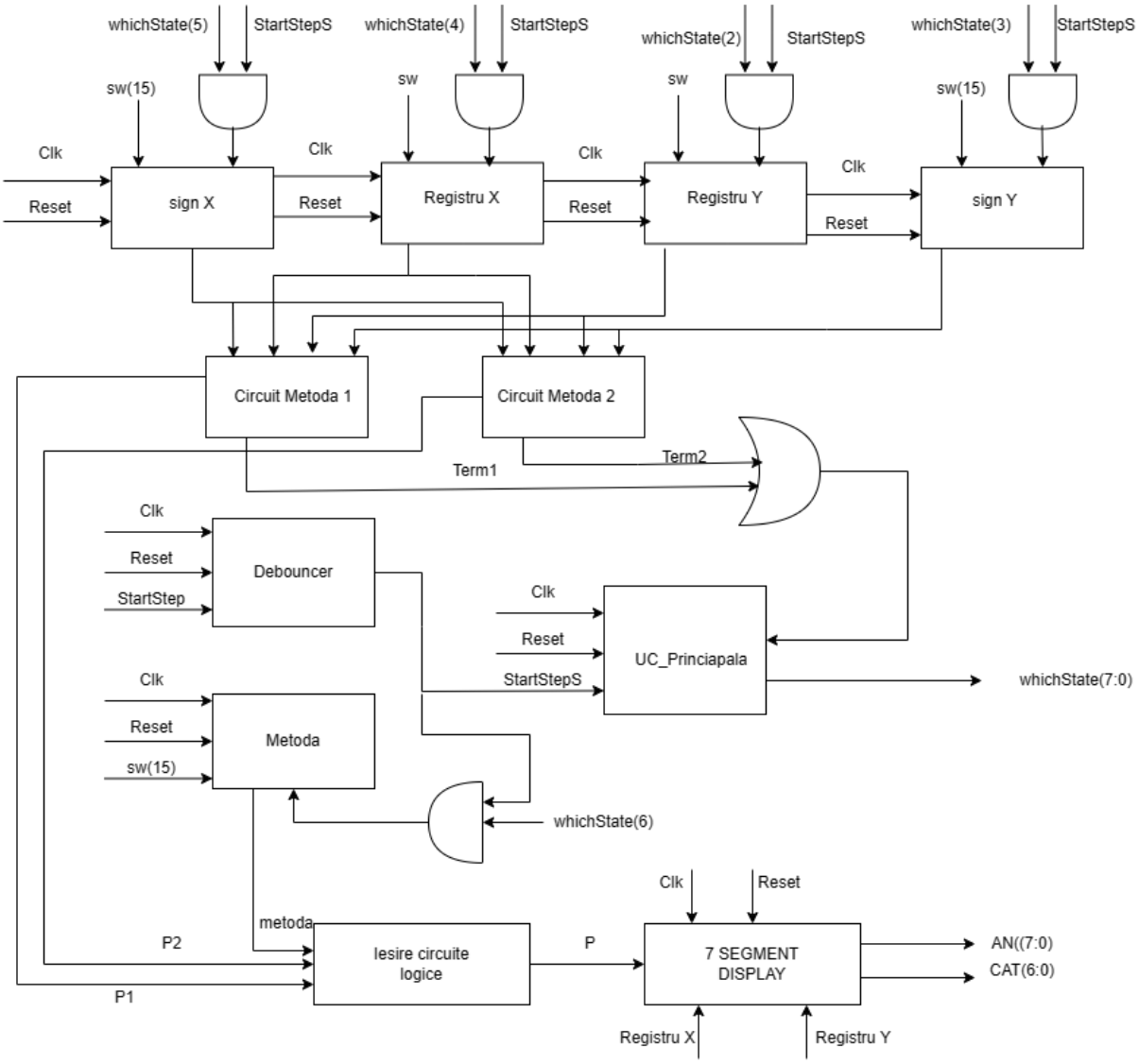


Fig. 3: Schema cirucitului care implementează cele 2 metode de înmulțire zecimală (facuta in draw.io)

Organigrama FSM ului:

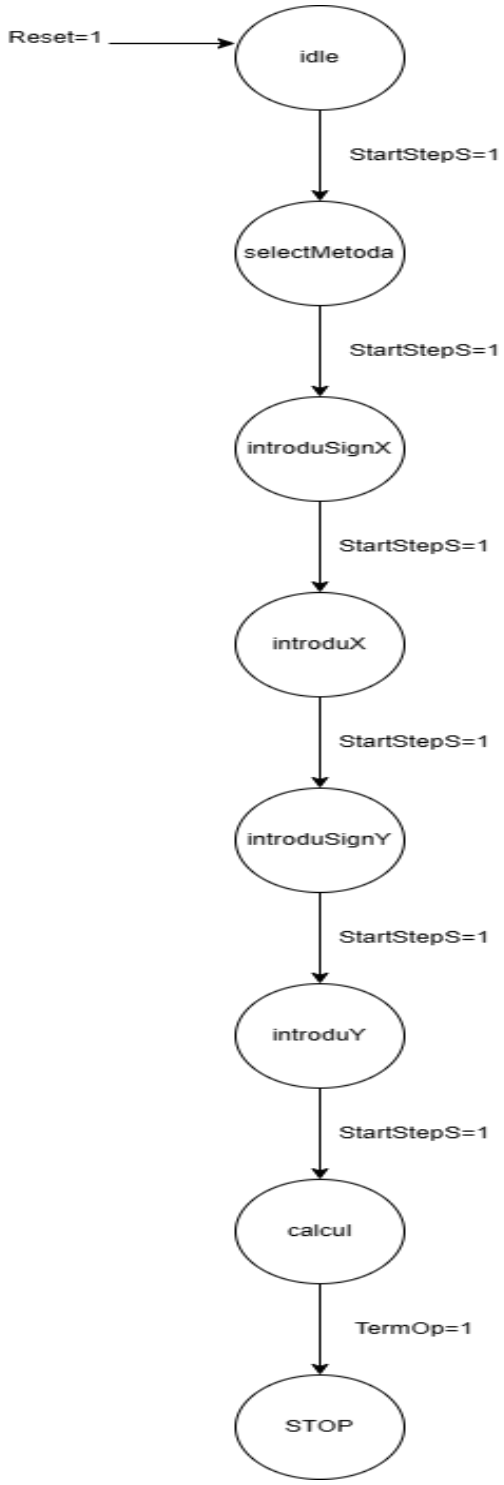
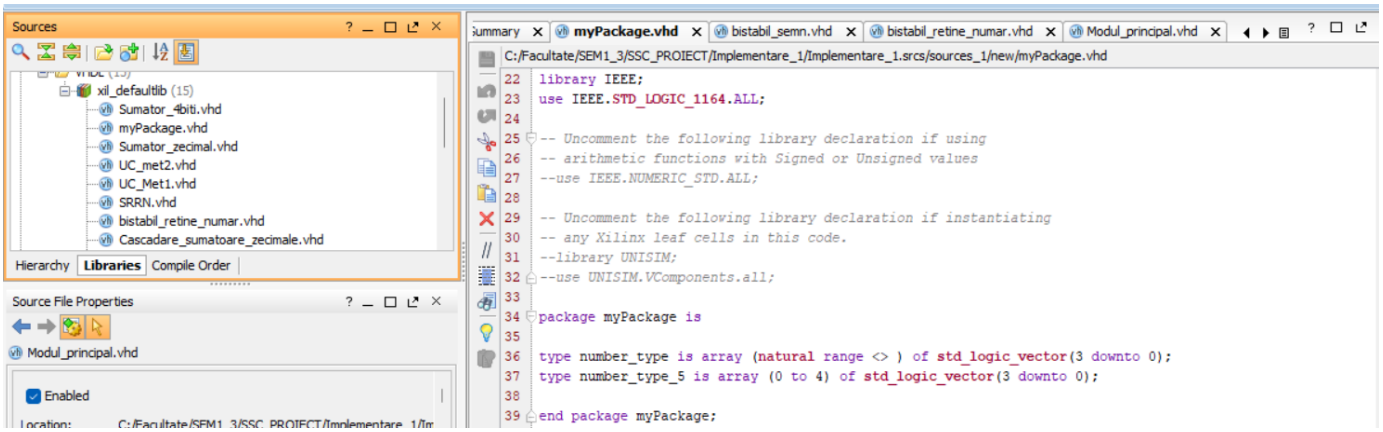


Fig.4: Diagrama de tranziții pentru FSM al sistemului general(facuta in acelasi program)

Avand in vedere organigrama lucrurile merg astfel:
whichState(6) inseamna alegem metoda
whichState(5) alegem semnul lui X (presupunand ca putem avea si numere poz si numere negative)
whichState(4) alegem numarul X
whichState(3) alegem semnul lui Y
whichState(2) alegem numarul Y
whichState(1) inseamna ca se calcueaza inmultirea . Practic se trece la modulul pt metoda 1 si 2 de inmultire
whichState(0) o sa ne anute daca s-a terminat inmultirea . Se afiseaza rezultatul pe SSD
Exemplu : in momentul in care whichState e de forma : "00000001" s-a ajung in STOP

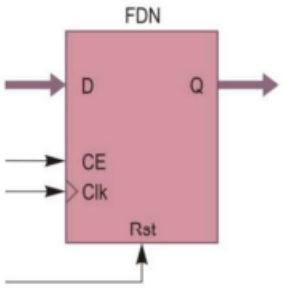
Definirea pachetului care va fi utilizat in majoritatea surselor noastre vhdl:



Alte componente necesare sunt :

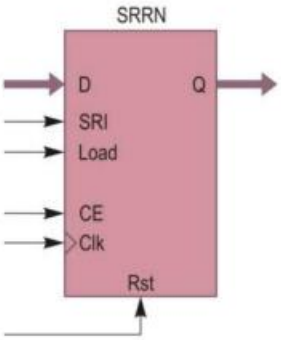
- **B – registru utilizat pentru stocarea deînmulțitului (pe n biti)**

Insa nu vom declara D ca fiind de tip std_logic_vector ci il vom declara de tipul definit in pachetul nosstru , adica number_type (atuci cand vom vrea sa fie in fapt de **16 biti , n va fi 3** si astfell vom pastra in D(0) <-cifra unitatilor , D(1) <-cifra zecilor, D(2) <-cifra sutelor si D(3) <-cifra miilor



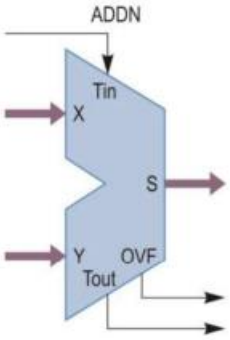
- **A și Q – registre de deplasare cu încarcare paralelă și intrare serială utilizate pentru stocarea produsului parțial**

Definite similar registrului de mai sus , acesta vor avea D de tipul number_type.
Pentru A n=4 (ca sa putem vedea si cifra zecilor de mii) , in timp ce pentru Q n se pastreaza 3



- **Sumator zecimal pentru numere zecimale de 5 cifre utilizat pentru calculul produsului parțial**

Acest sumator ne permite să calculăm suma a două numere de câte 4 cifre, fiind construit prin interconectarea a 4 sumatoare zecimale de o cifră.



Unde n=4 . Si cascadam cu port map uri 4 astfel de numaratoare carora le oferim pe rand X(3:0) pt unitate

- **Unitate de comandă pentru fiecare metoda in parte – automat cu stări finite care generează semnale responsabile pentru încarcare paralelă în registre, shiftare, terminare operație**

4.2. Metoda adunării repetate

Acest circuit este compus din mai multe elemente esențiale:

- Registru B: folosit pentru a stoca rezultatul înmulțirii
- Registrele A și Q: sunt registre de deplasare cu încărcare paralelă și intrare serială utilizate pentru a stoca rezultatul parțial al înmulțirii
- Sumator zecimal: conceput pentru a procesa numere zecimale de 5 cifre și a calcula rezultatul parțial al înmulțirii
- Unitatea de comandă: este un automat cu stări finite responsabil pentru generarea semnalelor necesare pentru operațiile de încărcare paralelă în registre, deplasare și terminare a operației

Deoarece în circuit se înmulțesc două numere zecimale de 4 cifre, registrele B (pentru stocarea rezultatului) și Q (pentru rezultatul parțial) vor avea n=3 (adica vor fi de 4*e biti definite).Inițial, în registrul Q este stocat multiplicatorul. După evaluarea unei cifre din multiplicator, conținutul registrului este deplasat spre dreapta. Registrul acumulator (A) va avea nevoie de n=4(adica 5*4 biti), deoarece pozitia D(4) conține transportul, exprimat ca cifră zecimală, rezultat în urma adunărilor repetate ale rezultatului înmulțirii. Această reprezentare este necesară deoarece adunările repetate pot genera un overflow mai mare decât 1. De exemplu, adunarea repetată a valorii 9999 de 5 ori este 49995, care trebuie reprezentată pe array de 5 std_logic_vector. În acest scop, sumatorul zecimal este utilizat pentru adunarea a două numere zecimale de 5 cifre, unde unul dintre operandi este registrul Acumulator, iar celălalt este rezultatul înmulțirii, completat în față cu cifra 0.

4.3. Metoda celor nouă multipli ai deînmulţitului

Această metodă implică generarea celor 9 multipli ai multiplicatorului și stocarea acestora în registre specializate. Aceste registre sunt au n=4, echivalând cu 5 cifre zecimale. Sumatorul zecimal este utilizat pentru adunarea a două numere de 5 cifre, deoarece unul dintre operanzi provine din aceste registre menționate anterior, iar celălalt vine din registrul acumulator. Registrul acumulator este un **registru de deplasare dreapta**, utilizând încărcare paralelă sincronizată și resetare sincronizată. Aici sunt stocate produsele parțiale și, la final, împreună cu registrul Q, formează rezultatul înmulțirii. Registrul Acumulator are n=4(adica 5*4 biti), având o cifră zecimală adăugată pentru a stoca transportul de la sumatorul zecimal. În rezultatul final, primele 4 poziții din registrul acumulator nu sunt luate în considerare. Registrul Q este un registru de deplasare dreapta, utilizând încărcare paralelă și resetare sincronizată. Inițial conține multiplicatorul și la fiecare iterație este deplasat spre dreapta împreună cu registrul acumulator, după evaluarea unei cifre din multiplicator și realizarea operațiilor de adunare la produsul parțial.

4.4. Manual de utilizare

Avem 2 butoane folosite : pentru a trece un pas următor, respectiv pentru a reseta circuitul.

Pentru **a reseta** circuitul apasam butonul **BTNQ(P18)** .

Pentru **a parcurge fiecare pas al organigramei de mai sus** (idle , selectMetoda etc) apasam butnonul **BTNR(M17)** .

Ca sa probam daca ne aflam unde trebuie , putem urmari ledurile.

Codificare leduri:

- Led 15 – este aprins dacă operația de înmulțire a luat sfârșit
- Led 14 ne spune ce metoda am selectat (0 daca este prima metoda de adunari repetate sau 1 daca este a doua cu multiplii;
- Led 13 afiseaza semnul rezultatului (pentru ca putem avea rezultate pozitive (este stins-0) sau negative (este aprins-1));
- Led 10 afiseaza semnul introdus pentru X (aceasi codificare ca cea pentru semnul rezultatului general;
- Led 9 afiseaza semnul introdus pentru Y (aceasi codificare ca cea pentru semnul rezultatului general;
- Led 8 afiseaza eroarea introducerii lui X (daca exista)
- Led 7 afiseaza eroarea introducerii lui Y (daca exista)
- Ledurile de la 0 pana la 6 ne arata whichState de la 1 la 7 si in functie de care dintre aceste leduri sunt aprinse , putem vedea in care stare a programului ne aflam

Cum sa incepi sa probezi programul:
Se **apasă butonul BTNR(M17).**
În acest moment, ne aflăm în starea de introducere a metodei. În această etapă, ajutorul **switchului 15,** se va alege metoda (0 pentru metoda adunarii repetate sau 1 pentru metoda celor nouă multipli ai deînmulţitului).
După selecția metodei, se apasă butonul BTNR pentru trecerea la următoarea etapă.
In mod similar , se introduc pe rand , respectand **ordinea din fsm ul de mai sus** , toate informatiile necesare efectuarii inmultirii .
In partea urmatoare a documentatiei am sa atasez poze , pentru a putea vizualiza mai bine modul in care acestea functioneaza.

5. Rezultate experimentale

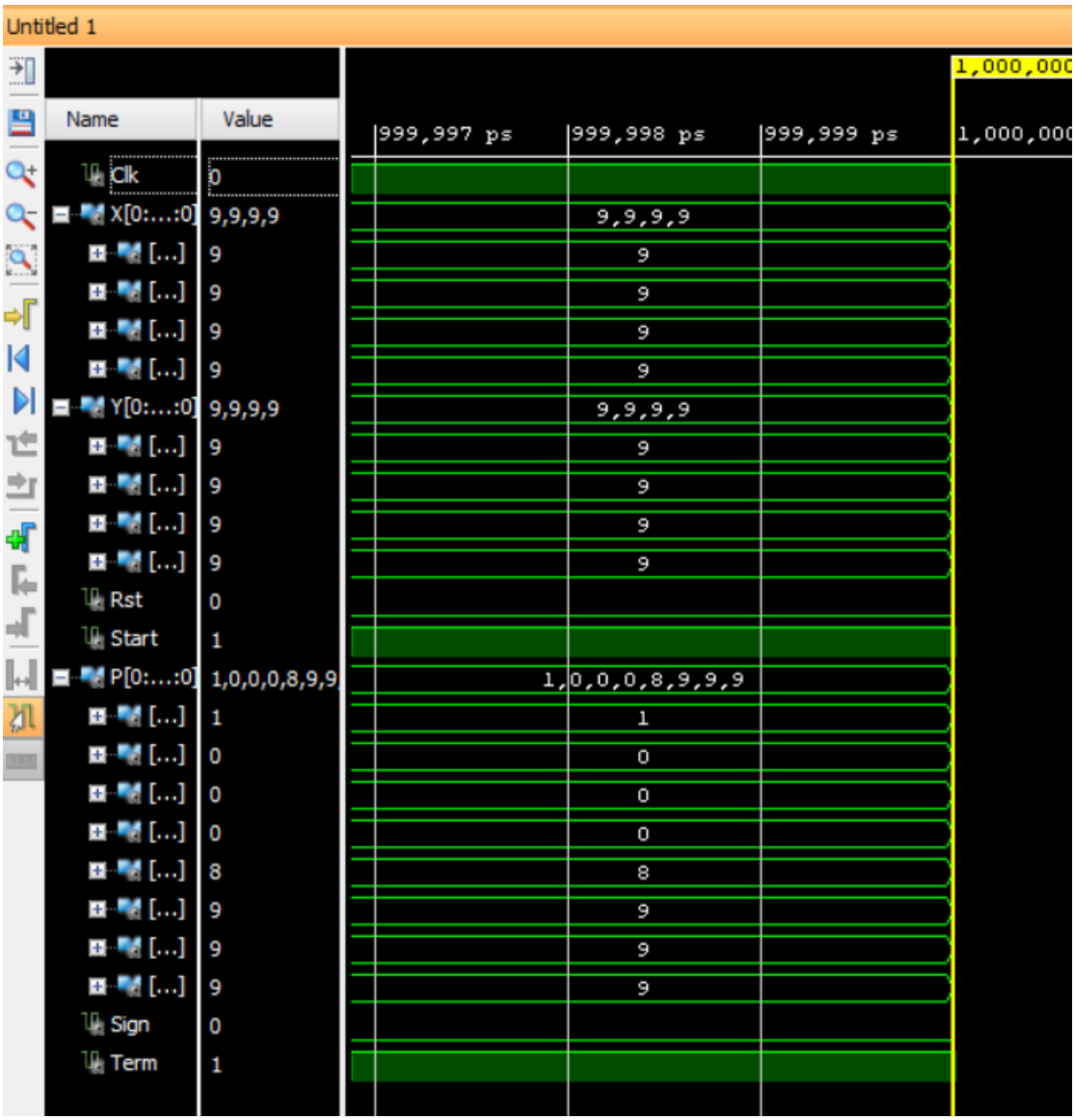
Am sa atasez initial rezultatele test banchurilor atat pentru metoda 1 cat si pentru metoda 2 , urmand ca dupa aceea sa atasez pozele cu probarea unei astfel de metode si pe placuta de la laboratorul de proiect .

Metoda 1 :

- Ce probam -> 9999*9999 (proba care s-a facut si pe placuta la laborator)

Rezultatul trebuie sa fie **99980001**

- Cum o sa ne arate rezultatul si de ce **-> 1,0,0,0,8,9,9,9 .** Dar in fapt este corect
  Deoarece am decis sa pastram pe **pozitia 0** a array ului nostru **cifra unitatilor** , pe pozitia 1 cifra zecilor si asa mai departe . Noi in TB in felul acesta il vom vedea **(P(0),P(1),..P(7))**
  Pe placuta maparea sa este facuta corespunzator , astfel ca vom putea vedea rezultatul corect

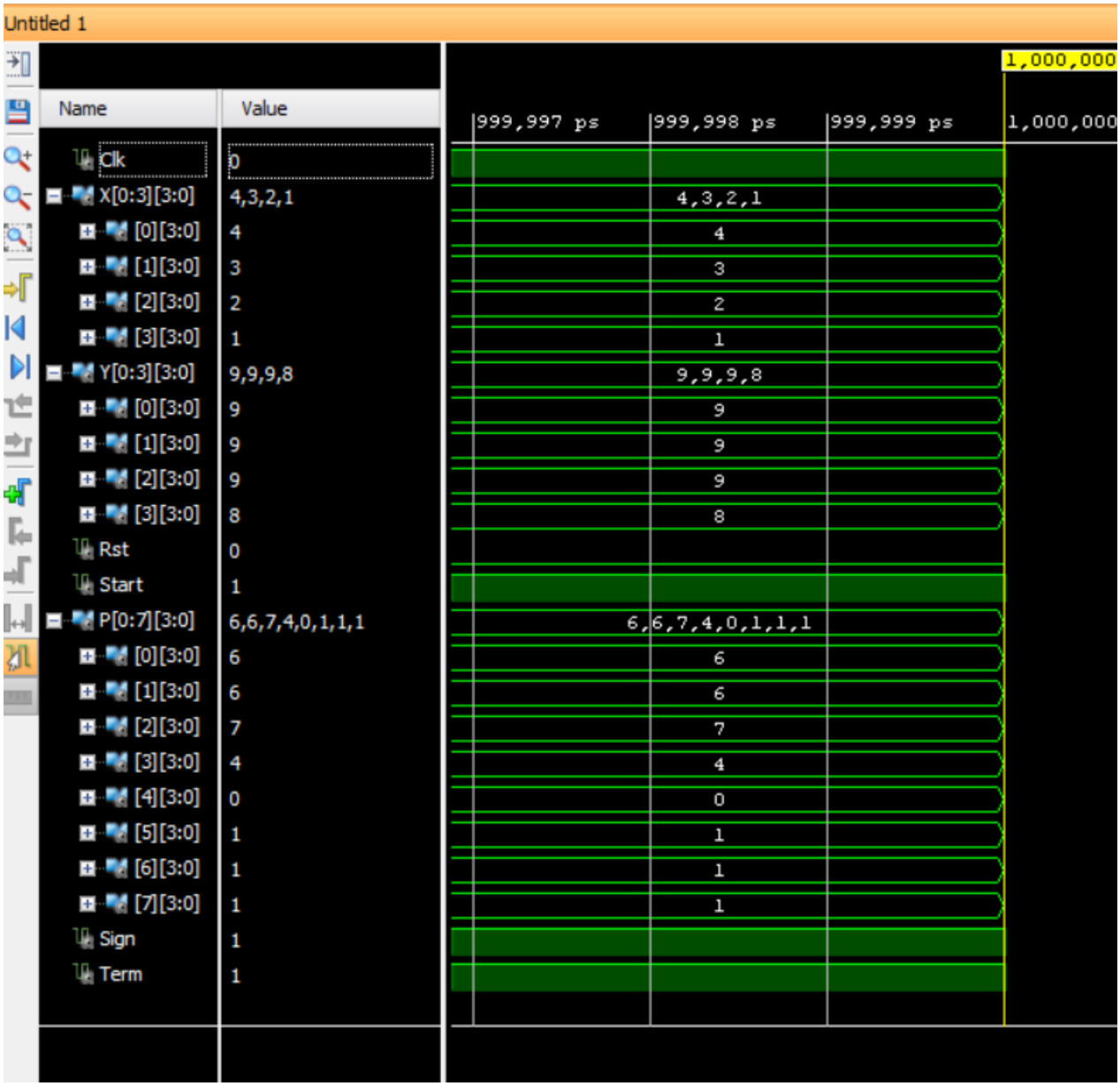


Metoda 2:

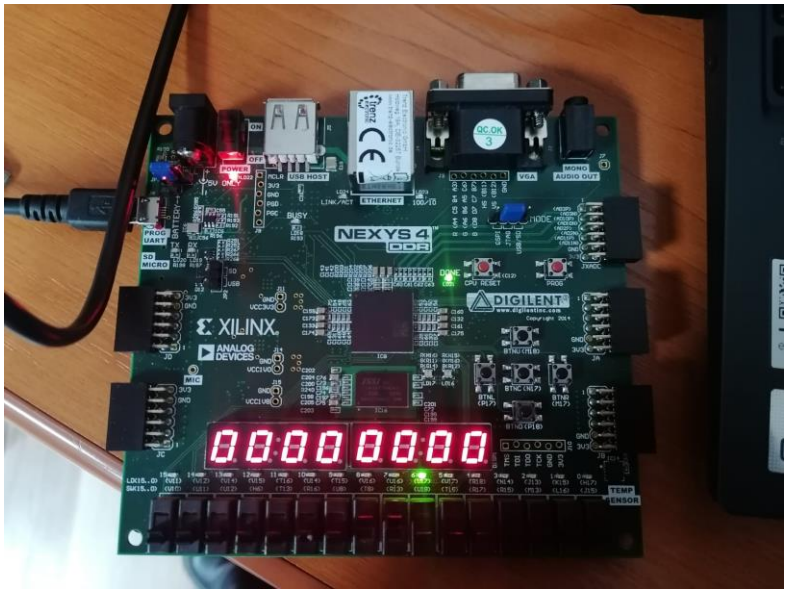
- Ce probam -> (-1234)*8999

Rezultatul trebuie sa fie **11104766**

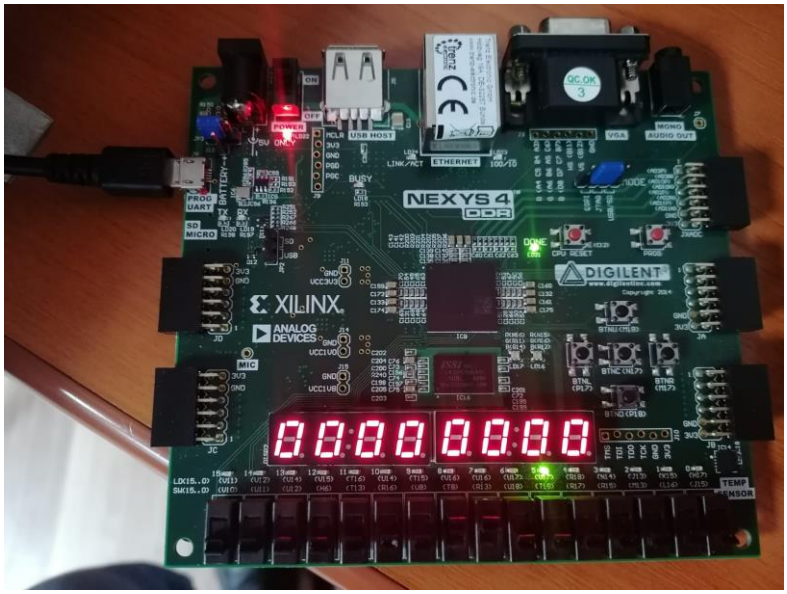
- Cum o sa arate numerele noastre in TB: X o sa fie de forma 4,3,2,1 si Y de forma 9,9,9,8
- Cum o sa ne arate rezultatul si de ce **-> 6,6,7,4,0,1,1,1 .** Dar in fapt este corect
  Deoarece am decis sa pastram pe **pozitia 0** a array ului nostru **cifra unitatilor** , pe pozitia 1 cifra zecilor si asa mai departe . Noi in TB in felul acesta il vom vedea **(P(0),P(1),..P(7))**
  Pe placuta maparea sa este facuta corespunzator , astfel ca vom putea vedea rezultatul corect
- Semnul nu mai este pe 0 de data aceasta , deaorece rezultatul nostru este negativ



Punerea pe placuta :
->Stare idle : codificare 1000000 (pe led avem in ordinea asta de la 6 la 0 : 100000)

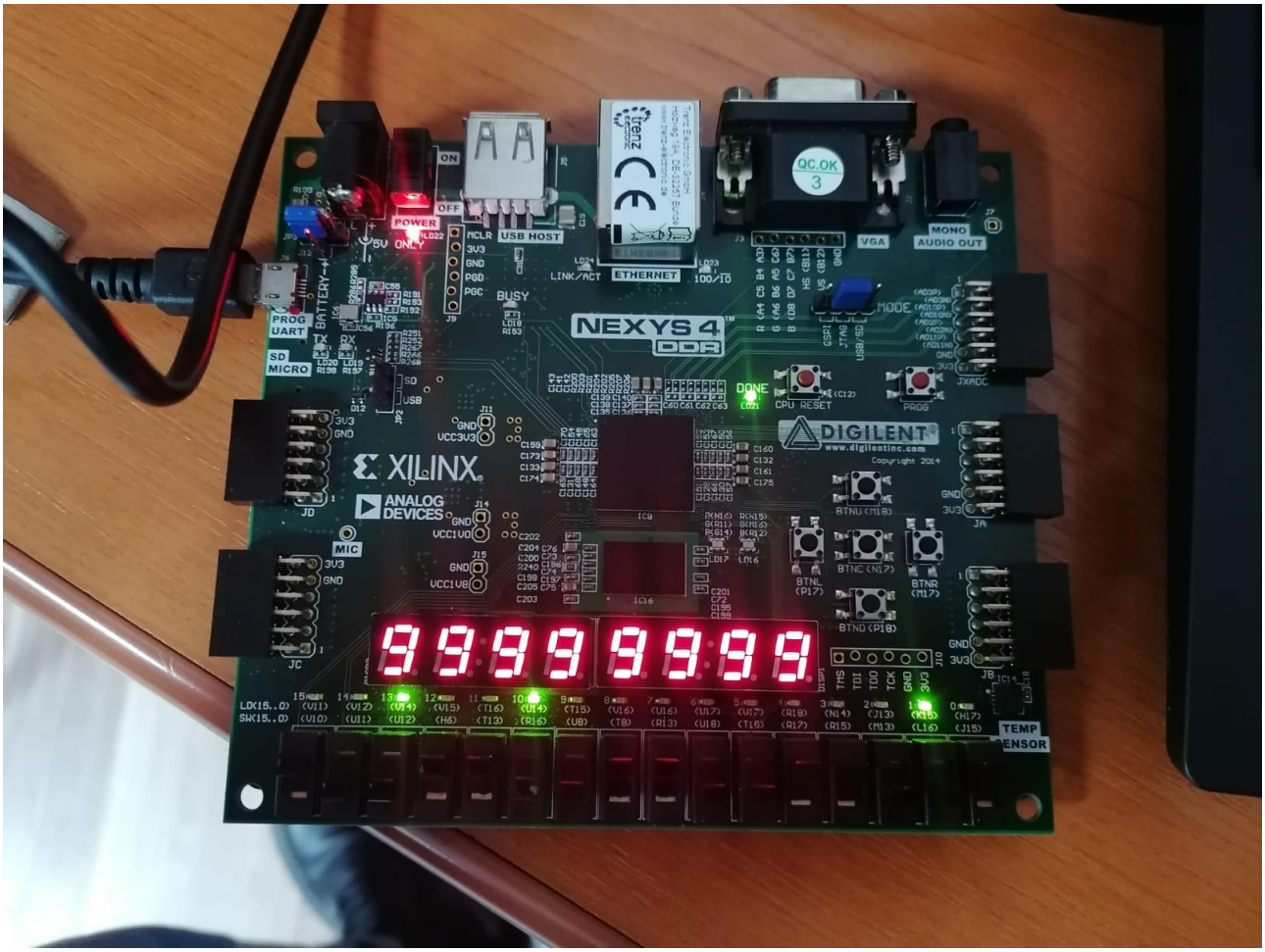


->Selectare metoda (de pe sw 15) , se poate veadea ca am ales metoda 0 (adunari repetate)

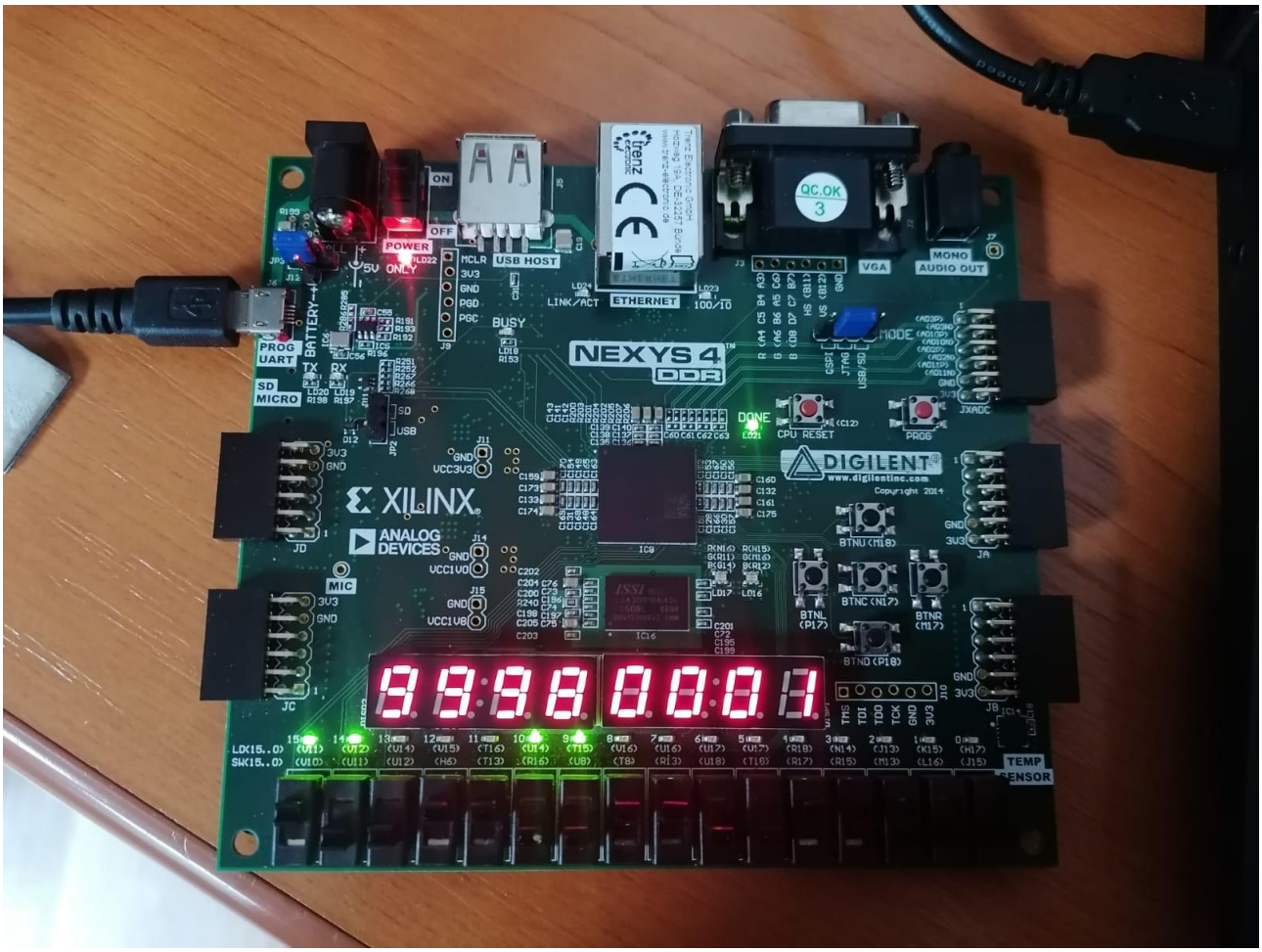


->Am sa sar etapa de selectare a semnului X , deoarece putem vedea in punctul in care introducem X si codificare semnului sau (led 10). De asemenean codificarea whichState se schimba

->Sar din nou etapa de introducere semn Y , pentru ca se poate vedea semnul codificat (led9) si putem vedea de asemenea deja semnul rezultatului final(led13).



->Rezultatul final



6. Concluzii

Acest proiect a presupus **proiectarea și implementarea celor 2 circuite de înmulțire zecimală** pentru numere de patru cifre: **Metoda "Adunărilor repetate"** si **metoda "Celor 9 multiplii ai deînmulțitului"**. Totodată, a fost proiectat și implementat pe placa de dezvoltare FPGA Nexys 4 DDR un sistem care înglobează toate cele 2 metode, prin care un utilizator este capabil să aleagă ce metodă să folosească, să introducă numere de 4 cifre și să observe rezultatele obșinute. Fiecare metodă din cele enumerate au fost implementate utilizând o descriere structurală cu ajutorul limbajului de descriere hardware VHDL și a mediului de dezvoltare oferit de aplicația Vivado Design Suite. Pentru fiecare metodă, a fost concepută **o unitate de comandă proprie**, un

circuit esențial în defășurarea sistemului. Fiecare unitate de comandă reprezenta un autmoat de stări finite, fiecare stare fiind reprezentativă pentru o etapă a algoritmului prin semnalele de ieșire active. Pe lângă unitatea de comandă, fiecare circuit are în componență regiștri speciali, de tip accumulator și pentru stocarea deînmulțitului, înmulțitorului, capabili să fie deplasați la dreapta pentru calcularea produselor parțiale la fiecare pas. Un circuit care a avut un rol esențial în rezolvarea acestor cerințe a fost sumatorul zecimal pentru numere de mai multe cifre. Acesta a fost construit prin cascadarea mai multor sumatoare zecimale pentru cifre zecimale.

7.Bibliografie

[1] Rekha K. James, K. Pouluse Jacob, Sreela Sasi , "Decimal Floating Point Multiplication using RPS Algorithm" în International Conference on VLSI, Communication & Instrumentation(ICVCI) 2011. Proceedings published by International Journal of Computer Applications (IJCA). (link: https://www.ijcaonline.org/icvci/number7/icvci1349.pdf)
[2] G. Jaberipur, A. Kaivani, "Binary-coded decimal digit multipliers" în IET Comput. Digit. Tech., Vol. 1, No. 4, July 2007 (link: https://pdfs.semanticscholar.org/8719/ed34825f114e30044cdc676d26997cb80993.pdf)
[3] Baruch Z.F., "Structure of Computer Systems with Applications", U.T. Press 2003