

Unitate de calcul în virgulă mobilă: adunarea si scaderea

Muntean Diana-Gratiana

Pop Andreea-Gabriela

Grupa 30236

Profesor îndrumător: Lisman Dragos

Data: 15.01.2024

**Cuprins**

[**1. Rezumat** 3](#_Toc510273654)

[**2. Introducere** 3](#_Toc510273655)

[**3. Fundamentare teoretică** 5](#_Toc510273656)

[**3.1. Tehnologii utilizate** 5](#_Toc510273657)

[**3.1.1. Nexys4 DDR** 5](#_Toc510273660)

[**3.1.2. PmodKYPD** 5](#_Toc510273661)

[**3.2. Reprezentarea numerelor în virgulă mobilă** 5](#_Toc510273662)

[**3.3. Standardul IEEE 754** 6](#_Toc510273663)

**3.4. Adunarea si scaderea in virgula mobila** 8

[**3.4.1. Alinierea mantiselor** 8](#_Toc510273664)

[**3.4.2. Adunarea sau scaderea mantiselor** 8](#_Toc510273665)

[**3.4.3. Normalizarea rezultatului** 8](#_Toc510273666)

[**3.4.4. Rotunjirea rezultatului** 8](#_Toc510273667)

[**4. Proiectare şi implementare** 10](#_Toc510273668)

[**4.1. Schema bloc** 10](#_Toc510273669)

[**4.2. Descrie module** 1](#_Toc510273670)1

**4.2.1. CompExp** 11

**4.2.2. ShiftRight** 11

**4.2.3. Adder** 11

**4.2.3.1 Singout**  12

**4.2.3.2 FractionAdd** 12

**4.2.4. DepExpBlock** 13

**4.2.5. Normalizare** 13

**4.2.5.1. CountLeadingZero** 14

**4.2.5.2. ShiftLeft** 14

**4.2.6. UC** 14

**4.2.7. Main** 16

[**5. Rezultate experimentale** 17](#_Toc510273672)

[**6. Concluzii** 19](#_Toc510273673)

**7**. [**Bibliografie** 19](#_Toc510273674)

# **Rezumat**

Aritmetica în virgulă mobilă reprezintă una dintre cele mai frecvente metode de aproximare a numerelor reale pentru a permite efectuarea de calcule numerice pe calculatoarele moderne. Scopul acestui proiect constă în implementarea unui sumator/scăzător pentru numerele în virgulă mobilă, conform standardului IEEE 754 privind reprezentarea numerelor în virgulă mobilă. Acest standard definește trei formate: precizie simplă (32 de biți), precizie dublă (64 de biți) și precizie extinsă (80 de biți). În cadrul acestui proiect, am optat pentru formatul de precizie simplă, implementat în limbajul VHDL.

Pentru testarea funcționalității proiectului, am dezvoltat un "test bench" în care am efectuat diverse operații asupra a 17 perechi de numere diferite, permițând observarea rezultatelor obținute. În diagrama de undă ("waveform"), am inclus și stările intermediare pentru a evidenția întregul proces prin care trec aceste numere în funcție de formatul în care sunt reprezentate. În concluzie, am reușit să implementez cu succes acest sumator/scăzător pentru numere în virgulă mobilă. Rezultatele obținute în cadrul testului reflectă conformitatea proiectului cu obiectivele inițiale.

# **2. Introducere**

Tema propusă spre implementare este cea de adunare/scadere a numerelor reprezentate în virgulă mobilă prin standardul IEEE 754. Majoritatea unităţilor centrale de procesare lucrează cu două tipuri de numere: în virgulă fixă şi în virgulă mobilă[1] . Acest lucru este datorat faptului că nu toate informațiiile necesară UCP-ului pentru a-și realiza operațiile specifice sunt numere întregi. Reprezentarea internă în calculator a datelor este realizată printr-un format în virgulă mobilă și pentru asigurarea portabilității programelor între calculatoare diferite, această reprezentare internă respecta un standard(IEEE 754) utilizat de marea majoritate a unităților de calcul și a coprocesoarelor matematice.

Multe aplicații necesită numere care nu sunt întregi. Există mai multe posibilități pentru reprezentarea acestor numere. Una dintre ele este reprezentarea în *virgulă fixă*. În acest caz, se poate utiliza aritmetica pentru numere întregi, plasând apoi virgula binară în poziția predefinită, de exemplu, după bitul de semn. În cazul reprezentării numerelor in acest format, deși virgula nu este reprezentată fizic în calculator, poziția virgulei binare, care este stabilită la proiectare, nu mai poate fi schimbată[2]. Astfel, este necesară transformarea tuturor numerelor în acest format, lucru ce necesită o serie de operații de de scalare sau deplasare, atașând numerelor *factori de scală.* Evidența acestora trebuie realizată prin program, ceea ce mărește timpul de calcul.

O solutie la problemele aduse de reprezentarea numerelor în virgulă fixă este utilizarea unei tehnici de scalare automată, cunoscută sub numele de reprezentare în *virgulă mobilă*. În acest caz, factorul de scală devine o parte a cuvântului din calculator, poziția virgulei variind pentru fiecare număr în mod automat. Un număr reprezentat în virgulă mobilă are două componente: *mantisa(*M) și *exponentul(*E). Această reprezentare poate fi memorată într-un cuvânt cu trei câmpuri: semnul, mantisa și exponentul. În general, câmpul exponentului conține o valoare pozitivă, valoare obținută prin adunarea unui *deplasament*: în felul acesta, câmpul exponentului va avea întotdeauna o valoare pozitivă.

Reprezentarea numerelor în virgulă mobilă poate fi realizată în mai multe moduri prin diferite modalități de tratare a cazurilor de excepție, numărul de biți alocați fiecărui câmp în reprezentare, modul de rotunjire, etc. Din cauza multiplelor moduri de tratare a acestei reprezentări, programele scrise nu sunt portabile întrucât două calculatoare cu două moduri diferite de reprezentare vor realiza scrierea acestuia în moduri diferite.

Pentru a asigura portabilitatea programelor, *Societatea Calculatoarelor a IEEE(Institute of Electrical and Electronics Engineers* a elaborat un standard pentru reprezentarea numerelor în virgulă mobilă și pentru operații aritmetice în virgulă mobilă. Acest standard prevede o metodă de calcul cu numere în virgulă mobilă, care va produce același rezultat dacă prelucrarea se face în hardware, software sau o combinație a celor două. Rezultatele vor fi identice, independent de punere în aplicare, având în vedere aceleași date de intrare. De asemenea, erorile și condițiile de eroare  
produse în operații matematice vor fi raportate într-o manieră consistentă, indiferent de  
modul de implementare.

O problemă întâlnită la calculele în acest mod de reprezentare este modul de tratare a depășirilor inferioare și superioare. Depășirea superioară apare în momentul în care un exponent depășește valoarea maximă, iar depășirea inferioară apare atunci când exponentul are o valoare mai mică decât cea minimă.

Problema propusă spre soluționare, operația de adunare/scadere a numerelor în virgula mobilă, presupune [3] :

1. Alinierea mantiselor
2. Adunarea sau scaderea mantiselor
3. Normalizarea rezultatului
4. Rotunjirea rezultatului

Obiectivele principale ale acestui proiect sunt: realizarea unei reprezentări ale numerelor corespunzătoare cu formatul IEEE 754 pe 32 biți, implementarea algoritmului de adunare/scadere corespunzător pentru numere reprezentate în virgulă mobilă, realizarea operației de rotunjire pentru obținerea rezultatelor cu o precizie cât mai mare și tratarea cazurilor excepționale ce pot apărea în cadrul acestei operații (depășire inferioară, depășire superioară, rezultat inexact).

În cele ce urmează, capitolul “Fundamentare teoretică” va conține informații mai detaliate referitoare la formatul standard IEEE 754 pe 32 biți și modul de realizare a operatiilor de adunare/scadere pentru numere reprezentate în acest format.

# **3. Fundamentare teoretică**

## **3.1. Tehnologii utilizate**

### **3.1.1. Nexys4 DDR**

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. [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.

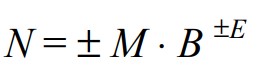
### **3.1.2. PmodKYPD**

PmodKYPD este o tastatură de 16 butoane aranjate într-un format hexadecimal (0-F). Aducand pe rand fiecare coloana a tastaturii la un nivel logic scazut(logic level low voltage), utilizatorii pot citi nivelul logic al fiecarei linii pentru a determina care buton(daca exista) este apasat.

Placa gazdă comunică cu tastatura prin intermediul protocolului GPIO [5] .Această tastatură va fi utilizată pentru introducerea operanzilor în virgulă mobilă ai operației de adunare/scadere și va facilita utilizarea unitatii de calcul incarcata pe placa Nexys4 DDR, permitand introducerea operanzilor intr-un format zecimal si nu unul binar.

## **3.2. Reprezentarea numerelor in virgula mobila**[6]

Un număr N poate fi reprezentat în sistemul de virgulă mobilă (VM) conform următoarei formule:



Un număr reprezentat în virgulă mobilă constă din două componente. Prima componentă este mantisa (M), care indică valoarea exactă a numărului într-un anumit domeniu și este reprezentată de obicei sub formă de număr fracționar cu semn. A doua componentă este exponentul (E), care indică ordinul de mărime al numărului în sistemul de virgulă mobilă. În expresia de mai sus, B reprezintă baza exponentului.

Această reprezentare poate fi memorată într-un cuvânt binar cu trei câmpuri: semnul, mantisa şi exponentul. De exemplu, presupunând un cuvânt de 32 de biţi, o asignare posibilă a biţilor la fiecare câmp poate fi următoarea:

Shape, rectangle

Description automatically generated

De obicei, câmpul rezervat exponentului nu conţine exponentul real, ci o valoare numită caracteristică, care se obţine prin adunarea unui deplasament la exponent, astfel încât să rezulte întotdeauna o valoare pozitivă. Astfel, nu este necesar să se rezerve un câmp separat pentru semnul exponentului.

Caracteristica C este deci exponentul deplasat:

C = E + deplasament; deplasament = 127

Unul din avantajele utilizării exponentului deplasat constă în simplificarea operaţiilor executate cu exponentul, datorită lipsei exponenţilor negativi.

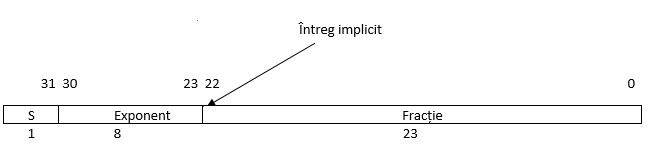
Pentru simplificarea operaţiilor cu numere în VM şi pentru creşterea preciziei acestora, se utilizează reprezentarea sub forma normalizată. Un număr în VM este normalizat dacă din stanga virgulei este 1. Deoarece bitul din fata virgulei al unui număr normalizat în VM este întotdeauna 1, acest bit nu este de obicei memorat, fiind un bit ascuns la dreapta virgulei binare. Numarul normalizat 1.75 va avea urmatoarea forma:



## **3.3. Standardul IEEE 754**

Reprezentare numerelor in formatul IEEE 754 poate fi memorată într-un cuvânt binar cu trei câmpuri: semn, mantisă și exponent [2]. În cele ce urmează, va fi detaliat modul de reprezentare, intervalul numerelor definite și modul de tratare a unor condiții speciale și de eroare pentru formatul IEEE 754 pe 32 de biți.

În cazul reprezentării pe 32 de biți, standardul impune ca dimensiunea celor trei câmpuri componente ale reprezentării interne să fie următoarea: un bit pentru semn, 8 biți pentru exponent și 23 biți pentru mantisă. Acest format este numit format cu *precizie simplă* și poate fi vizibil mai jos :



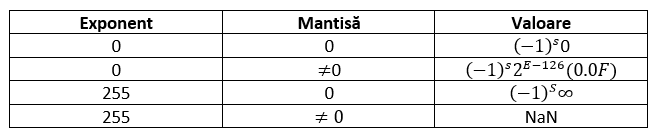
Aceasta este o reprezentare în mărime și semn, întrucât semnul are un câmp separat pentru reprezentare. Acest câmp este 0 pentru numerele pozitive și 1 pentru cele negative. Numerele reprezentate în acest format sunt normalizate, adică au bitul cel mai semnificativ al mantisei egal cu 1.

În formatul precizie simplă, este introdus conceptul de "bit ascuns," fixând o valoare pentru bitul cel mai semnificativ al mantisei, implicit cu valoarea 1, pentru a reprezenta partea întreagă a numărului. Această strategie îmbunătățește precizia operațiilor și reprezentărilor numerelor în virgulă mobilă, extinzând intervalul numeric de la 1,18\*și

3,4\*.

Deoarece valoarea exponentului poate sa fie pozitiva sau negativa in locul acesteia este folosita o valoare numita caracteristica, ce este obtinuta prin adaugarea unui deplasament de 127 la exponent, astfel ca intotdeauna va fi pozitiva.

Valoarea minimă și maximă ale acesteia sunt folosite pentru reprezentarea unor valori speciale. Mai jos sunt prezentate valorile speciale ale formatului pe 32 de biți.



Valoarea 0 va fi reprezentată prin atribuirea câmpului de exponent și cel al mantisei a valorii 0. În acest caz, bitul ascuns este implicit 0, și nu 1.

Formatul cu precizie simplă întâmpină probleme în reprezentarea numerelor care depășesc gama suportată de un cuvânt de 32 de biți, manifestându-se sub formă de depășire superioară și depășire inferioară. Depășirea superioară apare când exponentul depășește valoarea maximă (127), iar depășirea inferioară apare atunci când exponentul este mai mic decât valoarea minimă (-126).

În cazul unei depășiri inferioare, standardul permite utilizarea numere care nu sunt normalizate, numite *numere denormalizate.* Acestea sunt generate printr-o tehnică care constă în deplasarea mantisei la dreapta și incrementarea exponentului până când acesta ajunge la valoarea minimă permisă, tehnică numită *depășire inferioară graduală.*

În cazul unei depășiri superioare, există o reprezentare specială pentru infinit. Aceasta constă din atribuirea valorii maxime exponentului și valorii 0 mantisei. Bitul de semn va distinge în acest caz, reprezentarea între . Utilizatorul va putea decide dacă depășirea superioară va fi tratată ca și o condiție de eroare sau va continua calculele cu valoarea infinit.

O altă valoare specială a acestui format, este valoarea NaN. Aceasta va fi utilizată pentru a indica diferite condiții de excepție.

## **3.4** **Adunarea si scaderea in virgula mobila [7]**

Adunarea şi scăderea în VM sunt destul de complexe, deoarece pentru adunarea sau scăderea corectă a celor două numere, trebuie să se realizeze egalizarea exponenţilor acestora. Aceasta implică compararea mărimii exponenţilor şi apoi alinierea mantisei numărului cu exponentul mai mic. Proiectul este impartit pe mai multe nivele, fiecare nivel avand rolul lui pentru atingerea obiectivului. Nivelele amintite sunt urmatoarele:

3.4.1. Alinierea mantiselor

3.4.2. Adunarea sau scaderea mantiselor

3.4.3. Normalizarea rezultatului

3.4.4. Rotunjirea rezultatului

3.4.1. Alinierea mantiselor

In aceasta etapa se compara exponentii celor doua numere, si se deplaseaza spre dreapta mantisa care are exponentul mai mic.

Compararea exponentilor se face prin diferenta acestora. Daca rezultatul este negativ, atunci exponentul al doilea este mai mare, in caz contrar, este mai mare primul exponent. Deplasarea la dreapta se face cu un numar de biti egal cu diferenta dintre exponenti.

3.4.2 Adunarea sau scaderea mantiselor

In aceasta etapa se aduna sau se scad mantisele in functie de operatia selectata de catre utlizator.

3.4.3 Normalizarea rezultatului

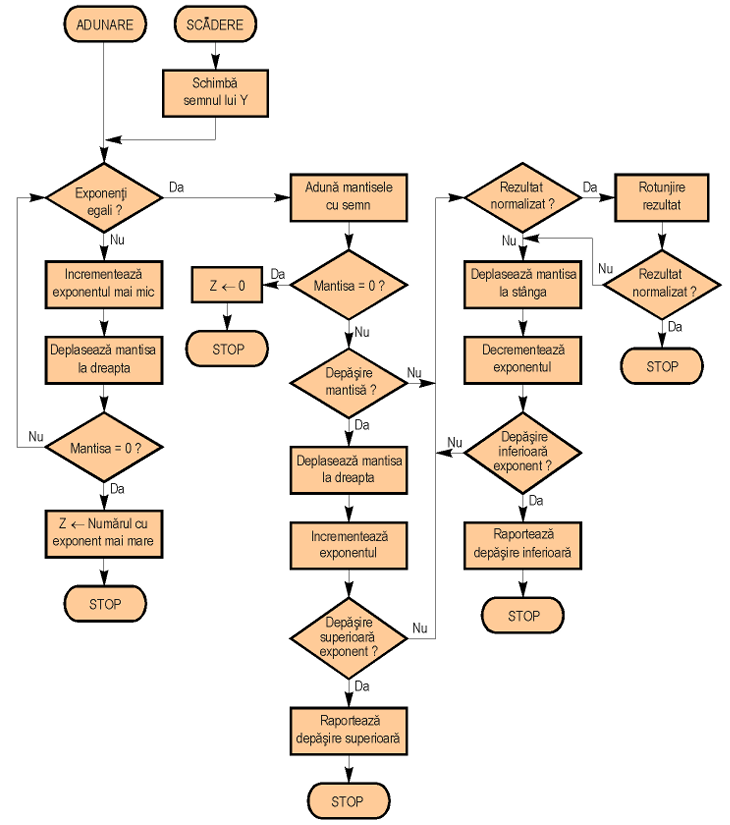
Dupa adunarea mantiselor, se verifica daca rezultatul este normalizat sau nu. Daca nu este normalizat, atunci urmeaza sa fie deplasat spre stanga pana cand primul bit inaintea virgulei devine egal cu 1.

3.4.4 Rotunjirea rezultatului

In cadrul proiectului am folosit trunchierea.

Organigrama proiectului:

-este preluata din [7]:

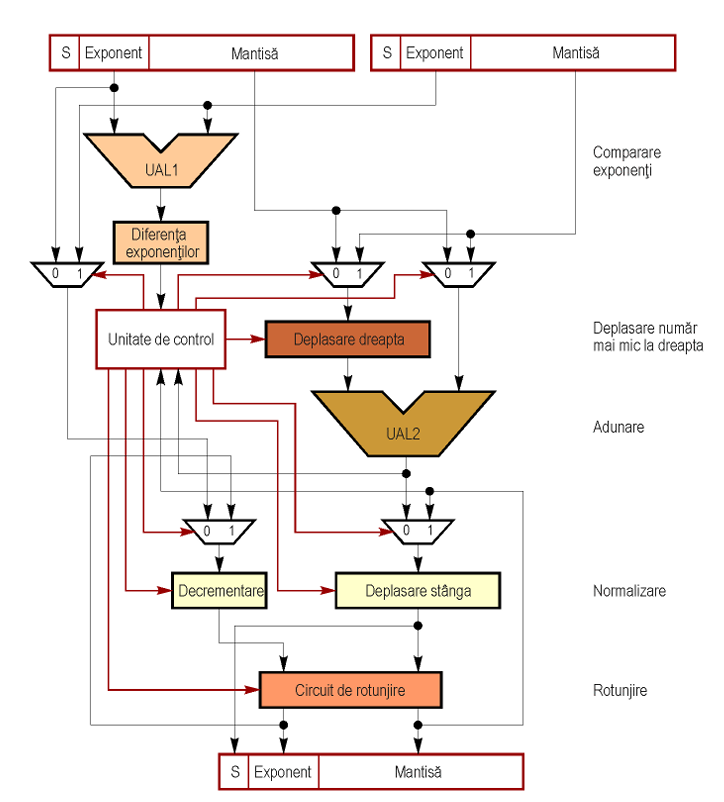


.

# **4.** **Proiectare şi implementare**

Proiectul a fost realizat in limbaj VHDL, iar formatul utilizat este simpla precizie.

## **4.1. Schema bloc**



Intrarea de ceas a registrului este semnalul Clk, intrarea de date este semnalul tip vector D de n biti, iar iesirea registrului este semnalul tip vector Q. Dacă semnalul Rst este 1 logic, registrul este resetat în mod sincron, ieșirile sale fiind resetate la 0 logic. Dacă semnalul CE (Clock Enable) este 1 logic, la frontul crescător al semnalului de ceas se realizează încărcarea paralelă a registrului cu datele aplicate la intrare.

## **4.2. Descrie module**

**4.2.1. CompExp**

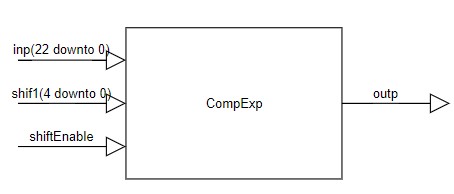
Acest modul realizeaza compararea celor 2 exponenenti.

A picture containing chart

Description automatically generated

Modulul returneaza diferenta exponentilor si Comp. Comp este o valoare pe 2 biti, si are valoarea “00” cand expA>expB, “01” cand expA<expB si “10” pentru cazul in care sunt egali exponentii.

* + 1. **ShiftRight**



(

22

downto

0

)

Modulul shifteaza la dreapta un numar pe 23 de biti, prin utilizarea unui case in fuctie de shift1. Primul bit ce este adaugat in fata este 1, restul sunt 0, deoarece in fata virgulei avem un 1. Intrarea inp este mantisa care trebuie shiftata, shift1 reprezinta numarul de biti cu care vrem sa shiftam.

* + 1. **Adder**

Acest modul realizeaza adunarea/scaderea mantiselor si calcularea semnului pentru numarul rezultat.

Diagram

Description automatically generated

Contine doua alte module: SignOut si FractionAdd.

**4.2.3.1 Singout**

Acest modul determina semnul operatiei, semnul numarului ce are exponentul mai mic si semnul numarului rezultat.

Diagram

Description automatically generated

**4.2.3.2 FractionAdd**

Acest modul realizeaza operatia propriu-zisa, uitilizand sumatoare elementare.

Diagram

Description automatically generated

## **4.2.4. DepExpBlock**

Acest modul este folosit pentru cazul in care acel 1 din fata virgulei se deplaseaza la stanga (ne dam seama cu ajutorul bitilor p si Cout, iesirile de la Adder, unde p= primul bit din fata virgulei, Cout= al doilea bit din fata virgulei). Modulul consta in incrementarea exponentului si in deplasarea mantisei la dreapta cu un bit de valoare p.

Diagram

Description automatically generated

# **Normalizare**

Acest modul realizeaza normalizarea rezultatului prin decrementarea exponentului si prin shiftarea mantisei la stanga pana ajunge 1 inaintea virgulei.

Diagram

Description automatically generated

Acest module contine alte 2 module: CountLeadingZero si ShiftLeft

* + - 1. **CountLeadingZero**

Acest modul Numara bitii de zero de la stanga mantisei pana la intalnirea primului bit de 1.

Diagram

Description automatically generated

* + - 1. **ShiftLeft**

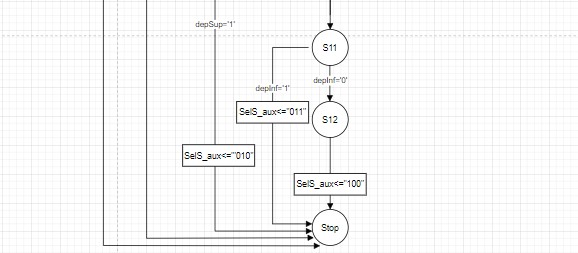
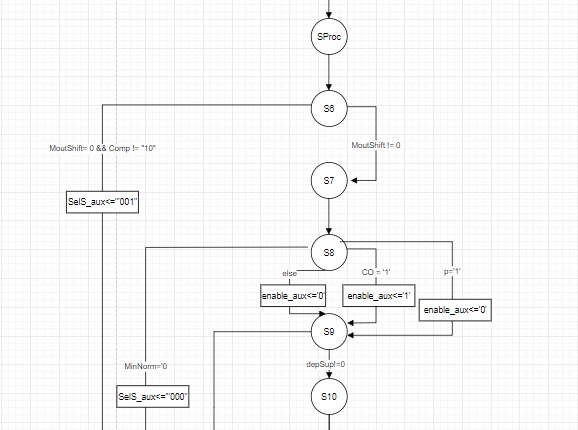
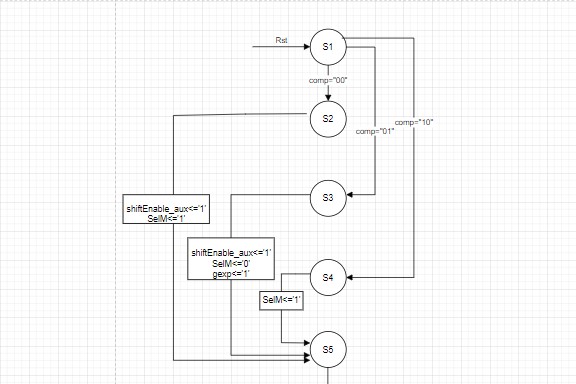
Acest modul realizeaza shiftarea mantisei la stanga cu un numar dat de biti.

A picture containing chart

Description automatically generated

* + 1. **UC**

Acest modul implementeaza urmatorul aparat cu stari finite(dupa principiul organigramei):



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1 | S2 | S3 | S4 | S5 | SProc | S6 | S7 | S8 | S9 | S10 | S11 | S12 | Stop |
| SelExp |  |  |  |  |  |  | gexp | | |  |  |  |  |  |
| SelMantisa |  |  |  |  |  |  | SelM | | |  |  |  |  |  |
| enableLoad | ‘0’ | ‘1’ | ‘1’ | ‘1’ | ‘1’ | ‘1’ | ‘0’ | ‘0’ | ‘0’ | ‘0 | ‘0’ | ‘0’ | ‘0’ | ‘0’ |
| enable |  |  |  |  |  |  | enable\_aux | | |  |  |  |  |  |
| SelS | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 | Sels\_aux |
| Term | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘1’ |
| shiftEnable |  |  |  |  |  |  | shiftEnable\_aux | | |  |  |  |  |  |
| normEnable | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘1’ | ‘1’ | ‘1’ | ‘1’ |
| EnableMoutShift | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘1’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ | ‘0’ |

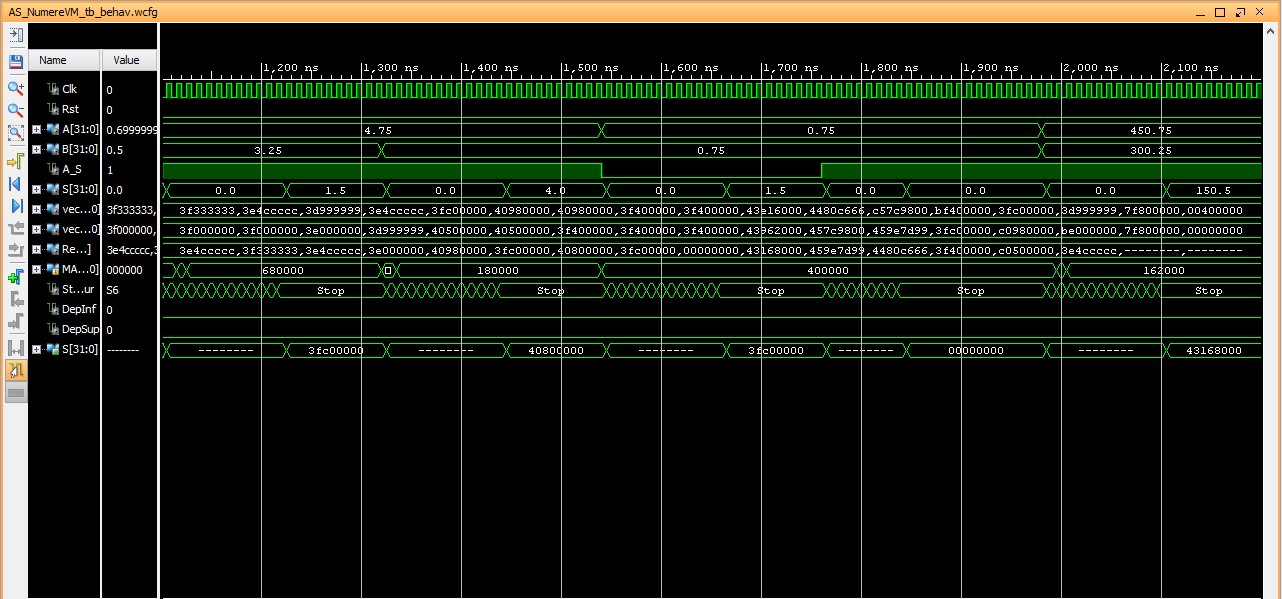
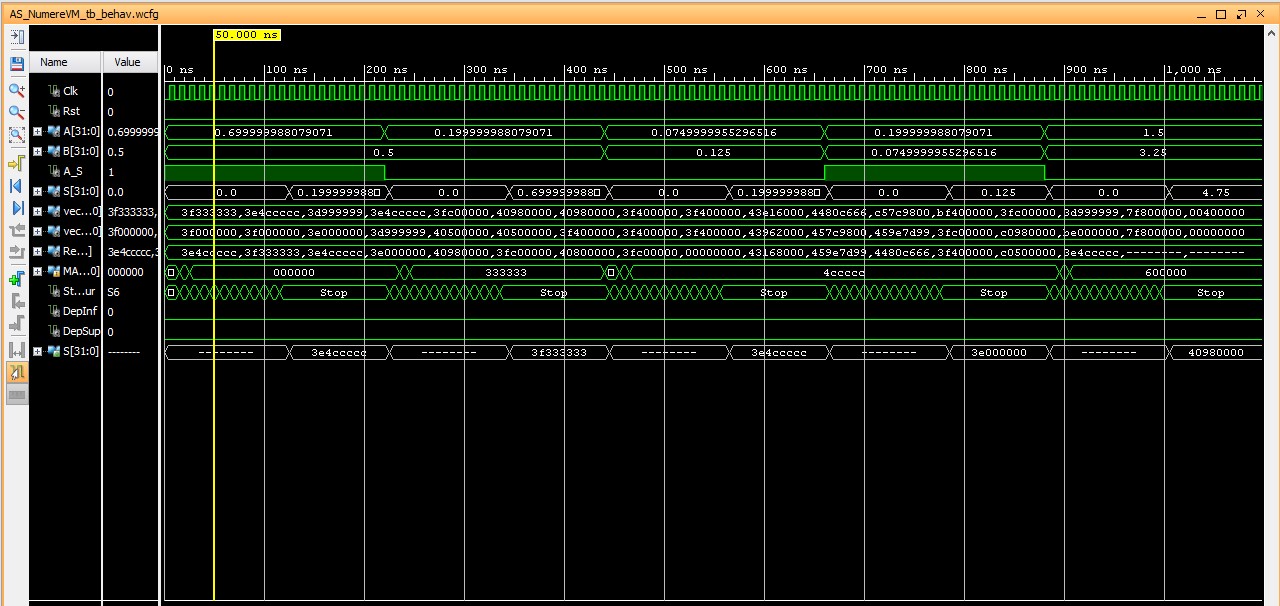
|  |  |
| --- | --- |
| S1 | Initializare + comparare exponenti |
| S2 | Tratare expA > expB |
| S3 | Tratare expA < expB |
| S4 | Tratare expA = expB |
| S5 | Shiftare dreapta |
| SProc | Procesare rezoltat shiftare |
| S6 | Decizie in functie de razultatul shiftarii |
| S7 | Adunare/Scadere |
| S8 | Decizie in functie de rezultatul adunarii |
| S9 | Verificare depasire superioara |
| S10 | Normalizare |
| S11 | Verificare depasire inferioara |
| S12 | Procesare rezultat final |
| Sop | Gata |

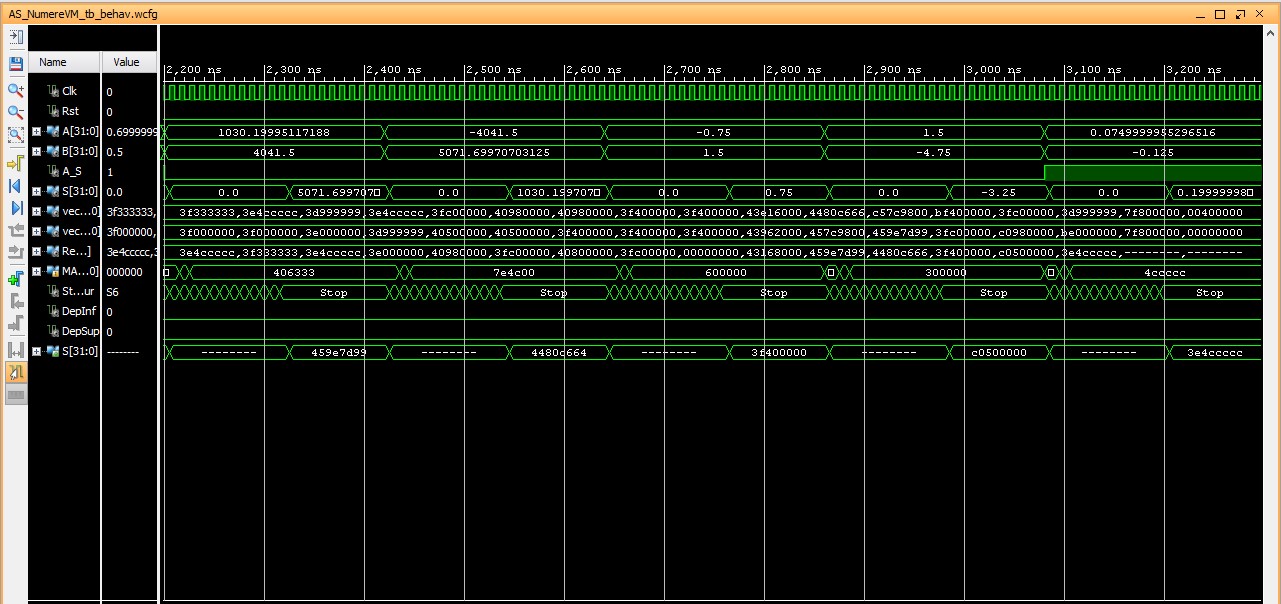
* + 1. **Main**

Acest modul este modulul principal, care conecteaza toate modulele prezentate mai sus.

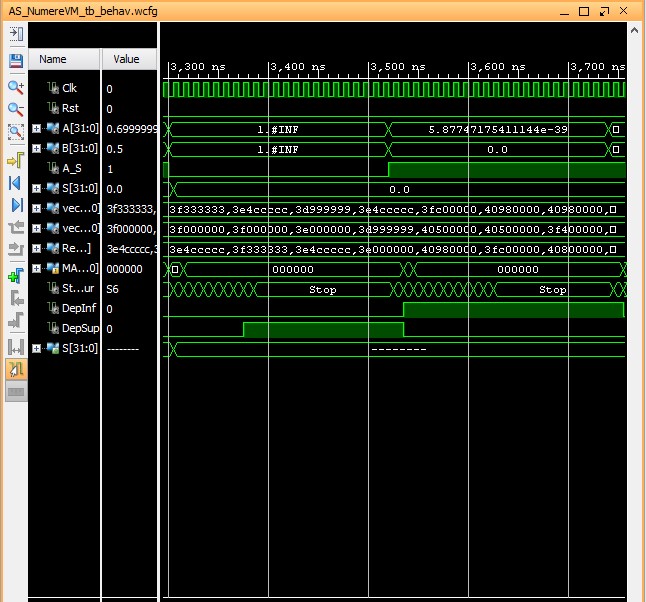
# **Rezultate experimentale**

Pentru testarea aplicatiei, am creat un modul de Test Bench al modulului principal. Rezulatele pot fi vizualizate in imaginile de mai jos. \





In urmatoarele imagini se pot observa cazurile de depasire superioara si depasire inferioara:



# **Concluzii**

In concluzie, in cadrul acestui proiect, am implementat operațiile de adunare și scădere a numerelor în virgulă mobilă, acumulând cunoștințe despre Standardul IEEE 754 și modalitățile de realizare a acestor operații. În paralel, am îmbunătățit abilitățile de codare în limbajul VHDL. Pentru evoluții ulterioare, proiectul ar putea fi integrat într-o aplicație extinsă, cum ar fi un calculator, în care să fie necesare aceste operații. Acest calculator ar permite introducerea numerelor în format binar și efectuarea diverselor operații aritmetice cu ele.

# **7. Bibliografie**

[1] “Unitatea centrală de prelucrare (Central Processing Unit – CPU)”, [https://computerplusro.wordpress.com](https://computerplusro.wordpress.com/tag/numere-in-virgula-fixa-si-in-virgula-mobila/)

[2] Baruch Zoltan Francisc, “Structura sistemelor de calcul”, Editura U. T. PRES, Cluj-Napoca, 2002.

[3] http://users.utcluj.ro/~baruch/book\_ac/AC-Adunare-VM.pdf

[4] “Nexys4 DDR™ FPGA Board Reference Manual ”, 2016, [https://reference.digilentinc.com/nexys4ddr\_rm.pdf](https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf)

[5] “PmodKYPD™ Reference Manual”, 2016,[https://reference.digilentinc.com/pmodKYPD\_rm.pdf](https://reference.digilentinc.com/_media/pmod:pmod:pmodKYPD_rm.pdf)

[6] <https://users.utcluj.ro/~baruch/book_ac/AC-Reprez-VM.pdf>

[7] <https://users.utcluj.ro/~baruch/book_ac/AC-Adunare-VM.pdf>