

Proiectarea unei unități aritmetico-logice în virgulă flotantă

Student : Rujac Roxana

-Structura Sistemelor de Calcul-

Universitatea Tehnica din Cluj-Napoca
-9 ianuarie 2025-

Cuprins

1. Introducere

1.1 Domeniu si context	3
1.2 Obiective.....	3

2. Studiu bibliografic

2.1 Ce este ALU	4
2.2 Tehnologii	5
2.3 Algoritmi	6

3. Analiza

3.1. Propunerea de proiect.....	8
3.2. Analiza proiectului.....	8

4. Design

4.1 Proiectare.....	9
4.2 Implementare.....	11

5. Testare.....13

6. Bibilografie16

1. Introducere

1.1 Domeniu si context

Unitățile Aritmetice și Logice (ALU) reprezintă componente esențiale ale arhitecturii moderne a calculatoarelor, fiind responsabile pentru realizarea operațiunilor aritmetice și logice. Într-o lume din ce în ce mai dependentă de date, procesarea eficientă a numerelor reale este crucială, iar utilizarea virgulei mobile permite gestionarea unor intervale largi de valori și precizii variabile. Aceasta se dovedește a fi deosebit de importantă în aplicații precum simulările științifice și algoritmi de învățare automată.

Proiectul de față își propune să exploreze și să implementeze o ALU cu virgula mobilă, analizând eficiența și precizia operațiunilor aritmetice.

1.2 Obiective

Proiectul are ca scop dezvoltarea unei unități aritmetice și logice (ALU) capabilă să efectueze operații cu virgula mobilă. Principalele obiective ale acestui proiect sunt:

- Proiectarea ALU cu Virgula Mobilă în VHDL: Crearea unei ALU care să implementeze funcționalități de adunare și înmulțire utilizând VHDL pentru o descriere hardware clară și eficientă.

- Implementarea pe Placă FPGA: Utilizarea mediului Xilinx pentru a sintetiza și implementa designul ALU pe o placă FPGA, validând astfel funcționalitatea în condiții practice.

2. Studiu bibliografic

2.1 Ce este ALU

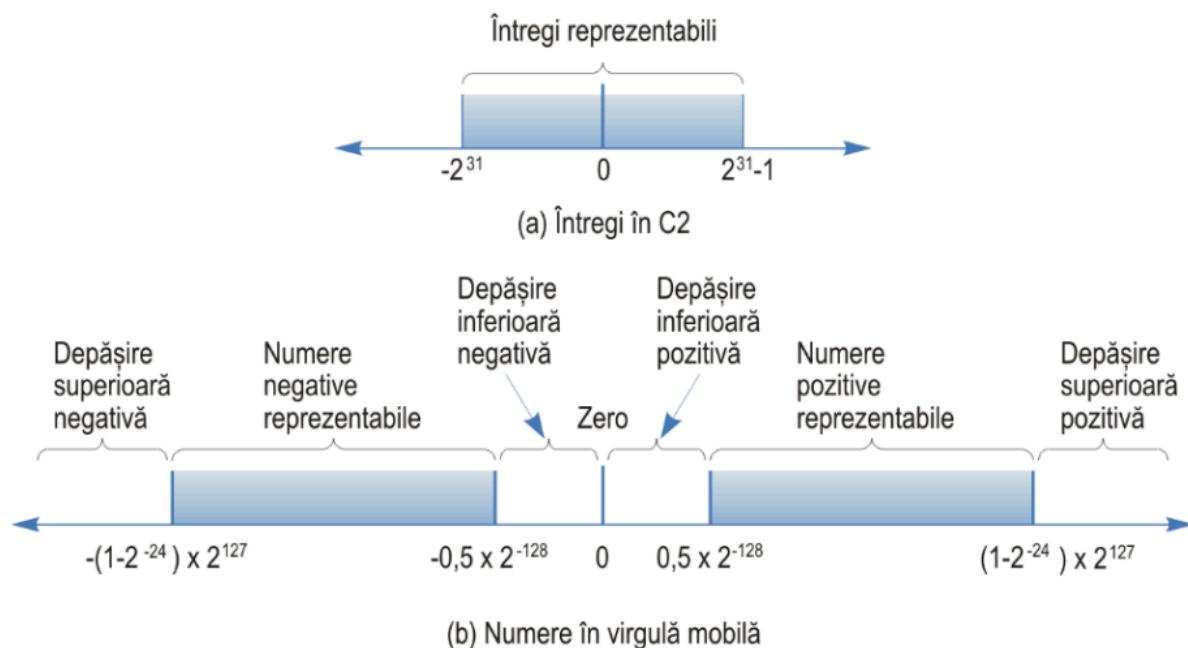
ALU, sau **Unitatea Aritmetică și Logică** (din engleză: **Arithmetic Logic Unit**), este o componentă fundamentală a arhitecturii unui procesor sau a unui sistem de calcul. Aceasta îndeplinește sarcini esențiale pentru procesarea datelor.

ALU cu virgula mobilă (Floating Point ALU) este o extensie a unității aritmetice și logice (ALU) care se ocupă cu manipularea numerelor în virgula mobilă.

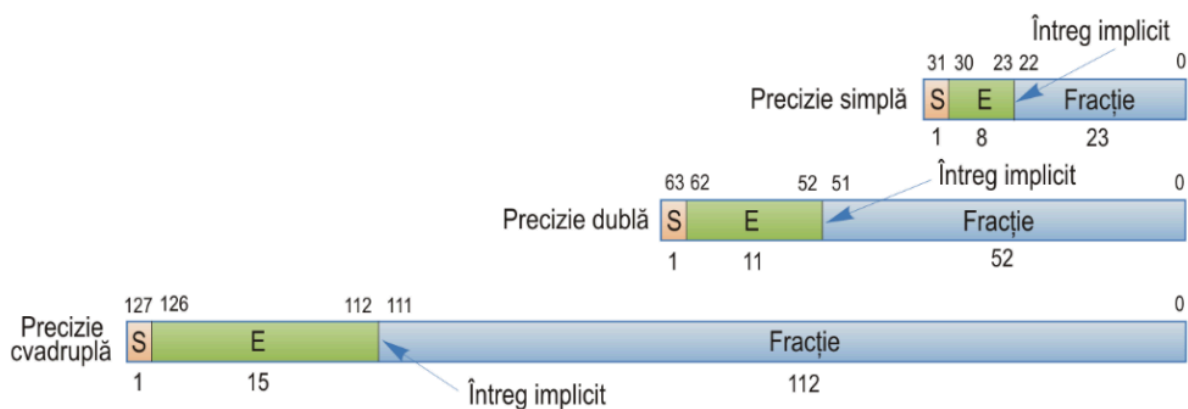
Virgula mobilă este un sistem de reprezentare a numerelor reale care permite stocarea atât a valorilor foarte mari, cât și a celor foarte mici. Forma generală a unui număr în virgula mobilă este:

$$\text{număr} = \pm \text{mantisa} \times \text{baza}^{\text{exponent}}$$

- **Mantisa:** Partea semnificativă a numărului.
- **Baza:** De obicei, 2 pentru calculatoarele care folosesc sistemul binar.
- **Exponent:** Oferă o scalare a mantisei, determinând poziția punctului zecimal (sau binar).



Standardul IEEE 754 pentru reprezentarea în virgula mobilă:



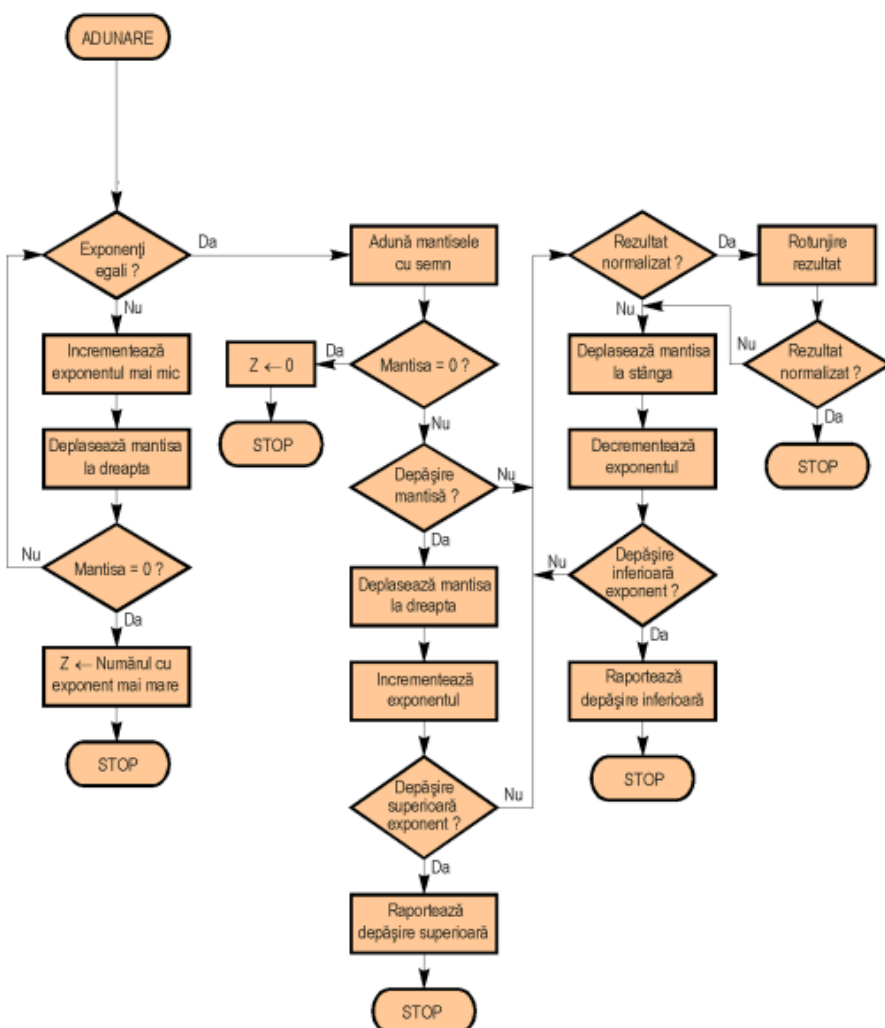
2.2 Tehnologii

- **VHDL (VHSIC Hardware Description Language)** este un limbaj de descriere a hardware-ului care ne permite modelarea, simularea și sintetizarea circuitelor digitale. VHDL este esențial pentru definirea comportamentului și structurii ALU-ului nostru, facilitând verificarea logicii designului înainte de implementare.
- **FPGA (Field-Programmable Gate Array)** reprezintă un dispozitiv semiconductor reconfigurabil, ideal pentru prototiparea circuitelor digitale. Prin utilizarea FPGA-urilor, putem implementa rapid și eficient ALU-ul cu virgula mobilă, beneficiind de execuția simultană a operațiilor pentru o performanță optimizată.
- **Xilinx** este un lider în producția de FPGA-uri și oferă instrumente software avansate, precum Vivado Design Suite. Acest mediu de dezvoltare ne permite să sintetizăm și să implementăm designul ALU-ului în mod eficient, asigurându-ne că funcționalitatea acestuia este corect testată și validată în medii reale.

2.3 Algoritmi

Pentru ALU-ul cu virgula mobilă, algoritmi folosiți pentru **adunare** și **înmulțire** trebuie să țină cont de reprezentarea numerelor în formatul standard IEEE 754. Acești algoritmi sunt ceva mai complecși decât în cazul operațiilor cu numere întregi, deoarece implică manipularea separată a **mantisei**, **exponentului** și **semnului** numerelor.

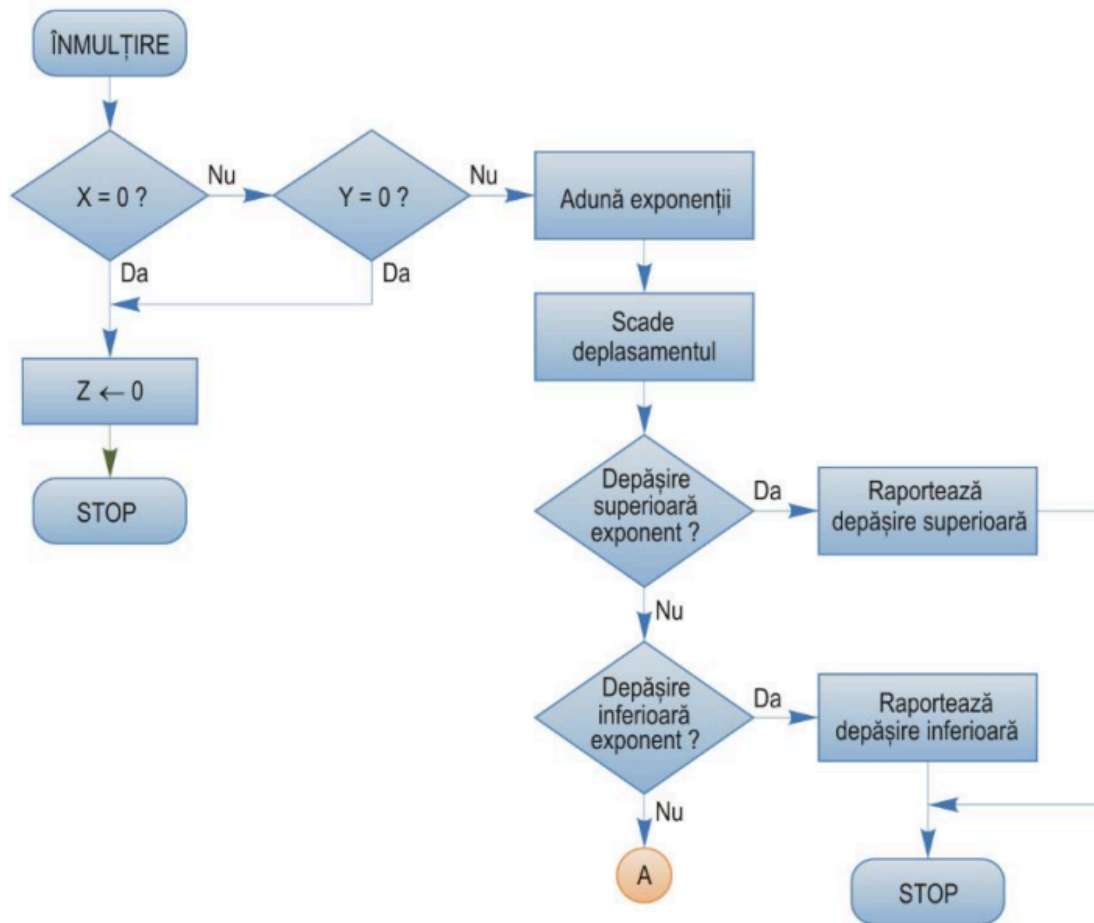
1. Adunarea



Etape ale algoritmului de adunare:

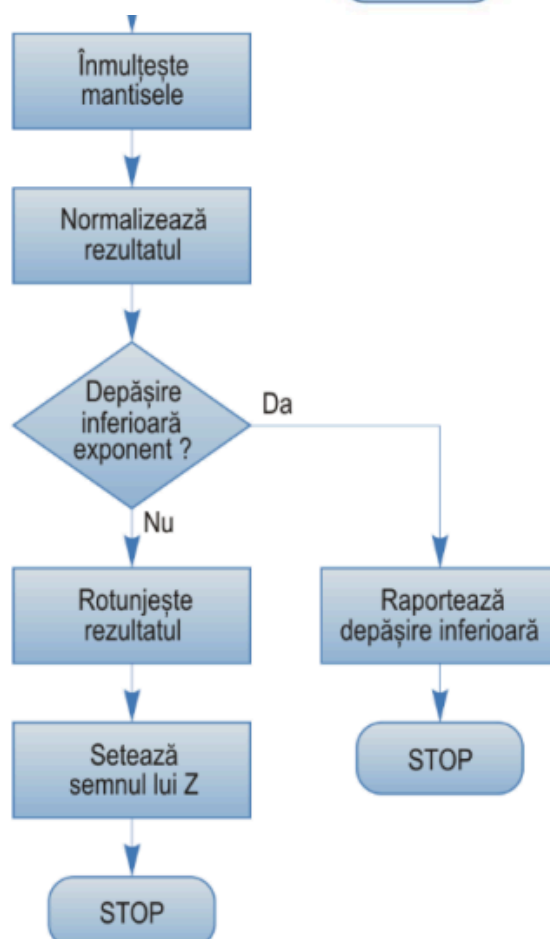
1. Alinierea mantiselor;
2. Adunarea sau scăderea mantiselor;
3. Normalizarea rezultatului;
4. Rotunjirea rezultatului.

2. Înmulțirea



Etape ale algoritmului de înmulțire:

1. Determinarea semnului rezultatului
2. Adunarea exponenților
3. Înmulțirea mantiselor
4. Normalizarea rezultatului
5. Detectarea erorilor de tip overflow și underflow
6. Formarea rezultatului final



3. Analiza

3.1. Propunerea de proiect

Scopul acestui proiect este să creez o unitate aritmetico-logică (ALU) capabilă să efectueze operații aritmetice pe numere în virgula flotantă, implementată în VHDL și implementată pe un FPGA. ALU-ul va suporta operațiile de **adunare** și **înmulțire** cu virgula flotantă, în conformitate cu standardul IEEE 754, care definește modul de reprezentare a numerelor în virgula flotantă.

În cadrul acestui proiect, se va urmări o abordare detaliată pe două niveluri importante:

- Proiectarea arhitecturală a ALU-ului
- Implementarea și simularea în VHDL pe FPGA

3.2 Analiza proiectului

Pentru ca proiectul să fie viabil, sunt necesare anumite cerințe tehnice și funcționale, care trebuie respectate în timpul dezvoltării:

- **Reprezentarea în virgula flotantă:** ALU-ul va trebui să manipuleze numere în formatul IEEE 754 (32 sau 64 de biți), ce include semn, exponent și mantisă.
- **Operații aritmetice:** Proiectul va implementa cel puțin două operații fundamentale cu virgula flotantă: adunare și înmulțire.
- **Selector de operații:** selectarea dintre adunare și înmulțire
- **Gestionarea excepțiilor:** ALU-ul va trebui să poată gestiona cazuri speciale, cum ar fi overflow, underflow, NaN (Not a Number) și Infinity.
- **Performanță:** Proiectul trebuie să fie optimizat pentru performanță, cu un timp de răspuns rapid, minimizând latența operațiilor.
- **Consum de resurse:** Designul trebuie să fie optimizat pentru a utiliza cât mai puține resurse hardware, mai ales având în vedere că implementarea se va face pe un FPGA, unde resursele sunt limitate.
- **Simularea și testarea:** Proiectul va include teste pentru validarea corectitudinii operațiunilor ALU-ului, folosind testbench-uri în VHDL.

4. Design

4.1 Proiectare

Arhitectura ALU-ului va fi împărțită în mai multe componente funcționale care să permită implementarea corectă a operațiilor de adunare și înmulțire. Structura principală a ALU-ului va include următoarele unități:

1. **Unitatea de adunare (Floating Point Adder):** Această unitate va efectua operația de adunare pentru numerele în virgula flotantă. Ea va conține:
 - **Alinierea mantiselor:** Compararea exponentilor și mutarea mantiselor pentru a le aduce la aceeași ordine de magnitudine.
 - **Adder pentru mantise:** Adunarea efectivă a mantiselor, având în vedere semnul fiecărui operand.
 - **Normalizare:** Rezultatul adunării va fi normalizat astfel încât mantisa să fie plasată în intervalul $[1, 2)$.
 - **Rotunjire:** Dacă este necesar, rezultatul va fi rotunjit pentru a păstra precizia.
2. **Unitatea de înmulțire (Floating Point Multiplier):** Această unitate va efectua operația de înmulțire pentru numerele în virgula flotantă. Ea va conține:
 - **Determinarea semnului:** Determinarea semnului rezultatului pe baza semnelor operanzilor.
 - **Adunarea exponenților:** Exponenții celor două numere vor fi adunați, având în vedere bias-ul specific.
 - **Înmulțirea mantiselor:** Multiplicarea mantiselor celor două numere.
 - **Normalizare:** Rezultatul înmulțirii va fi normalizat.
 - **Overflow/Underflow:** Verificarea erorilor de overflow și underflow în urma operațiilor.
3. **Bloc de ieșire:** După efectuarea operației, rezultatul va fi pregătit și ieșit într-un format corect de virgula flotantă, fiind disponibil la ieșirea ALU-ului.

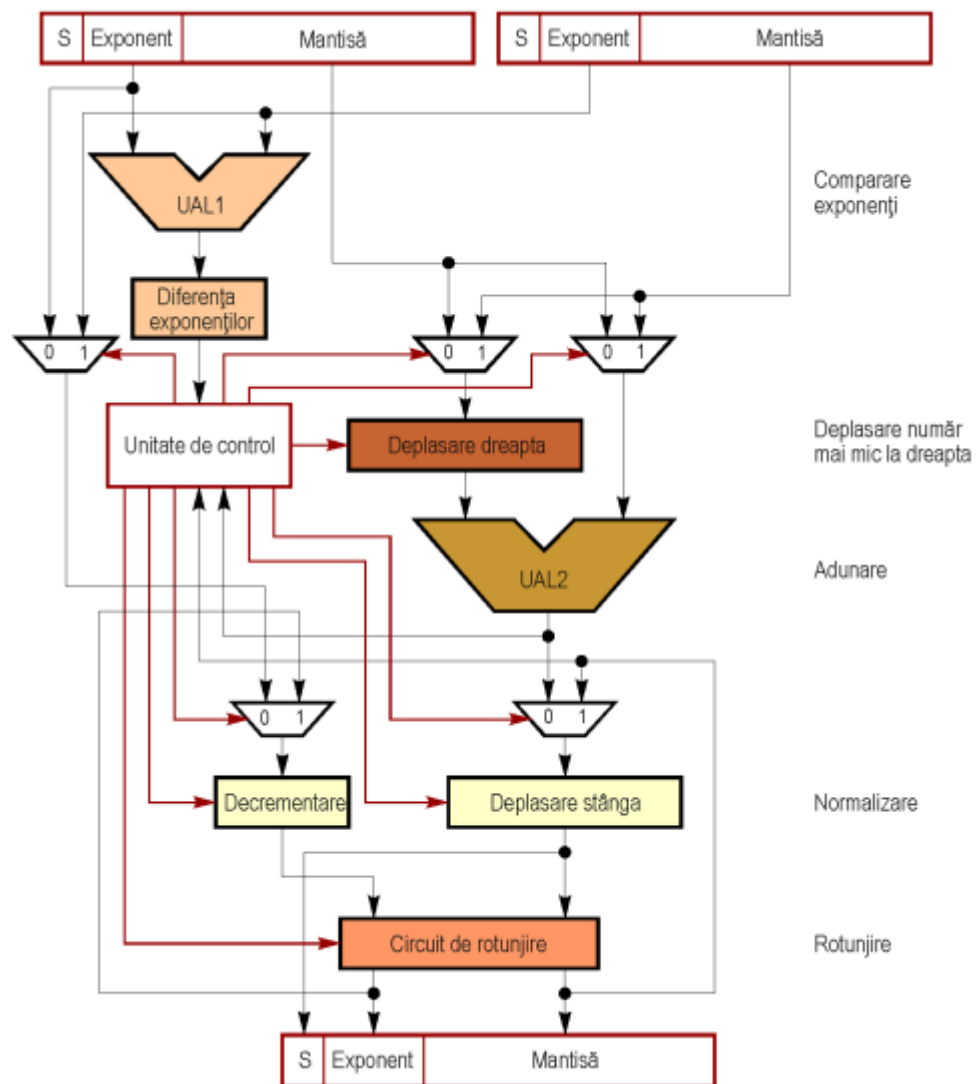
Intrări:

- **Semnal de control (1 bit):** Indică ce operație trebuie realizată. De exemplu:
 - 0 = Operație de adunare.
 - 1 = Operație de înmulțire.

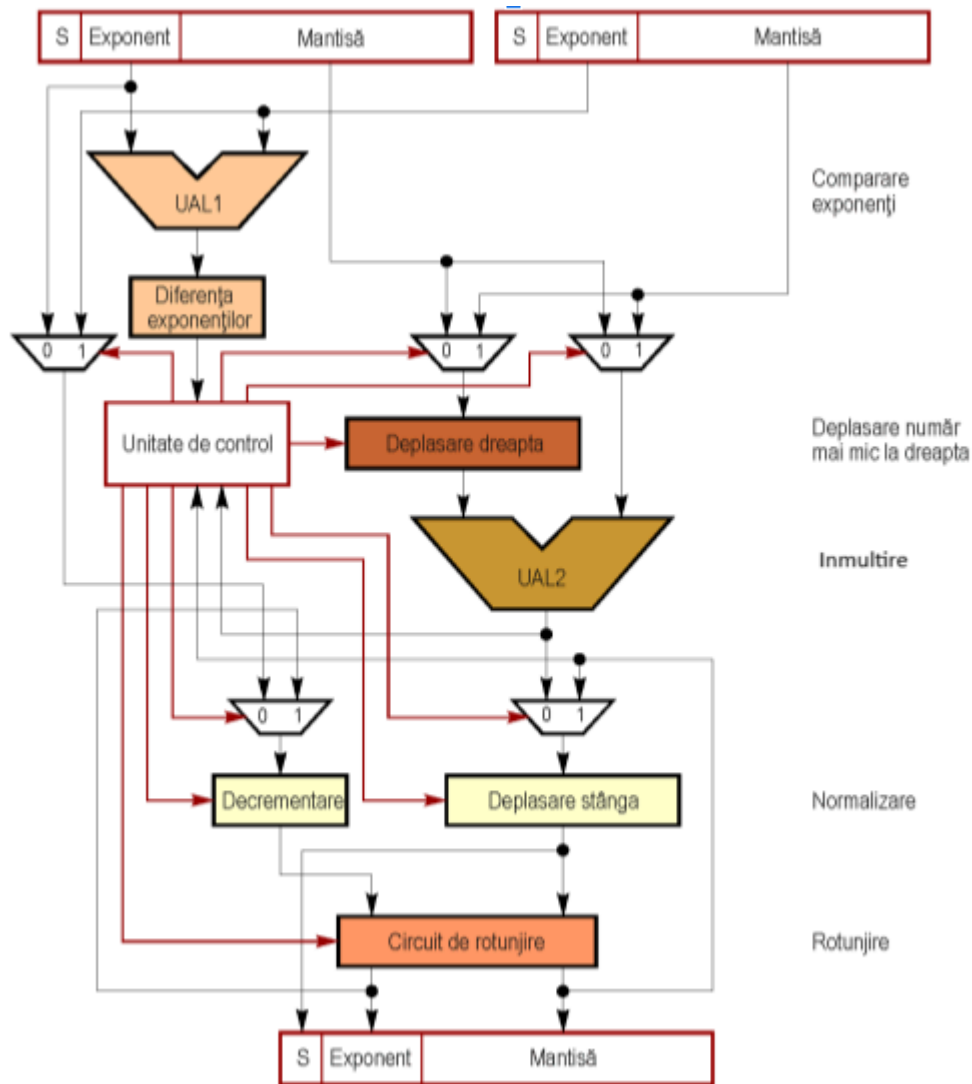
Ieșiri:

- **Rezultatul (Z):** Rezultatul operației efectuate va fi un număr în virgula flotantă, având aceeași structură ca intrările:
 - **Semn (1 bit):** Semnul rezultatului.
 - **Exponent (8 biți pentru IEEE 754 Single Precision):** Exponentul rezultatului.
 - **Mantisa (23 biți pentru IEEE 754 Single Precision):** Mantisa rezultatului, normalizată și rotunjită, dacă este necesar.

4.2 Implementare



Schema bloc a unui circuit de adunare în virgula mobilă



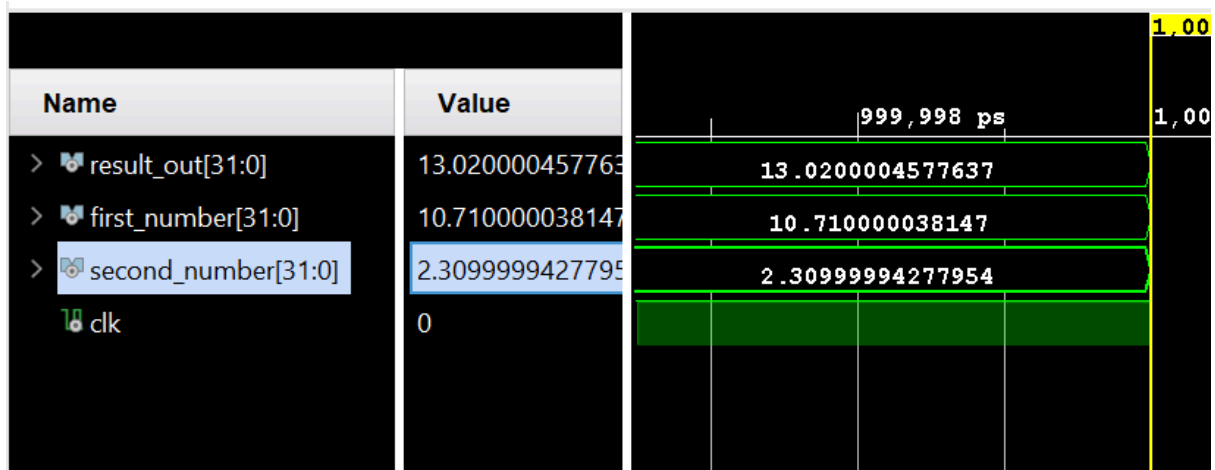
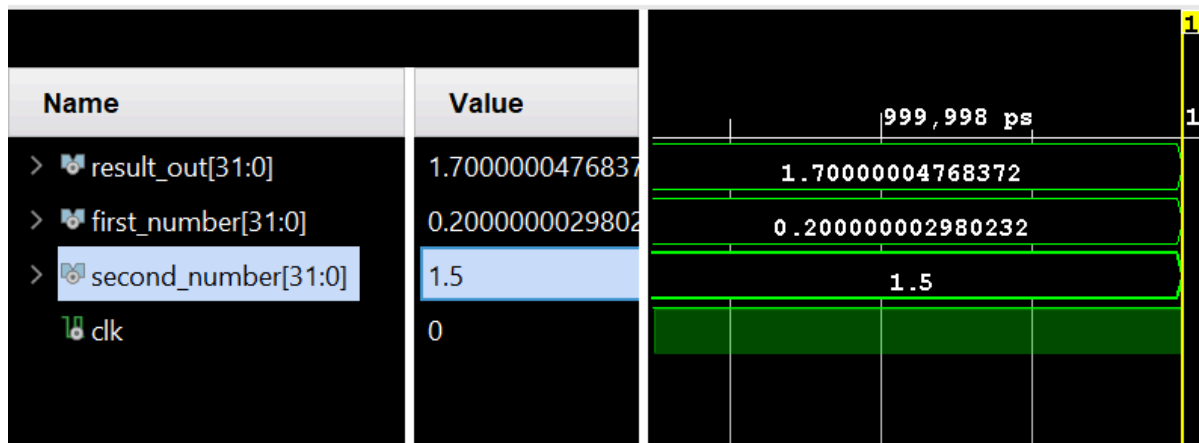
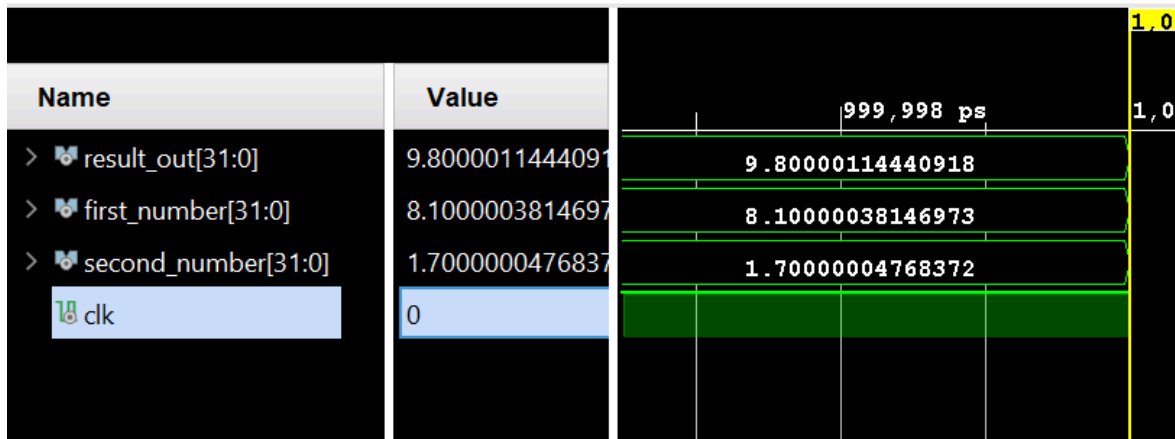
Schema bloc a unui circuit de înmulțire în virgulă mobilă





Pentru implementare, am împărțit structura ALU-lui în mai multe componente, astfel:

- un registru: aici avem numerele în virgulă flotantă (reprezentate pe 32 de biți)
- compare exponents: aici comparăm exponenții celor 2 numere
- right shift: aici shiftăm la dreapta numărul ales
- alu add: aici adunăm mantisele
- alu multiply: aici înmulțim mantisele și calculăm exponentul final
- noemalization : aici normalizăm rezultatul
- rounding: aici rotunjim rezultatul
- ssd: entitate pentru a afișa pe un Seven-Segment Display rezultatul obținut





5. Testare

Aici sunt cateva exemple de output-uri pentru operatia de adunare (primele 3) si respectiv cea de inmultire (urmatoarele 3) :







Name		Value		
>  result_out[31:0]		14.7599983215332		
>  first_number[31:0]		3.59999990463257		
>  second_number[31:0]		4.09999990463257		
 clk		0		

	999,998 ps	1,000,000 ps	1,000,002
14.7599983215332			
3.59999990463257			
4.09999990463257			

Name		Value		
>  result_out[31:0]		15.6200008392334		
>  first_number[31:0]		7.09999990463257		
>  second_number[31:0]		2.20000004768372		
 clk		0		

	999,998 ps	1,000,000	
15.6200008392334			
7.09999990463257			
2.20000004768372			

Name		Value		
>  result_out[31:0]		7.35999965667725		
>  first_number[31:0]		2.29999995231628		
>  second_number[31:0]		3.20000004768372		
 clk		0		

	999,998 ps	1,000	
7.35999965667725			
2.29999995231628			
3.20000004768372			

Bibliografie

1. TechTarget - arithmetic-logic unit (ALU)
<https://www.techtarget.com/whatis/definition/arithmetic-logic-unit-ALU>
2. GeekForGeeks - Introduction of floating point representation
<https://www.geeksforgeeks.org/introduction-of-floating-point-representation/>
3. Cristian Vancea - Cursuri Arhitectura Calculatoarelor
<https://users.utcluj.ro/~vcristian/AC.html>
4. Universitatea Tehnica din Cluj-Napoca
https://users.utcluj.ro/~baruch/book_ac/AC-Adunare-VM.pdf
5. <https://users.utcluj.ro/~baruch/media/ssc/curs/SSC-VM.pdf>