ADUNAREA SI SCADEREA NUMERELOR IN VIRGULA MOBILA

UNIVERSITATEA TEHNICA CLUJ-NAPOCA
AUTOR: DANCI IONELA
GRUPA: 6
INDRUMATOR: LISMAN DRAGOS FLORIN

Cuprins

1 Rezumat

Aritmetica in virgula mobila este unul dintre cele mai utilizate moduri de a aproxima numere reale pentru a putea performa calcule numerice pe computerele moderne.
Scopul acestui proiect este implementarea unui sumator/scazator pentru numerele in virgula mobila, in concordanta cu standardul IEEE 754 de reprezentare a numerelor in virgula mobila. Standardul defineşte trei formate: precizie simplă (32 de biţi), precizie dublă (64 de biţi) şi precizie extinsă (80 de biţi). In acest proiect este folosit formatul simpla precizie, in limbaj VHDL.
Pentru testarea proiectului am creat un test bench in care am facute diferite operatii pentru 17 perechi de numere diferite, prin care se pot observa rezultatele. In waveform am adaugat si starile intermediare pentru a se evidentia tot procesul prin care trec aceste numere in functie de formatul pe care il au.
Ca si concluzie, am reusit sa implementez acest sumator/ scazator pentru numere in vrigula mobila. Pentru exemplele pe baza carora am facut test bench-ul proiectul functioneaza in concordanta cu scopul initial.

2 Introducere

Adunarea reprezinta cea mai utilizata operatie de catre sistemele de calcul, chiar si unele operatii complexe pot fi reduse la o serie de adunari efectuate de catre Unitatea Aritmetica Logica (UAL).
Exista mai multe modalitati de implementare a operatiei de adunare, fiecare avand avantajele si dezavantajele sale legate de viteza de executie a operatiilor si resurse hardware necesare, mai multe detalii privind acestea vor fi detaliate in sectiunea de Fundamentare teoretică .
in capitolul de Proiectare si Implementare am vorbit despre modul in care am realizat implementarea proiectului si scopul in care am folosit fiecare modul. In rezultate experimentale am generat exemple prin care sa testez operatii pe diferite valori. Concluziile legate de acest proiect se gasesc in capitolul de Concluzii, iar in capitolul de Bibliografie se gasesc paginile pe baza carora am scris capitolul de Fundamentare teoritica.

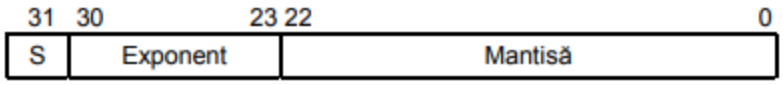
3.Fundamentare teoretica

3.1. Reprezentarea numerelor in virgula mobila (III in bibliografie)

În general, un număr N se poate reprezenta în virgulă mobilă (VM) în forma următoare:

$$N = \pm M \cdot B^{\pm E}$$

Un număr reprezentat în VM are două componente. Prima componentă este mantisa (M), care indică valoarea exactă a numărului într-un anumit domeniu, fiind reprezentată de obicei ca un număr fracţionar cu semn. A doua componentă este exponentul (E), care indică ordinul de mărime al numărului. În expresia de sus, B este 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:



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:

$+1.11 \cdot 2^0$

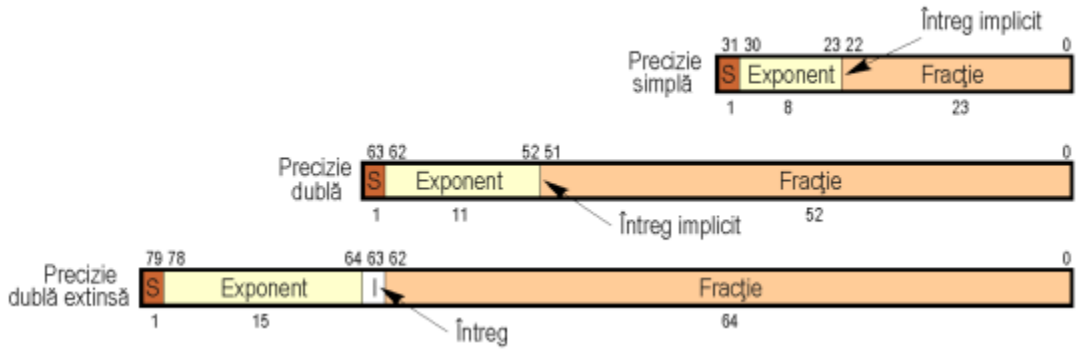
0	1000 0000	1100 0000 0000 0000 0000 000

3.2 Standardul IEEE 754 (I in bibliografie)

Standardul era destinat în primul rând microprocesoarelor şi microcalculatoarelor, unde unii producători pun la dispoziţie doar posibilităţi limitate pentru calcule numerice. Ca rezultat al acestui standard, au fost dezvoltate circuite sau procesoare care implementează standardul.
Standardul IEEE 754 defineşte următoarele formate sau precizii: precizie simplă, precizie simplă extinsă, precizie dublă şi precizie dublă extinsă. Parametrii principali ai acestor formate sunt prezentaţi în urmatorul tabel:

	Precizie simplă	**Precizie simplă extinsă**	**Precizie dublă**	**Precizie dublă extinsă**
Biți ai mantisei	24	≥ 32	53	≥ 64
Exponent real maxim	127	≥ 1023	1023	≥ 16383
Exponent real minim	-126	≤ -1022	-1022	≤ -16382
Deplasament exponent	127	Nespecificat	1023	Nespecificat

Pentru toate formatele, baza implicită este 2. Formatele cu precizie simplă, precizie dublă şi precizie dublă extinsă sunt prezentate în urmatoarea figura:



3.3 Adunarea si scaderea in virgula mobila (II in bibliografie)

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:

3.3.1. Alinierea mantiselor
3.3.2. Adunarea sau scaderea mantiselor
3.3.3. Normalizarea rezultatului
3.3.4. Rotunjirea rezultatului

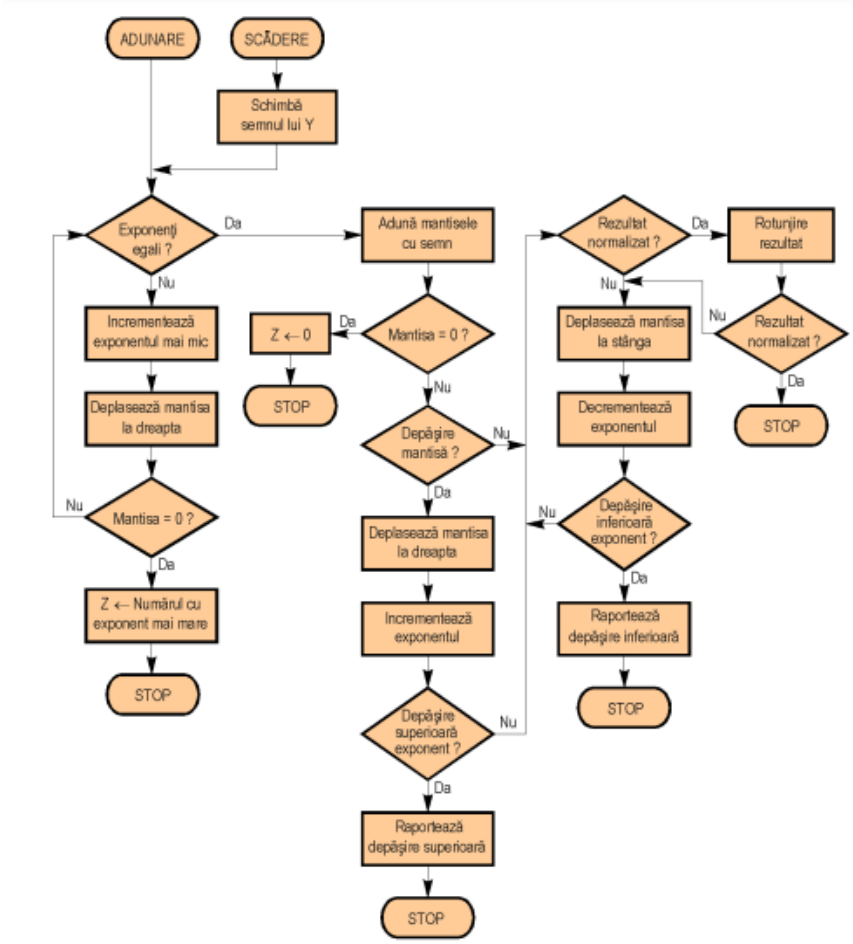
3.3.1.In aceasta etapa se compara exponentii celor doua numere, si se deplaseaza spre dreapta mantisa care are exponentul mai mic.
Compararea exponentilor se face prin diferenta acestora. Daca rezultatul este negativ, atunci exponentul al doilea este mai mare, in caz contrar, este mai mare primul exponent. Deplasarea la dreapta se face cu un numar de biti egal cu diferenta dintre exponenti.

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

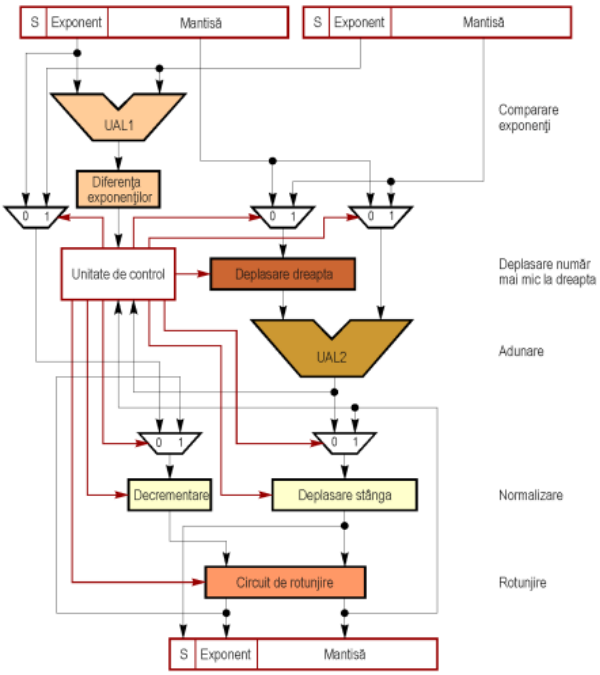
3.3.3 Normalizarea rezultatului
Dupa adunarea mantiselor, se verifica daca rezultatul este normalizat sau nu. Daca nu este normalizat, atunci urmeaza sa fie deplasat spre stanga pana cand primul bit inaintea virgulei devine egal cu 1.

3.2.4 Rotunjirea rezultatului
In cadrul proiectului am folosit trunchierea.

Organigrama proiectului:
-este preluata din II din bibliografie



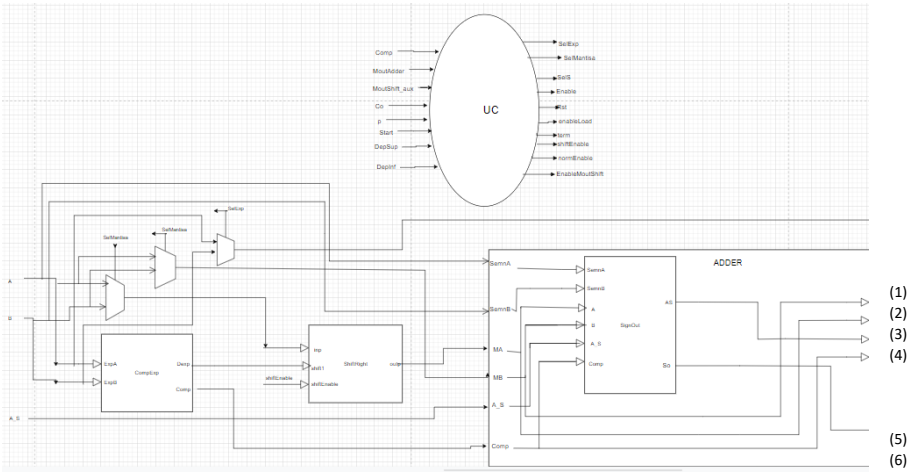
Schema pe baza caruia am gandit proiectul este urmatoarea (face parte din II din bibliografie) :

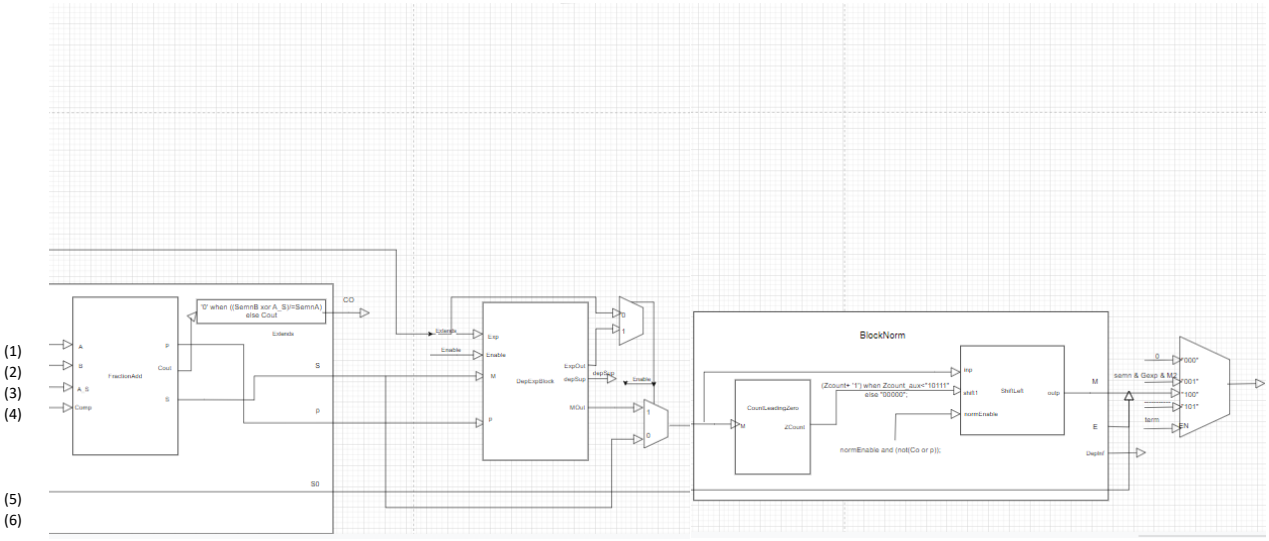


4. Proiectare si Implementare

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

4.1.Schema bloc a aplicatiei





4.2. Descriere module

4.2.1. CompExp

Acest modul realizeaza compararea celor 2 exponenenti.

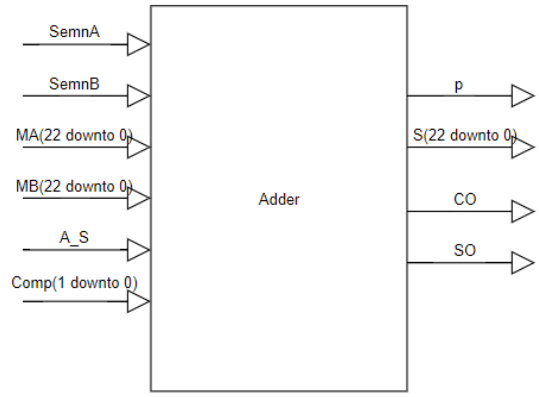
Modulul returneaza diferenta exponentilor si Comp. Comp este o valoare pe 2 biti, si are valoarea “00” cand expA>expB, “01” cand expA<expB si “10” pentru cazul in care sunt egali exponentii.

4.2.2. ShiftRight

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.

4.2.3. Adder

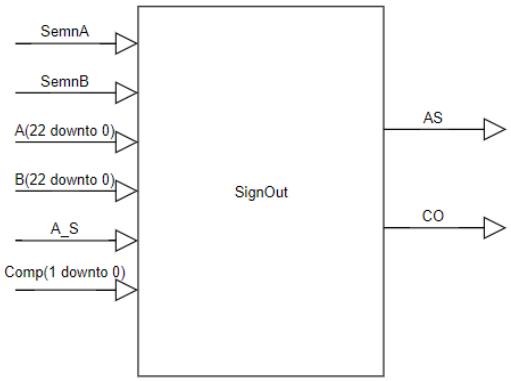
Acest modul realizeaza adunarea/scaderea mantiselor si calcularea semnului pentru numarul rezultat.



Contine doua alte module: SignOut si FractionAdd.

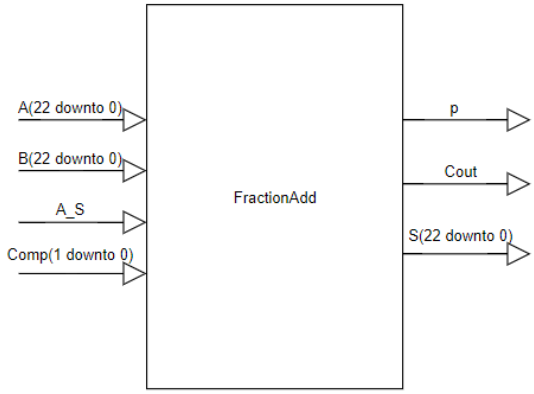
4.2.3.1. Signout

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



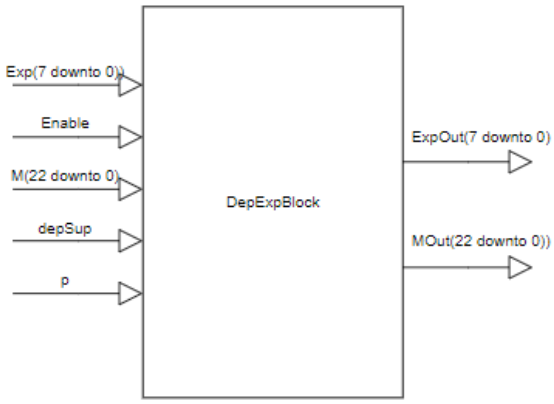
4.2.3.2. FractionAdd

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



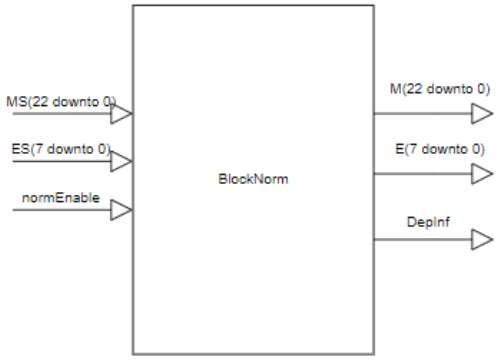
4.2.4. DepExpBlock

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.



4.2.5. BlockNorm

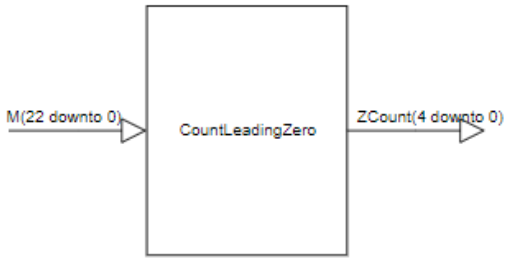
Acest modul realizeaza normalizarea rezultatului prin decrementarea exponentului si prin shiftarea mantisei la stanga pana ajunge 1 inaintea virgulei.



Acest module contine alte 2 module: CountLeadingZero si ShiftLeft

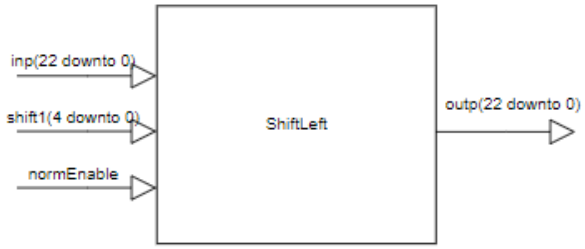
4.2.5.1. CountLeadingZero

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



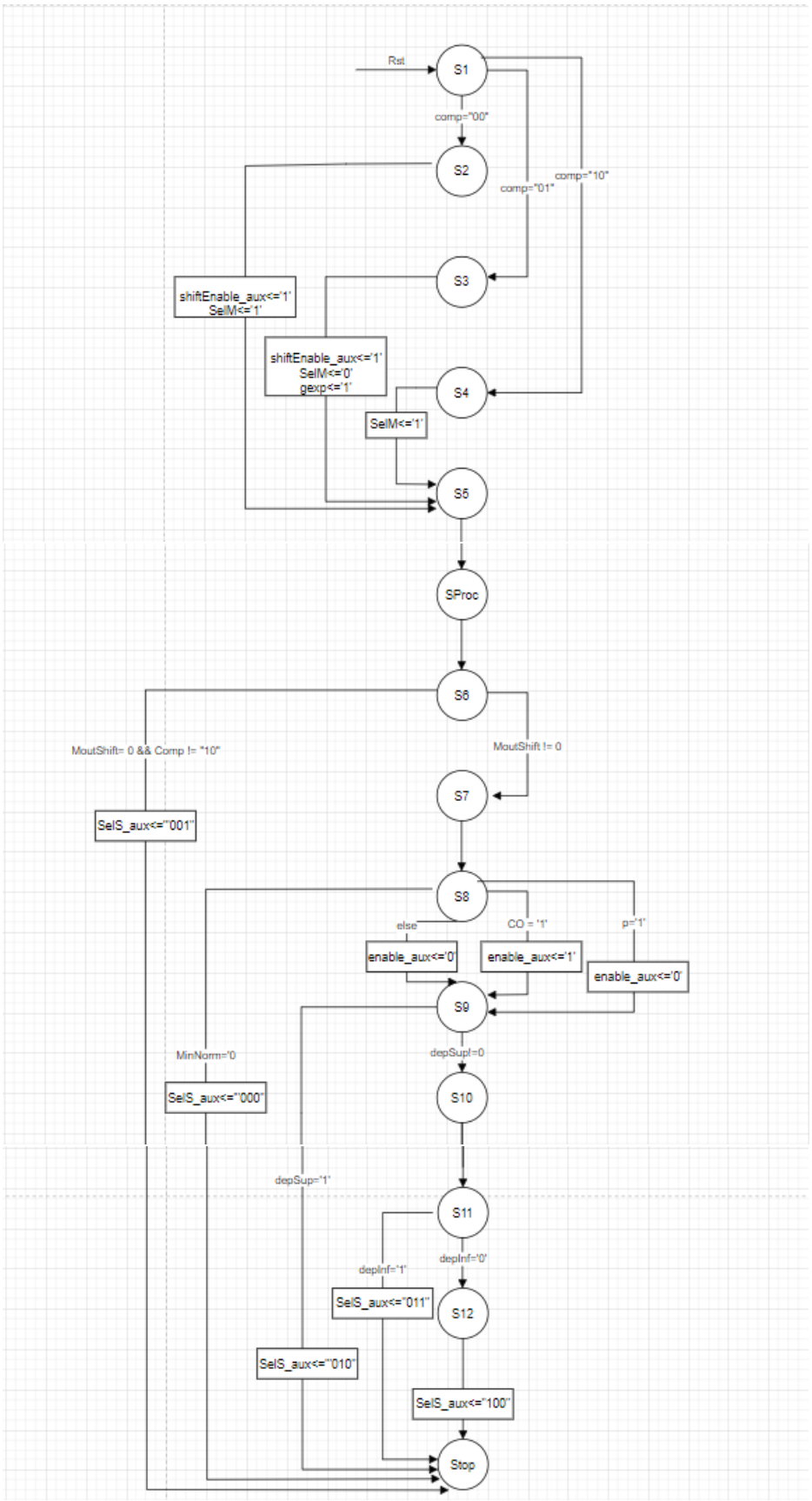
4.2.5.2. ShiftLeft

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



4.2.6. UC

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


Rst
S1
comp="00"
S2
comp="01"
comp="10"
S3
shiftEnable_aux<='1'
SelM<='1'
shiftEnable_aux<='1'
SelM<='0'
gexp<='1'
S4
SelM<='1'
S5
SProc
S6
MoutShift= 0 && Comp != "10"
MoutShift != 0
S7
SelS_aux<="001"
S8
else
CO = '1'
p='1'
enable_aux<='0'
enable_aux<='1'
enable_aux<='0'
S9
MinNorm='0
depSup!=0
SelS_aux<="000"
S10
depSup='1'
S11
depInf='1'
depInf='0'
SelS_aux<="011"
S12
SelS_aux<="010"
SelS_aux<="100"
Stop

	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’

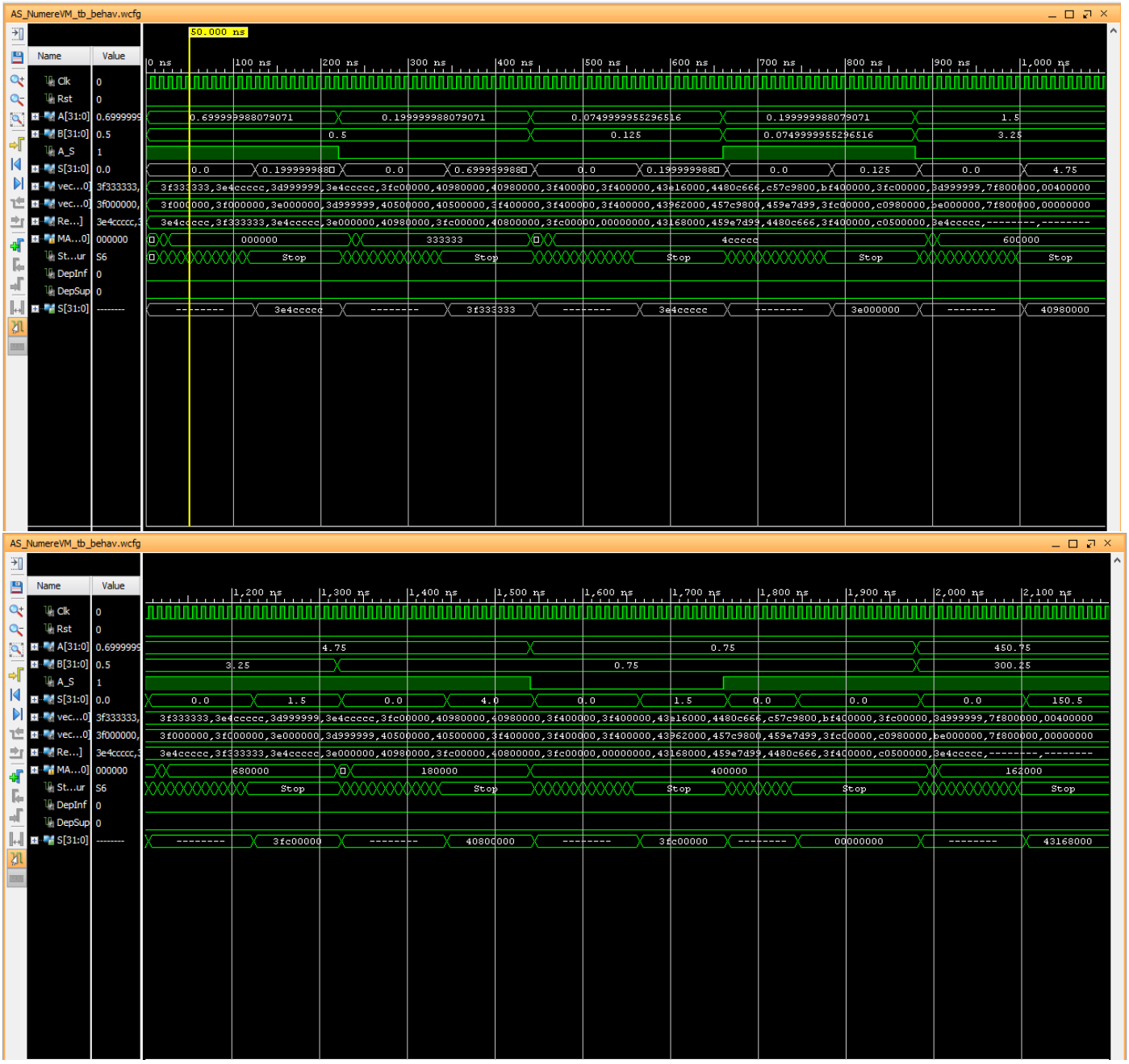
S1	Initializare + comparare exponenti
S2	Tratare expA > expB
S3	Tratare expA < expB
S4	Tratare expA = expB
S5	Shiftare dreapta
SProc	Procesare rezoltat shiftare
S6	Decizie in functie de razultatul shiftarii
S7	Adunare/Scadere
S8	Decizie in functie de rezultatul adunarii
S9	Verificare depasire superioara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

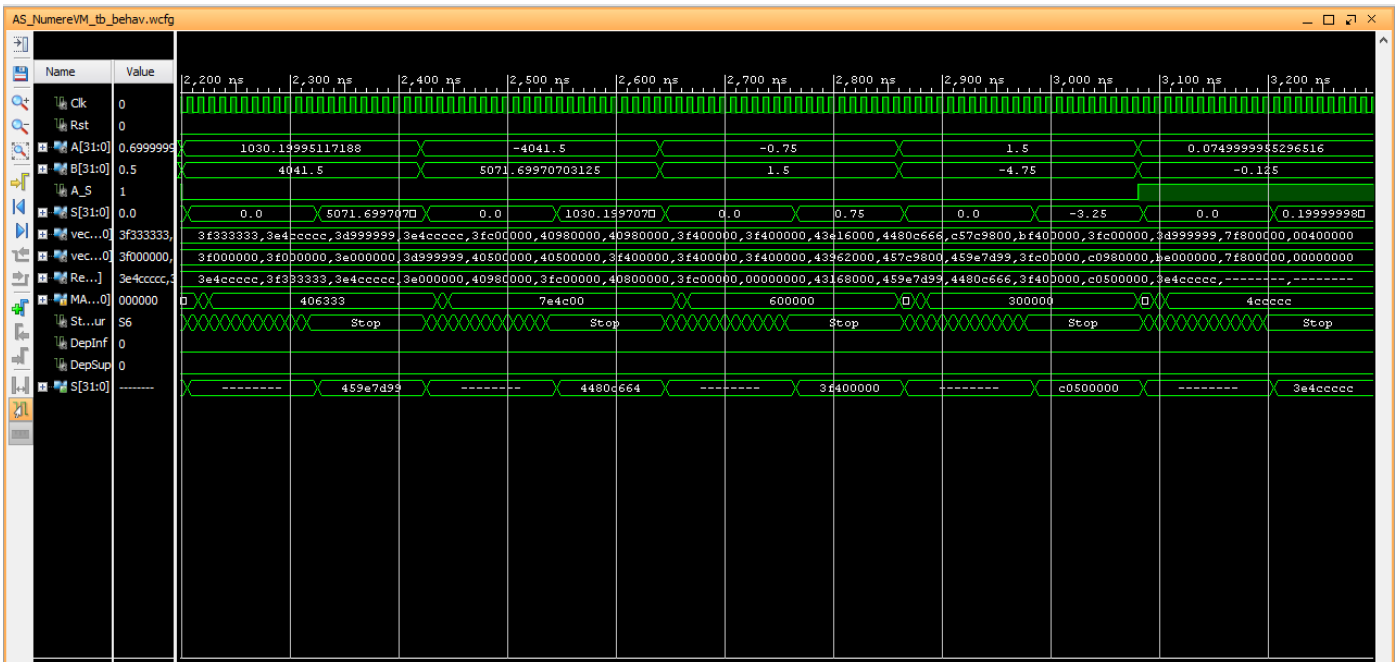
4.2.7. Main

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

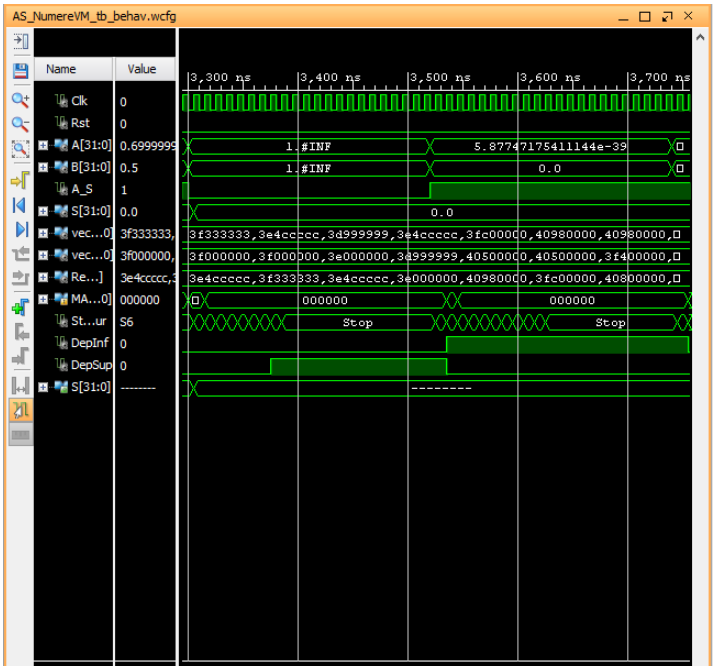
5. Rezultate experimentale

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


AS_NumereVM_tb_behav.wcfg
Name
Value
Clk
Rst
A[31:0]
B[31:0]
A_S
S[31:0]
vec...0]
vec...0]
Re...]
MA...0]
St...ur
DepInf
DepSup
S[31:0]
50.000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
1,000 ns
1,200 ns
1,300 ns
1,400 ns
1,500 ns
1,600 ns
1,700 ns
1,800 ns
1,900 ns
2,000 ns
2,100 ns
Stop



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



6. Concluzii

In acest proiect am realizat implementarea operatiilor de adunare si scadere a numerelor in virgula mobila, prin intermediu caruia am acumulat informatii despre Standardul IEEE 754, despre modul prin care se fac aceste operatii, dar in acelasi timp mi-am si imbunatatit capacitatea de a scrie cod VHDL. Ca dezvoltari ulterioare, acest proiect ar putea sa fie inclus intr-o aplicatie mai mare, in care sa fie necesare aceste operatii, cum ar fi un calculator in care sa fie introduse numere in binar si realizate diferite operatii cu ele.

7. Bibliografie

I. file:///C:/Users/Public/Desktop/~An3~/~Sem1~/SSC/Proiect/ieee-754-2008.pdf

II. https://users.utcluj.ro/~baruch/book_ac/AC-Adunare-VM.pdf

III. https://users.utcluj.ro/~baruch/book_ac/AC-Reprez-VM.pdf