Universitatea Tehnica din Cluj-Napoca

Calculatoare si tehnologia informatiei



Unitate de calcul in virgula mobila : adunarea si scaderea

Narita Catalin-Ioan

Tandea Alin-Dan

Grupa : 30235

Indrumator de proiect : Cristi Mocan

Data:

Contents

[1.Introducere 3](#_Toc482543925)

[2.Fundamente teoretice 4](#_Toc482543926)

[2.1 Reprezentarea numerelor in virgula mobila. Standardul IEEE 754 4](#_Toc482543927)

[2.2 Valori speciale 5](#_Toc482543928)

[2.3 Adunarea si scaderea in virgula mobila 6](#_Toc482543929)

[2.4 Organigrama operatiei de adunare 7](#_Toc482543930)

[2.5 Schema bloc a unitatii de adunare si scadere in virgula mobila 7](#_Toc482543931)

[2.6 Schema bloc a circuitului implementat 9](#_Toc482543932)

[3. Proiectare si implementare 9](#_Toc482543933)

[3.1 Descrierea componentelor 10](#_Toc482543934)

[3.1.1 Unitate aritmetico – logica (UAL1) 10](#_Toc482543935)

[3.1.2 Circuit de rotunjire 10](#_Toc482543936)

[3.1.3 Calcul exponent 11](#_Toc482543937)

[3.1.4 Deplasare mantisa 12](#_Toc482543938)

[3.1.5 Multiplexor generic 12](#_Toc482543939)

[3.1.6 Calcul mantisa 13](#_Toc482543940)

[4. Rezutate experimentale 13](#_Toc482543941)

[4.1 Unitate aritmetico-logica (UAL1) 13](#_Toc482543942)

[4.2 Circuit de rotunjire 14](#_Toc482543943)

[4.3 Circuit de calcul al exponentului 14](#_Toc482543944)

[4.4 Circuit de deplasare a mantisei 15](#_Toc482543945)

[4.5 Multiplexor generic 15](#_Toc482543946)

[4.6 Calcul mantisa 15](#_Toc482543947)

[Bibliografie 15](#_Toc482543948)

# 1.Introducere

Nevoia de a avea o gama mai larga de valori rationale care totodata sa fie si foarte precise a dus la aparitia unei alte metode de reprezentare a numerelor rationale , si anume reprezentarea in virgula flotanta. Tendinta in domeniu este de a se putea efectua cat mai multe operatii in virgula mobila pe secunda .

Operatia de adunare se afla la baza tuturor sistemelor de calcul moderne . Operatia de scadere este similara cu cea de adunare , diferenta constand in faptul ca scazatorul este reprezentat cu semn schimbat. Virgula mobila se refera la faptul ca in interiorul numarului virgula care separa partea intreaga si partea fractionara se poate deplasa oriunde relativ la cifrele semnificative ale numarului . [[1](#_Bibliografie)]

Obiectivul principal al proiectului este implementarea opeartiilor de adunare si scadere a numerelor reprezentate in virgula mobila respectand standardul IEEE 754 pentru numere de 32 de biti (simpla precizie) .

# 2.Fundamente teoretice

# 2.1 Reprezentarea numerelor in virgula mobila. Standardul IEEE 754

Standardul IEEE 754 reglementeaza operatiile aritmetice in virgula mobila , fiind specificat formatul , operatiile , conversiile , reguli de rotunjire si conditii exceptionale

În general, un număr N se poate reprezenta în virgulă mobilă în forma urmatoare:

N = ± M · B±E

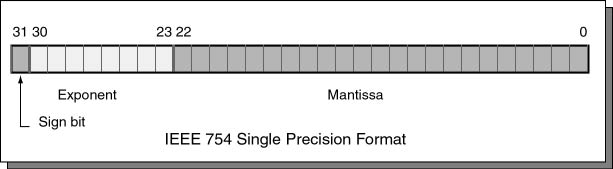
Un număr reprezentat în virgulă mobilă are următoarele componente:

* s semnul
* *c*  caracteristica
* *e*  exponentul
* m mantisa
* *d*  deplasarea

Exponentul nu este reprezentat de obicei in forma lui normala ci este folosit exponentul deplasat care este intotdeauna pozitiv

Pentru reprezentarea numerelor in precizie simpla ( 32 de biti ) sunt folositi 23 de biti pentru partea fractionara , 8 biti pentru exponent si 1 bit de semn . Reprezentarea poate fi observata in **figura 1**. Valoare numarului se determina astfel :

* daca 0 < exponent < 255 => numar = (-1)semn X c.m X 10e
* daca e = 0 si m = 0 => numar = 0
* daca e = 0 si m 0 sau e = 255 => eroare [[3](#_Bibliografie)]



**figura 1.** Reprezentare in virgula mobila (simpla precizie) [[2](#_Bibliografie)]

In cadrul reprezentarii numerelor in virgula mobila, partea fractionara nu poate avea o rezolutie la fel de mare ca reprezentarea numerelor intregi, doar cu cei 24 de biti, insa aceasta valoare este aproximata.

# 2.2 Valori speciale

Signed zero

In standardul IEEE 745 zero este considerat ca avand un semn, deci exista atat +0 cat si -0.

Cele doua reprezentari se comporta normal in cadrul comparatiilor numerice, dar unele operatii dau rezultate diferite pentru +0 si -0. De exemplu, 1/(-0) returneaza -infinit, iar 1/(+0) +infinit.

Infinit

In reprezentarea in virgula mobila, valorile de +infinit si -infinit sunt reprezentate ca un numar cu bitii exponentului 1 si bitii mantisei 0. Bitul de semn face distinctia intre cele doua valori,

NaN(Not a Number)

NaN este o valoare speciala in standardul IEEE 754 care este returnata ca si rezultat in urma onr operatii invalide cum ar fi impartire la 0, infinit x 0, sau radacina patrata din numere negative.

Aceasta valoare este reprezentata ca un numar cu bitii exponentului 1 si bitul semnificativ al mantisei setat pentru a se face distinctia intre acesta si o valoare infinita.

Exista doua tipuri de NaN: quiet NaN(denota operatii nedeterminate) si signaling NaN(determina operatii invalide).

# 2.3 Adunarea si scaderea in virgula mobila

Pentru adunarea a doua numere in virgula mobila trebuie urmati urmatorii pasi : [[4](#_Bibliografie)]

1. se convertesc cele doua numere in notatia stiintifica normalizata , determinand semnul exponentul si mantisa
2. se aduc cei doi operanzi la acelasi exponent : se calculeaza dimensiune celor doi operanzi si se compara aceste dimensiuni . Mantisa operandului cu exponentul de dimensiuni mai mici este deplasata la dreapta cu un numar de pozitii care reprezinta diferenta in modul dintre dimensiunile exponentilor
3. se aduna mantisele obtinute la punctul anterior
4. se converteste numarul obtinut inpoi in formatul IEEE 754 pentru numere de 32 de biti

Sa prespunem ca avem urmatoarele numere reprezentate in standardul IEEE 745:

X = 0100 0010 0000 1111 0000 0000 0000 0000

Y = 0100 0001 1010 0100 0000 0000 0000 0000

1. cele doua numere in notatia stiintifica

pentru X:

S = 0

e = 1000 0100 = 132 – 127 = 5

deci notatia stiintifica pentru X : 1.0001111 x 25

pentru Y:

S = 0

e = 1000 0011 = 131 -127 = 4

deci notatia stiintifica pentru Y : 1.01001 x 24

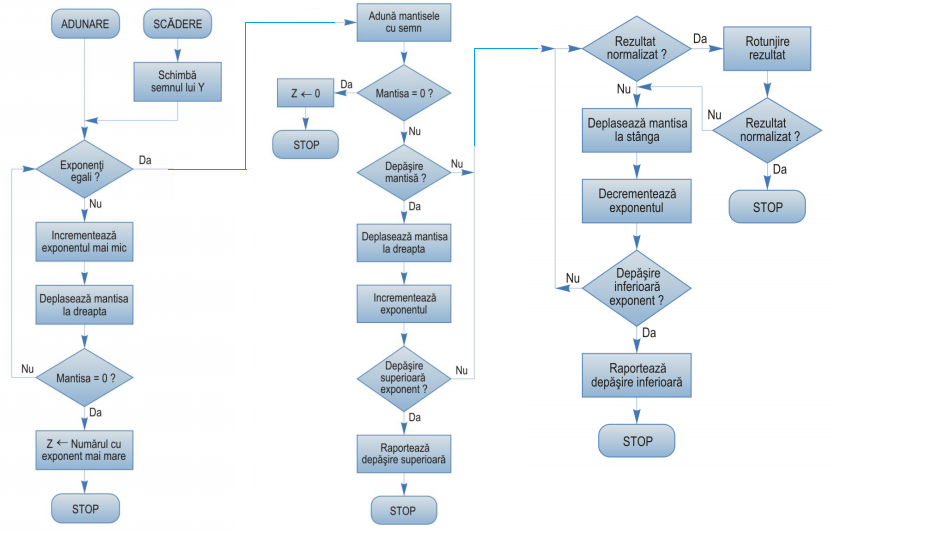
1. deoarece Y are un exponent mai mic , mantisa acestuia este deplasata cu o pozitie

Y : 0.1010010 x 25

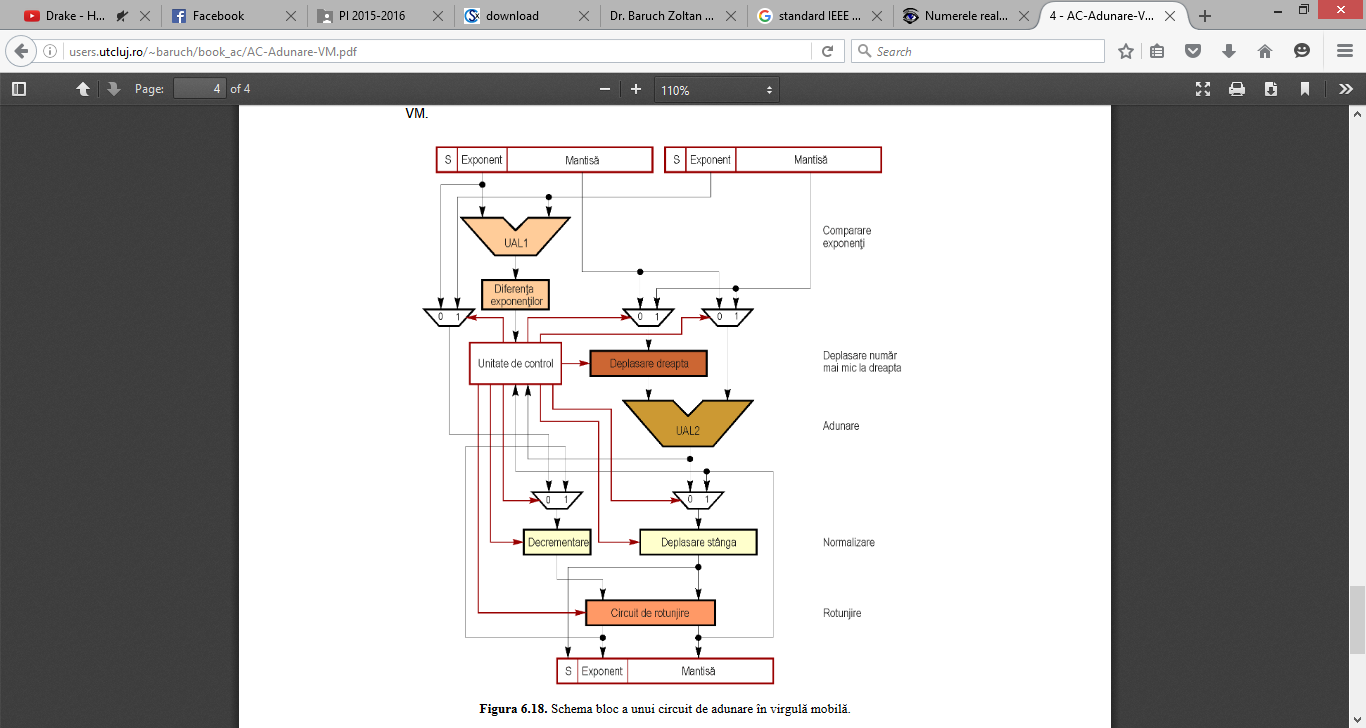
1. se aduna cele doua numere si obtinem X + Y = 1.1100001 x 25
2. rezultatul convertit inapoi in standardul IEEE 754 este

S = 0100 0010 0110 0001 0000 0000 0000 0000

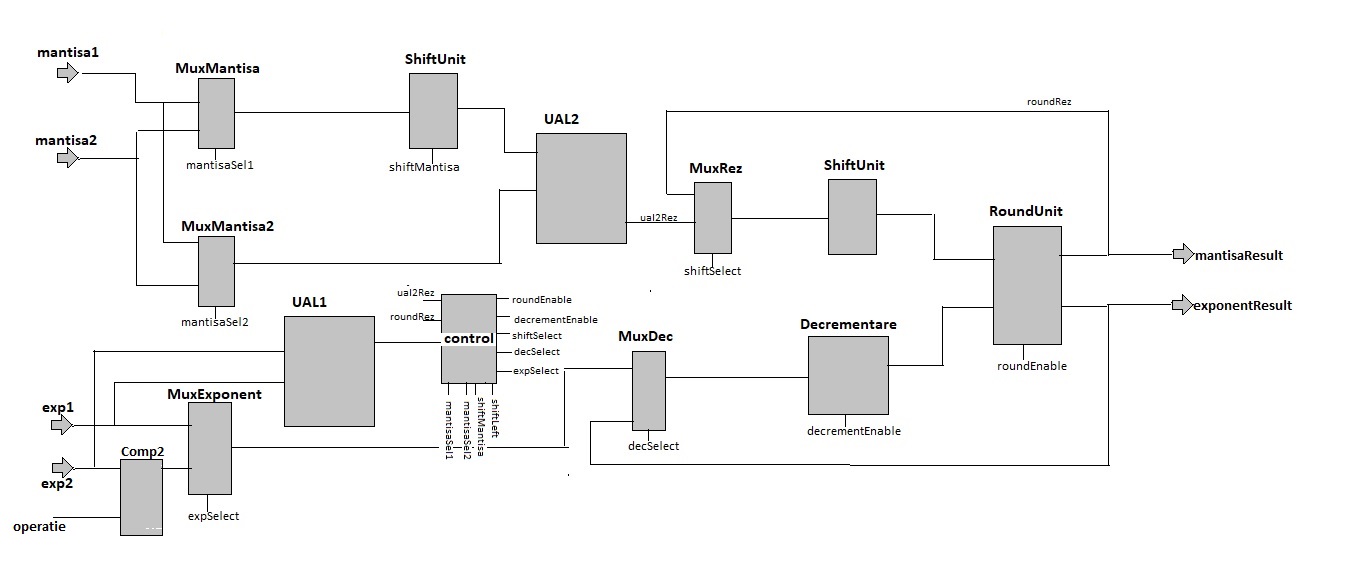
# 2.4 Organigrama operatiei de adunare



# 2.5 Schema bloc a unitatii de adunare si scadere in virgula mobila



# 2.6 Schema bloc a circuitului implementat



# 2.7 Introducerea datelor - modulul Pmod KYPD

Pentru o mai usoara furnizare a informatiilor la intrarile circuitului de adunare si scadere in virgula mobila datele vor fi introduse cu ajutorul moduluilui **Pmod KYPD** .

Modulu este o tastatura cu 16 butoane aranjate in format hexazecimal . Aceasta se conecteaza la unul dintre porturile **pmod** cu 12 pini ale placii Nexys 4 DDR folosita pentru implementarea proiectului .

Cei 12 pini ai modulului sunt descrisi in urmatorul tabel :

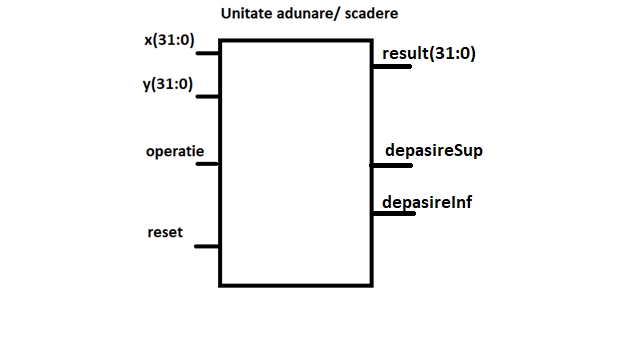
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pin | Semnal | Descriere | Pin | Semnal | Descriere |
| 1 | COL4 | Coloana 4 | 7 | ROW4 | Rând 4 |
| 2 | COL3 | Coloana 3 | 8 | ROW3 | Rând 3 |
| 3 | COL2 | Coloana 2 | 9 | ROW2 | Rând 2 |
| 4 | COL1 | Coloana 1 | 10 | ROW1 | Rând 1 |
| 5 | GND | Ground sursa de tensiune | 11 | GND | Ground sursa de tensiune |
| 6 | VCC | Sursa de tensiune(3.3V/5V) | 12 | VCC | Sursa de tensiune(3.3V/5V) |

# 3. Proiectare si implementare

Unitatea de adunare si scadere a numerelor in virgula mobila va primi ca intrare doua semnale pe 32 de biti reprezentand cele doua numere , un semnal pe un bit de selectare a operatiei ( adunare / scadere) si un semnal pe un bit de resetare . Circuitul va furniza ca iesire un semnal pe 32 de biti care reprezinta rezultatuloperatiei efectuate precum si doua flag-uri pentru depasire superioara sau inferioara.

Semnale :

* intrari
  + x : primul operand pe 32 de biti
  + y : al doilea operand pe 32 de biti
  + operatie : determina ce operatie se va efectua ( adunare / scadere )
  + reset : semnal care aduce circuitul in starea initiala
* iesiri
  + result : semnal pe 32 de biti care reprezinta rezultatul operatiei
  + depasireSup : flag care indica o depasire superioara
  + depasireInf : flag care indica o depasire inferioara



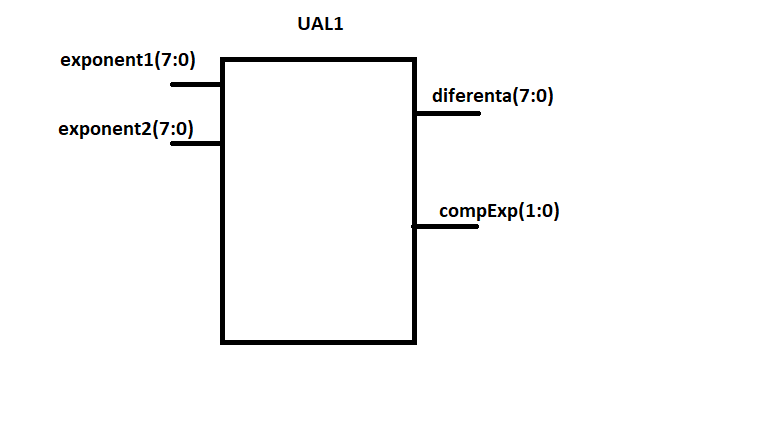
# 3.1 Descrierea componentelor

# 3.1.1 Unitate aritmetico – logica (UAL1)

Folosita pentru calcularea diferentei dintre cei doi exponenti pe 8 biti . De asemenea cu ajutorul unui semnal de iesire (compExp) se determina care dintre cei doi exponenti este mai mare

Semnale :

* intrari :
  + exponent1 : exponentul primului operand
  + exponent2 : exponentul celui de – al doilea operand
* iesiri
  + diferenta : rezultatul operatie de scadere al exponentilor celor doua numere primite ca semnale de intrare
  + compExp : semnal pe doi biti care determina rezultatul compararii celor doi exponenti

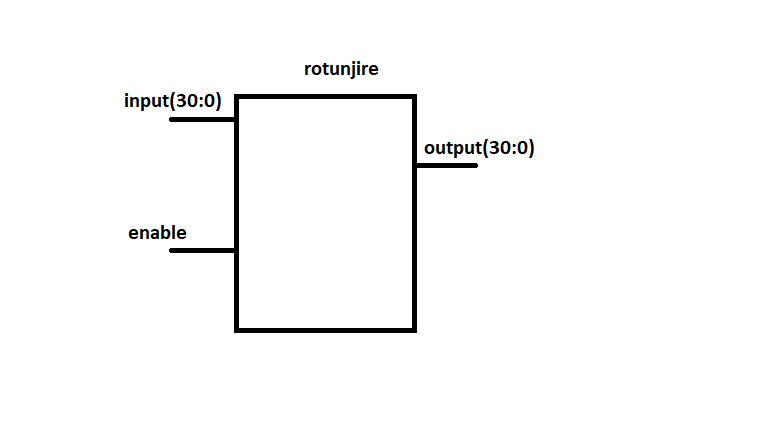


# 3.1.2 Circuit de rotunjire

Este folosit pentru rotunjirea unui numar la cel mai apropiat numar par prin schimbarea unlimului bit din ‘1’ in ‘0’ in cazul in care numarul este impar . Circuitul primeste ca intrare un numar fara semn pe 31 de biti (**input)** , un semnal de **enable** si furnizeaza ca iesire rezultatul operatiei de rotunjire pe 31 de biti ( **output** )

Semnale :

* intrari :
  + input : numar fara semn care urmeaza sa fie rotunjit
  + enable : semnal pe un bit care permite sau nu efectuarea operatiei
* iesiri :
  + output : rezultatul operatiei de rotunjire

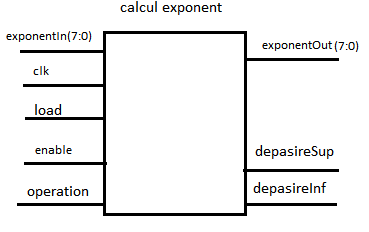


# 3.1.3 Calcul exponent

Componenta este folosita pentru incrementarea eponentului la operatia de aliniere a mantiselor si la cea de normalizare(incrementare resprectiv decrementare). Pentru fiecare deplasare la dreapta a mantisei, exponentul este incrementat cu unu. Pentru fiecare deplasare la stanga a mantisei, exponentul se decrementeaza cu unu. Aceste operatii pot produce depasiri superioare sau inferioare, depasiri ce sunt semnalate prin semnalele de iesire **depasireSup**  si depasireInf. Circuitul primeste un semnal **exponentIn**  si genereaza o iesire **exponentOut** cu rezultatul operatiei de alese prin intrarea **operation**. Circuitul va face o incarcare a operandului prin semnalul **load**, iar operatia propriuzisa se realizeaza prin setarea semnalului de enable.

Semnale :

* intrari :
  + exponentIn : exponentul care trebuie calculat
  + clk : semnal de ceas
  + load : se incarca operandul in circuit
  + enable : se efectueaza operatia propriu zisa
  + operation : determina operatia efectuata asupra operandului
* iesiri :
  + exponentOut : rezultatul operatiei
  + depasireSup : flag care indica o depasire superioara
  + depasireInf : flag care indica o depasire inferioara



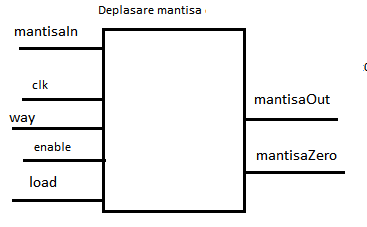
# 3.1.4 Deplasare mantisa

Circuitul de deplasare mantisa spre dreapta este folosit pentru alinierea punctelor zecimelale ale operanzilor inainte de a fi efectuata operatia selectata. Acest circuit este de asemenea folosit pentru deplasarea mantisei la stanga in cadrul operatiei de normalizare. Circuitul se comporta in principiu ca si cel de calculExponent, singura diferenta fiind faptul ca se realizeaza deplasari si nu adunari sau scaderi.

Circuitul primeste ca semnal de intrare (**mantisaIn**) mantisa asupra careia vom avea nevoie sa aplicam deplasarea. Semnalul **way** ne va da directia de deplasare(0 – stanga, 1 – dreapta).

Semnale :

* intrari :
  + mantisaIn : operandul care urmeaza sa fie deplasat
  + clk : semnal de ceas
  + way : determina directia deplasarii
  + enable : la activarea semnalului se efectueaza operatia de deplasare
  + load : la activarea semnalului se incarca operandul
* iesiri :
  + matisaOut : operandul dupa efectuarea operatiei de deplasare
  + mantisaZero : flag care indica faptul ca operandul este 0 dupa efectuarea operatiei de deplasare

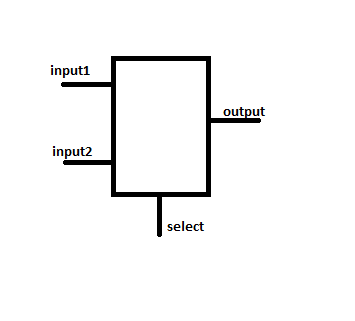


# 3.1.5 Multiplexor generic

Va fi nevoie de mai multe multiplexoare care primesc la intrari semnale de dimensiuni diferite . De aceea vom folosi un parametru generic care va indica dimensiunea intarilor si iesirilor multiplexorului . Astfel componenta va putea fi refolosita pentru toate multiplexoarele din schema bloc . Semnalele **input1**  si  **input2**  reprezinta intrarile circuitului iar semnalul **output** reprezinta intrarea selectata de multiplexor pentru a trece mai departe . Semnalul pe 1 bit **select** determina care intrare trece mai departe.

Semnale :

* intrari :
  + input1 : intrare de dimensiune variabila
  + input2 : intrare de dimensiune variabila
  + select : semnal care determina care intrare va fi trasmisa mai departe
* iesiri :
  + output : iesire determinata de semnalul **select**

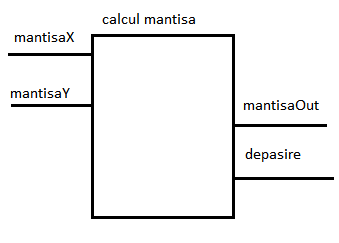


# 3.1.6 Calcul mantisa

Circuitul va aduna(scadea) mantisele in functie de operatie. Circuitul in sine face doar o adunare. In cadrul scaderii se presupune ca al doilea operand este in complement fata de 2. Iesirea **mantisaOut** va contine rezultatul operatiei asupra mantiselor. Semnalul **depasire** va fi 1 in cazul in care in urma operatiei va ramane un bit de carry in plus cu valoarea 1.

Semnale :

* intrari :
  + mantisaX : mantisa primului operand
  + mantisaY : mantisa celuilalt operand
* iesiri :
  + matisaOut : rezultatul calculului mantisei
  + depasire : flag care indica o depasire

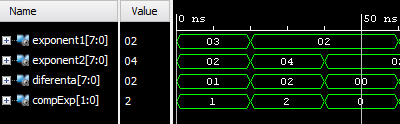


4. Rezutate experimentale

Pentru testarea componentelor unitatii de adunare si scadere in virgula mobila am elaborat module de simulare pentru fiecare dintre componentele unitatii .

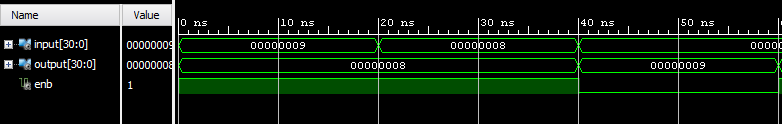
# 4.1 Unitate aritmetico-logica (UAL1)

Pentru unitatea aritmetico logica am folosit 3 cazuri de testare : cand primul numar este mai mare , cand al doilea numar este mai mare si cand numerele sunt egale . Functionarea corecta a acestei componente poate fi observata in urmatoarea imagine care ne prezinta forma de unda a semnalelor de intrare si iesire.



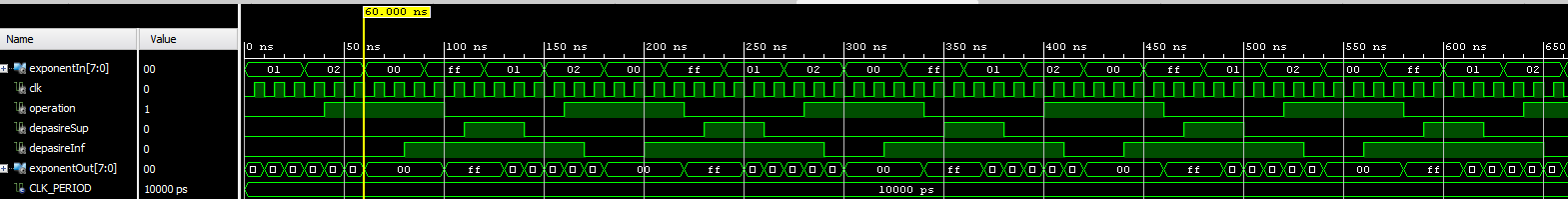
# 4.2 Circuit de rotunjire

Pentru circuitul de rotunjire am testat 3 cazuri : numar impar si semnal de enable activat , numar par si semnal de enble activat , numar impar si semnal de enable dezactivat . Rezultatul poate fi obseervat in urmatoarea imagine .



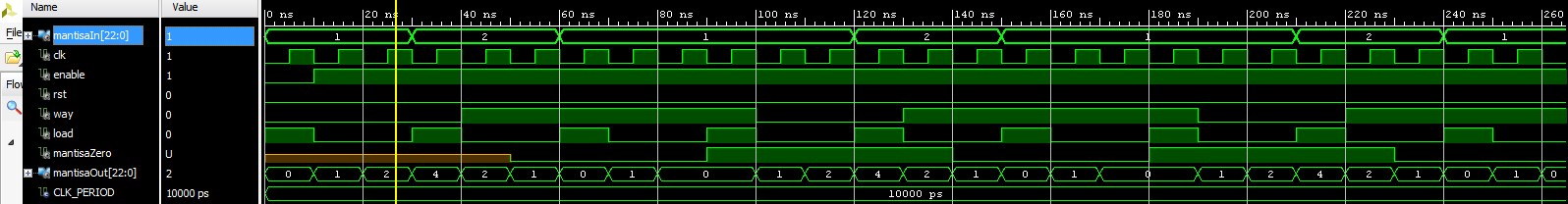
# 4.3 Circuit de calcul al exponentului

Pentru testarea acestui circuit am folosit doua valori normale(prin normale ma refer la valori ce nu produc depasiri in cadrul acestei simulari) si doua valoari ce produc depasiri(inferioare si superioare).



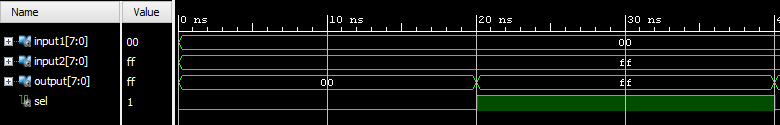
# 4.4 Circuit de deplasare a mantisei

Deplasarile cu o pozitie pot fi private ca o inmultire(deplasare stanga) sau o impartire(deplasare dreapta) a valorii initiale cu 2.



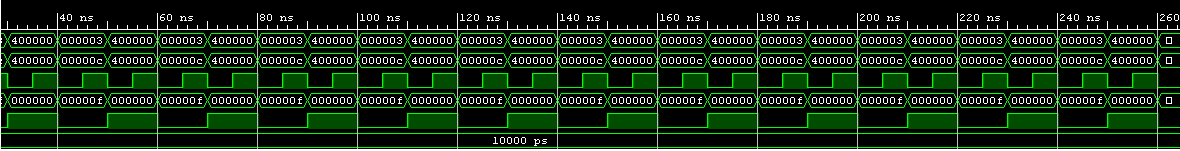
# 4.5 Multiplexor generic

In experimentul de mai jos se poate observa modul de functionare a multiplexorului generic instantiat in asa fel incat sa poata primi un numar pe 8 biti :



# 4.6 Calcul mantisa

Am folosit doua seturi de valori pentru simulare: cu sau fara generare de depasire.



# Bibliografie

[1] Informatii virgula mobila : <https://ro.wikipedia.org/wiki/Virgulă_mobilă>

[2]Figura reprezentare virgula mobila : <http://chortle.ccsu.edu/assemblytutorial/Chapter-30/IEEE754.jpg>

[3] https://en.wikipedia.org/wiki/Floating-point\_arithmetic

Informatii operatia de adunare :

[4]<http://andrei.clubcisco.ro/cursuri/1ii/cursuri/Aritmetica%20in%20virgula%20mobila..PDF>

[5]<https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/BinMath/addFloat.html>