**Facultatea de Automatica si Calculatoare**

**Domeniul: Calculatoare si Tehnologia Informatiei**

**Unitate de calcul în virgulă mobila: înmultirea**

**Disciplina: Structura sistemelor de calcul**

Student: Coman Vasile

Grupa: 30231

Profesor curs: Dr. Baruch Zoltan Francisc

Profesor indrumator: Eng. Dan Butiri

**Cuprins**

1. Rezumat.....................................................................................................3
2. Introducere................................................................................................3
3. Fundamentare teoretica............................................................................4
4. Proiectare si implementare........................................................................7
5. Rezultate experimentale............................................................................7
6. Concluzii.....................................................................................................7

Bibliografie......................................................................................................7

1. **Rezumat**

Standardul IEEE 754 este cea mai des intalnita modalitate de a reprezenta numere in virgula flotanta avand un rol important in operatiile efectuate de unitatile de calcul.

In implementarea proiectului s-au utilizat numere in simpla precizie pe 32 de biti si s-a descris algoritmul de inmultire a doua numere de acest tip.

S-a utilizat un sumator generic format din sumatoare elementare pe un bit pentru adunarea exponentilor si a bias-ului, un inmultitor matricial pentru inmultirea mantiselor si un circuit de normalizare.

1. **Introducere**

Tema proiectului consta in implementarea unui algoritm de inmultire a doua numere pe 32 de biti in virgula flotanta folosind limbajul de descriere hardware VHDL.

VHDL (abrevierea VHSIC HDL) este acronimul folosit pentru Very High Speed Integrated Circuit Hardware Description Language.

Este vorba despre un limbaj de descriere a hardware-ului (Hardware Description Language -HDL), destinat descrierii comportamentului și/sau arhitecturii unui "modul" electronic logic, cu alte cuvinte al unei funcțiuni logice combinatorii și/sau secvențiale. Alături de Verilog, este cel mai utilizat limbaj de proiectare a sistemelor electronice digitale. E una din uneltele principale pentru proiectarea circuitelor integrate moderne, aplicat cu succes in campul microprocesoarelor (DSP, acceleratoare grafice), în telecomunicatii (TV, celulare), automobile (navigație, sisteme de control al stabilitații) și altele.

Standardul, pentru VHDL, a apărut in 1987, denumit IEEE 1076 . Iar în 1993 iese o versiune actualizată.

Este utilizat în concepția asistată pe calculator (CAD din engleză, computer aided design) a circuitelor integrate (de exemplu ASIC) sau pentru configurarea FPGA-urilor.

În tehnologia informației, virgula mobilă (în engleză floating point) este unul din sistemele folosite pentru reprezentarea numerelor raționale ca șiruri de biți. Termenul de virgulă mobilă se referă la faptul că virgula care separă partea întreagă de cea fracționară se poate deplasa, adică poate fi plasată oriunde relativ la cifrele semnificative ale numărului. Poziția virgulei este indicată separat în cadrul reprezentării interne. Astfel, reprezentarea în virgulă mobilă poate fi considerată o adaptare la calculator a notației științifice. De-a lungul timpului, au fost folosite diverse reprezentări în virgulă mobilă. Astăzi, cea mai des întâlnită modalitate de reprezentare a numerelor în virgulă mobilă este cea definită de standardul IEEE 754-1985.

Avantajul reprezentării în virgulă mobilă față de cea în virgulă fixă este gama mai largă de valori reprezentate. Dacă în virgulă fixă se pot reprezenta numere cu un număr fix de cifre de o parte și de alta a virgulei, virgula mobilă permite sacrificarea preciziei (numărului de cifre fracționare cunoscut) pentru reprezentarea unor numere mai mari, și invers, reprezentarea mai precisă a numerelor mici. De exemplu, presupunând că într-o reprezentare în virgulă fixă numărul de cifre ale părții întregi este 6 și cel de cifre fracționare este 2, se pot reprezenta valori ca 123.456,75 sau 984,57, pe când în virgulă mobilă pot fi reprezentate la fel de bine numere ca 1,2345678 sau 1.234.567,8.

Viteza cu care un calculator efectuează calcule în virgulă mobilă este o măsură a performanței în multe domenii de aplicații. Aceasta se măsoară în megaFLOPS (milioane de operațiuni în virgulă mobilă pe secundă) sau gigaFLOPS. Performanțele supercalculatoarelor de la nivelul anului 2008 sunt de ordinul teraFLOPS sau petaFLOPS. Supercalculatorul IBM Roadrunner a atins 1,026 petaFLOPS, adică 1,026 milioane de miliarde de operațiuni în virgulă mobilă pe secundă.

Necesitatea reprezentarii in calculator a numerelor foarte mari sau foarte mici, cu o precizie ridicata a condus la reprezentarea in virgula mobila (VM) . Reprezentarea in format VM utilizeaza reprezentarea stiintifica :

r = m x b^E

b - baza

E – exponent

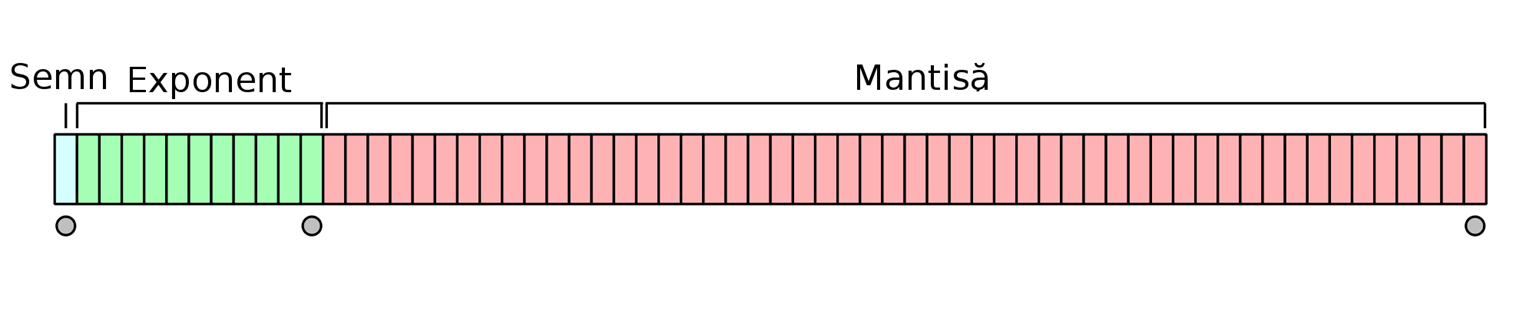
m – mantisa

Pentru implementare folosim placa de dezvoltare Nexys 4 DDR rezultatul operatiei fiind vizibil pe afisorul cu 7 segmente iar pentru introducerea operanzilor vom folosi extensia PmodKYBD.

In continuare vom descrie urmatoarele capitole:

* Fundamentare teoretica: detaliaza notiunile expuse anterior
* Proiectare si implementare: prezinta etapele de implementare a proiectului
* Rezultate experimentale: validarea proiectului prin testare si verificare
* Concluzii: contine observatii cu privire la desfasurarea proiectului

1. **Fundamentare teoretica**

Reprezentarea in virgula mobila (virgula flotanta) este utilizata pentru a reprezenta si opera cu numere reale, permitand o precizie ridicata si o gama larga de valori. Un numar este reprezentat in virgula mobila printr-un cuvant de n biti, continand urmatoarele campuri:Semn reprezinta bitul de semn, 0 pentru pozitiv si 1 pentru negativ.

Mantisa reprezinta cifrele fractiei.

Expoenent reprezinta puterea la care trebuie ridicata baza si care inmulteste cifrele semnificative ale nr.

R= M\*b^E

B = baza, este 2 sau o putere a lui 2, nu este reprezentata in calculatorul numeric, dar se tine cont de valoarea bazei atunci cand se efectueaza operatii aritmetice. La un anumit calculator numeric valoarea bazei este aceeasi pentru toate numerele reale reprezentate. Reprezentarea numerelor in calculator se face prin cuvinte binare cu lungime egala cu lungimea registrelor de stocare.

Precizia de reprezentare a unui numar real este data in primul rand de numarul de biti folositi pentru reprezentarea mantisei. Domeniul maxim de reprezentare este determinat de valoarea adoptata pentru baza si de numarul de biti folositi pentru a reprezenta exponentul. Spre deosebire de numerele intregi, numerele reale se reprezinta cu o anumita aproximatie in calculator. Daca pentru reprezentarea numerelor reale in calculator se folosesc n biti, se vor putea reprezenta maxim

2^n numere, dintr-un anumit interval continuu ±R al numerelor reale.

Ca urmare, numerele reale sunt reprezentate cu o anumita eroare, determinata de numarul limitat de biti folosit pentru reprezentare. Eroarea de reprezentare depinde de distanta dintre doua numere succesive reprezentabile cu cei n biti. Toate numerele reale cuprinse intre cele doua valori vor trebui aproximate prin una din cele doua valori.

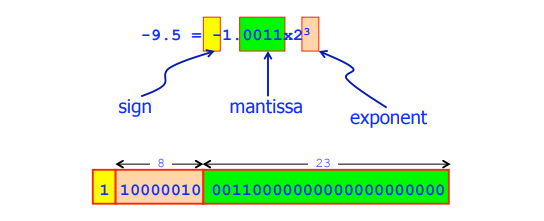
Pentru reprezentarea mantisei numerelor in virgula mobila se foloseste forma normalizata in care, pentru reprezentarea in marime si semn, prima pozitie binara dupa virgula este diferita de zero. Normalizarea restrange marimea valorii absolute a mantisei binare. Normalizarea in binar este usor de facut, prin deplasarea mantisei catre dreapta sau stanga si incrementarea respectiv decrementarea corespunzatoare a exponentului.

Valoarea numarului real reprezentat in virgula mobila este data de expresia:

R=(-1)^s\*m\*B^e

Pentru alegerea unui format in VM trebuie realizat un compromis intre dimensiunea mantisei si cea a exponentului. Cresterea dimensiunii mantisei va conduce la cresterea preciziei numerelor, iar cresterea dimensiunii exponentului va conduce la cresterea domeniului numerelor care pot fi reprezentate. Singura cale de a creste atat precizia, cat si domeniul numerelor, este de a utiliza un numar mai mare de biti pentru reprezentare. Cele mai multe calculatoare utilizeaza cel putin doua formate, in simpla precizie (pe 32 de biti), si dubla precizie (pe 64 de biti)

Standardul IEEE 754 defineste urmatoarele formate sau precizii: precizie simpla, precizie simpla extinsa, precizie dubla si precizie dubla extinsa.



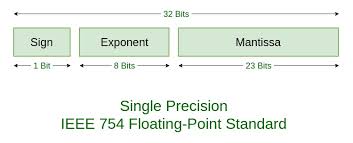
Numarul in binar trebuie normalizat, partea intreaga trebuie tot timpul sa fie egala cu 1.

La exponent se aduna 127( bias).

00000000 este folosit pt valori nenormalizate.

11111111 este flosit pentru infinit sau NaN(not a number).

Valoarea 0 nu poate fi normalizata. Pentru 0 toti bitii sunt 0.



Valoarea numarului poate fi:

Daca 0<Exponentul<255 atunci R=(-1)^Sign\*Mantissa\*2^(Exponent-127)

Daca Exponent=0 si Mantissa=0 atunci R=0

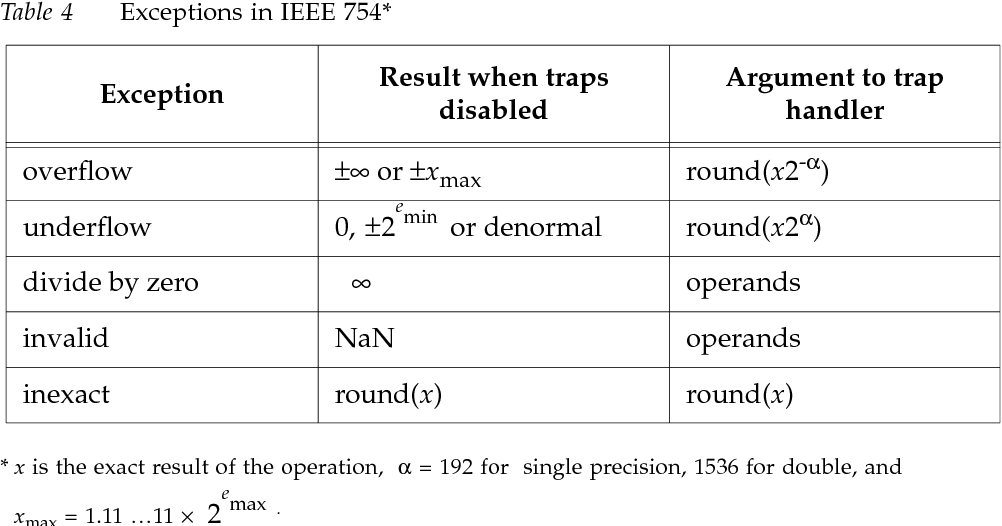
Daca Exponent=0 si Mantissa!=0 sau Exponent=255 atunci eroare

Tipuri de rotunjire:

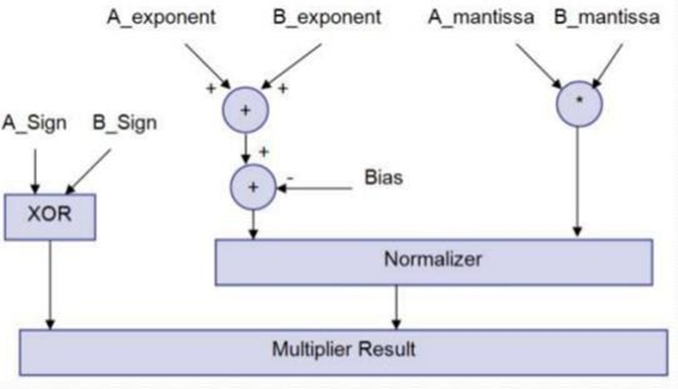
* Rotunjire spre 0;
* Rotunjire spre +infinit;
* Rotunjire spre -infinit;
* Rotunjire la cel mai apropiat nr reprezentabil;

Ultimul mod este cel implicit, prevazut pentru situatiile in care numarul se afla exact la jumatatea intervalului dintre doua reprezentari in VM. Acest mod efectueaza rotunjirea la un numar par.

Standardul IEEE defineste cinci tipuri de exceptii: depasire inferioara, depasire superioara, impartire la zero, rezultat inexact si operatie invalida. In mod implicit, la aparitia unei asemenea exceptii, este setat un indicator si calculele continua. Exceptiile de depasire inferioara, depasire superioara si impartire la zero sunt prevazute la majoritatea sistemelor de calcul in VM. Exceptia de rezultat inexact apare atunci cand rezultatul unei operatii trebuie rotunjit.



Operatia de inmultire



Aceasta operatie se desfasoara in 4 pasi:

* Inmultirea(fara semn) a mantiselor, trebuie sa tinem cont de partea intreaga, implicit de normalizare, nr de biti rezultat este de 2 ori mai mare decat dimensiunea operanzilor(48 biti);
* Normalizarea rezultatului, exponentul poate fi modificat;
* Adunarea exponentilor, se tine cont de bias;
* Calcularea semnului;

Exemplu

A=-18.0;

B=9.5;

Reprezentarea binara:

A=-10010.0;

B=+1001.0;

Reprezentarea normalizata:

A=-1.001x2^4;

B=+1.0011x2^3;

Reprezentarea IEE:

A=1 10000011 00100000000000000000000;

B=0 10000010 00110000000000000000000;

Inmultirea mantiselor: extragem mantisele si adaugam un 1 la cel mai semnificativ bit pentru normalizare:

100100000000000000000000 100110000000000000000000

Rezultatul de 48 biti este:

0x558000000000

Doar cei mai semnificativi biti sunt utili: dupa normalizare( eliminare celui mai semnificativ 1) obtinem un rezulta de 23 de biti. Aceasta normalizare poate duce la o corectie a rezultatului exponentului.

In acest caz obtinem:

01 01010110000000000000000 0000000000000000000000

Adunarea exponentilor: exponentul rezultatului este egal cu suma exponentilor operanzilor. Un 1 poate fi adaugat daca este nevoie de normalizarea mantisei rezultatului( in acest caz nu).

Exponentii avand bias adunat trebuie scazut inainte de a face adunarea si apoi trebuie adunat din nou.

Er = (Ea-127) + (Eb-127) + 127 = Ea + Eb – 127

Ea 10000011

Eb 10000010

-127 10000001

Er 10000110

Calcularea semnului: se face sau exclusiv intre semnele operanzilor

Sr=Sa xor Sb=1 xor 0=1

Rezultatul final:

1 10000110 01010110000000000000000

AxB = -18.0x9.5 = -1.0101011x2^(134-127) = -10101011.0 = -171.0

1. **Proiectare si implementare** Implementarea acestui proiect a necesitat descrierea urmoarelor componente hardware:

* Un sumator generic;
* Un inmultior matricial;
* Un circuit de normalizare a mantisei si a exponentului;
* Componenta principala.

Sumator generic

Acest sumator generic este format din mai multe sumatoare elementare pe 1 bit. Prezinta 3 intrari( operanzii si transportul de intrare) si 2 iesiri(rezultatul si transportul de iesire). Este implementat generic pentru a putea modifica dimensiunea semnalelor de intrare si de iesire in functie de nevoi.

Inmultitor matricial

Acest circuit calculeaza produsul dintre operanzi utilizand o structura matriciala (fiecare bit a lui Y este inmultit cu fiecare bit a lui X) si dupa calcularea produselor partiale se cascadeaza sumatoare elementare pentru a determina rezultatul final. Algoritmul este foarte asemanator cu cel al inmultirii a 2 numere pe foaie.

Circuitul de normalizare

Acest circuit verifica daca exponentul si mantisa este normalizata.

Componenta principala

Aceasta componenta primeste semnalul de ceas si cei 2 operanzi avand ca iesire produsul si un semnal care ne indica eroarea in cazul in care s-a depasit limita.

Pasi:

Se selecteaza exponentii de la bitul 30 la bitul 23 la care se concateneaza cate un bit de 0 la inceput

Se calculeaza semnul ca si un sau exclusiv intre bitii 31 ai operanzilor

Se aduna exponentii folosindu-se sumatorul generic

La acest rezultat se aduna bias-ul in complement fata de 2 adica se aduna -127 in binar

Dupa care se selecteaza mantisele de la bitul 22 la 0 la care se concateneaza cate un 1

Mantisele sunt inmultite folosindu-se inmultitorul matricial.

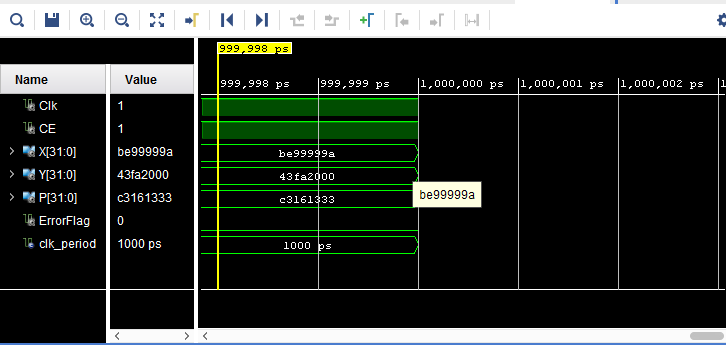
Se normalizeaza mantisa si exponentul astfel: daca mantisa rezultata pe 48 de biti in urma inmultirii celor doua mantise are bitul cel mai semnificativ 0 atunci inseamna ca este deja normalizata si se iau cei mai semnificativi 23 de biti de la (45 la 23) in caz contrar mantisa se deplaseaza spre stanga si se selecteaza bitii(46-24) si se aduna la exponent valoarea 1. Aici poate aparea overflowul in urma incrementarii exponentului.

La final rezultatul este transcris in semnalul de iesire dupa aceeasi regula: 31-bit de semn, 30-23 exponentul calculat si 22-0 mantisa. Semnalul de detectare a erorii poate fi activat(1) in cazul in care in urma normalizarii apare depasire peste limita superioara(overflow).

1. **Rezultate experimentale**

Testarea acestui proiect a fost realizata doar in simulator utilizandu-se un banc de test. In aceasta componenta am initializat 2 operanzi pentru calcularea produsului acestor(-0.3 si 500.25) rezultatul fiind -150.075(c3161333)

In urmatoarea imagine putem observa ca echivalentul acestor numere doar ca in hexazecimal apar si in simulator deci implementarea acestui proiect este corecta.



Pentru implementarea si pe placa sunt necesare componente suplimentare cum ar fi:

-debouncer pentru butoane;

-afisor;

-unitate de control;

-convertire operanzi dati;

-registre de stocare.

1. **Concluzii**

In concluzie aceasta implementare ofera rezultate corecte in simulator dar pentru a putea fi implementat pe o placa este necesar de a fi implementate componentele mentionate mai sus.

Ca si dezvoltari ulterioare se poate extinde formatul din simpla precizie in dubla precizie sau se poate adauga aceasta operatie ca facand parte din setul de instructiuni al unui microprocesor.

**Bibliografie**

[http://users.utcluj.ro/~baruch](http://users.utcluj.ro/~baruch/ro/pages/cursuri/structura-sistemelor-de-calcul)

<http://lslwww.epfl.ch/pages/teaching/cours_lsl/sl_info/FPMultiplier.pdf>

<https://www.doc.ic.ac.uk/~eedwards/compsys/float/>

<http://ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-7-130-136.pdf>