Unitate de calcul în virgulă mobilă: extragerea radăcinii pătrate

Mureșan Alexandru-Dorian

Cuprins

**1. Introducere**.................................................................................................................,................................1

1.1 Context...........................................................................................................................................1

1.2 Soluție propusă...............................................................................................................................1

1.3 Plan de proiect.................................................................................................................................2

**2. Studiul bibliografic**.....................................................................................................................................2

2.1 Prezentare generală.........................

2.2 Utilizări..............

2.3 Arhitectura setului de instructiuni...............

2.4 Date si memorie............

2.5 Componente.................

2.6 Endiannes.............

2.7 Adresarea memoriei................

**3. Analiză**..........................................................................................................................................................3

**4. Plan de proiectare**.......................................................................................................................................5

**5. Implementare**..............................................................................................................................................7

**6. Testare și validare**.......................................................................................................................................8

**7. Concluzii**......................................................................................................................................................9

**8. Bibliografie** ...............................................................................................................................................10

1. **Introducere**
   1. **Context**

Descriere generală: Unitatea de calcul în virgulă mobilă reprezintă o componentă esențială a procesorului (CPU) care este responsabilă pentru efectuarea operațiilor matematice care implică numere în format virgulă mobilă. Extracția rădăcinii pătrate dintr-un număr în virgulă mobilă este o operație matematică complexă și de mare importanță în domeniul calculului numeric, științific și graficii computerizate. Această operație necesită algoritmi și hardware specializați pentru a asigura acuratețe și eficiență.

* 1. **Soluție propusă**

Proiectarea și dezvoltarea unei unități de calcul în virgulă mobilă pentru extragerea rădăcinii pătrate reprezintă un proces complex care necesită expertiză în arhitectura procesoarelor și matematică numerică. O atenție deosebită la detaliile hardware este esențială pentru a obține un FPU dedicat eficient și precis pentru această operație matematică specifică.

1. Algoritmi Optimizați: Pentru a realiza extragerea rădăcinii pătrate într-un mod eficient, este esențial să dezvoltați sau să selectați algoritmi optimizați care să minimizeze numărul de operații necesare și să ofere precizie ridicată. Un astfel de algoritm poate implica utilizarea metodei Newton-Raphson sau a altor metode iterative care converg rapid către valoarea corectă a rădăcinii pătrate.
2. Arhitectura Hardware Dedicată: Pentru a implementa algoritmul de extragere a rădăcinii pătrate, este necesară dezvoltarea unei arhitecturi hardware dedicate în cadrul FPU. Aceasta ar putea include unități funcționale specializate, registre de control, și circuite pentru efectuarea operațiilor matematice.
3. Precision Hardware: Precizia este un aspect crucial în extragerea rădăcinii pătrate, mai ales când lucrăm cu numere în virgulă mobilă. Hardware-ul trebuie să fie capabil să gestioneze cu precizie virgula mobilă, să evite eroarea de trunchiere și să ofere rezultate corecte, cu cât mai multe cifre semnificative.
4. Control de Excepții: FPU trebuie să fie capabil să trateze excepțiile care pot apărea în timpul extragerii rădăcinii pătrate. De exemplu, în cazul unui număr negativ, poate fi necesar să se declanșeze o excepție și să se gestioneze corect această situație.

Pentru indicarea diferitelor condiţii de excepţie, ca în cazul operaţiilor nedefinite de forma ∞/∞, ∞–∞, ∞ ∗ 0, 0/∞, 0/0, sau extragerea rădăcinii pătrate dintr-un număr negativ, s-a prevăzut un format special, care nu reprezintă un număr obişnuit, fiind numit NaN (Not a Number). Exponentul are valoarea maximă posibilă, iar mantisa este diferită de 0. Astfel, există o clasă întreagă de valori NaN.

1. Optimizare a Performanței: Într-un mediu de calcul modern, performanța este esențială. Dezvoltarea hardware-ului ar trebui să includă optimizări pentru a reduce latența și pentru a permite FPU să efectueze extrageri de rădăcini pătrate într-un timp cât mai scurt.
2. Integrare cu Arhitectura CPU: FPU trebuie să fie perfect integrată cu arhitectura generală a CPU-ului. Aceasta implică dezvoltarea unor mecanisme de interfațare adecvate pentru a permite transferul datelor între FPU și restul CPU-ului.
3. Testare și Validare: Odată ce hardware-ul este dezvoltat, acesta trebuie supus unor proceduri de testare și validare riguroase pentru a se asigura că funcționează corect și furnizează rezultate precise.
4. Documentare Tehnică: Un aspect important este crearea de documentație tehnică adecvată pentru a descrie arhitectura FPU, instrucțiunile suportate și modul de utilizare.

**1.3 Plan de proiect**

1.3.1 Studiul Bibliografic:

Efectuarea unui studiu aprofundat al literaturii pentru a identifica algoritmii eficienți și optimizați pentru extragerea rădăcinii pătrate în format virgulă mobilă.

Investigarea metodelor matematice care pot fi aplicate într-o unitate de calcul în virgulă mobilă pentru a obține rezultate precise.

1.3.2 Analiză:

Analiza cerințelor de performanță și precizie pentru operația de extragere a rădăcinii pătrate în diverse aplicații.

Evaluarea resurselor hardware disponibile pentru implementarea algoritmului într-o unitate de calcul în virgulă mobilă.

**Reprezentarea numerelor în virgulă mobilă:**

Primul aspect pe care trebuie să înțelegem este cum putem să reprezentăm numerele cu virgulă în formă binară. Ca și în cazul numerelor întregi, valorile fracționare se reprezintă pe un anumit număr de biți.

Principalele obiective ale unei reprezentări pe un anumit număr de biți sunt:

* posibilitatea de a reprezenta cât mai multe valori intre valoarea minimă și valoarea maximă
* o precizie cât mai bună a valorilor (numărul maxim de cifre după virgulă)

Pentru a reprezenta valorile fracționare vom folosi **Reprezentarea în virgulă mobilă** (**Floating Point Representation**). În această reprezentare, numerele au următoarea structură:

[](http://elf.cs.pub.ro/asm/wiki/_detail/laboratoare/lab12-equation1.png?id=laboratoare%3Alaborator-12)

După cum putem observa mai sus, valorile trebuie transformate astfel încât partea întreagă să fie 1. Această formă poartă numele de formă normală, iar operația de transformare în această formă poartă numele de **normalizare**.

În forma binară, valorile se reprezintă astfel:[A black and pink rectangle with black lines

Description automatically generated](http://elf.cs.pub.ro/asm/wiki/_detail/laboratoare/2000px-ieee_754_double_floating_point_format.png?id=laboratoare%3Alaborator-12)

**Semnul** este dat de primul bit din reprezentarea binară: 1 = negativ, 0 = pozitiv.  
**Mantisa** dă partea fracționară a numărului în forma normală. Numărul de biți pe care mantisa este reprezentată dă precizia maximă a reprezentării.  
**Baza** este de obicei 2, 10 sau 16 și este dată de standardul de reprezentare ales.  
**Exponentul** dă valoarea la care este ridicată baza și numărul de biți pe care este reprezentat dă valorile valorile maxime și minime ce pot fi reprezentate.

**Standarde de reprezentare în virgulă mobilă:**

Cele mai folosite standarde de reprezentare în virgulă mobilă sunt cu precizie simplă (**Single Precision**) și cu precizie dublă (**Double Precision**).

Reprezentarea cu **precizie simplă** presupune folosirea a 32 de biți și corespunde valorilor float din limbajul C. În acest caz baza folosită este 2, exponentul are 8 biți, iar restul de 23 de biți corespund mantisei.

Reprezentarea cu **precizie dublă** presupune folosirea a 64 de biți și corespunde valorilor de tip double. În acest caz, baza este 2, exponentul are 11 biți, iar restul de 52 de biți corespund mantisei.

1.3.3 Descriere și Implementare:

Această secțiune a proiectului este crucială și necesită expertiză în dezvoltarea hardware-ului și în evaluarea matematică. Scopul final este de a oferi o unitate de calcul în virgulă mobilă dedicată și eficientă pentru extragerea rădăcinii pătrate, care să îndeplinească standardele de precizie și performanță cerute pentru aplicația respectivă.

1. Dezvoltarea sau Adaptarea Algoritmului:

* În acest pas, se alege sau se dezvoltă un algoritm optimizat pentru extragerea rădăcinii pătrate în format virgulă mobilă. Algoritmul trebuie să fie adecvat pentru implementarea hardware și să aibă un nivel de precizie corespunzător.
* Sunt luate în considerare aspecte matematice, cum ar fi metoda Newton-Raphson sau metode iterative, pentru a ajunge la o soluție eficientă. Adaptarea algoritmului pentru formatul în virgulă mobilă este esențială pentru a asigura precizia și performanța.

1. Proiectarea Unității de Calcul în Virgulă Mobilă (FPU):

* Această etapă implică proiectarea hardware-ului pentru a implementa algoritmul selectat. Componentele hardware necesare includ unități funcționale pentru operațiile matematice, registre de control, unități de stocare temporară a datelor, și unități pentru gestionarea preciziei.
* FPU trebuie să fie integrată în arhitectura generală a CPU-ului și să aibă interfețe corespunzătoare pentru a comunica cu restul sistemului.

1. Implementarea și Testarea Hardware-ului:

* După proiectare, se trece la etapa de implementare, care presupune construcția fizică a hardware-ului. Acest lucru poate fi realizat fie pe o placă de dezvoltare, fie pe un cip (chip) dedicat.
* După implementare, se efectuează teste hardware pentru a verifica corectitudinea funcționării. Aceste teste implică furnizarea datelor de intrare și verificarea rezultatelor obținute de hardware pentru a se asigura că algoritmul este implementat corect.

1. Evaluarea Performanței, Acurateții și Eficienței

* După testarea hardware, se efectuează o evaluare detaliată a performanței, acurateții și eficienței soluției propuse. Se compară rezultatele obținute de hardware cu valorile teoretice corecte pentru a verifica precizia.
* Se măsoară și se evaluează performanța în termeni de latență și putere consumată. Dacă este necesar, se fac optimizări pentru a îmbunătăți performanța sau eficiența.

1. **Studiul bibliografic**
   1. **Prezentare generală**

În cadrul studiului bibliografic pentru proiectarea unei unități de calcul în virgulă mobilă (FPU) specializate pentru extragerea rădăcinii pătrate, este crucial să se analizeze în detaliu aspectele hardware care sunt relevante pentru această operație. Studiul bibliografic ar trebui să înceapă cu o prezentare generală a arhitecturii hardware, care include caracteristicile de bază ale unui FPU, cum ar fi tipul de virgulă mobilă (de exemplu, IEEE 754), capacitatea de calcul, precizia și performanța.

**Metoda Newton-Raphson:**

Metoda Newton-Raphson este o tehnică iterativă pentru găsirea rădăcinilor unei funcții. Pentru a extrage rădăcina pătrată a unui număr x, formula este:

A math equation with numbers and symbols

Description automatically generated

**Metoda de căutare binară:**

Metoda de căutare binară este o altă tehnică eficientă pentru extragerea rădăcinii pătrate. Algoritmul funcționează prin reducerea intervalului în care se găsește rădăcina pătrată prin compararea cu jumătatea intervalului curent. Acest lucru se repetă până când precizia dorită este atinsă.

* 1. **Utilizări**

Trebuie să se exploreze utilizările potențiale ale unei unități de calcul în virgulă mobilă specializate pentru extragerea rădăcinii pătrate. Aceasta poate include aplicații din domenii precum grafica computerizată (pentru a accelera operațiile de transformare și rasterizare), simulări științifice (pentru calcule complexe în fizică sau inginerie) și în alte contexte în care extragerea rădăcinii pătrate este o operație frecventă.

* 1. **Arhitectura setului de instrucțiuni**

Studiul bibliografic trebuie să se concentreze pe arhitectura setului de instrucțiuni (ISA - Instruction Set Architecture) folosită pentru FPU. Acest lucru poate include detalii despre tipurile de instrucțiuni utilizate pentru operațiile în virgulă mobilă, modul în care instrucțiunile sunt codificate și cum sunt executate în hardware.

* 1. **Date si memorie**

Analiza ar trebui să acopere modul în care datele în virgulă mobilă sunt reprezentate și stocate în memoria fizică. Este important să se înțeleagă cum valorile în virgulă mobilă sunt normalizate, cum se realizează conversiile între formatul intern și reprezentarea externă și cum este gestionată precizia în timpul operațiilor.

* 1. **Componente**

Trebuie să se examineze componentele hardware ale FPU-ului, inclusiv registrele speciale, unitățile de calcul, registrele de stare și orice alte module care contribuie la operația FPU. De asemenea, este important să se studieze modul în care aceste componente lucrează împreună pentru a realiza extragerea rădăcinii pătrate.

* 1. **Endiannes**

Studiul bibliografic trebuie să abordeze problema endianness-ului, care se referă la modul în care datele cu mai multe octeți sunt stocate în memorie. Este important să se știe cum este gestionată endianness-ul în FPU și cum poate influența operațiile cu numere în virgulă mobilă.

* 1. **Adresarea memoriei**

Trebuie să se investigheze modul în care FPU accesează și adresează memoria pentru a citi sau scrie date. Acest aspect poate influența performanța și eficiența operațiilor de extragere a rădăcinii pătrate, în special atunci când se lucrează cu mari volume de date.

Un studiu bibliografic detaliat asupra acestor aspecte hardware este esențial pentru proiectarea unei unități de calcul în virgulă mobilă eficiente și precise pentru extragerea rădăcinii pătrate. Aceste cunoștințe vor servi ca fundație solidă pentru dezvoltarea hardware-ului specializat necesar pentru această operație matematică complexă.

1. **Analiză**

**3.1 Metode de implementare**

**Metoda Newton-Raphson:**

Metoda Newton-Raphson este o tehnică iterativă pentru găsirea rădăcinilor unei funcții. Pentru a extrage rădăcina pătrată a unui număr x, formula este:

A math equation with numbers and symbols

Description automatically generated

Unde:

A black background with white text

Description automatically generated

Aplicarea metodei Newton-Raphson pentru numărul 12:

A screenshot of a math problem

Description automatically generated

In VHDL:

A screenshot of a computer code

Description automatically generated

**Metoda de căutare binară:**

Metoda de căutare binară este o altă tehnică eficientă pentru extragerea rădăcinii pătrate. Algoritmul funcționează prin reducerea intervalului în care se găsește rădăcina pătrată prin compararea cu jumătatea intervalului curent. Acest lucru se repetă până când precizia dorită este atinsă.

A screenshot of a computer program

Description automatically generated

Aplicarea metodei căutării binare pentru numărul 12:

A screenshot of a math test

Description automatically generated

In VHDL:

A screenshot of a computer code

Description automatically generated

* 1. **Componente**

Implementarea algoritmilor de extragere a rădăcinii pătrate într-un limbaj de descriere hardware precum VHDL presupune un set de instrucțiuni hardware. În esență, pentru a descrie funcționarea acestor algoritmi în hardware, vom avea nevoie de componente precum:

1. **Unitatea de Comparare**: Pentru metoda de căutare binară, va fi nevoie de o unitate care să poată compar cu numărul dat și să genereze semnale care să indice dacă este mai mare, mai mic sau egal cu numărul dat.

2. **Unitatea Aritmetică și Logică (ALU):** În ambele metode, vom avea nevoie de operații de adunare, înmulțire și împărțire. Aceste operații sunt folosite pentru a actualiza valorile în cadrul iterațiilor.

3. **Unitatea de Împărțire:** Pentru metoda Newton-Raphson, care implică împărțire, va fi necesară o unitate de împărțire pentru a realiza operația de / *xn*

4. **Registrul:** Pentru stocarea și actualizarea valorilor *xn* , *xn+1* ,left, right, mid și a altor variabile temporare utilizate în algoritmi.

5. **Unitatea de Control:** Aceasta coordonează fluxul de date și controlează secvența de execuție a operațiilor. Ea decide care operații trebuie să fie realizate în fiecare etapă și controlează actualizarea registrului.

6. **Unitatea de Comparare și Decizie:** Pentru a lua decizii în funcție de rezultatul comparațiilor, cum ar fi determinarea dacă trebuie să ajustăm intervalul în cazul metodei de căutare binară.

Acestea sunt componente generale care pot fi utilizate pentru a implementa algoritmii menționați. Totuși, implementarea detaliată depinde de arhitectura specifică a sistemului și de tehnologiile utilizate. De exemplu, dacă implementați aceste algoritmi pe un FPGA (Field-Programmable Gate Array), puteți utiliza resurse specifice FPGA, cum ar fi blocurile DSP (Digital Signal Processing) pentru operații aritmetice eficiente.

**3.2.1 Tipuri de instrucțiuni:**

Pentru realizarea proiectul vom avea nevoie de urmatoarele instructiuni care ne vor ajuta la implementarea acestuia:

* Instrucțiuni de încărcare(LD): Pentru a încărca datele BCD de intrare în registrele de lucru ale circuitului.
* Instrucțiuni de împărțire(DIV): Pentru a iniția operația de împărțire zecimală și pentru a specifica divizorul.
* Instrucțiuni de comparare(CMP): Pentru a compara restul parțial cu valori de referință și pentru a decide următoarea cifră a câtului.
* Instrucțiuni de adunare și scădere BCD(ADD / SUB): Pentru a efectua adunări și scăderi de numere BCD în funcție de cifrele câtului calculate.
* Instrucțiuni de deplasare și transfer de date(MOV): Pentru a manipula datele și pentru a muta informații între registre.
* Instrucțiuni de control(CTRL): Pentru a gestiona execuția circuitului, cum ar fi începerea sau oprirea operației de împărțire.
* Instrucțiuni de salt(JMP): Pentru a controla fluxul de execuție al programului și pentru a gestiona situații speciale sau cazuri de eroare.

**3.2.2 Formatul instrucțiunilor:**

Formatul instrucțiunilor menționate este:

* Instrucțiuni de încărcare (LD):

Format: LD destinatie, sursa

Explicație: Încarcă valoarea de la adresa specificată de sursă în destinație.

* Instrucțiuni de împărțire (DIV):

Format: DIV destinatie, operand1, operand2

Explicație: Împarte valorile specificate de operand1 și operand2, rezultatul fiind stocat în destinație.

* Instrucțiuni de comparare (CMP):

Format: CMP operand1, operand2

Explicație: Compară valorile dintre operand1 și operand2, setează starea de flag-uri corespunzătoare.

* Instrucțiuni de adunare și scădere BCD (ADD / SUB):

Format ADD: ADD destinatie, operand1, operand2

Explicație ADD: Adună valorile specificate de operand1 și operand2, rezultatul fiind stocat în destinație.

Format SUB: SUB destinatie, operand1, operand2

Explicație SUB: Scade operand2 din operand1, rezultatul fiind stocat în destinație.

* Instrucțiuni de deplasare și transfer de date (MOV):

Format: MOV destinatie, sursa

Explicație: Transferă valoarea de la sursă în destinație.

* Instrucțiuni de control (CTRL):

Format: CTRL operatie

Explicație: Realizează o operație de control specifică, cum ar fi pornirea sau oprirea unui circuit sau a unei componente.

* Instrucțiuni de salt (JMP):

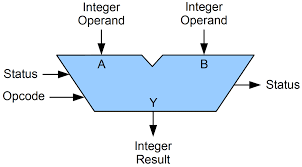
Format: JMP adresa

Explicație: Transferă controlul la adresa specificată.

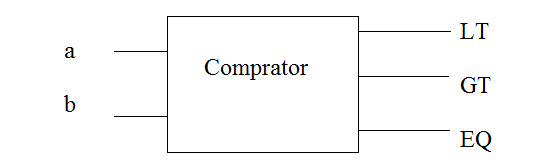
1. **Plan de proiectare**

Pentru a implementa algoritmii menționați (Newton-Raphson, căutare binară) și instrucțiunile precizate (încărcare, împărțire, comparare, adunare și scădere BCD, deplasare și transfer de date, control, salt) într-un limbaj de descriere hardware precum VHDL, veți avea nevoie de diverse componente hardware pentru a realiza operațiile necesare. Iată o listă generală de componente pe care le-ați putea utiliza în proiectul dumneavoastră:

1. ALU (Unitate Aritmetică și Logică): Componenta care efectuează operații aritmetice (adunare, înmulțire, împărțire), logice (și, sau, xor) și alte operații matematice.



1. Comparator: O componentă care compară două valori și generează semnale sau stări de flag-uri în funcție de rezultatul comparației.



1. Unitate de Împărțire: Pentru a realiza operațiile de împărțire necesare în metoda Newton-Raphson.
2. Registrul: O componentă de memorie care stochează temporar datele în timpul execuției algoritmului.
3. Unitate de Control: O componentă care coordonează secvența de operații, decidând care operații trebuie să fie realizate în fiecare ciclu de ceas.
4. Multiplexor (MUX) și Demultiplexor (DEMUX): Pentru a multiplexa și demultiplexa semnale între diferitele componente ale sistemului.
5. Unitate de Stocare a Instrucțiunilor: O componentă care stochează instrucțiunile de program.
6. Contor de Program: Un contor care ține evidența instrucțiunilor care trebuie să fie executate într-o anumită ordine.
7. Unitate de Intrare/Ieșire: Dacă proiectul implică interacțiunea cu mediul exterior (cum ar fi citirea datelor sau afișarea rezultatelor), veți avea nevoie de componente pentru aceste operațiuni.
8. Blocuri DSP (Procesare Digitală a Semnalului): Dacă se dorește accelerarea operațiilor aritmetice și de împărțire, se pot utiliza blocurile DSP disponibile pe unele dispozitive FPGA.

**4.1 Diagrame de flux pentru algoritmi**

Diagrama de Flux pentru Metoda Newton-Raphson:

* Inițializare:
* Se începe cu o formă de decizie pentru a alege valoarea de pornire.
* Se trece la un bloc care reprezintă procesul de inițializare a variabilelor.
* Iterație:
* Se utilizează o buclă pentru a reprezenta iterațiile algoritmului.
* În fiecare iterație, se efectuează operații de adunare, înmulțire și împărțire conform metodei Newton-Raphson.
* Se actualizează variabilele pentru a reflecta valorile calculate.
* Ieșire:
* altă formă de decizie verifică dacă s-a atins o precizie suficientă sau dacă s-a atins numărul maxim de iterații.
* Dacă da, se iese din buclă și se afișează rezultatul.

Diagrama de Flux pentru Metoda de Căutare Binară:

* Inițializare Interval:
* Se inițializează valorile pentru a delimita intervalul de căutare.
* Se efectuează o operație de împărțire pentru a calcula mijlocul intervalului.
* Iterație:
* Se intră într-o buclă care reprezintă iterațiile algoritmului.
* În fiecare iterație, se efectuează operații de însumare, înmulțire și comparație.
* Se actualizează intervalul în funcție de rezultatul comparației.
* Ieșire:
* formă de decizie verifică dacă s-a atins o precizie suficientă sau dacă s-a atins numărul maxim de iterații.
* Dacă da, se iese din buclă și se afișează rezultatul.

**4.2 Diagrame de flux pentru componente**

* ALU:
* Blocul ALU este conectat la registrul și primește operații precum adunare, înmulțire și împărțire.
* Ieșirile ALU sunt conectate la un multiplexor pentru a selecta rezultatul corect.
* Comparator:
* Comparatorul primește două valori și compară rezultatul.
* Rezultatul comparării este utilizat pentru a seta sau reseta starea de flag.
* Registrul:
* Registrul stochează temporar valorile în timpul execuției.
* Ieșirile registrului sunt conectate la intrările ALU și la multiplexoare.
* Unitate de Împărțire:
* Această unitate primește valorile și efectuează operațiile de împărțire necesare pentru metoda Newton-Raphson.
* Unitate de Control:
* Unitatea de control dirijează fluxul de date și controlează operațiile fiecărei componente în funcție de instrucțiuni.

Diagrame de sistem

Input swich

Output ssd

**8. Bibliografie**

* <http://elf.cs.pub.ro/asm/wiki/laboratoare/laborator-12>
* <https://www.slideserve.com/yardley-hoover/2-unitatea-aritmetic-i-logic>
* <https://users.utcluj.ro/~baruch/book_ac/AC-Reprez-VM.pdf>