

Unitate de calcul în virgulă mobilă: adunarea si scaderea

Matasariu Gabriel

Grupa 30238

Profesor îndrumător: Mocan Cristi

Data: 11.03.2017

**Cuprins**

[**1. Rezumat** 3](#_Toc510273654)

[**2. Introducere** 3](#_Toc510273655)

[**3. Fundamentare teoretică** 4](#_Toc510273656)

[**3.1. VHDL** 4](#_Toc510273657)

[**3.2. Tehnologii utilizate** 5](#_Toc510273658)

[**3.2.1. FPGA** 5](#_Toc510273659)

[**3.2.1. Nexys4 DDR** 5](#_Toc510273660)

[**3.2.1. PmodKYPD** 6](#_Toc510273661)

[**3.3. Reprezentarea numerelor în virgulă mobilă, formatul IEEE 754** 6](#_Toc510273662)

[**3.4. Sumatorul elementar** 8](#_Toc510273663)

[**3.4.1. Sumatorul cu propagarea succesivă a transportului** 10](#_Toc510273664)

[**3.4.2. Sumatorul cu anticiparea transportului** 11](#_Toc510273665)

[**3.4.3. Sumatorul cu salvarea transportului** 13](#_Toc510273666)

[**3.5. Adunarea/Scaderea numerelor in virgula mobila** 14](#_Toc510273667)

[**4. Proiectare şi implementare** 18](#_Toc510273668)

[**4.1. Registru de n biti cu resetare sincrona** 18](#_Toc510273669)

[**4.2. Registru de deplasare stanga de n biti cu resetare sincrona** 18](#_Toc510273670)

[**4.3. Registru de deplasare dreapta de n biti cu resetare sincrona** 19](#_Toc510273671)

[**5. Rezultate experimentale** 20](#_Toc510273672)

[**7. Concluzii** 20](#_Toc510273673)

[**Bibliografie** 20](#_Toc510273674)

# **1. Rezumat**

# **2. Introducere**

Tema propusă spre implementare este cea de adunare/scadere a numerelor reprezentate în virgulă mobilă prin standardul IEEE 754. Majoritatea unităţilor centrale de procesare lucrează cu două tipuri de numere: în virgulă fixă şi în virgulă mobilă[[1]](#_okwn25hjzs29) . Acest lucru este datorat faptului că nu toate informațiiile necesară UCP-ului pentru a-și realiza operațiile specifice sunt numere întregi. Reprezentarea internă în calculator a datelor este realizată printr-un format în virgulă mobilă și pentru asigurarea portabilității programelor între calculatoare diferite, această reprezentare internă respecta un standard(IEEE 754) utilizat de marea majoritate a unităților de calcul și a coprocesoarelor matematice.

Multe aplicații necesită numere care nu sunt întregi. Există mai multe posibilități pentru reprezentarea acestor numere. Una dintre ele este reprezentarea în *virgulă fixă*. În acest caz, se poate utiliza aritmetica pentru numere întregi, plasând apoi virgula binară în poziția predefinită, de exemplu, după bitul de semn. În cazul reprezentării numerelor in acest format, deși virgula nu este reprezentată fizic în calculator, poziția virgulei binare, care este stabilită la proiectare, nu mai poate fi schimbată[[2]](#_okwn25hjzs29). Astfel, este necesară transformarea tuturor numerelor în acest format, lucru ce necesită o serie de operații de de scalare sau deplasare, atașând numerelor *factori de scală.* Evidența acestora trebuie realizată prin program, ceea ce mărește timpul de calcul.

O solutie la problemele aduse de reprezentarea numerelor în virgulă fixă este utilizarea unei tehnici de scalare automată, cunoscută sub numele de reprezentare în *virgulă mobilă*. În acest caz, factorul de scală devine o parte a cuvântului din calculator, poziția virgulei variind pentru fiecare număr în mod automat. Un număr reprezentat în virgulă mobilă are două componente: *mantisa(*M) și *exponentul(*E). Această reprezentare poate fi memorată într-un cuvânt cu trei câmpuri: semnul, mantisa și exponentul. În general, câmpul exponentului conține o valoare pozitivă, valoare obținută prin adunarea unui *deplasament*: în felul acesta, câmpul exponentului va avea întotdeauna o valoare pozitivă.

Reprezentarea numerelor în virgulă mobilă poate fi realizată în mai multe moduri prin diferite modalități de tratare a cazurilor de excepție, numărul de biți alocați fiecărui câmp în reprezentare, modul de rotunjire, etc. Din cauza multiplelor moduri de tratare a acestei reprezentări, programele scrise nu sunt portabile întrucât două calculatoare cu două moduri diferite de reprezentare vor realiza scrierea acestuia în moduri diferite.

Pentru a asigura portabilitatea programelor, *Societatea Calculatoarelor a IEEE(Institute of Electrical and Electronics Engineers* a elaborat un standard pentru reprezentarea numerelor în virgulă mobilă și pentru operații aritmetice în virgulă mobilă. Acest standard prevede o metodă de calcul cu numere în virgulă mobilă, care va produce același rezultat dacă prelucrarea se face în hardware, software sau o combinație a celor două. Rezultatele vor fi identice, independent de punere în aplicare, având în vedere aceleași date de intrare. De asemenea, erorile și condițiile de eroare  
produse în operații matematice vor fi raportate într-o manieră consistentă, indiferent de  
modul de implementare.

O problemă întâlnită la calculele în acest mod de reprezentare este modul de tratare a depășirilor inferioare și superioare. Depășirea superioară apare în momentul în care un exponent depășește valoarea maximă, iar depășirea inferioară apare atunci când exponentul are o valoare mai mică decât cea minimă.

Problema propusă spre soluționare, operația de adunare/scadere a numerelor în virgula mobilă, presupune[[3]](#_okwn25hjzs29) :

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

Obiectivele principale ale acestui proiect sunt: realizarea unei reprezentări ale numerelor corespunzătoare cu formatul IEEE 754 pe 32 biți, implementarea algoritmului de adunare/scadere corespunzător pentru numere reprezentate în virgulă mobilă, realizarea operației de rotunjire pentru obținerea rezultatelor cu o precizie cât mai mare și tratarea cazurilor excepționale ce pot apărea în cadrul acestei operații (depășire inferioară, depășire superioară, rezultat inexact).

În cele ce urmează, capitolul “Fundamentare teoretică” va conține informații mai detaliate referitoare la formatul standard IEEE 754 pe 32 biți și modul de realizare a operatiilor de adunare/scadere pentru numere reprezentate în acest format.

# **3. Fundamentare teoretică**

## **3.1. VHDL**

VHDL nu este un limbaj de programare, ci un limbaj de descriere a sistemelor electronice hardware pornind de la structura lor modulară şi de la interconexiunile dintre acestea. El a fost definit şi integrat în rândul instrumentelor de CAD (Computer-Aided Design) din domeniul electronicii, pentru a introduce o metodologie riguroasă de proiectare în ciclul de dezvoltare al sistemelor hardware.

VHDL a devenit un limbaj industrial standardizat, utilizat pentru descrierea hardware de la nivelul abstract până la nivelul concret. VHDL a fost rapid asimilat ca un mediu universal de comunicatie în proiectare. Toti producătorii de statii de lucru şi de software CAE (Computer-Aided Engineering) îşi standardizează produsele pentru a avea intrări şi ieşiri standard VHDL. Aceste produse includ software pentru simulare, sinteză şi trasare de cablaj imprimat.

Limbajul provine din programul VHSIC (Very High Speed Integrated Circuit) initiat de Departamentul Apărării din Statele Unite ale Americii în 1980. În fata importantei creşteri a complexitătii sistemelor electronice şi mai ales a costurilor de întretinere rezultante, s-a făcut simtită nevoia aparitiei unui limbaj modern şi standardizat.

Necesitatea unei descrieri lipsite de ambiguitate a sistemelor hardware a apărut în mod pregnant la Departamentul Apărării (DOD) al sStatelor Unite la începutul anilor 1980. Efortul de standardizare a fost eşalonat între anii 1983 şi 1987 sub egida DOD. Efortul prestat de numeroase societăti americane, grupate în jurul lui INTERMETRICS, IBM şi TEXAS INSTRUMENTS, a dus la aparitia normei IEEE 1076B, aprobată în 10 decembrie 1987. Îmbunătătirea standardului VHDL este supervizată de IEEE. Ultima variantă a fost publicată în 2009.

VHDL conŃine toate elementele de descriere algoritmică proprii limbajelor de programare. De aceea este adeseori considerat drept un limbaj informatic sau chiar drept un limbaj de informaticieni. Este deci preferabil ca programatorul începător în VHDL să posede cunoştinŃe minime într-un limbaj de programare structurată de nivel înalt, pe lângă cunoştinŃe de proiectare a sistemelor numerice.

## **3.2. Tehnologii utilizate**

### **3.2.1. FPGA**

Un **FPGA** (**Field Programmable Gate Array**) este un [circuit integrat](https://ro.wikipedia.org/wiki/Circuit_integrat) digital configurabil, de către utilizator, după ce a fost fabricat (spre deosebire de dispozivele a căror funcție este implementată in procesul de fabricație). Configurarea FPGA se face, în general, cu ajutorul unui limbaj de descriere hardware [HDL](https://ro.wikipedia.org/w/index.php?title=HDL&action=edit&redlink=1), similar cu cel folosit pentru dispozivele [ASIC](https://ro.wikipedia.org/wiki/ASIC), dezvoltându-se recent și compilatoare care traduc instrucțiuni din limbajul C în limbaje HDL. Un astfel de compilator este [Impulse C](https://ro.wikipedia.org/wiki/Impulse_C).

FPGA-urile sunt alcătuite din blocuri logice configurabile (programabile) legate între ele de o serie de conexiuni configurabile la rândul lor.

### **3.2.1. Nexys4 DDR**

Placa Nexys4 DDR este o platformă completă, gata de utilizare pentru dezvoltarea de circuite digitale bazate pe cele mai recente FPGA-uri Artix-7 de la Xilinx®. Cu o mare capacitate FPGA, memorii externe generoase, precum și o colecție de USB, Ethernet și alte porturi, Nexys4 DDR poate găzdui modele variind de la circuite combinaționale introductive la procesoare puternice încorporate. Mai multe tipuri periferice încorporate, inclusiv un accelerometru, senzor de temperatură, MEMS microfon digital, un amplificator, și mai multe dispozitive I / O permit utilizarea plăcii Nexys4 DDR pentru o gamă largă de modele, fără a avea nevoie de alte componente[[4]](#_okwn25hjzs29).

Unitatea de calcul in virgula mobila pentru adunare si scadere ce urmeaza a fi implementata, utilizând mediul de proiectare Vivado, va fi încărcata pe placa FPGA Nexys4 DDR. Astfel functionarea corecta a acestui circuit poate fi evaluata fizic, si nu doar cu ajutorul simulatorului pus la dispozitie de mediul de proiectare Vivado.

### **3.2.1. PmodKYPD**

PmodKYPD este o tastatură de 16 butoane aranjate într-un format hexadecimal (0-F). Aducand pe rand fiecare coloana a tastaturii la un nivel logic scazut(logic level low voltage), utilizatorii pot citi nivelul logic al fiecarei linii pentru a determina care buton(daca exista) este apasat.

Placa gazdă comunică cu tastatura prin intermediul protocolului GPIO[[5]](#_okwn25hjzs29). Această tastatură va fi utilizată pentru introducerea operanzilor în virgulă mobilă ai operației de adunare/scadere și va facilita utilizarea unitatii de calcul incarcata pe placa Nexys4 DDR, permitand introducerea operanzilor intr-un format zecimal si nu unul binar.

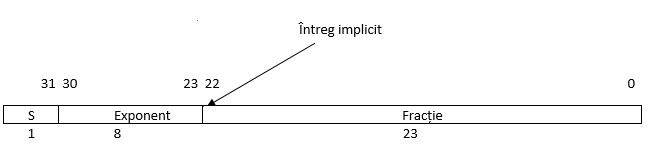
## **3.3. Reprezentarea numerelor în virgulă mobilă, formatul IEEE 754**

Multe operații necesită numere care nu sunt întregi. Aceste numere pot fi reprezentate atât în virgulă fixă, cât și în virgulă mobilă. Este preferată ultima variantă menționată, întrucât în celălalt mod de reprezentare poziția virgulei binare nu poate fi modificată și reprezentarea numerelor în acest format necesită unele operații de scalare sau deplasare, care fac timpul de execuție să crească.

În virgulă mobilă, un număr poate fi reprezentat prin două componente: mantisa, care indică valoarea exactă a numărului într-un anumit domeniu, și exponentul, care indică ordinul de mărime al numărului. Ținând cont de aceste componente, un număr în virgulă mobilă este reprezentat în felul următor:

Reprezentarea numerelor în formatul IEEE 754 asigură portabilitatea programelor de la un calculator la altul, întrucât acestea conțin aceeași reprezentare internă a numerelor în virgulă mobilă. Această reprezentare poate fi memorată într-un cuvânt binar cu trei câmpuri: semn, mantisă și exponent[[2].](#_okwn25hjzs29) În cele ce urmează, va fi detaliat modul de reprezentare, intervalul numerelor definite și modul de tratare a unor condiții speciale și de eroare pentru formatul IEEE 754 pe 32 de biți.

În cazul reprezentării pe 32 de biți, standardul impune ca dimensiunea celor trei câmpuri componente ale reprezentării interne să fie următoarea: un bit pentru semn, 8 biți pentru exponent și 23 biți pentru mantisă. Acest format este numit format cu *precizie simplă* și poate fi vizibil în figura 3.1.



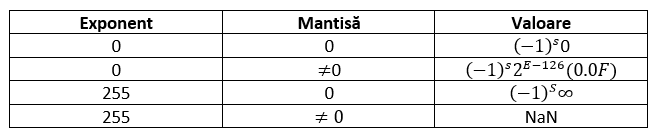
**Figura 3.1.**Formatul cu precizie simplă

Aceasta este o reprezentare în mărime și semn, întrucât semnul are un câmp separat pentru reprezentare. Acest câmp este 0 pentru numerele pozitive și 1 pentru cele negative. Numerele reprezentate în acest format sunt normalizate, adică au bitul cel mai semnificativ al mantisei egal cu 1.

În acest format apare noțiunea de *bit ascuns.* Acesta se referă fixarea unei valori pentru bitul cel mai semnificativ al mantisei prin intermediul formatului de reprezentare ,valoare ce nu poate fi modificată, motiv pentru care acest bit nu mai este reprezentat. În formatul precizie simplă, mantisa este alcătuită dintr-un bit ascuns cu valoarea 1, reprezentând partea întreagă a numărului), virgula binară implicită, urmată de biții de fracție. Utilizarea acestui bit implicit crește precizia operațiilor și reprezentărilor numerelor în virgulă mobilă, întrucât permite mantisei să aibă un bit în plus. Intervalul numerelor care pot fi reprezentate în precizie simplă este cuprins între 1,18\*și

3,4\*.

Câmpul exponentului în acest format, nu conține valoarea exponentului, ci o valoare numită *caracteristică.* Aceasta se obține prin adunarea unui deplasament de 127 la exponent, astfel încât valoarea caracteristicii este întotdeauna pozitivă. Deplasamentul este ales la valoarea 127 pentru ca cel mai număr posibil în reprezentarea cu semn să aibă caracteristica 0. Avantajul utilizării caracteristicii constă în simplificarea operațiilor asupra exponentului, pentru că valorile acesteia sunt numere pozitive și nu mai este necesară utilizarea operațiilor cu semn. Valoarea minimă și maximă ale acesteia sunt folosite pentru reprezentarea unor valori speciale. În tabelul 3.1 sunt prezentate valorile speciale ale formatului pe 32 de biți.



**Tabelul 3.1.**Valori speciale ale numerelor în IEEE 754

Valoarea 0 va fi reprezentată prin atribuirea câmpului de exponent și cel al mantisei a valorii 0. În acest caz, bitul ascuns este impicit 0, și nu 1.

Formatul cu precizie simplă prezintă probleme în cazul reprezentării unor numere ce nu se află în gama numerelor care pot fi reprezentate pe un cuvânt de 32 biți, probleme cunoscute sub numele de depășire superioară și depășire inferioară. Depășirea superioară apare atunci când exponentul depășește valoarea maximă(127), în timp ce depășirea inferioară apare atunci când exponentul este mai mic decât valoarea minimă(-126).

În cazul unei depășiri inferioare, standardul permite utilizarea numere care nu sunt normalizate, numite *numere denormalizate.* Acestea sunt generate printr-o tehnică care constă în deplasarea mantisei la dreapta și incrementarea exponentului până când acesta ajunge la valoarea minimă permisă, tehnică numită *depășire inferioară graduală.*

În cazul unei depășiri superioare, există o reprezentare specială pentru infinit. Aceasta constă din atribuirea valorii maxime exponentului și valorii 0 mantisei. Bitul de semn va distinge în acest caz, reprezentarea între . Utilizatorul va putea decide dacă depășirea superioară va fi tratată ca și o condiție de eroare sau va continua calculele cu valoarea infinit.

O altă valoare specială a acestui format, este valoarea NaN. Aceasta va fi utilizată pentru a indica diferite condiții de excepție.

## **3.4. Sumatorul elementar**

Sumatorul este blocul operațional de bază implicat în majoritatea operațiilor complexe ale majorității circuitelor, fiind utilizat în special în structura internă a UAL- Unității Aritmetico-logice. Acestea fiind spuse, optimizarea sumatoarelor se traduce în optimizarea întregului sistem, deoarece viteza de operare a acestora va influența direct proporțional viteza de calcul a circuitului final.

Sumatorul elementar este un bloc combinațional ce primește trei intrări și are ca rezultat două ieșiri. Intrările sunt compuse din cei doi biți de adunat , respectiv , și bitul de transport de la nivelul precedent - bitul din poziția mai puțin semnificativă - notat cu . Ieșirile sunt reprezentate de bitul suma, , respectiv bitul generat de transport către următorul bit din poziție mai semnificativă, .[[2]](file:///C:\Users\Mati\Desktop\Proiect_doc.docx#_okwn25hjzs29)

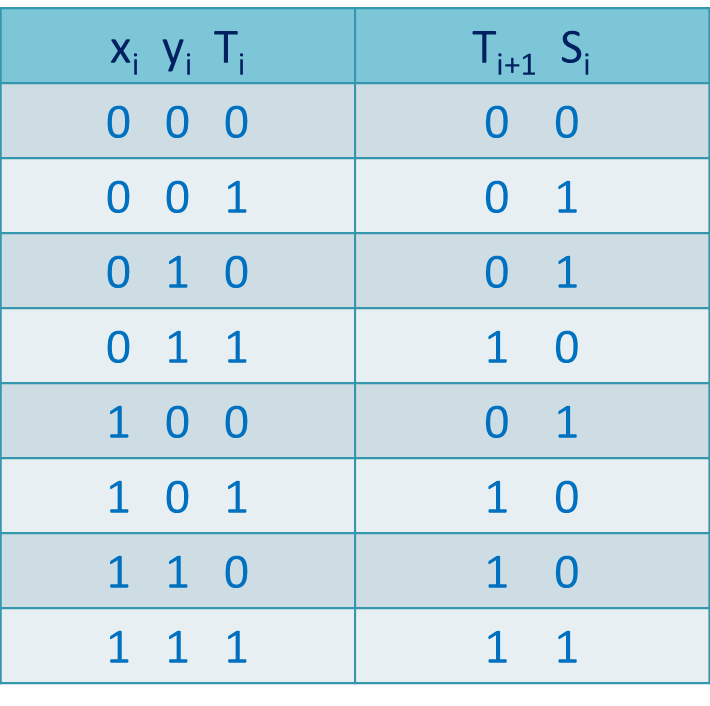
Expresiile booleene ale ieșirilor sunt după cum urmează:

.

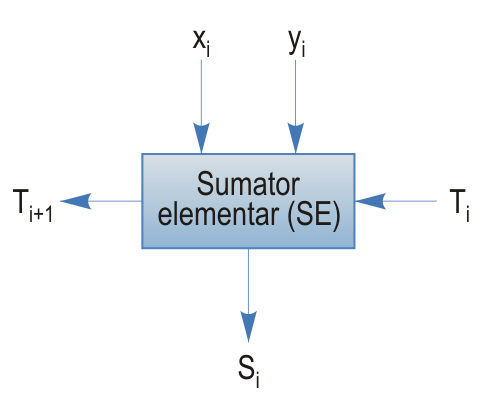
Operația “sau exclusiv” () are următoarea formulă de calcul:

.

Tabelul 3.2. Reprezintă tabelul de adevăr a operației de adunare cu ajutorul sumatorului elementar. Schema bloc a sumatorului elementar este dată de figura 3.2.



**Tabel 3.2.** Tabelul de adevăr al sumatorului elementar



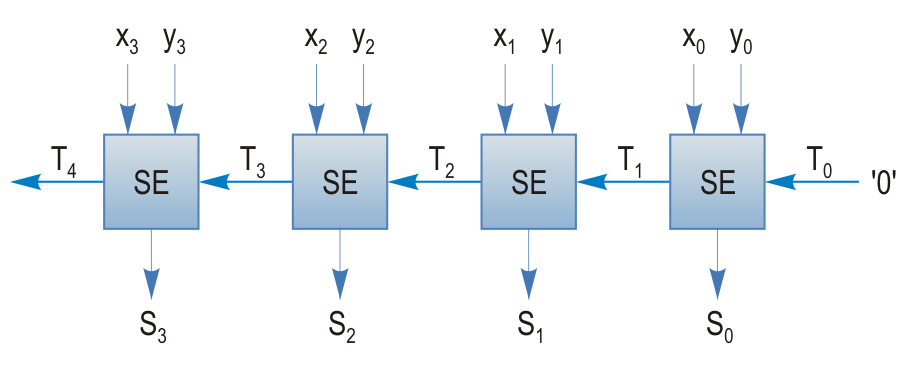
**Figura 3.2** Simbolul sumatorului elementar

Un sumator particular este *semisumatorul elementar* care primește ca și intrări cei doi biți de adunat (se omite bitul de transport) și generează la ieșire un bit de sumă și unul de transport.

În mod similar, *semisumatorul elementar* are ca și intrări doi biți - scăzătorul și descăzutul, iar la ieșire este generat un bit diferență și unul de împrumut. Scăzătorul elementar are în plus ca și intrare bitul de împrumut de la bitul mai puțin semnificativ.

### **3.4.1. Sumatorul cu propagarea succesivă a transportului**

Acest sumator este format dintr-o serie de sumatoare elementare conectate în cascadă și se realizează adunarea bit cu bit a două numere. Se utilizează câte un sumator pentru fiecare pereche de biți de adunat de la fiecare poziție. Spre exemplu adunarea a două numere reprezentate pe patru biți, și va utiliza patru sumatoare elementare; primul sumator va aduna biții , respectiv și va genera bitul cel mai puțin semnificativ al sumei, și un transport spre bitul următor, mai semnificativ (în cazul acesta, bitul de la poziția 1). Asftel ieșirea de transport a primului sumator este conectată la intrarea de transport a celui de-al doilea, realizându-se conectarea în serie a sumatoarelor. Primul sumator, deoarece are intrarea de transport setată pe 0, va putea fi implementat ca un semisumator. Ieșirea de transport a ultimului sumator (), în cazul în care va fi setată, va reprezenta cel mai semnificativ bit (de pe poziția 4, ). Schema bloc a acestui tip de sumator este redată în figura 3.3.



**Figura 3.3.** Sumator de 4 biți cu propagarea succesivă a transportului

Acest tip de sumator este avantajos ca și simplitate și datorită faptului ca nu necesită logică combinațională suplimentară, având un cost redus. Dezavantajul major, însă, este dat de viteza foarte redusă a acestuia. Această latență se datorează faptului că sumatorul de la fiecare etaj va trebui să aștepte după transportul de intrare propagat de etajul anterior, acest lucru fiind foarte costisitor pentru timpul de execuție.

Acest sumator se poate utiliza și cu funcționalitatea unui scăzător, în cazul în care se vor aduna unul din termeni (cel pozitiv) cu cel de-al doilea (negativ) reprezentat în complement față de doi. Transportul din poziția cea mai semnificativă a rezultatului se neglijează.

### **3.4.2. Sumatorul cu anticiparea transportului**

În sumatorul cu transport anticipat (Carry Look-Ahead), fiecare adunare pe bit elimină dependenţa de semnalul transportului generat anterior şi impune în schimb utilizarea valorilor celor doi operanzi de intrare. El funcționează prin generarea a două noi semnale (P și G) pentru fiecare rang binar în funcție de starea intrărilor. [[6]](file:///C:\Users\Mati\Desktop\Proiect_doc.docx#_okwn25hjzs29)

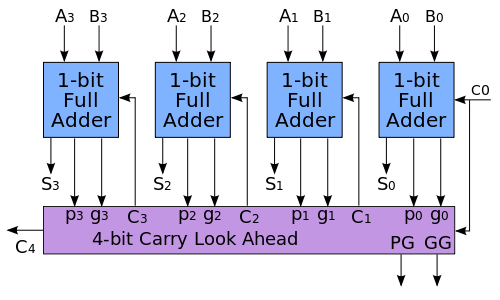
Fiecare etaj va avea un bloc de transport, conținând o logică combinațională ce va calcula transportul de intrare pentru acel etaj în funcție de transportul inițial și toți biții din etajele precedente (deci independent de alte semnale de transport).

Expresiile booleene pentru fiecare etaj de transport sunt date după cum urmează:

Semnalele P și G menționate anterior vin în ajutorul simplificării expresiei, referindu-se la *propagarea,* respectiv *generarea* transportului la ieșire. Acestea sunt definite astfel:

Funcția de *generare*  este denumită astfel deoarece etajul *i* va produce un transport () dacă ambii operanzi vor fi 1 (), independent de . Alternativ, *propagarea*  se referă la prezența unui transport () dacă având ca și intrare , unul din cei doi operanzi este setat (). Noua expresie devine:

Schema bloc a unui sumator de 4 biți cu anticiparea transportului care utilizează un generator de transport anticipat:



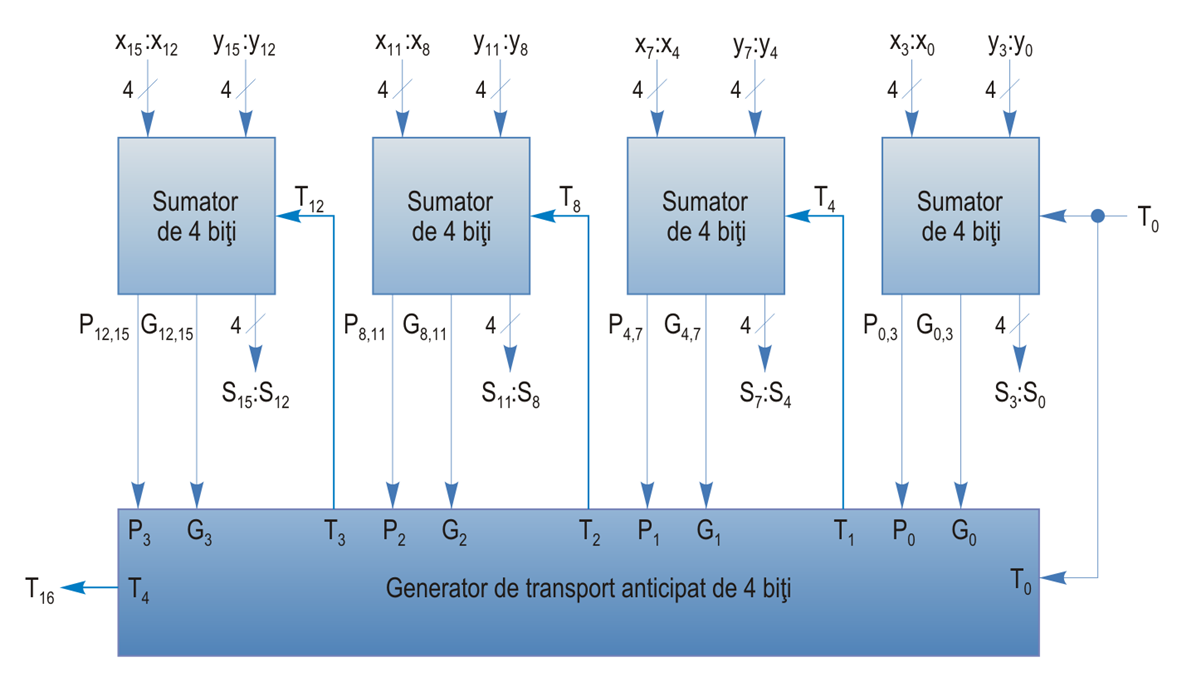
**Figura 3.4.** Sumator de 4 biți cu anticiparea transportului

Următoarele ecuații definesc ieșirile semnalelor de transport în cazul sumatorului pe 4 biți:

În cazul în care dorim să reducem complexitatea circuitului putem pune limitări la numărul porților și al intrărilor existente. Acest lucru se va traduce, însă, în necesitatea adăugării la circuit a unei logici suplimentare. Pentru a realiza acese limitări în cazul adunării a 4 biți (și nu numai), se definesc două noi funcții: pentru generarea transportului pe grup, , respectiv pentru propagarea transportului pe grup, , corespunzând bițiilor de la i până la k. Următorul exemplu ilustrează aceste două funcții în cazul sumatorului pe 4 biți:

Vom obține astfel:

Figura următoare arată transformarea sumatorului utilizând noile funcții, în cazul extinderii acestuia la un sumator de dimensiune mai mare. În locul utilizării sumatoarelor pe 1 bit se vor utiliza sumatoare pe grupuri de 4 biți.



**Figura 3.5.** Schema bloc a unui sumator de 16 biți format din sumatoare de 4 biți conectate prin semnale de transport generate anticipat

Intrarea de transport () a celui de-al treilea sumator se calculează după o ecuație similară cu următoarea:

, unde:

Ecuațiile pentru intrările și sunt asemănătoare calculelor pentru semnalele și utilizate în schema anterioară fără funcțiile de propagare, respectiv generare pe blocuri.

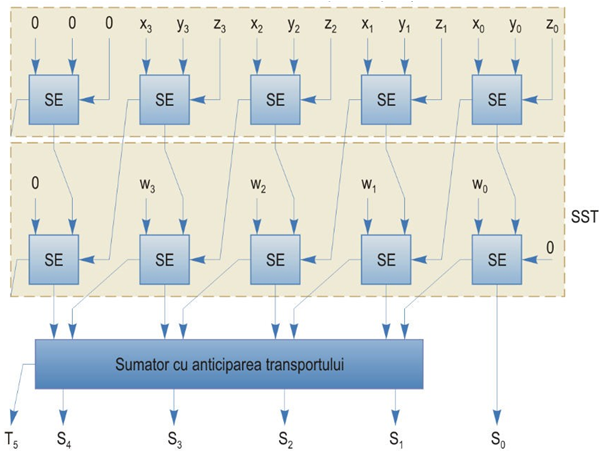
Acest tip de sumator este superior celui cu propagare succesivă datorită vitezei de execuție, însă necesită circuite combinaționale adiționale datorită complexității sale

### **3.4.3. Sumatorul cu salvarea transportului**

Adunarea cu salvarea transportului este o tehnică utilizată pentru reducerea timpului de propagare al semnalelor de transport, fiind o tehnică utilizată pentru creșterea vitezei operației de adunare atunci când trebuie adunate mai mult de două numere.

Un sumator bazat pe adunarea cu salvarea transportului este reprezentat printr-o colecție de n sumatoare elementare independente, n fiind numărul de biți al sumatorului. Semnalele de transport nu sunt propagate între sumatoarele elementare, viteza de adunare fiind substanțial mai ridicată prin generarea biților de sumă și de transport în paralel. Rezultatul final este obținut prin adunarea sumei și a transportului prin utilizarea unui sumator, diferit de cel cu salvarea transportului, numit *sumator cu propagarea transportului*.

Pe baza celor două faze existente în acest mod de adunare, cea de adunare independentă a biților și cea de adunare a sumei și transportului obținut, se poate realiza un sumator cu operanzi multipli. Acesta poate fi utilizat pentru realizarea circuitelor de înmulțire pentru acumularea produselor parțiale. Sumatorul standard cu salvarea transportului adună independent, în paralel, trei numere. Suma și transportul acestora pot fi folosite în altă operație de adunare, metodă prin care acest sumator poate fi folosit pentru adunarea a patru numere. Procesul acestei adunări constă în adunarea primelor trei numere, urmată de operția de adunare a sumei și produsului acestora cu cel de-al patrulea număr. Această adunare are ca rezultat o sumă și un transport ce vor fi adunate prin intermediul sumatorului cu propagarea transportului. În figura 3.4.1 este vizibil procedeul descris de adunare a patru numere utilizând două sumatoare cu salvarea transportului.



**Figura 3.6** Sumator cu salvarea transportului pentru adunarea a patru numere

Acest sumator este utilizat pentru adunarea a mai mult de două numere, acest număr nefiind limitat de vre constrângere. Schema bloc din figura 3.4.1 prezintă modul în care este realizată adunarea pentru patru numere. Numărul termenilor ce urmează a fi adunați poate fi mai mare decât cel prezentat în schemă, procedeu ce utilizează aceeași metodă a sumatorului pentru patru numere. Diferența în folosirea sumatorului pentru adunarea a mai mult de patru termeni constă în numărul de etape necesare pentru obținerea sumei și a transportului ce urmează a fi adunate în sumatorul cu propagarea transportului. Acestea se obțin după adunarea termenilor în sumatorul cu sumele și transporturile lor intermediare.

## **3.5. Adunarea/Scaderea numerelor in virgula mobila**

Adunarea şi scăderea în VM sunt mai complexe decât înmulţirea şi împărţirea. Aceasta 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. Algoritmul pentru adunare şi scădere are patru etape principale:

1. Alinierea mantiselor;

2. Adunarea sau scăderea mantiselor;

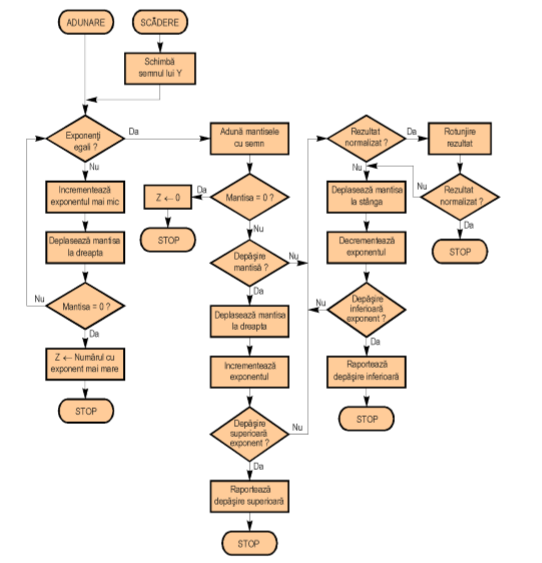
3. Normalizarea rezultatului;

4. Rotunjirea rezultatului.

Înaintea operaţiei, cei doi operanzi trebuie transferaţi în registrele care vor fi utilizate de UAL. Dacă formatul de reprezentare în VM cuprinde un bit implicit al mantisei, acest bit trebuie reprezentat în mod explicit pentru operaţie. În mod tipic, exponenţii şi mantisele vor fi păstrate în registre separate, şi vor fi reunite după obţinerea rezultatului.

Deoarece adunarea şi scăderea sunt identice cu excepţia semnului diferit al operandului al doilea, în cazul operaţiei de scădere se schimbă semnul scăzătorului. Următoarea etapă este alinierea mantiselor. Aceasta necesită compararea celor doi exponenţi şi apoi alinierea mantisei numărului cu exponentul mai mic. Alinierea se realizează prin deplasarea repetată la dreapta a mantisei cu câte o poziţie şi incrementarea exponentului până când cei doi exponenţi devin egali. Dacă în urma acestui proces mantisa care a fost deplasată devine 0, atunci se raportează ca rezultat celălalt număr. Astfel,dacă cele două numere au exponenţi care diferă în mod semnificativ, numărul mai mic va fi neglijat.

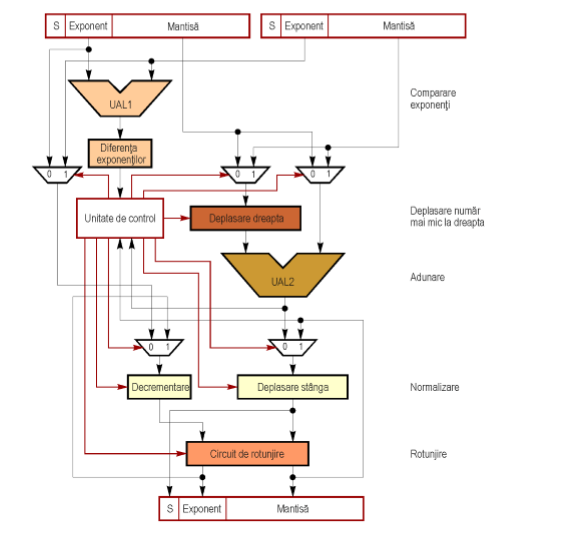
În continuare, se adună mantisele, ţinând cont de semnele acestora. Deoarece semnele pot fi diferite, rezultatul poate fi 0. Există de asemenea posibilitatea apariţiei unei depăşiri cu un bit a mantisei rezultatului. În acest caz, mantisa rezultatului este deplasată la dreapta cu o poziţie şi exponentul este incrementat. În urma deplasării, poate apare o depăşire a exponentului; în acest caz, se raportează o depăşire, iar operaţia este oprită.



**Figura 3.7** Adunarea si scaderea in virgula mobila

În următoarea etapă se normalizează rezultatul. Normalizarea constă în deplasarea mantisei rezultatului la stânga până când bitul c.m.s. devine 1. După fiecare deplasare, se decrementează exponentul, ceea ce poate determina o depăşire inferioară a exponentului. În final, rezultatul este rotunjit. Procedura de rotunjire este prezentată în secţiunea 6.3.3. Dacă în urma rotunjirii rezultatul nu mai este normalizat, procedura de normalizare este repetată.

Multe calculatoare sau procesoare dispun de circuite dedicate pentru creşterea vitezei operaţiilor în VM. Figura 6.18 prezintă o schemă bloc a unui circuit de adunare în VM.

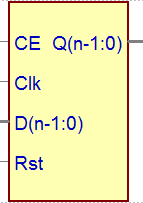


**Figura 3.8** Adunarea si scaderea in virgula mobila

Mai întâi se scade exponentul unui operand din exponentul celuilalt operand utilizând UAL1 pentru a determina care este exponentul mai mare. Diferenţa dintre exponenţi controlează cele trei multiplexoare. Acestea selectează (de la stânga la dreapta) exponentul mai mare, mantisa numărului mai mic, respectiv mantisa numărului mai mare. Mantisa numărului mai mic se deplasează la dreapta, iar apoi se adună mantisele utilizând UAL2. Pentru normalizare se deplasează apoi suma la stânga şi se decrementează exponentul. Circuitul de rotunjire generează rezultatul final.

# **4. Proiectare şi implementare**

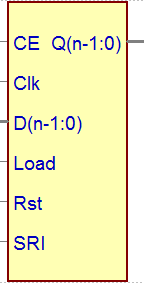
## **4.1. Registru de n biti cu resetare sincrona**



**Figura 4.1** Registru de n biti cu resetare sincrona

Intrarea de ceas a registrului este semnalul Clk, intrarea de date este semnalul tip vector D de n biti, iar iesirea registrului este semnalul tip vector Q. Dacă semnalul Rst este 1 logic, registrul este resetat în mod sincron, ieșirile sale fiind resetate la 0 logic. Dacă semnalul CE (Clock Enable) este 1 logic, la frontul crescător al semnalului de ceas se realizează încărcarea paralelă a registrului cu datele aplicate la intrare.

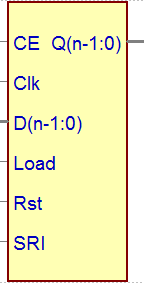
## **4.2. Registru de deplasare stanga de n biti cu resetare sincrona**



**Figura 4.2** Registru de deplasare stanga de n biti cu resetare sincrona

Intrarea de ceas a registrului este semnalul Clk, intrarea de date este semnalul tip vector D de n biti, iar iesirea registrului este semnalul tip vector Q. Dacă semnalul Rst este 1 logic, registrul este resetat în mod sincron, ieșirile sale fiind resetate la 0 logic. Dacă semnalul CE (Clock Enable) este 1 logic,si semnalul Load este 1 logic, la frontul crescător al semnalului de ceas se realizează încărcarea paralelă a registrului cu datele aplicate la intrare.Daca semnalul CE este 1 logic dar semnalul Load are valoarea logica 0, atunci are loc deplasarea la stanga cu o pozitie a continutului registrului.

## **4.3. Registru de deplasare dreapta de n biti cu resetare sincrona**



**Figura 4.3** Registru de deplasare dreapta de n biti cu resetare sincrona

Intrarea de ceas a registrului este semnalul Clk, intrarea de date este semnalul tip vector D de n biti, iar iesirea registrului este semnalul tip vector Q. Dacă semnalul Rst este 1 logic, registrul este resetat în mod sincron, ieșirile sale fiind resetate la 0 logic. Dacă semnalul CE (Clock Enable) este 1 logic,si semnalul Load este 1 logic, la frontul crescător al semnalului de ceas se realizează încărcarea paralelă a registrului cu datele aplicate la intrare.Daca semnalul CE este 1 logic dar semnalul Load are valoarea logica 0, atunci are loc deplasarea la dreapta cu o pozitie a continutului registrului.

# **5. Rezultate experimentale**

# **7. Concluzii**

# **Bibliografie**

[1] “Unitatea centrală de prelucrare (Central Processing Unit – CPU)”, [https://computerplusro.wordpress.com](https://computerplusro.wordpress.com/tag/numere-in-virgula-fixa-si-in-virgula-mobila/)

[2] Baruch Zoltan Francisc, “Structura sistemelor de calcul”, Editura U. T. PRES, Cluj-Napoca, 2002.

[3] http://users.utcluj.ro/~baruch/book\_ac/AC-Adunare-VM.pdf

[4] “Nexys4 DDR™ FPGA Board Reference Manual ”, 2016, [https://reference.digilentinc.com/nexys4ddr\_rm.pdf](https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf)

[5] “PmodKYPD™ Reference Manual”, 2016,[https://reference.digilentinc.com/pmodKYPD\_rm.pdf](https://reference.digilentinc.com/_media/pmod:pmod:pmodKYPD_rm.pdf)