

DOCUMENTAȚIE

Proiectarea unei unități de calcul în virgulă flotantă: inmultirea

NUME STUDENT: MURESAN IOANA DANINA

GRUPA: 30236

# CUPRINS

[1. Introducere 3](#_Toc117899591)

[1.1 Context 3](#_Toc117899592)

[1.2 Specificații 4](#_Toc117899593)

[1.3 Obiective 4](#_Toc117899594)

[2. Studiu bibliografic 5](#_Toc117899595)

[2.1 Numerele în virgulă flotantă 5](#_Toc117899596)

[2.2 Înmulțirea numerelor în virgulă flotantă 5](#_Toc117899597)

[3. Design 8](#_Toc117899599)

[4. Implementare 9](#_Toc117899600)

[5. Testare 14](#_Toc117899601)

[6. Concluzii 16](#_Toc117899602)

[7. Bibliografie 18](#_Toc117899603)

# Introducere

# Context

În domeniul IT, virgula flotantă este unul dintre principalele mecanisme de reprezentare a numerelor raționale sub formă de șiruri de biți. Acest termen face referire la faptul că virgula care separă partea întreagă de partea fracționară poate fi plasată oriunde relativ la cifrele semnificative ale numărului. Unul dintre cele mai mari avantaje ale reprezentării în virgulă mobilă față de cea în virgulă fixă este gama mai largă de valori care pot fi reprezentate.

Scopul acestui proiect este de a proiecta, implementa și testa o unitate de calcul ce realizează operația de înmulțire.Operatia propusa pentru unitatea de calcul se va realiza pe numere în virgulă flotantă pe 32 de biți. Aceste operații sunt mai complexe decât cele pe numere întregi deoarece implică aritmetica în virgulă flotantă.

Aplicația poate fi folosită de utilizatori pentru a calcula simpla înmulțire a două numere în virgulă flotantă pe 32 de biți. Mai mult decât atât, mai poate fi integrată și în alte aplicații de o complexitate mai ridicată. De exemplu, se poate integra în procesoare sau microprocesoare ce necesită utilizarea operațiilor menționate. De asemenea, poate fi conectat la alte dispozitive sau aplicații care să dea comanda realizării uneia dintre operații.

# Specificații

Aplicația va fi realizată și simulată/testată în IDE-ul Vivado. De asemenea, ea va fi implementată în limbajul VHDL. Aceasta va trebui să aibă următoarele specificații:

* Să reprezinte intern numerele în formatul standard IEEE
* Să realizeze conversii ce vor fi necesare în procesul de implementare a operațiilor
* Să suporte operația de înmulțire, iar acestea să fie implementata într-un mod cât mai simplu și eficient
* Să producă rezultate corecte și cu acuratețe
* Rezultatele să fie prezentate într-un format specific

# Obiective

În ceea ce privește obiectivele proiectului, acestea sunt următoarele:

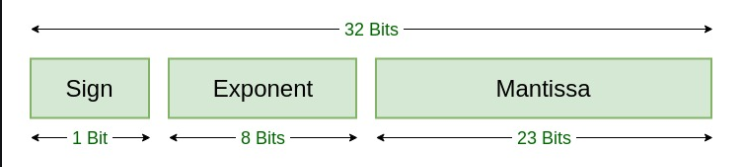
* Implementarea unei metode de a introduce ca input numere raționale
* Stocarea acestora în reprezentarea în virgulă mobilă
* Afișarea rezultatului operațiilor într-un anumit format specific
* Implementarea operațiilor de înmulțire specifice numerelor reprezentate în virgulă flotantă
* Testarea și simularea operațiilor de înmulțire
* Corectitudinea operațiilor, cât și a întregii aplicații

# Studiu bibliografic

# Numerele în virgulă flotantă

În concordanță cu standardul IEEE, numerele în virgulă flotantă cu simplă precizie sunt reprezentate astfel folosind:

* 1 bit pentru semn
* 8 biți pentru exponent
* 23 de biți pentru mantisă



Numărul se calculează cu ajutorul următoarei formule:

O imagine care conține text

Descriere generată automat

# Înmulțirea numerelor în virgulă flotantă

În ceea ce privește algoritmul de înmulțire al numerelor în virgulă flotantă, acesta are următorii pași:

* Conversia numerelor (x și y) în formatul IEEE
* Fie a exponentul lui x și b exponentul lui y
* Fie c = a + b exponentul rezultat
* Se înmulțește mantisa lui x cu mantisa lui y, iar rezultatul este m
* Se ajustează exponentul dacă este cazul
* Pentru a vedea care este semnul înmulțirii, se adună biții de semn si se face mod 2. Rezultatul este semnul
* Se convertesc numerele înapoi și se trunchiază dacă este cazul

Exemplu:

Numărul a = 0 0111 110 și numărul b = 0 1001 010

Urmând algoritmul avem:

* a = 1.11 x 2^0 și b = 1.01 x 2^2
* c = 0 + 2 = 2 exponent
* Înmulțind 1.11 cu 1.01 obținem 10.0011
* Normalizăm 10.0011 la 1.00011 și ajustăm exponentul la 3
* Bitul de semn este 0, deci rezultatul va fi pozitiv
* Trunchiem rezultatul și îl normalizăm: 1.00011 x 2^3 -> 1.000 x 2^3

# Design

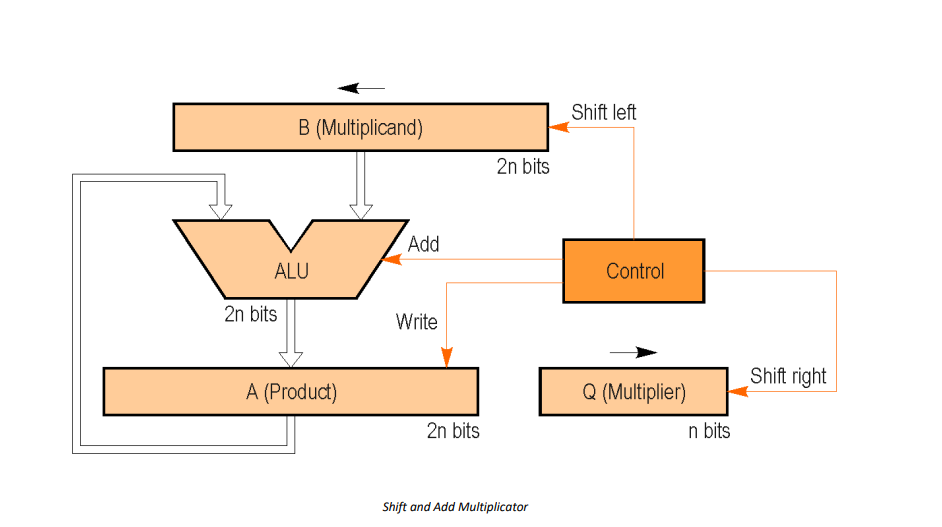
Înainte să vorbim despre design – ul propriu – zis al proiectului, trebuie făcută o analiză a operatiei.

 În figura din stânga avem flowchart-ul operației de înmulțire. Pentru această operație vom avea nevoie de o componentă de înmulțire a mantiselor și de un sumator pentru exponenți.

Pentru înmulțirea mantiselor, concatenam 1 ca cel mai semnificativ bit pentru ambele mantise și trimitem rezultatele la înmulțitor. Rezultatul va fi astfel pe 48 de biți. Avem astfel 2 cazuri: dacă primul bit de la rezultat este 1 atunci luăm următorii 23 de biți și normalizăm rezultatul prin adăugarea unui 1 la exponent, celălalt caz, când bitul cel mai semnificativ este 0, luăm următorii 23 de biți.

Pentru a realiza adunarea exponenților se va folosi un sumator pe 8 biți, iar pentru aflarea semnului operației, se va folosi operația XOR între semnele numerelor.

În ceea ce privește componenta de înmulțire, se va folosi tehnica de Shift and Add, descrisă de figura de mai jos:



În ceea ce privește componenta de adunare, se va folosi un Ripple Carry Adder pe 8 biți.

Întreaga componentă de înmulțire a numerelor în virgulă flotantă este descrisă de urmatoarea diagramă.



Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated

La înmulțire, dacă ambii operanzi sunt foarte mari, adunarea exponenților va cauza overflow, dând un rezultat foarte mic, iar dacă sunt foarte mici, se va obține un rezultat foarte mare.

# Implementare

Implementarea va consta în crearea unei componente care să îi permită user-ului să introducă numerele dorite, cifră cu cifră, dar și să aleagă poziția virgulei din cadrul numărului. Apoi va fi nevoie de o altă componentă pentru a transforma input–ul user–ului în format binar, apoi în formatul standard IEEE.

Alta componenta va fi cea de înmulțire, care va avea la rândul ei alte componente auxiliare ajutătoare integrate.

După efectuarea operațiilor de înmulțire în formatul standard IEEE, o altă componentă va facilita conversia din IEEE înapoi în BCD pentru a putea fi afișate utilizatorului.

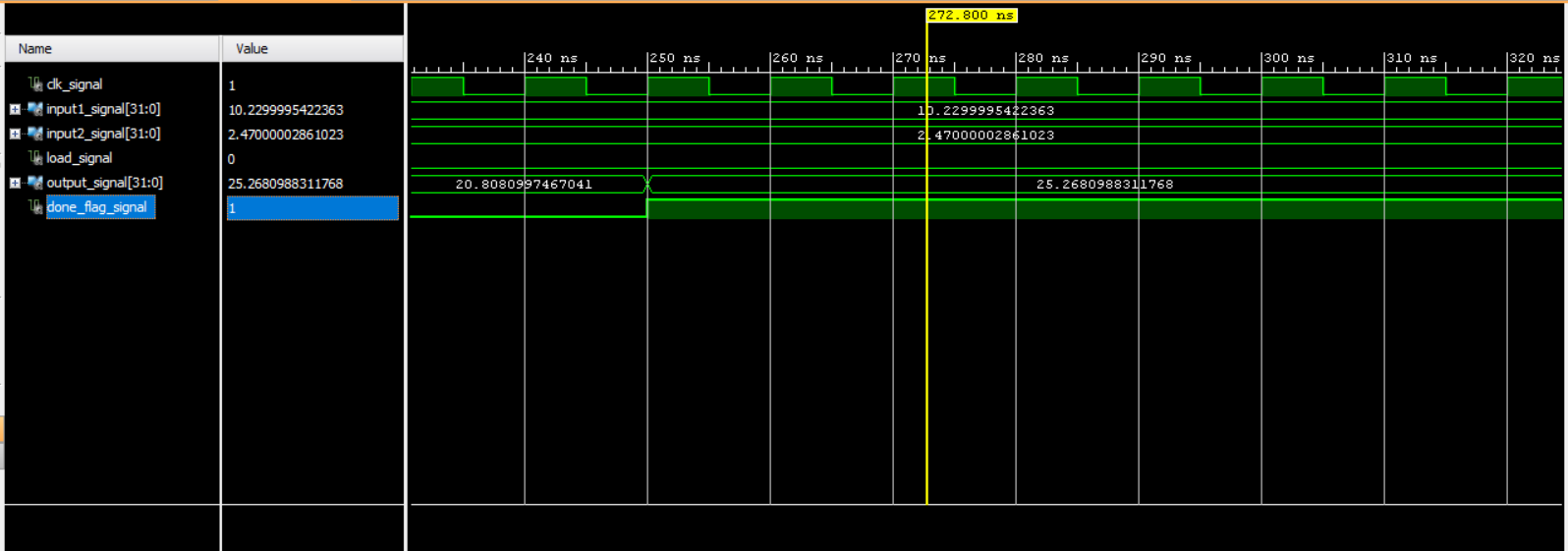
Pe lângă elementele descrise mai sus, se vor mai folosi debouncere pentru butoane, dar și un afișor pe șapte segmente.

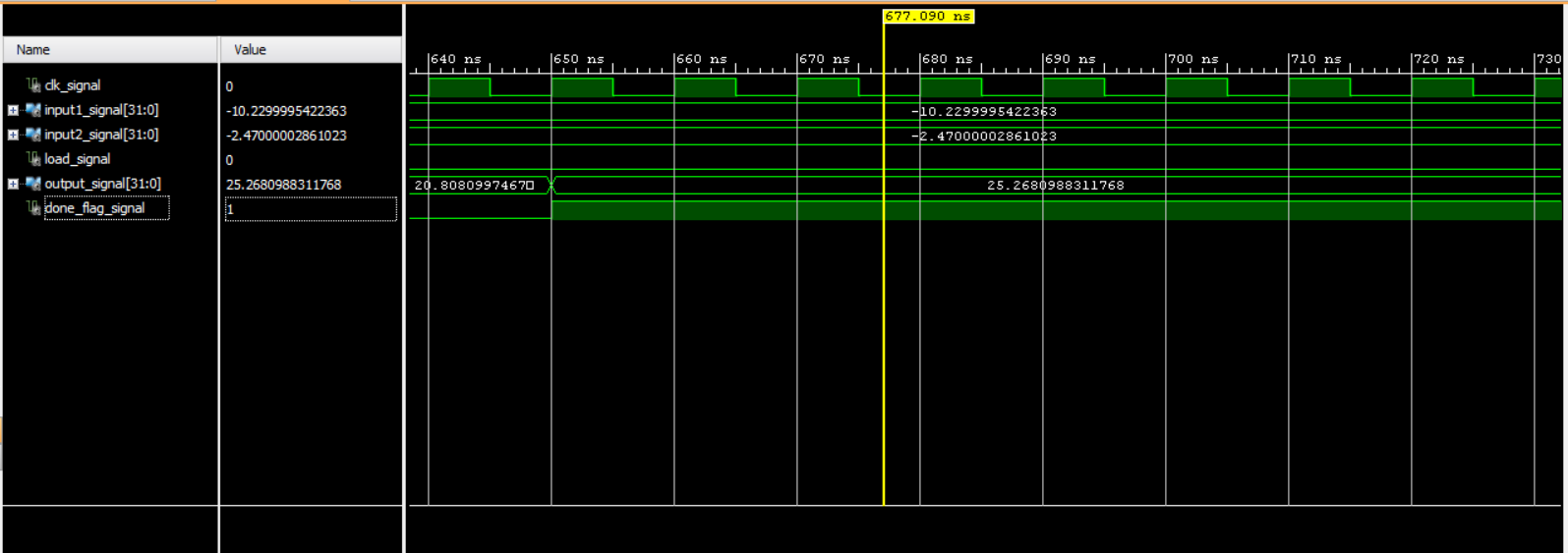
# Testare

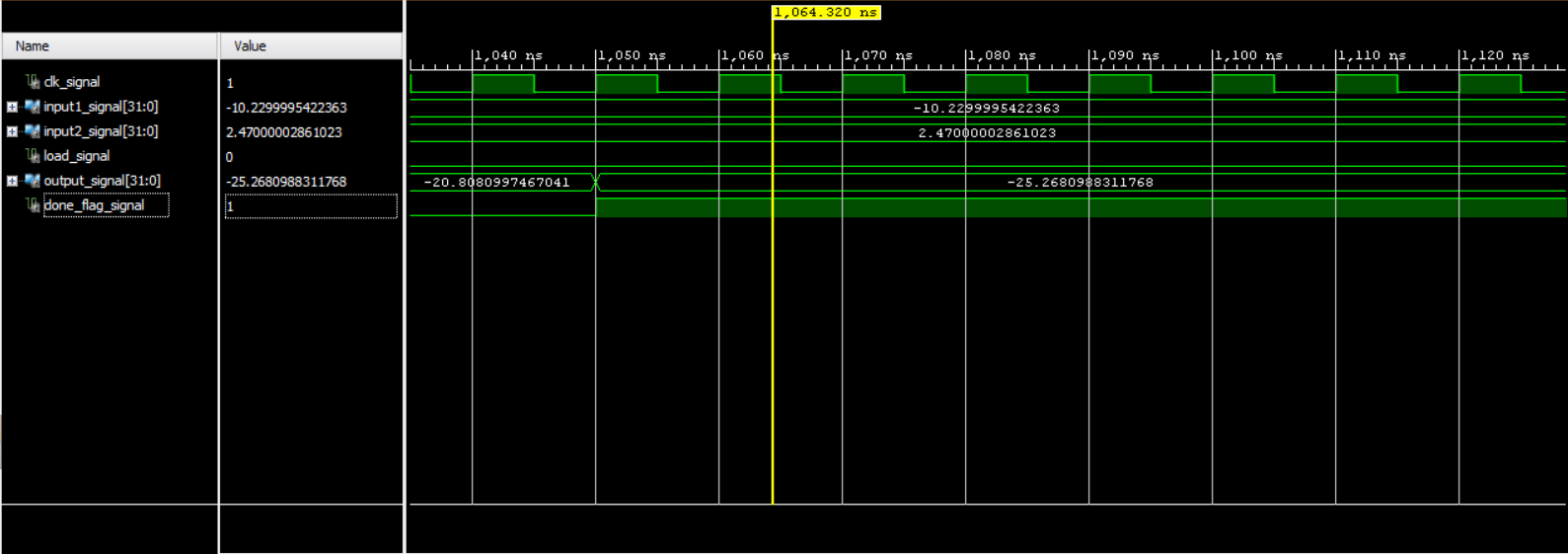
Din punct de vedere al testării aplicației, s-au realizat teste pe plăcuța FPGA Basys3, dar s-au realizat și testbench-uri pentru componentele de înmulțire .Acestea din urmă au fost simulate în Vivado.

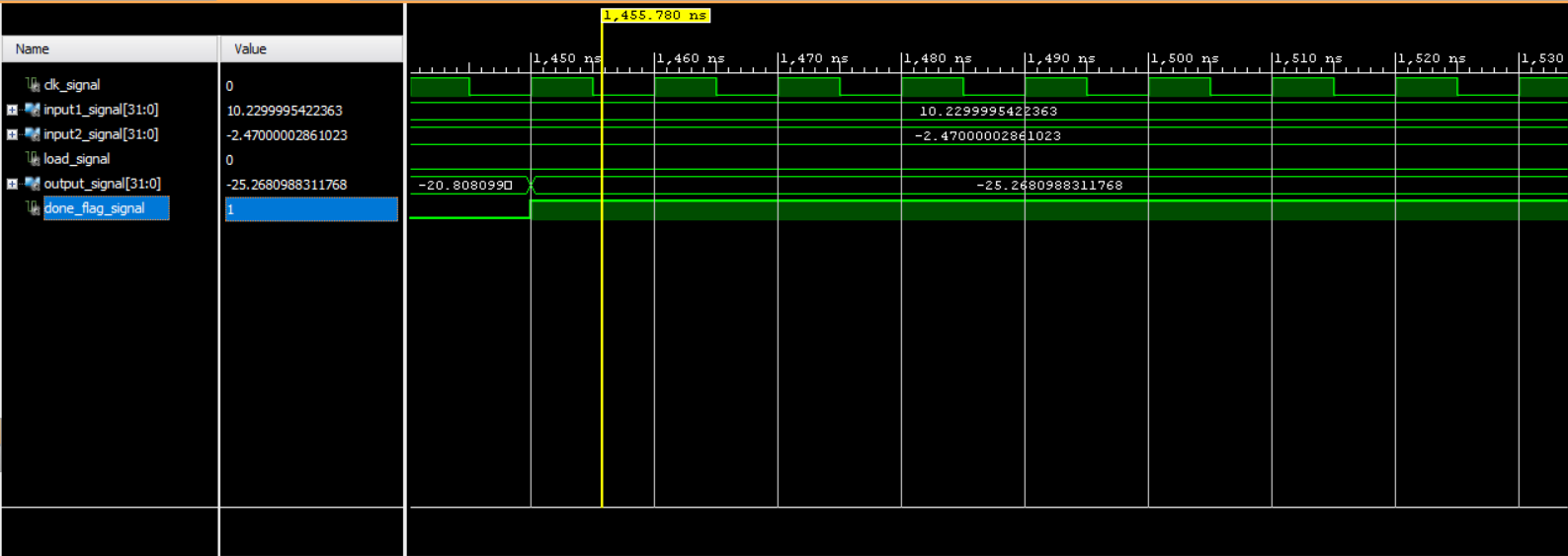
Un exemplu este următorul:

Înmulțire:









Tot din punct de vedere al testării, pentru testarea aplicației pe plăcuța Basys3, se vor folosi următoarele switch-uri/led-uri/butoane:

* V17 – sw0 – controlează numărarea crescătoare/descrescătoare
* W16 – sw2 – semnul de la al doilea număr
* W17 – sw3 – semnul de la primul număr
* W15 – sw4 – selectează numărul (1 primul număr, 0 al doilea număr)
* R2 – sw15 – load (1 încarcă, 0 afișează rezultatul)
* U16 – ld0 – done
* W3 – ld10 – cifra curentă 0
* U3 – ld11 – cifra curentă 1
* P3 – ld12 – cifra curentă 2
* N3 – ld13 – cifra curentă 3
* L1 – ld15 – semnul rezultatului
* U18 – btnc – numărare incrementare/decrementare
* T17 – btnr – selectare următoarea cifră
* U17 – btnd – schimbarea locației punctului

# Concluzii

Scopul acestui proiect a fost proiectarea și implementarea unei aplicații ce suportă înmulțirea și împărțirea numerelor în virgulă flotantă pe 32 de biți.

Entitatea principală Main are mai multe componente ce îndeplinesc diferite sarcini. De asemenea, are intrări pentru acțiunile utilizatorilor, cum ar fi încărcarea numerelor, selectarea numărului pe care doresc să îl manipuleze și alegerea unei operații (înmulțire sau împărțire). De asemenea, are o intrare pentru semnalul de ceas și ieșiri pentru diverse componente de afișare, cum ar fi display-ul cu șapte segmente și indicatorul curent al cifrei.

În entitatea principală sunt instanțiate mai multe componente, inclusiv o unitate de logică aritmetică (ALU) pentru a efectua calcule, două componente de convertor BCD la IEEE pentru a converti reprezentarea în zecimal bazat pe codul binar (BCD) a numerelor în reprezentarea în virgulă mobilă IEEE și două componente de convertor IEEE la BCD pentru a converti reprezentarea în virgulă mobilă IEEE înapoi în BCD. Există de asemenea două componente de înregistrare a numerelor pentru a stoca și a manipula reprezentarea BCD a numerelor introduse de către utilizator. Există o componentă debouncer pentru debouncing-ul intrărilor comutatorului și o componentă de afișare cu șapte segmente pentru afișarea rezultatului calculatorului.

În arhitectura entității principale, există un proces care este sensibil la semnalul de ceas. Acest proces se ocupă de încărcarea și manipularea numerelor, precum și de calcule și actualizările afișajului. Când utilizatorul încarcă un nou număr, procesul trimite un semnal de încărcare în registrul de numere corespunzător și în convertorul BCD - IEEE. Convertorul BCD - IEEE convertește reprezentarea BCD a numărului în reprezentarea IEEE cu virgulă mobilă și o trimite la ieșirea entității principale. După ce au fost încărcate ambele numere, procesul trimite un semnal de încărcare către ALU și selectează înmulțirea sau împărțirea în funcție de intrarea utilizatorului. ALU efectuează calculele și trimite rezultatul la ieșirea principală a entității. Rezultatul este apoi convertit înapoi la reprezentarea BCD de către convertorul IEEE - BCD și afișat pe afișorul cu șapte segmente.

Concluzionând, acest proiect a fost o oportunitate de a studia mai bine operațiile în virgulă flotantă, conversiile IEEE-BCD și BCD-IEEE, dar și reprezentarea numerelor în formatul specific. De asemenea, provocarea cea mai mare a fost implementarea algoritmilor propriu-ziși de înmulțire și împărțire, dusă la bun sfârșit datorită căutării de soluții în surse variate.

# Bibliografie

<https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/>

<https://www.geeksforgeeks.org/multiplying-floating-point-numbers/?ref=gcse>

<https://www.rfwireless-world.com/Tutorials/floating-point-tutorial.html>

<https://electronics.stackexchange.com/questions/20791/what-is-the-sequence-and-how-it-is-going>

<https://www.semanticscholar.org/paper/Implementation-of-Single-Precision-Floating-Point-Chaitanya/cfd118aa92262dc032b3e2ac4ecd12715aa1f947>

<https://allaboutfpga.com/>

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