GENERAREA DE MICRO-OPERATII DE VIRGULA FLOTANTA UTILIZATE IN GRAFICA, IMPLEMENTATE PE FPGA

REZUMAT

Ovidiu SICOE

Coordonator:
Prof. Dr. Ing. Mircea POPA

Facultatea de Automatică și Calculatoare
Universitatea Politehnica Timișoara

Timisoara
2018

1 Introducere

O mare parte a dispozitivelor mobile din ziua de astăzi au ecrane pentru interacțiunea cu utilizatorul uman. Astfel, se duce o muncă permanentă de îmbunătățire a performanțelor acestor dispozitive, din punct de vedere al vitezei de execuție, al vitezei de calcul și al miniaturizării. În plus, reutilizarea resurselor hardware ar contribui semnificativ la scăderea costurilor de producție. O posibilă implementare a acestei soluții ar fi utilizarea unor dispozitive fizice reprogramabile, cum ar fi FPGA-urile(Filed Programmable Gate Array).

De asemenea, conținutul grafic prezentat a devenit tot mai bogat și, astfel, a fost nevoie de o permanentă îmbunătățire a procesoarelor grafice. Pentru a realiza acest lucru, procesoarele grafice și nu numai, au început să utilizeze algoritmi tot mai performanți, dar și să încorporeze operatori aritmetici pentru operații matematice compuse.

În plus, se pot chiar folosi operatori aritmetici imprecișI, acolo unde fidelitatea rezultatului final permite acest lucru. Extinzând această idee, se pot utiliza și reprezentări cu o precizie mai mică de numere reale, astfel încât rezultatul final să fie acceptabil pentru un utilizator uman. Aceste pierderi de precizie vin cu un câștig în timpul de execuție al algoritmilor, dar și cu o economie de putere.

Având în vedere cele descrise anterior, teza al cărei rezumat urmează propune o serie de idei studii și soluții care să ajute la dezvoltarea și îmbunătățirea procesoarelor grafice folosite de către dispozitivele mobile.

În primă fază am dezvoltat o serie de arhitecturi pentru operatori matriciali folosiți într-un pipeline grafic, implementați pe FPGA. Aceste arhitecturi sunt generice și pot fi configurate din punct de vedere al dimensiunii operatorilor folosiți, al gradului de paralelizare și al frecvenței țintă.

De asemenea, am studiat impactul utilizării a multiple formate de numere reale în diferitele stagii ale unui pipeline grafic. Pentru acest lucru am utilizat o implementare proprie a specificației Open GL ES 1.1 pe care am adaptat-o astfel încât să poată utiliza diferite reprezentări pentru numerele reale utilizate de-a lungul pipeline-ului. Pentru acest studiu am proiectat și implementat o serie de module software cu ajutorul cărora am creat și executat un proces prin care am reușit să generăm un volum mare de date de analizat. De asemenea, am folosit o parte din aceste module software pentru analiza datelor și pentru extragerea a diferite statistici.

La final, după analiza datelor rezultate, pe baza unor metrici deja existente pentru compararea a două imagini similare, am simțit nevoia introducerii unei noi metrici. Noua metrică introdusă are o semantică a valorilor ușor de înțeles și interpretat și este in același timp ușor de calculat.

2 State Of The Art

În acest capitol al tezei se prezintă concluziile desprinse din analiza unor articole relevante pentru tematica tezei, în principal din trei mari categorii:

- Algoritmi și operatori implementați pe FPGA pentru diferite operații aritmetice simple sau compuse
- Operatori aritmetici imprecișI
- Metrici pentru compararea unor imagini similare

În prima categorie, am prezentat o serie de operatori aritmetici pentru operații aritmetice simple dar și îmbunătățirile aduse de utilizarea unor operatori aritmetici compuși, cum ar fi operatorul de înmulțire-adunare. Astfel, acesta are următoarele avantaje clare:

- Nu este nevoie de rotunjiri separate pentru operația de înmulțire respectiv pentru cea de adunare.
- Este suficient să se realizeze o singură despachetare și o singură împachetare din formatul IEEE 754 în formatul intern utilizat de unitate și invers, din formatul intern în formatul IEEE 754, pentru fiecare operație compusă.
- Reducerea costurilor de implementare prin folosirea acelorași componente atât pentru operația de înmulțire, cât și pentru cea de adunare.

De asemenea, am analizat o serie de articole care prezentau algoritmi specializați pentru diferite tipuri de aplicații particulare. Acești algoritmi au fost special concepuți și implementați pentru particularitățile FPGA-urilor, astfel încât să profite de avantajele acestora. Ca și analiză a performanțelor acestor algoritmi, ei au fost comparați cu versiuni implementate pe procesoare de uz general și de fiecare dată, variantele sintetizate pe FPGA au avut rezultate superioare, din punct de vedere al timpilor de execuție sau al lărgimii de bandă în accesul memorie, chiar dacă rulau la frecvențe mult mai reduse.

În continuare, am analizat o serie de articole care evidențiau consumul redus de putere și uneori timpii reduși de execuție al unor algoritmi imprecișI, implementați pe FPGA, comparativ cu alternativele de precizie ridicată. Rezultatele acestor operatori imprecișI erau analizate din punct de vedere al acceptabilității lor prin prisma unor metrici care analizau diferențele față de rezultatele precise echivalente.

Acești operatori imprecișI au fost validați și verificați pe aplicații din domeniul graficii digitale, în algoritmi de procesare a imaginilor. Din cauză că rezultatul final este destinat evaluării de către un utilizator uman, în foarte puține articole am găsit o evaluare matematică a pierderii de precizie regăsite în imaginea finală.

Astfel, am atins și ultima categorie de articole studiate, cele referitoare la metricile utilizate pentru compararea unor imagini similare, rezultate din folo-

sirea unor algoritmi sau operatori aritmetici mai puțin precișI. Acest mod de evaluare a diferențelor, prin metrici, este unul obiectiv, care este dispus la o analiză automatizată și poate fi ușor scalat la un număr mare de rezultate.

Pe baza unor metrici des folosite în aceste articole am realizat analiza rezultatelor obținute în studiul nostru și plecând de la acestea am introdus o nouă metrică.

3 Noțiuni preliminare

În acest capitol din teză am introdus o serie de concepte utilizate pe parcursul tezei, pentru dezvoltarea soluțiilor concepute și pentru realizarea studiului propus. Astfel, am introdus următoarele concepte:

- Pipeline grafic și specificația OpenGL ES 1.1
- Reprezentarea obiectelor tridimensionale prin poligoane distribuite pe suprafața acestora.
- Transformări de coordonate folosite în pipeline-ul grafic, cum ar fi translația și scalarea
- Reprezentări de numere reale în virgulă fixă sau mobilă

Pentru pipeline-ul grafic și specificația OpenGL ES 1.1 am introdus concepte de baza, cum ar fi principalele stagii și legătura dintre ele. De asemenea am prezentat pe scurt cum se ajunge de la date de intrare tridimensionale la proiecții bidimensionale ale acestora. În continuare, am detaliat aspecte legate de reprezentarea obiectelor tridimensionale date ca intrări. Această specificație descrie intrările prin triunghiuri într-un spațiu tridimensional. Triunghiurile sunt date prin coordonatele vârfurilor sale. Astfel, vârfurile fiecărui triunghi pot fi furnizate independent, sau pentru triunghiurile care au vârfuri comune există un mod de reprezentare prin fâșii.

Pentru transformările de coordonate am prezentat conceptele matematice ale acestor transformări geometrice afine și am exemplificat prin prezentarea matricilor corespunzătoare translației și scalării într-un spațiu bidimensional.

În finalul acestui capitol de concepte am introdus noțiunile de bază și formulele matematice de calcul pentru două tipuri de reprezentări de numere reale, respectiv virgulă fixă și virgulă mobilă.

4 Operatori matriciali folosiți în pipeline-ul grafic

Folosind modelele matematice prezentate în capitolul anterior, am exploatat particularitățile matricilor aferente translației și scalării și am conceput o arhi-

tectură de operatori matriciali, sintetizabili pe FPGA, care să poată fi configurați după următorii parametrii:

- Dimensiunea numerelor reale. Am folosit reprezentarea cu virgulă mobilă descrisă în specificația IEEE, putând configura dimensiunea mantisei, precum și a exponentului:
  - exponent 5 biți; mantisă: 10 biți(half precision float)
  - exponent 8 biți; mantisă: 23 biți(float)
  - exponent 11 biți; mantisă: 52 biți(double precision float)
- Frecvența țintă pentru generare. Acest parametru influențează numărul de stagii ale pipeline-ului operatorului:
  - 100 Mhz
  - 200 Mhz
  - 300 Mhz
  - 400 Mhz
- Gradul de paralelizare. Acest parametru influențează direct resursele utilizate în implementarea operatorului:
  - Maxim
  - Mediu
  - Minim

Acești parametrii de configurare permit obținerea unor operatori particularizați pentru o gamă largă de aplicații, în funcție de nevoile fiecăreia. Pentru fiecare operator, respectiv scalare și translație, în funcție de fiecare tripletă de parametrizare, am obținut un număr de 36 de operatori.

Astfel, în funcție de gradul de paralelizare, pentru fiecare transformare geometrică, am creat operatori cu:

- Șase unități de procesare, în care fiecare unitate calculează rezultatul aferent unei poziții din matrice. O matrice rezultat are nouă poziții, însă trei dintre ele sunt identice cu intrările, din cauza formei particulare a acestor matrici.
- Două unități de procesare. Fiecare unitate calculează rezultatele pentru câte una dintre cele două linii diferite din matricea rezultat. Astfel, în ciclul de tact k va fi pregătită prima poziție din fiecare linie, în ciclul $k+1$ va fi produsă următoarea poziție, iar în ciclul $k+2$ va fi gata ultima poziție a liniei.
- O singură unitate de procesare. În acest caz unitatea de procesare produce rezultatele pentru toate cele șase poziții ale matricii rezultat. În mod analog, după modelul anterior, va fi nevoie de șase cicluri de tact pentru a produce rezultatele.

Pentru implementarea acestor arhitecturi, am folosit biblioteca FloPoCo, care oferă o infrastructură de generare a codului VHDL necesar sintezei pentru FPGA a acestor operatori. Modelul necesar generării a fost descris în C++ și a inclus și o parte de validare și verificare a operatorilor generați. Această parte de verificare și validare a rezultat în generarea testbench-ului aferent fiecărui operator în parte.

La finalul fiecărei secțiuni aferente operatorilor de translație și, respectiv, scalare se prezintă un tabel cu rezultatele sintezei acestor operatori. Aceste rezultate susțin ideea că acești operatori pot fi ușor asociați specificului diferitor clase de aplicații, în funcție de nevoile și constrângerile fiecăreia.

5 Implementare calcule aproximative

În acest capitol al tezei se prezintă modulele software concepute și procesul creat pentru realizarea studiului impactului utilizării a multiple formate de numere reale pe parcursul aceluiași pipeline grafic.

În prima fază a pipeline-ului grafic descris de specificația OpenGL ES 1.1 se procesează vârfurile triunghiurilor prin care este descris obiectul tridimensional. Această procesare constă în aplicarea unor transformări geometrice afine, prin înmulțiri matriciale succesive. Am ales astfel să modificăm o implementare software proprie a specificației OpenGL ES 1.1 pentru a putea utiliza un format de reprezentare a numerelor reale în primul stagiu și un alt format în restul pipeline-ului.

Pentru a avea un volum satisfăcător de date de intrare, am creat un modul software care scana o serie de pagini web dedicate obiectelor tridimensionale produse de numeroși artiști grafici și astfel am colectat aproximativ 3000 de fișiere conținând obiecte tridimensionale. Având în vedere faptul că specificația OpenGL ES 1.1 nu suportă decât triunghiuri ca poligoane de reprezentare a suprafețelor obiectelor tridimensionale, iar unele din aceste fișiere colectate conțineau poligoane cu mai mult de trei vârfuri, am fost nevoiți să creăm un alt modul software pentru transformarea acestor fișiere în unele care să conțină date de intrare în formatul acceptat. Unele din fișierele care nu au putut fi transformate acceptabil au fost eliminate. În urma acestei etape, am rămas cu aproximativ 2400 de obiecte tridimensionale valide.

Ulterior, aceste date au fost furnizate aplicației create să proiecteze aceste obiecte tridimensionale pe o suprafață plană, utilizând implementarea proprie a pipeline-ului grafic după specificația OpenGL ES 1.1. Astfel, aplicația constă din următorii pași:

1. ștergem suprafața de desenare
2. desenăm obiectul rotit cu unghiul de rotație curent, începând de la zero

3. obținem o imagine PNG aferentă proiecției obiectului tridimensional rotit
4. rotim obiectul tridimensional cu 2 grade în jurul vectorului (-1, -1, -1), obținând un nou unghi de rotație ce va fi folosit în cadrul următor
5. reluăm procesul de desenare(pasul 1) pentru desenarea următorului cadru

Pentru rularea aplicației pentru toate cele aproximativ 2400 de obiecte tridimensionale obținute am avut nevoie de 21 de zile de procesare, împărțite între trei stații care au lucrat în paralel. Aceste stații au avut următoarea configurație minimală:

- procesor Intel Core i7 la ≈2500 MHz, cu patru core-uri, opt thread-uri
- 16 GB RAM
- SSD

În urma rulării aplicației, am obținut mai mult de trei milioane de cadre, din care aproximativ 450 000 sunt imagini de referință. Astfel, pentru a putea analiza diferențele imaginilor alterate care au rezultat în urma utilizării unor formate de reprezentare mai puțin precise, relativ la imaginile de referință, am creat un modul de comparare a două imagini care să producă o imagine diferență, dar și un fișier cu valorile diferitor metrici.

La final, am avut nevoie de un modul software care să centralizeze datele din fișierele cu metrici create la pasul anterior și astfel am creat un modul special care are două subsisteme: unul pentru citirea datelor și unul destinat procesării acestora. Din cauza timpului mare de citire a datelor, modulul de citire era încărcat static de către aplicația de procesare, iar modulul de procesare era încărcat dinamic. Astfel, am putut citi datele o singură dată la rularea aplicației, iar modulul de procesare putea fi adaptat continuu, recompilat și reîncărcat de aplicație, fără să fie nevoie de recitirea datelor.

Datele au fost obținute în urma utilizării combinațiilor de formatelor din următorul tabel:

	F_0	F_1
1	FP[1]	FP
2	16:16[2]	16:16
3	16:16	FP
4	FP	16:16
5	32:32	32:32
6	32:32	FP
7	FP	32:32

Tabela 1: Formate folosite

[1]Floating Point - virgulă flotantă]
[2]format de virgulă fixă specificat ca dimensiune parte întreagă : dimensiune parte fracționară

Astfel, F_0 reprezintă formatul folosit în primul stagiu al pipeline-ului grafic, iar F_1 formatul folosit în restul stagiilor. Prima linie din tabel reprezintă combinația de precizie maximă care a fost utilizată pentru obținerea imaginilor de referință. Celelalte linii descriu combinații de precizie în care cel puțin unul dintre formate este mai puțin precis.

6 Rezultate calcule aproximative

Acest capitol al tezei de doctorat prezintă rezultatele obținute în urma procesului de analiză al influenței folosirii mai multor formate de reprezentare a numerelor reale în cadrul stagiilor unui pipeline grafic.

În introducerea capitolului am prezentat pe scurt biblioteca de obiecte tridimensionale create, cu caracteristicile acestor obiecte precum și o serie de grafice și histograme care prezintă distribuția acestora. S-a putut astfel observa varietatea de obiecte utilizate în studiu. Astfel am avut de la obiecte cu un volum mic, în jurul unității de volum, până la obiecte cu un volum de ordinul milioanelor de unități de volum. De asemenea, am avut de la obiecte reprezentate prin 2 triunghiuri, până la obiecte reprezentate prin aproximativ 450000 de triunghiuri.

Ulterior, am prezentat rezultate unei scurte analize subiective, realizate de către un subiect uman, prin analiza comparativă a imaginilor alterate obtinute, relativ la corespondentele imagini de referință. Am putut astfel observa că o astfel de analiză subiectivă nu e suficientă pentru a sublinia diferențele uneori nesemnificative între imaginile obținute. Am continuat astfel prin introducerea în analiză a unor metrici des utilizate în lucrările de specialitate din domeniu: MSE(Mean Square Error) și PSNR(Peak Signal-to-Noise Ratio).

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \tag{1}$$

$$PSNR(f,g) = 10\log_{10}(\frac{255^2}{MSE(f,g)}) \tag{2}$$

În ecuațiile precedente, f și g reprezintă mulțimea de pixeli ai celor două imagini, iar M și N reprezintă lățimea și înălțimea imaginilor. Valorile metricii MSE dau media diferenței pătratice între pixelii celor două imagini comparate. În același timp, PSNR este definită logaritmic pe baza MSE și a fost introdusă pentru a limita valorile luate de metrica MSE.

Am încercat urmărirea acestor valori și extragerea unor anumite tipare, iar varianta cea mai potrivită de urmărire a acestora a reprezentat-o graficul valorilor pentru fiecare obiect, pentru fiecare cadru în parte, în funcție de combinația

de precizie utilizată. Astfel, am observat o dependință între anumite caracteristici ale obiectelor tridimensionale proiectate, cum ar fi volumul acestora, între forma graficului valorilor și combinația de precizie utilizată. De asemenea, am realizat că pentru valorile acestor metrici variază foarte mult de la un obiect la altul, pentru diferite combinații de precizie, făcând dificilă comparația între rezultatele obținute prin utilizarea diferitelor combinații de reprezentare.

În urma acestei concluzii, am introdus o nouă metrică, pe care am definit-o relativ la numărul de pixeli utili ai proiecției obținute prin utilizarea combinației de formate de reprezentare cu cea mai mare precizie, astfel:

$$M(f,g) = \frac{1}{P(f)} \sum_{i=1}^{M} \sum_{j=1}^{N} D(f_{ij}, g_{ij}) \tag{3}$$

unde

$$D(f_{ij}, g_{ij}) = \begin{cases} 1, f_{ij} \neq g_{ij} \\ 0, f_{ij} = g_{ij} \end{cases} \tag{4}$$

și

$$P(f) = \textit{numărul de pixeli utili ai matricii de pixeli } f \tag{5}$$

Pentru o mai bună semantică a valorilor acestei metrici, am simțit nevoie să definim procentual această metrică și astfel am introdus următoarea definiție a acesteia, pe care am utilizat-o în continuare:

$$M\%(f,g) = M(f,g) * 100 \tag{6}$$

Metrica astfel definită prin ecuația de mai sus redă procentul de pixeli utili care sunt diferiți în imaginea alterată, relativ la imaginea de referință. Astfel, graficele acestei metrici aveau o formă similară cu cele ale metricilor MSE și PSNR, dar plaja valorilor este restrânsă.

La finalul prezentării rezultatelor studiului am introdus o serie de grafice în care am utilizat agregări ale celor trei metrici enumerate anterior pentru toate obiectele tridimensionale luate în considerare. S-au putut astfel observa anumite puncte de maxim și minim, conturate în jurul anumitor cadre. Aceste cadre corespund proiecției obiectelor rotite cu 60, 120, 180 și respectiv 240 de grade. Una dintre principalele explicații ale acestor puncte de maxim și minim o reprezintă imprecizia în reprezentarea valorilor funcțiilor trigonometrice sinus și cosinus utilizate în calcularea valorilor matricii de rotație utilizată în pipeline-ul grafic.

7 Contribuții, concluzii și perspective

În capitolul de încheiere am prezentat contribuțiile originale ale acestei teze de doctorat, concluziile acesteia, precum și câteva direcții de dezvoltare ale stu-

diilor prezentate.

Astfel, în prima parte a tezei am prezentat o serie de operatori matriciali, implementați pe FPGA, care pot fi configurați în momentul sintezei pe următoarele planuri:

- Dimensiunea numerelor reale.
- Frecvența țintă pentru generare.
- Gradul de paralelizare.

În continuare, pentru a putea extinde setul de operatori creați, am studiat influența folosirii mai multor formate de reprezentare a numerelor reale în stagiile unui pipeline grafic. Astfel, am modificat o implementare software proprie a specificației OpenGL ES 1.1 care să permită schimbarea formatelor de numere reale folosite, chiar în timpul rulării. De asemenea, am creat o suită de programe software care să ne ajute să realizăm următoarele:

- Colectarea obiectelor tridimensionale folosite ca date de intrare pentru pipeline-ul grafic.
- Transformarea datelor de intrare într-un format recunoscut de pipeline-ul grafic.
- O aplicație software care încorporează pipeline-ul grafic și îi furnizează datele de intrare pentru a produce proiecțiile obiectelor tridimensionale.
- Compararea imaginilor alterate, obținute prin folosirea unor combinații imprecise de precizie, cu cele de referință, obținute prin folosirea unui format considerat precis.
- Analizarea imaginilor obținute și generarea de valori ale unor metrici
- Centralizarea, filtrarea, prezentarea și interpretarea valorilor metricilor obținute

Folosind aceste module, am definit un proces care să ajute la studiul propus. în urma obținerii datelor rezultate din aplicarea acestui proces am prezentat concluziile studiului prin exemple semnificative, metrici concludente și grafice cu valorile acestor metrici.

Concluzionând, pe parcursul cercetării aferente elaborării acestei teze, am obținut următoarele rezultate concrete:

- Am realizat o arhitectură de operatori geometrici de translație și scalare care pot opera cu diferite formate de numere reale reprezentate în virgulă flotantă, inclusiv formate nestandard. De asemenea, aceste arhitecturi se pot configura astfel încât să rezulte diferiți operatori, în funcție de formatul ales, adâncimea dorită a pipeline-lor interne sau frecvența țintă.
- Am creat o bibliotecă de obiecte tridimensionale care pot fi folosite ca date de intrare pentru un pipeline grafic.

- Am creat un proces și suita de module software aferente cu ajutorul cărora am studiat și înțeles influența combinațiilor de precizie folosite în pipeline-ul grafic modificat, asupra rezultatelor finale.
- Am propus o nouă metrică cu o semantică a valorilor clară, care să ajute la evaluarea obiectivă a diferențelor dintre imaginile alterate și cele de referință.

La final, am prezentat câteva acțiuni care pot fi întreprinse în viitor pentru a continua cercetarea începută și prezentată în teza de doctorat:

- Crearea de operatori matriciali configurabili pentru transformarea geometrică de rotație, precum și pentru alte transformări geometrice afine utilizate în pipeline-urile grafice.
- Crearea de unități de calcul implementate pe FPGA care să transpună hardware conceptul prezentat prin modificarea pipeline-ului grafic software.
- După implementarea hardware a conceptului, să efectuăm o analiză comparativă cu soluții deja existente, din punct de vedere al performanțelor obținute.
- Utilizarea combinațiilor de formate pe o scară mai restrânsă; astfel, ar putea fi posibilă utilizarea de mai multe formate de reprezentări ale numerelor fracționare în cadrul aceluiași pipeline.
- Definirea unui prag pentru M% astfel încât valori mai mari decât acest prag să indice o imagine inacceptabilă de către un utilizator uman.