PARALELIZAREA ALGORITMULUI TRANSFORMATEI WAVELET RAPIDE

ing. Voicu Horaţiu

Universitatea Tehnică "Gh. Asachi" Iaşi

Rezumat: Lucrarea prezintă o modalitate de paralelizare a algoritmului transformatei wavelet rapide. După o scurtă introducere a transformatei wavelet şi a maşinilor paralele, se porneşte de la algoritmul secvenţial, bazat pe utilizarea filtrelor de tipul QMF (Quadrature Mirror Filter) şi se continuă cu implementări pe două arhitecturi de calcul diferite: reţea de transputere, respectiv reţea de staţii Sun. Se prezintă avantajele şi dezavantajele fiecărei arhitecturi pentru algoritmul paralel implementat. De asemenea, se prezintă performanţele de convergenţă, pe care le are transformata wavelet pentru semnale neperiodice, specifice fenomenelor tranzitorii. Utilitatea paralelizării algoritmului transformatei wavelet rapide este determinată de posibilitatea construirii unor analizoare spectrale în timp real cât mai performante.

Cuvinte cheie: filtrare digitală, decimare, funcţii wavelet, transformata wavelet, QMF (quadrature mirror filter).

1. Introducere

Formularea algoritmului Transformatei Fourier Rapide (TFR) de către J.W.Cooley şi J.W. Tukey la începutul anilor '60 a însemnat un pas important în utilizarea tehnicii de calcul la analiza şi studiul în timp real ale mediului înconjurător. Inventată la începutul secolului XIX, transformata Fourier este un instrument de bază în analiza modernă a semnalelor. Algoritmul Transformatei Fourier Discrete (TFD) necesită n^2 înmulţiri şi adunări pentru o intrare de dimensiunea n, în timp ce algoritmul FFT necesită doar $n \log_2 n$ astfel de operaţii.

Transformata Fourier s-a dovedit incredibil de versatilă în aplicaţii cum ar fi recunoaşterea formelor sau procesarea imaginilor. Totuşi, ea suferă de anumite restricţii. Limitările transformatei Fourier sunt determinate de condiţiile pe care trebuie să le satisfacă o funcţie pentru a fi transformabilă. În general, majoritatea semnalelor din natură pot fi exprimate ca funcţii ce admit transformată Fourier. O problemă care apare este aceea a unor semnale care au componente tranzitorii sau admit transformată Fourier, dar seria Fourier converge foarte încet la semnalul iniţial. Evident, acest lucru nu este convenabil în cazul utilizării calculatoarelor care pot folosi un număr finit de coeficienţi ai seriei Fourier. Acesta este unul din motivele pentru care s-au dezvoltat transformatele wavelet, care sunt destinate unei mulţimi mult mai largi de semnale. Relativ recent, au fost descoperite noi familii de funcţii ortonormate, care permit construirea unor transformate ce înlătură condiţiile severe ale transformatei Fourier, [2], [1].

2. Introducere în paralelism

Maşinile paralele au ca principal obiectiv exploatarea paralelismului inerent aplicaţiilor informatice. Nu toate aplicaţiile au acelaşi profil în ceea ce priveşte paralelismul: unele conţin mult paralelism, altele prea puţin. Pe lângă acest factor cantitativ, trebuie considerat şi un factor calitativ: maniera în care acest paralelism poate fi exploatat. Se disting trei surse principale:

- paralelismul de date
- paralelismul de control
- paralelismul de flux

Exploatarea *paralelismului de date* are la bază faptul că anumite aplicaţii tratează structuri de date regulate (vectori, matrici) repetând aceeaşi acţiune asupra fiecărui element al structurii. Resursele de calcul sunt asociate datelor. Deseori, datele de tip identic sunt în număr foarte mare.

Exploatarea *paralelismului de control* are la bază faptul că, o aplicaţie informatică este compusă din acţiuni care se pot executa în acelaşi timp. Acţiunile, numite şi task-uri, pot fi executate de o manieră mai mult sau mai puţin independentă pe resursele de calcul, numite procesoare elementare. Un procesor elementar poate fi văzut ca un calculator simplificat: el conţine, de obicei, o unitate de calcul, o unitate de control şi o memorie care stochează datele şi programele. Exploatarea paralelismului de control constă în a gestiona dependenţele între acţiunile unei aplicaţii pentru a obţine o alocare a resurselor de calcul cât mai eficace.

Exploatarea *paralelismului de flux* are la bază faptul că anumite aplicaţii funcţionează după modul de lucru în lanţ: se dispune de un flux de date, în general, similare, pe care se efectuează o suită de acţiuni în cascadă. Resursele de calcul sunt asociate acţiunilor şi sunt înlănţuite de o manieră astfel că rezultatul acţiunilor efectuate la timpul t sunt trecute la timpul $t+1$ la procesorul elementar următor. Acest mod de funcţionare se numeşte şi *pipe-line*.

3. Definirea şi clasificarea maşinilor paralele

O maşină paralelă este un ansamblu de unităţi de tratare, care comunică şi cooperează pentru rezolvarea unei aceleiaşi probleme. Există mai multe clasificări ale maşinilor paralele. Cea mai veche şi recunoscută la ora actuală este a lui M.J. Flynn, propusă la sfârşitul anilor 60, [3], [6]. Această clasificare a calculatoarelor este fondată pe analiza secvenţelor de instrucţiuni şi de date şi are următoarea formă:

Flux de instrucţiuni/ Flux de date	simplu	multiplu
simplu	**SISD** (von Neumann)	**SIMD** (Vectoriale şi celulare)
multiplu	**MISD** (Pipe-line)	**MIMD** (Multiprocesoare)

În această clasificare a calculatoarelor, maşinile **MIMD** corespund execuţiei concurenţiale în timp ce maşinile **SIMD** corespund execuţiei paralel. Maşinile **SISD** reprezintă ordinatorul secvenţial al lui von Neumann şi maşinile **MISD** constituie o clasă aparte, maşinile pipe-line.

1. Arhitectura SISD a lui Flynn

Modelul maşinii secvenţiale a lui von Neumann este apelat SISD (*Single Instruction stream Single Data stream*) în taxonomia lui Flynn. El este compus dintr-o memorie centrală unde sunt stocate datele şi programele unui procesor: unitatea de control citeşte instrucţiunile programului şi comandă unitatea de calcul care efectuează operaţiile asupra datelor.

2. Arhitectura SIMD a lui Flynn

În modelul SIMD (*Single Instruction stream Multiple Data stream*) al lui Flynn, numai unităţile de calcul sunt multiplicate: ele efectuează aceeaşi operaţie în acelaşi moment, însă asupra unor date diferite. Totul se execută în mod sincron, sub controlul unei unităţi centrale, care posedă ceasul maşinii SIMD.

3. Arhitectura MISD a lui Flynn

Modelul pipe-line, pe care îl regăsim la procesoarele RISC şi la supercalculatoarele vectoriale, conservă structura monoprocesor a modelului von Neumann. Totuşi, unităţile de calcul şi de control sunt decupate în etaje, fiecare fiind reponsabil de o parte a operaţiei ce trebuie efectuată. Deci, datele pot fi tratate la etajul 2, în timp ce altele sunt tratate la etajul 3. Fluxul de informaţii este continuu şi viteza de execuţie creşte cu numărul de etaje.

4. Arhitectura MIMD a lui Flynn

În modelul maşinii paralele de tip MIMD (*Multiple Instruction stream Multiple Data stream*) procesoarele elementare sunt multiplicate. Procesoarele sunt deci capabile să execute programe independente, apelate task-uri sau procese. Procesele care se execută pe procesoare distincte se comportă ca majoritatea calculatoarelor independente, funcţionând în mod asincron. Din nefericire, în majoritatea aplicaţiilor MIMD trebuie ca procesoarele să comunice date. Gestiunea accesului concurent la aceste date pune deseori probleme foarte dificile de sincronizare.

În funcţie de modul de organizare a memoriei, maşinile de tip MIMD sunt de două tipuri:

- **maşini cu memorie comună**, unde datele unei aplicaţii sunt plasate în acelaşi mediu, memoria comună. Astfel un proces p plasat pe un procesor P are posibilitatea fizică de a accesa datele unui alt proces q plasat pe un alt procesor Q. Această proprietate este foarte utilă, căci ea permite proceselor să schimbe informaţii:
- **maşini cu memorie distribuită**, unde trebuie adaugată o reţea externă de comunicaţie între procesoare pentru ca ele să poate schimba informaţii.

4. Algoritmul secvenţial al transformatei wavelet rapide

O funcţie din domeniul timp este translată de transformata Fourier într-o funcţie în domeniul frecvenţei. Acestă interpretare a funcţiei obţinută prin translare apare deoarece trasformata Fourier reprezintă o dezvoltare în serie după o bază de funcţii ortonormate, respectiv sinus şi cosinus, de durată infinită. Transformata Fourier Discretă este valabilă pentru funcţiile periodice în domeniul timpului. Dezvoltarea transformatelor wavelet a fost motivată la început de eforturile de a găsi un tip de transformată Fourier pentru fenomene tranzitive. Dacă o funcţie are multe vârfuri ascuţite, seriile Fourier asociate au o convergenţă foarte înceată. Acest lucru este de aşteptat, deoarece nu este surpinzător faptul că o funcţie care are vârfuri ascuţite să fie greu de exprimat cu ajutorul funcţiilor sinus şi cosinus. Această problemă se rezolvă dezvoltând astfel de funcţii în serii de funcţii care ele însele au astfel de vârfuri şi sunt nule în afara unui interval finit. Aceste funcţii se numesc *wavelets* şi se notează cu $w(x)$. Dezvoltarea în serie de funcţii wavelet impune cosiderarea următoarelor observaţii:

1. Cum sunt tratate funcțiile care au vârfuri mai ascuțite decât funcția wavelet principală? Aceasta problemă este rezolvată în seriile Fourier convenționale prin *înmulțirea* variabilei x cu întregi în dezvoltarea seriei. De exemplu, $sin(nx)$ are vârfuri care sunt de n ori mai înguste decât vârfurile lui $sin(x)$. Soluția care este folosită în seriile wavelet este de a înmulți variabila x cu o *putere* a lui 2. Această procedură de schimbare de scală se numește *dilatare*.

2. Deoarece funcția wavelet este nulă în afara unui interval finit, cum sunt tratate funcțiile care sunt diferite de zero pe un interval mare de pe axa numerelor reale? Această problemă nu apare la seriile Fourier convenționale, deoarece funcțiile folosite pentru dezvoltare sunt nenule pe întreaga axă reală. Soluția folosită în seriile de tip wavelet este de a decala funcția wavelet cu un număr întreg și de a forma combinații liniare: $\sum_{j=0}^{\infty} c_j w(2^j x-k)$. Prin această modalitate se poate controla intervalul pe care funcția se poate extinde.

O dezvoltare a unei funcții $f(x)$ în serie wavelet se face cu formula:

$$\sum_{\substack{-1\le j<\infty \\ -\infty<k<\infty}} A_{jk}\, w_{jk}(x)$$

unde:

$$w_{jk}(x)=\begin{cases} w\left(2^j x-k\right) & \text{daca } j\ge 0 \\ \phi(x-k) & \text{daca } j=-1 \end{cases}$$

Funcția $w(x)$ se numește funcție wavelet de bază a dezvoltării și $\phi(x)$ se numește funcția de scalare asociată lui $w(x)$.

Funcția de scalare trebuie să îndeplinească următoarele condiții [6]:

1. Funcția de scalare, ca și funcția wavelet, trebuie să fie nenulă pe un interval finit. Această condiție este o consecință a conceptului de bază a seriilor wavelet.

2. Funcția de scalare trebuie să satisfacă următoarea ecuație de dilatare:

$$\phi(x)=\sum_{i>-\infty}^{\infty} \xi_i \phi(2x-i) \qquad (1)$$

Această condiție nu este suficientă pentru determinarea completă a funcției de scalare. Ecuația (1) implică faptul că doar un număr finit de elemente al șirului ξ_i pot fi nenule.

3. Se poate observa că orice multiplu al unei soluții a condiției anterioare este deasemenea soluție. Se selectează o soluție preferată prin impunerea următoarei condiții:

$$\int_{-\infty}^{\infty} \phi(x)\,dx = 1$$

Aceste condiții pot fi privite și din punctul de vedere al științei calculatoarelor. Cu ajutorul ecuației (1) se poate determina complet $\phi(x)$ în toate punctele de forma p/q, unde q este o putere a lui 2. Pentru toate calculatoarele moderne astfel de puncte sunt singurele care există, deci $\phi(x)$ este complet determinată. Funcția wavelet asociată lui $\phi(x)$ este următoarea:

$$w(x)=\sum_{i>-\infty}^{\infty} (-1)^i \xi_{1-i}\phi(2x-i). \qquad (2)$$

Setul de funcții format de $\phi(x)$ și $w(x)$ este un sistem de funcții wavelet scalate și translate. Nu orice set de valori ξ_i poate fi folosit drept coeficienți ai funcției wavelet. Câteva condiții trebuie respectate pentru ca valorile ξ_i să poată fi considerate coeficienții unei funcții wavelet:

$$\sum_{k\in Z} \xi_{2k}=1, \quad \sum_{k\in Z} \xi_{2k+1}=1,$$

$$\sum_{k\in Z} \xi_k \xi_{k+2l}=1 \quad \text{pentru } l\neq 0, \quad \sum_{k\in Z} \overline{\xi_k}\,\xi_k=2.$$

Există formule de generare automată a coeficienților pentru dezvoltările în serie wavelet [1]. O dată ce un sistem de funcții wavelet este creat, poate fi utilizat pentru dezvoltarea unei funcții $g(t)$ cu ajutorul funcțiilor de bază.

Daubechies [6] a demonstrat că transformata wavelet poate fi implementată cu un set special de filtre cu răspuns finit la impuls (RFI) numite Quadrature Mirror Filter (QMF). Un filtru RFI execută o sumă de înmulțiri între coeficienții filtrului și eșantioanele cuantizate. Trecerea unui set de eșantioane prin filtru înseamnă de fapt operația de convoluție discretă a semnalului de intrare și coeficienții filtrului. Filtrele QMF sunt folosite în analiza discretă a vorbirii și analiza spectrală. Ele se disting față de alte filtre prin aceea că separă frecvențele înalte de cele joase ale semnalului de intrare. Punctul de divizare este între 0 Hz și jumătate din frecvența de eșantionare.

Ieșirea filtrului QMF este decimată la jumătate, adică din două în două eșantioanele sunt înlăturate. Ieșirea filtrului trece-jos este conectată la intrarea unei noi perechi de filtre trece-jos respectiv trece-sus (vezi figura 1). Această operație este repetată recursiv generând un algoritm de tip arbore sau

Figura 1. Arborele sau piramida algoritmului transformatei wavelet rapide. FTJ - filtru trece jos, FTS - filtru trece sus, ↓2 - decimare cu factor 2.

Figura 2. Împărţirea spectrului în benzi. Fiecare bandă acoperă jumătate din intervalul de frecvenţă şi jumătate din timpul de rezoluţie a benzii superioare.

piramidă, furnizând la ieşire un grup de semnale care divide spectrul semnalului iniţial ca în figura 2.

Mallat [1] a arătat că algoritmul de mai sus poate fi aplicat la transformata wavelet prin utilizarea coeficienţilor funcţiei wavelet ca parametri ai filtrelor QMF. Aceiaşi coeficienţi ai funcţiei wavelet sunt folosiţi atât în filtrele trece-sus (FTS), cât şi cele trece-jos (FTJ).

5. Paralelizarea algoritmului transformatei wavelet rapide

Se prezintă mai jos algoritmul secvenţial pentru calculul transformatei wavelet:

```
void TW(double *DataIn, int LungIn, double
*FiltruJ,
double *FiltruS, char lungfilt,
char niveluri, double **Datales)

/* se presupune că şirul de intrare
are 2 ^ (niveluri) puncte pe interval */
{
char i;
for (i = 0; i < niveluri; i++)
{
LungIn = DR(DataIn, LungIn,FiltruJ, FiltruS,
lungfilt, Datales[2 * i], Datales[(2 * i) + 1]);
DataIn = Datales[2 * i];
}
}

int DR(double *In, int LungIn,
double *FiltruJ,double *FiltruS, char lungfilt,
double *IesJ, double *IesS)
/* formează două ramuri */
{
Convoluţie(In, LungIn, FiltruJ, lungfilt, IesJ);
Convoluţie(In, LungIn, filtruS, lungfilt, IesS);
return (LungIn / 2);
}
```

O paralelizare adecvată pentru algoritmul în cauză este folosirea a două procesoare fiecare calculând un produs de convoluţie. Presupunem că procesorul care aplică filtrele trece jos este cu rol de master. Pseudocodul pentru execuţia paralelă este umătorul:

Stăpân:

```
void TWST(double *DataIn, int LungIn,
double *FiltruJ,double *FiltruS, char lungfilt,
char niveluri, double **Datales)
```

```
/* se presupune că şirul de intrare are
2 ^ (niveluri) puncte pe interval */
{
char i;
trimite_procesorului_sclav(LungIn);
for (i = 0; i < niveluri; i++)
{
trimite_procesorului_sclav(DataIn);
Convoluţie(DataIn, LungIn, FiltruJ, lungfilt, IesJ);
recepţionează(IesS);
DataIn = DataIes[2 * i];
LungIn /= 2;
}
}
```

Sclav:

```
void TWSC(double *DataIn, int LungIn,
double *FiltruJ,double *FiltruS, char lungfilt,
char niveluri, double **DataIes)

/* se presupune ca şirul de intrare are
2 ^ (niveluri) puncte pe interval */
{
char i;
recepţionează(LungIn);
for (i = 0; i < niveluri; i++)
{
recepţionează(DataIn);
Convoluţie(DataIn, LungIn, FiltruJ, lungfilt, IesJ);
trimite_procesorului_stăpân(IesS);
LungIn /= 2;
}
}
```

Se prezintă în continuare semnficaţiile funcţiilor folosite în algoritm precum şi acţiunea pe care o execută:

TW (Transformata Wavelet) conţine întreg algorimul transformatei pe care îl particularizează un anumit şir de intrare şi anumite filtre trece-jos, respectiv trece-sus.

DR (Descompunerea ramurilor) aplică semnalului de intrare filtrul trece-jos, respectiv trece-sus, furnizând la ieşire două semnale decimate cu factorul 2.

Convoluţie realizează filtrarea semnalului de intrare ţinând cont de filtrul care s-a dat ca parametru de intrare după formula:

$$y[k] = \sum_{l=0}^{\infty} x[k]h[l-k]. \qquad (3)$$

unde h este răspunsul la impuls al filtrului.

TWST (Transformata Wavelet STăpâan) reprezintă codul executat de procesorul master în varianta paralelă a algoritmului.

TWSC (Transformata Wavelet SClav) reprezintă codul executat de procesorul sclav în varianta paralelă a algoritmului.

Trimite, Recepţionează au implementări diferite în limbajul C paralel de pe transputere, respectiv din mediul de programare paralelă PVM (Paralel Virtual Machine), însă semantica lor este aceeaşi.

6. Rezultate obţinute pe diferite arhitecturi paralele

6.1 Implementarea pe transputere

Având în vedere puterea relativ mare de calcul a transputerelor, cât şi vitezele mari de comunicaţie ce pot fi atinse într-o reţea de transputere, am considerat utilă implementarea algoritmului transformatei wavelet pe aceste maşini de calcul. În cazul utilizării a două transputere, s-a obţinut reducerea la jumătate a timpului de execuţie faţă de programul secvenţial.

Spre deosebire de mediul PVM, limbajul C implementat pe transputere nu oferă facilităţi directe de transfer asincron al mesajelor. Pentru realizarea acestui lucru este nevoie să se creeze procese speciale, care să preia controlul transmiterii mesajelor [3], [4]. Implementarea realizată pe transputere este sincronă şi din acest motiv încărcarea procesoarelor pe etape de lucru a algoritmului, din punctul de vedere al calculului, trebuie să fie cât mai echilibrată pentru a obţine un timp minim de execuţie. În cazul implementării asincrone, după comanda de transmitere a mesajului, procesorul îşi poate continua execuţia lăsând pe seama procesului special transmiterea mesajului.

6.2 Implementarea pe reţea de staţii Sun

Natura algoritmului de calcul al transformatei de tip wavelet nu impune folosirea a mai mult de două procesoare care sunt relativ bine conectate între ele (viteza de comunicaţie este suficient de mare). În cazul

rețelei de stații Sun, comunicația între procese nu atinge performanțe foarte mari, astfel că paralelizarea unor relații recurente prin această metodă poate fi costisitoare datorită faptului că, la fiecare iterație, trebuie efectuat un schimb de date între procesoare. Având în vedere cazul special pe care-l comportă algoritmul de calcul al transformatei de tip wavelet prezentată în această lucrare, și anume, echivalarea unui timp de calcul al unui vector cu transmiterea aceluiași vector prin rețeaua mașinii virtuale, am considerat utilă o implementare în mediul PVM a acestui algoritm. În cazul rulării algoritmului pe două stații Sun, interconectate printr-un cablu Ethernet, timpul de execuție este mai mare decât în cazul secvențial, aceasta datorându-se timpului de comunicație care este prohibitiv.

6.3 Performanțele de convergență ale transformatei wavelet

Reconstrucția funcției inițiale se face cu ajutorul formulei de dezvoltare în serie:

$$f(x) = \sum_{\substack{-1 \le j < \infty \\ -\infty < k < \infty}} A_{jk}\, w_{jk}(x).$$

Pentru a putea face comparație între funcția inițială și cea obținută prin dezvoltare în serie, se introduc seriile parțiale:

$$S_j(x) = \sum_{-\infty < k < \infty} A_{jk}\, w_{jk}(x)$$

Aceste serii (ca și seriile Fourier) sunt convergente în sensul următor:

$$\lim_{n \to \infty} \int_{-\infty}^{\infty} \left| f(x) - S_n(x) \right|^2 dx = 0 \qquad (4)$$

Aceasta înseamnă că aria dintre graficele lui $f(x)$ și $S_n(x)$ se apropie de zero când n se apropie de ∞. Aceasta nu înseamnă că $S_n(x) \to f(x)$ pentru toate valorile lui x. Este interesant faptul că există puncte x_n unde $\lim_{n \to \infty} S_n(x)\ f \neq f(x)$. Ecuația (4) implică faptul că aria totală a acestui set de puncte este zero. Din fericire, multe aplicații ale transformatelor de tip wavelet necesită acest tip de convergență.

În continuare, sunt prezentate graficele pentru o funcție wavelet (care face parte din familia W4 inventată de Daubechies) (figura 3) și funcția de scalare (figura 4) atașată împreună cu dezvoltarea în serie wavelet a unui semnal rampă (figura 5). Din aceste figuri se poate observa faptul că șirul seriei S aproximează funcția inițială și se observă de asemenea, formarea unor puncte de tipul celor discutate mai sus.

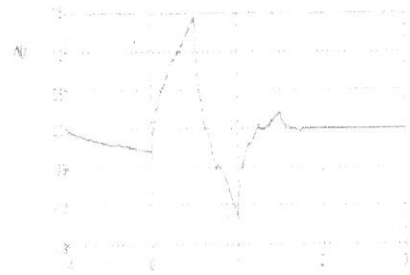

Figura 3. Funcția wavelet $w(x)$

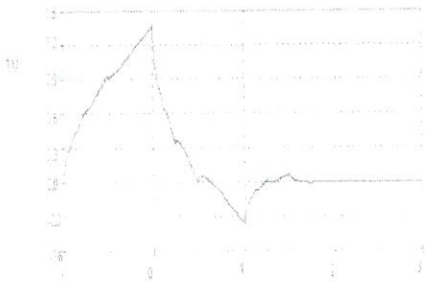

Figura 4. Funcția de scalare $\varphi(x)$

Algoritmul transformatei wavelet și corespondentul de reconstrucție al semnalului sunt mai simple și mai directe decât algoritmul FFT , complexitatea primului fiind $O(n)$. Transformatele de tip wavelet sunt folosite în foarte multe aplicații cum ar fi: procesarea de imagini, compresia de imagini, detecția contururilor în imagini, determinarea structurii la scară largă a universului.

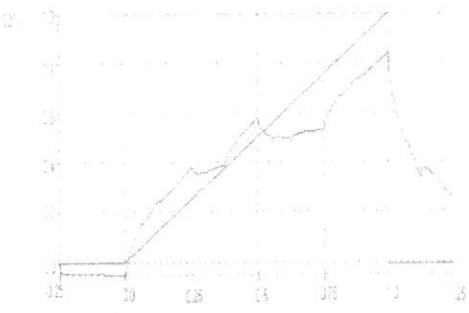

Figura 5. Funcția $S_8(x)$ pentru un semnal de tip rampă

7. Concluzii

În acest articol, s-a studiat paralelizarea calculului transformatei wavelet rapide. Lucrarea conține două părți: implementarea pe transputere și, respectiv, implementarea pe o rețea de stații Sun a algoritmului transformatei wavelet rapide. Prima a fost o abordare sincronă, iar a doua asincronă. Realizarea pe transputere a îmbunătățit timpul de execuție la jumătate a algoritmului secvențial deși acesta nu conține un grad mare de paralelism. Este evident faptul că mărirea numărului de procesoare nu va aduce o ameliorare a timpului de execuție datorită timpului de overhead, necesar gestionării informației între mai multe procesoare. Implementarea pe rețeaua de stații Sun a suferit datorită faptului că viteza de comunicație între stații este reletiv mică (cablu Ethernet) și rețeaua de stații Sun este un mediu nedeterminist din punctul de vedere al aplicației care se execută pe ea (în orice moment, resursa critică care o constituie cablul Ethernet poate fi cedată oricărui alt proces ce se execută pe rețea).

De fapt, rețeaua de interconectare de tip bus nu este bine adaptată pentru algoritmul de calcul al transformatei wavelet directe. Acesta poate fi ameliorat ținând cont de specificitatea rețelei. Datorită mediului nedeterminist pe care-l constituie rețeaua de statii Sun, măsurătorile de timp care se efectuează trebuie tratate printr-o analiză statistică fină.

Prin compararea rezultatelor celor două implementări se trage concluzia că sistemele bine conectate între ele (rețea de transputere) pot avea performanțe chiar paralelizând la nivel de iterație în programul secvențial inițial, în timp ce sistemele slab conectate (rețea de stații Sun) sunt mai puțin performante la acest nivel, compensând cu o bună comportare la aplicațiile de tip distribuit care necesită putere mare de calcul și schimb de mesaje de lungime mare.

Bibliografie

1. **CODY, MAC A.**:The Fast Wavelet Transform, beyond Fourier transforms. În: Dr. Dobbs's Journal, April, 1992.
2. **SMITH, J. R.**: The Design and Analysis of Parallel Algorithms, Oxford University Press, 1993.
3. * * * :Logical Systems C for the transputer. În: Computer Systems Architects, 1990.
4. * * *: Transputer Architecture and Overview. În: Computer Systems Architects, 1990.
5. **GEIST, AL.** ș.a.: PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked parallel computing, Massachusetts Institute of Technology, 1994.
6. **AKL, S. G.**: The Design and Analysis of Parallel Algorithms, Prentice Hall, 1993.