УДК 004.3

Я.М. Клятченко, В.П. Тарасенко, О.В. Тарасенко-Клятченко, О.К. Тесленко
Національний технічний університет України
“Київський політехнічний інститут”,
кафедра спеціалізованих комп’ютерних систем

SOFT-ПРОЦЕСОРНИЙ ПРИСТРІЙ НА БАЗІ СУЧАСНИХ ПЛІС ДЛЯ РЕАЛІЗАЦІЇ АЛГОРИТМУ АДАПТИВНОГО ПОРІВНЯННЯ ІНФОРМАЦІЙНИХ ОБ’ЄКТІВ

Запропоновано для розгляду підходи до створення спеціалізованої системи з використанням особливостей soft-процесорних блоків на базі сучасних серій ПЛІС для реалізації алгоритму швидкісного адаптивного порівняння інформаційних об'єктів.

Ключові слова: SOFT-процесор, адаптивне порівняння, інформаційний об’єкт.

The approaches for consideration are proposed for PLD-based specialized system with SOFT-processor creating for the implementation an algorithm of high-speed adaptive comparison of information objects.

Key words: SOFT-processor, adaptive comparision, information object.

Вступ

Досягнення в розвитку сучасних комп’ютерних технологій дали змогу покращити якість освіти, але зумовили виникнення додаткових проблем, що стосуються визначення запозичень в студентських роботах, наукових статтях, мас-медіа, інтернет-ресурсах та інших текстах, які мають містити оригінальну складову. Виникненню цих проблем сприяє антропогенний вплив факторів широкого використання комп’ютерних інформаційно-пошукових сервісів, які дозволяють швидко знаходити та використовувати оригінальні тексти з порушенням чинних законодавчих та моральних норм. Відомі програмні системи, наприклад [1], що призначені автоматизувати процес знаходження запозичень, тобто визначення такої когнітивної властивості інформації, як її оригінальність [2]. Для значної частини таких систем, функціонування яких покладається на сучасні засоби обчислювальної техніки, однією з найважливіших базових операцій виступає операція порівняння. Більшість програмних сервісів бере її за основу, і тому значне підвищення швидкодії засобів порівняння інформаційних об’єктів є сьогодні надзвичайно актуальною науково-технічною задачею.

Постановка задачі

У попередніх роботах [3–5] було розглянуто концепцію та виконано оцінку з точки зору апаратних витрат та швидкодії реалізації алгоритму адаптивного порівняння двох символьних послідовностей; апаратні структури для створення пристрою на базі сучасних програмованих логічних пристроїв – інтегральних схем, що реалізує алгоритм детального адаптивного порівняння послідовностей символів; варіант оптимізації алгоритму порівняння інформаційних об’єктів за рахунок введення спеціалізованих інструкцій, які дозволяють досягти показників швидкодії оброблення символьних послідовностей, що перевищують показники програмної моделі алгоритму на базі стандартних інструкцій, і тому створюють умови для реалізації низки гібридних (програмно-апаратних) реалізацій інформаційно-аналітичних систем, призначених для оперативного і надоперативного інтелектуального оброблення великих масивів даних різноманітного типу (електронний текст, числові дані, графіка, мультимедіа тощо). У цій роботі ми розглянемо підходи для підвищення швидкодії цього останнього варіанта реалізації.

Метод розв'язання задачі

У [3] як основний алгоритм для виявлення запозичень було запропоновано алгоритм адаптивного порівняння, де запозичення виявляють за допомогою порівняння окремих символів (наприклад, байтів) з обмеженням деякої довжини збігу, яка не береться до уваги.

При розгляді алгоритму адаптивного порівняння був запропонований етап швидкісного порівняння [6] (швидкісне адаптивне порівняння), який дозволяє формувати не точне, а приблизне значення рівня запозичень, але яке гарантовано не менше за реально існуюче.

Суть методики швидкісного адаптивного порівняння полягає в тому, що розглядаються послідовність-зразок A довжиною в n_a символів (наприклад, байт) та B – послідовність символів, яка перевіряється, довжиною n_b, і нехай c – довжини підпослідовностей (кількість символів), які порівнюються (де $1 \le c \le d$), а d – мінімальна довжина збігу двох підпослідовностей символів, яка береться до уваги. Значення d визначається (адаптується) залежно від конкретних умов порівняння. У загальному випадку із послідовності B вибирають всі підпослідовності довжиною d та визначають їх входження в послідовність A. У разі такого входження всі відповідні символи послідовності B фіксують.

Далі вибирають підпослідовності довжиною c із послідовностей A з кроком x (тут позначаються як $\boldsymbol{S_j}$ (j=1,2,…, r), де $r=n_a \ div \ x$), та B з кроком y, (тут позначаються як $\boldsymbol{F_h}$ (h=1,2,…,t), де $t = n_b \ div \ y$). Будемо вважати, що індекси j підпослідовностей $\boldsymbol{S_j}$ впорядковані шляхом послідовного вибору цих підпослідовностей з А, тобто при будь-яких a та b, таких, що $a < b$, підпослідовність $\boldsymbol{S_a}$ розташована до початку послідовності А ближче, ніж $\boldsymbol{S_b}$. Аналогічно припускаємо, що індекси h підпослідовностей $\boldsymbol{F_h}$ впорядковані так само.

Розглянемо дві довільні сусідні підпослідовності (рис. 1) довжиною c символів. Якщо початок деякої підпослідовністі довжиною d символів (на рис. 1 відзначено пунктирною лінією) збігається з початком підпослідовності довжиною c символів, то така d-підпослідовність безумовно містить відповідну c-підпослідовність. Наступна d-підпослідовність (на рисунку виділена товстою лінією) вже не міститиме поточну c-підпослідовність, але повинна містити наступну c-підпослідовність. Для цього необхідно і достатньо, щоб $d \ge c+x-1$, звідки $x \le d - c + 1$. Будь-яка наступна d-підпослідовність буде містити цю наступну c-підпослідовність, зокрема випадок, аналогічний показаному на рисунку пунктирною лінією. Тоді, якщо значення x задовольнятиме умову $1 \le x < d - c+1$, будь-яка підпослідовність з А із довжиною d (мінімальною довжиною збігу) буде повністю містити одну з вибраних підпослідовностей з А довжиною c. У випадку, коли послідовності A та B мають спільну підпослідовність, довжиною не менше ніж d символів, згідно з вищевикладеним, серед c-підпослідовностей $\boldsymbol{S_j}$ (j=1,2,…, r) знайдеться принаймні одна (позначимо її як $\boldsymbol{S_a}$), що повністю міститься в цій d-підпослідовності. Якщо y=1, а d-підпослідовність спільна, то незалежно від її розташування в послідовності B, серед підпослідовностей $\boldsymbol{F_h}$ (h=1,2,…,t) існуватиме і підпослідовність, яка збігається з $\boldsymbol{S_a}$.

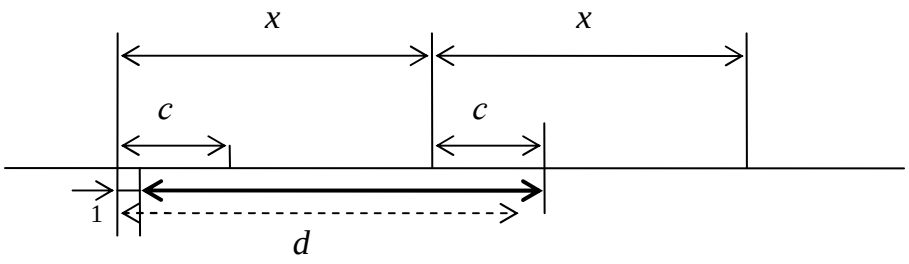

Рис. 1. Приклад розташування послідовностей

При швидкісному адаптивному порівнянні кожна підпослідовність $\boldsymbol{S_j}$ порівнюється з кожною підпослідовністю $\boldsymbol{F_h}$, і у випадку, хоча б одного збігу відповідна підпослідовність $\boldsymbol{S_j}$ відзначається. Якщо в результаті для деякого a ($a \in \{1,2,...,r-1\}$) $\boldsymbol{S_a}$ та $\boldsymbol{S_{a+1}}$ не відзначені, то це означає, що всі символи між цими підпослідовностями, а також символи самих підпослідовностей (всього $x+c$

символів), за умови, що y=1, не можуть бути запозиченими в A при заданій мінімальній довжині запозичення. Тобто, при $y=1$, у результаті порівняння кожної c-підпослідовності $\boldsymbol{S_j}$ із кожною c-підпослідовністю $\boldsymbol{F_h}$, будуть визначені всі спільні підпослідовності довжиною не менше d символів.

Використання пристроїв на базі сучасних ПЛІС для реалізації алгоритмів детального адаптивного порівняння послідовностей символів може дозволити отримати спеціалізований апаратний комплекс, що значно прискорить процес виявлення запозичень і визначення їх кількісної величини та може використовуватись або окремо з доступом до даних на дискових накопичувачах, або через обчислювальну мережу доступу до архівів даних тощо. Наприклад, сучасні ПЛІС мають в своєму складі вбудовані апаратні блоки, що реалізують функції доступу до мереж Ethernet 10/100/1000 Mbit/s. Також вони містять прийомопередавачі (послідовні високошвидкісні інтерфейси або трансивери), які мають різні показники швидкодії і дальності передачі. Різні варіанти цих послідовних прийомопередавачів застосовуються як для міжчіпового або міжплатного обміну, так і для високошвидкісного обміну даними (наприклад для Virtex-7 від Xilinx до 28Gbit/s на канал, або загальною пропускною здатністю 2,8Tbit/s) для мереж різного масштабу. Отже, висока швидкість обміну інформацією з зовнішніми джерелами обумовлює дуже високі вимоги до швидкодії засобів порівняння інформаційних об'єктів. Отже, підвищення швидкодії цих засобів прямо пропорційно впливає на забезпечення інформаційної стійкості [2] сучасних комп'ютерних технологій.

Основні виробники програмованих логічних пристроїв пропонують до використання в складі нових ПЛІС крім апаратних блоків (hard-процесори) ще й процесори, що можуть бути реалізовані програмним способом (soft-процесори). Ці процесорні soft ядра RISC архітектури, які мають свою систему команд, можуть бути оптимізовані за швидкодією для виконання таких завдань, як обробка символьних послідовностей за рахунок введення спеціалізовних інструкцій користувача. Зазвичай, спеціальні інструкції набагато краще враховують особливості алгоритмів для виконання спеціалізованих завдань. Також фірмові засоби проектування та оптимізації логіки ПЛІС можуть дозволити оптимізувати ці команди, а гнучкість архітектури FPGA – імплементувати в жорстку логіку. Спеціалізовані команди soft ядра для реалізації на ПЛІС мусять найбільшою мірою реалізовувати особливості алгоритму швидкісного адаптивного порівняння не тільки символьних, а й будь-яких послідовностей, якщо вважати, що для представлення кожного елемента цих послідовностей достатньо 1 байта (символу). А якщо врахувати можливість використання декількох soft-процесорних ядер в складі системи разом з апаратними блоками, то це дає можливість отримати багатопроцесорну систему на одній ПЛІС, яка має значну перевагу в швидкодії перед однопроцесорною системою (наприклад, сучасні ПЛІС компанії Xilinx® дозволяють створити систему з деякої кількості soft-процесорів MicroBlaze™).

Для створення пристрою на ПЛІС, що забезпечує ефективну реалізацію методики швидкісного адаптивного порівняння розглянемо команди, які було запропоновано в [5] :

Simple_Scan використовує стандартні 32-розрядні регістри загального призначення мікропроцесорного ядра MicroBlaze, які позначено:

-rD – в якому зберігаються елементи $\boldsymbol{S_j}$ (j=1,2,…, r) і кожний із яких є підпослідовністю довжиною 4 байти (символи) послідовності A;

-rS – в якому зберігаються елементи $\boldsymbol{F_h}$ (h=1,2,…,t), кожний елемент якої як і у випадку послідовності $\boldsymbol{S_j}$ є підпослідовністю довжиною 4 байти (символи) послідовності A.

-rC – регістр поточних даних лічильника кількості символів для порівняння.

-rA – регістр для зберігання початкового значення адреси 5-го символу послідовності B в пам'яті .

Псевдокод алгоритму команди *Simple_Scan*:

```
while (rD)<>(rS) and (rC)<>0 do
begin   (rS) ←(rS)<<8
  (rS)← (rS) V byte ptr MEM[rA]
   (rC)← (rC)-1
   (rA) ← (rA)+1
end
```

Double_Scan *з*а функціональним призначенням дана команда використовує дві групи регістрів. Перша група повністю відповідає команді *Simple_Scan*. Друга група регістрів це:

-rD2 – регістр для 4 байт для зберігання наступних чотирьох символів послідовності A, тобто наступного значення підпослідовності S_{j+1} (j=1,2,…, r-1).

-rS2 -регістр зсуву, що вміщає 4 байта для зберігання поточного F_{h+1} початкове значення –4 символи послідовності B , починаючи із символу за номером (d-4).

-rA2 –початкове значення адреси – d-го символу послідовності B.

Цикл порівняння команди виконується за два кроки. Далі наведений псевдокод алгоритму команди першого кроку:

(TEMP) ←1
While (TEMP=1) and ((rC)<>0) do
Begin
if (rD)<>(rS) then TEMP=1 else TEMP=0 одночасно rS2←(rS2) << 8
(rS))← (rS)<< 8 одночасно (rS2)← rS2 V *byte ptr MEM[rA2]*
(rS)← (rS) V byte ptr MEM[rA]
(TEMP) ←(TEMP) ∧ (rD2 <>rS2)
(rA)←(rA)+1
(rA2) ←(rA2)+1
(rC) ←(rC)-1
end

Команда одночасно виконує сканування двох сусідніх елементів (підпослідовностей) S_j (j=1,2,…, r), забезпечуючи точніше визначення граничного значення оригінальності при швидкісному адаптивному порівнянні.

Fast_Scan визначає входження підпослідовності S_j в підпослідовність довжиною c^* символів послідовності B, де $d \geq c^* > c$. Оскільки у швидкісному адаптивному порівнянні необхідно лише позначити (або не позначити) підпослідовність S_j з A , то за допомогою цієї інструкції виявляється лише факт входження (або не входження) S_j у відповідну підпослідовність із B . При використанні такого спеціалізованого пристрою значення кроку $y \leq c^* - c+1$.

Використання команди *Fast_Scan* дозволить використовувати значення $y > 1$. У випадку $c=4$ можна використати один із регістрів загального призначення MicroBlaze і відповідно команду для його завантаження. Щодо підпослідовності з B, то команда *Fast_Scan* повинна виконати завантаження багаторозрядного регістру з пам’яті. Під багаторозрядним регістром тут вважається або набір стандартних регістрів загального призначення мікропроцесорного ядра, “зчеплених” послідовно, або регістр, що реалізований, наприклад, на базі тригерів логічних комірок ПЛІС.

Тепер розглянемо варіанти використання в цій системі декількох процесорних ядер. Як відомо, швидкодія та функціональність є вагомими причинами для розроблення систем з декількома процесорами. Сьогодні використання багатопроцесорної програмованої логіки в існуючих системах, що пов'язані з обробленням великих масивів даних (системи зв'язку, мультимедіа тощо), практично підтвердило свою ефективність. Додаткові процесори можуть виступати у якості сопроцесорів до основного або керуючого процесора, або незалежно від нього обробляти дані. У нашому випадку можливі такі комбінації побудови багатопроцесорної системи для розв’язання поставленої задачі:

– система працює з єдиною послідовністю, оброблення якої розпаралелюється між декількома процесорами та керуючим модулем. Такий спосіб не вимагає зміни запропонованих мікроінструкцій користувальницьких команд;

– для оброблення єдиної піддослідної послідовності, що надходить в систему з високою швидкістю у вигляді потоку, організується процесорний конвеєр, де кожний процесор виконуватиме свій етап оброблення символьних даних або, інакше кажучи, виконувати визначену дію алгоритму адаптивного порівняння, та передавати проміжні результати іншому процесору для подальшого оброблення. Цей спосіб дозволяє збільшити пропускну спроможність системи. Кожне

процесорне ядро буде оптимізоване під інструкцію, яка відображає певний етап алгоритму швидкісного порівняння. Для цього випадку необхідні інші інструкції, суть апаратної реалізації яких для конвеєрного оброблення викладено в [7].

Висновки

Використання спеціалізованих процесорних команд користувача у складі системи, що реалізує алгоритм швидкісного адаптивного порівняння інформаційних об'єктів, дає змогу досягти показників швидкодії оброблення символьних послідовностей, що перевищують показники системи на базі стандартних інструкцій.

Один із способів удосконалення такої системи з погляду підвищення швидкодії полягає в можливостях сучасних ПЛІС імплементувати в систему декілька soft-процесорних ядер з використанням оптимізованих процесорних команд користувача. У розглянутих варіантах використання декількох процесорних ядер швидкість обробки послідовностей зростає пропорційно кількості процесорних ядер, але може бути обмежена швидкістю обміну даними між процесорами та керуючим модулем.

В якості подальшого підвищення швидкодії можна буде розглянути таку систему, яка виконує декілька незалежних завдань, тобто працює з декількома піддослідними послідовностями та зразками, що, наприклад, надходять від різних користувачів.

1. http://www.antiplagiat.ru 2. Тарасенко В.П. Автоматизація оцінки оригінальності інформації / В.П.Тарасенко, А.Ю. Михайлюк, О.К. Тесленко, О.С. Осипов // Наукові записки українського науково-дослідного інституту зв'язку. – 2007. – № 1. – С. 95–100. 3. Тарасенко В.П. Ефективність ПЛІС-реалізацій адаптивного порівняння послідовностей символів / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко // Науковий вісник Чернівецького університету: Зб. наук. праць. – Вип. 446. – Чернівці. – С. 23–29. –(Серія "Комп'ютерні системи та компоненти"). 4. Тарасенко В.П. Структури для ПЛІС реалізацій детального адаптивного порівняння послідовностей символів / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко // Міжвузівський збірник наукових праць "Наукові нотатки". – Вип. 27. – Луцьк, 2010. – С. 306–314. 5. Тарасенко В.П. Команди спеціалізованого процесора на ПЛІС для адаптивного порівняння інформаційних об'єктів / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко А.Ю. Михайлюк // НАУ "ХАІ", Наук.-техн. журн."Радіоелектронні і комп'ютерні системи". № 7. – 2010. – С. 220–225. 6. Тарасенко В.П. Швидкісне адаптивне порівняння / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко // Сучасні комп'ютерні системи та мережі: розробка та використання // Матер. 4-ї Міжнар. наук.-техн. конф. ACSN-2009. – Львів: НВФ "Українські технології", 2009. – С. 253–255. 7. Пат. № 61653 G06F 7/38. Пристрій для детального адаптивного порівняння символьних послідовностей / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко НТУУ КПІ; заявл. 29.22.2010; опубл. 25.07.2011 бюл. № 14.