УДК 004.3

Я.М. КЛЯТЧЕНКО, В.П. ТАРАСЕНКО, О.К. ТЕСЛЕНКО, А.Ю. МИХАЙЛЮК

Національний технічний університет України «КПІ», Україна

КОМАНДИ СПЕЦІАЛІЗОВАНОГО ПРОЦЕСОРА НА ПЛІС ДЛЯ АДАПТИВНОГО ПОРІВНЯННЯ ІНФОРМАЦІЙНИХ ОБ'ЄКТІВ

Розглянуто підходи для створення спеціалізованих команд для soft процесора на базі ПЛІС для реалізації алгоритму швидкісного адаптивного порівняння. Оптимізація алгоритму порівняння інформаційних об'єктів за рахунок введення спеціалізованих інструкцій дозволяє досягти показників швидкодії обробки символьних послідовностей, що перевищують показники програмної моделі алгоритму на базі стандартних інструкцій. Це створює умови для реалізації низки гібридних (програмноапаратних) реалізацій інформаційноаналітичних систем, призначених для оперативної і надоперативної інтелектуальної обробки великих масивів даних різноманітного типу (електронний текст, числові дані, графіка, мультимедіа тощо).

Ключові слова: *ПЛИС, адаптивне порівняння.*

Вступ

Характерною рисою сучасної, постіндустріальної, стадії розвитку суспільства є надзвичайно гостра залежність практично всіх сфер людської діяльності від високої ефективності взаємодії з глобальним електронним інформаційним ресурсом. У зв'язку з цим до розряду найбільш пріоритетних сегментів IT-ринку протягом кількох останніх років долучилися так звані інформаційно-аналітичні системи, котрі в залежності від конкретного напрямку застосування набирають форми інтелектуальних інформаційно-пошукових систем, інформаційно-моніторингових систем, BI-систем (Business Intelligence) тощо [1]. Очевидно, внаслідок різної спеціалізації згаданих систем можна відзначити суттєві відмінності у їх функціональності. Однак, сьогодні вимальовується деякий перелік функціональних сервісів, який можна вважати мінімально необхідним для комп'ютерних засобів даного класу. В загальному випадку до цього переліку можна віднести пошук даних з наступним упорядкуванням пошукового відгуку, кластеризацію і класифікацію інформаційних об'єктів за різноманітними ознаками, ліквідацію дублювання даних та інформаційної надлишковості взагалі, визначення міри подібності інформаційних об'єктів за різними ознаками і т.і. [2, 3]. При цьому слід відзначити, що для значної частини згаданих вище функцій однією з найважливіших базових операцій може виступати операція порівняння. В деяких же випадках власне порівняння є ключовим і самодостатнім сервісом програмного інструменту, як це має місце, наприклад, у системах автоматизованого виявлення та аналізу рівня запозичень інформації (у тому числі плагіату) для науково-освітньої галузі, бізнесу, масмедіа, політтехнологій, сфери безпеки тощо. Таким чином, очевидно, від ефективності засобів порівняння інформаційних об'єктів на початковому рівні (рівні символів) значною мірою залежить інтегральна ефективність відповідної інформаційно-аналітичної системи в цілому. З огляду на зазначене, підвищення швидкодії засобів порівняння інформаційних об'єктів є сьогодні надзвичайно актуальною науково-технічною задачею, деякі підходи до розв'язання котрої розглядаються у даній роботі.

Постановка задачі

У попередній роботі [4] розглянуто концепцію та виконано оцінку з точки зору апаратних витрат та швидкодії реалізації алгоритму адаптивного порівняння двох символьних послідовностей. Алгоритм швидкісного адаптивного порівняння [5], дозволяє встановити верхню границю кількісної величини запозичення, яка гарантовано буде більше ніж точна величина запозичення, що отримана за допомогою алгоритму детального адаптивного порівняння.

Апаратна реалізація цих алгоритмів на базі сучасних ПЛІС може дозволити отримати спеціалізований пристрій, що значно прискорить цей процес та зможе використовуватись разом із різними за потужністю комп'ютерами.

На сьогоднішній кожний із виробників ПЛІС пропонує готові мікропроцесорні soft ядра RISC архітектури, які мають свою систему команд та використовують спеціалізовані IP компоненти для

конструювання вбудованих систем. Так, наприклад, для компанії Xilinx® таким є ядро MicroBlaze™. Слід зауважити, що можливе використання тільки soft ядра Microblaze та його системи команд не буде відрізнятися від реалізації цього алгоритму на базі системи аналогічних команд будь-якого процесора (наприклад, Intel®). Але, зазвичай, спеціальні інструкції набагато краще враховують особливості алгоритмів для вирішення спеціалізованих задач. Також фірмові засоби проектування та оптимізації логіки ПЛІС можуть дозволити оптимізувати ці команди, а гнучкість архітектури FPGA – імплементувати в жорстку логіку. Наше завдання полягає в тому щоб створити спеціалізовані команди soft ядра для реалізації на ПЛІС, які б у найбільшій мірі реалізовували особливості алгоритму швидкісного адаптивного порівняння не тільки символьних, а й будь-яких послідовностей, якщо вважати, що для представлення кожного елемента цих послідовностей достатньо 1 байт. Так, розширивши набір інструкцій MicroBlaze, що є можливим, для того щоб вони були застосовними якнайкраще для цього алгоритму, ми значно оптимізуємо його по швидкодії.

Розглянемо ці команди в тій послідовності в якій вони можуть використовуватися алгоритмом швидкісного адаптивного порівняння [5].

Метод розв'язання задачі

Спочатку викладемо суть алгоритму швидкісного адаптивного порівняння для того, щоб спростити подальше пояснення особливостей команд. Нехай A – послідовність символів (послідовність - зразок) довжиною в n_a символів (наприклад – байт), B – послідовність символів, яка перевіряється, довжиною n_b, c ($1 \le c \le d$) – довжини підпослідовностей, які порівнюються, d – мінімальна довжина збігу двох підпослідовностей символів, яка береться до уваги. Значення d визначається (адаптується) у залежності від конкретних умов порівняння. В загальному випадку із послідовності B вибираються всі підпослідовності довжиною d та визначається їх входження в послідовність A. В разі такого входження всі відповідні символи послідовності B відмічаються.

Із послідовності A із кроком x, а з послідовності B із кроком y вибираються підпослідовності довжиною c. Позначаються далі – $\mathbf{S_j}$ (j=1,2,…, r) c-підпослідовності із A, $r=n_a$ div x, $\mathbf{F_h}$ (h=1,2,…,t) підпослідовності довжиною c символів, які вибираються із послідовності B з кроком y, $t = n_b$ div y. Будемо вважати, що індекси j підпослідовностей $\mathbf{S_j}$ впорядковані шляхом послідовного вибору цих підпослідовностей з A, тобто при будь-яких a та b, таких що, $a < b$, підпослідовність $\mathbf{S_a}$ розташована до початку послідовності A ближче, чим $\mathbf{S_b}$. Будемо вважати, що індекси h підпослідовностей $\mathbf{F_h}$ впорядковані аналогічно. Розглянемо дві довільні сусідні підпослідовності (рис.1) довжиною c символів. Якщо початок деякої підпослідовністі довжиною d символів (на рис.1 відмічено пунктирною лінією) співпадає з початком підпослідовності довжиною c символів, то така d-підпослідовність безумовно, містить відповідну c-підпослідовність. Наступна d-підпослідовність (на рисунку виділена товстішою лінією) вже не буде містити 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 містить цю наступну c-підпослідовність, включаючи випадок, аналогічний показаному на рисунку пунктирною лінією. Тоді, якщо значення x буде задовольнять умові $1 \le x < d - c+1$, будь-яка підпослідовність з A із довжиною d (мінімальною довжиною збігу) буде повністю містити одну з вибраних підпослідовностей з A довжиною c.

Розглянемо випадок , коли послідовності A та B мають спільну підпослідовність, довжиною не менше чим d символів. Згідно з вищевикладеним серед c-підпослідовностей $\mathbf{S_j}$ (j=1,2,…, r) знайдеться принаймні одна (позначимо її $\mathbf{S_a}$), яка повністю міститься в цій d-підпослідовності. Якщо y=1 , а d – підпослідовність спільна, то незалежно від її розташування в послідовності B, серед підпослідовностей $\mathbf{F_h}$ (h=1,2,…,t) буде існувати і підпослідовність, яка співпадає з $\mathbf{S_a}$.

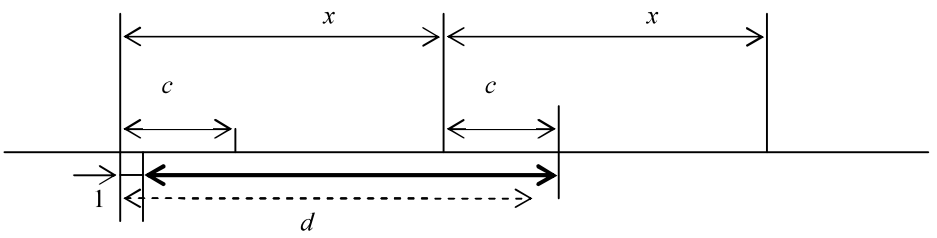

Рис. 1. Приклад розташування послідовностей

При швидкісному адаптивному порівнянні кожна підпослідовність $\mathbf{S_j}$ порівнюється з кожною підпослідовністю $\mathbf{F_h}$, і у випадку, хоча б одного співпадання відповідна підпослідовність $\mathbf{S_j}$ відмічається. Якщо в результаті для деякого a (a∈{1,2,...,r-1}) $\mathbf{S_a}$ та $\mathbf{S_{a+1}}$ не відмічені, то це означає, що всі символи між цими підпослідовностями, а також символи самих підпослідовностей (всього x+c символів), при умові, що y=1, не можуть бути запозиченими в A при заданій мінімальній довжині запозичення. Тобто, при y=1, у результаті порівняння кожної c-підпослідовності $\mathbf{S_j}$ із кожною c-підпослідовністю $\mathbf{F_h}$, будуть визначені всі спільні підпослідовності довжиною не менше d символів.

Для забезпечення ефективності реалізації методики швидкісного адаптивного порівняння розглянемо наступні команди:

Simple_Scan

Для цієї команди можуть використовуватися стандартні 32-розрядні регістри загального призначення мікропроцесороного ядра MicroBlaze, які мають номери r1-r13 та r18-r31. Звернемо увагу, на те, що розрядність регістрів та самого ядра впливає на довжину підпослідосностей, які ми порівнюємо, тобто c=4. Дамо позначення регістрам, що мають використовуватися цією командою та опишемо алгоритм роботи самої команди:

-rD – регістр який пропонується використати для зберігання елементів $\mathbf{S_j}$ (j=1,2,…, r) кожний із яких є підпослідовністю довжиною 4 байти (символи) послідовності A;

-rS – регістр (використовуються в якості регістра зсуву) для зберігання елементів поточної послідовності $\mathbf{F_h}$ (h=1,2,…,t), кожний елемент якої, як і у випадку послідовності $\mathbf{S_j}$ є підпослідовністю довжиною 4 байти (символи) послідовності A. Наприклад, початкове значення ($\mathbf{F_1}$) дорівнює першим 4 символам послідовності B;

-rC – регістр для зберігання поточних даних лічильника кількості символів для порівняння. Також зауважимо, що початкове значення rC повинно бути не менше ніж довжина послідовності B;

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

Псевдокод 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
```

Коментар 1. Команда *Simple_Scan* використовує лише стандартні блоки soft процесора Micro-Blaze і потребує попереднього завантаження чотирьох регістрів, що може бути виконано стандартними інструкціями мікропроцесорного ядра Micro-Blaze.

Коментар 2. Команда *Simple_Scan* подібна до команди Scan процесорів Intel. Принципова відмінність полягає в тому, що при порівнянні багатобайтних даних адреса змінюється лише на 1.

Double_Scan

За 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 $\mathbf{S_{j+1}}$ (j=1,2,…, r-1).

-rS2 – регістр зсуву, що вміщає 4 байта для зберігання поточного $\mathbf{F_{h+1}}$ 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
```

Коментар 3. Команда одночасно виконує сканування двох сусідніх елементів (підпослідовностей) $\boldsymbol{S_j}$ (j=1,2,…, r), забезпечуючи більш точне визначення граничного значення оригінальності при швидкісному адаптивному порівнянні.

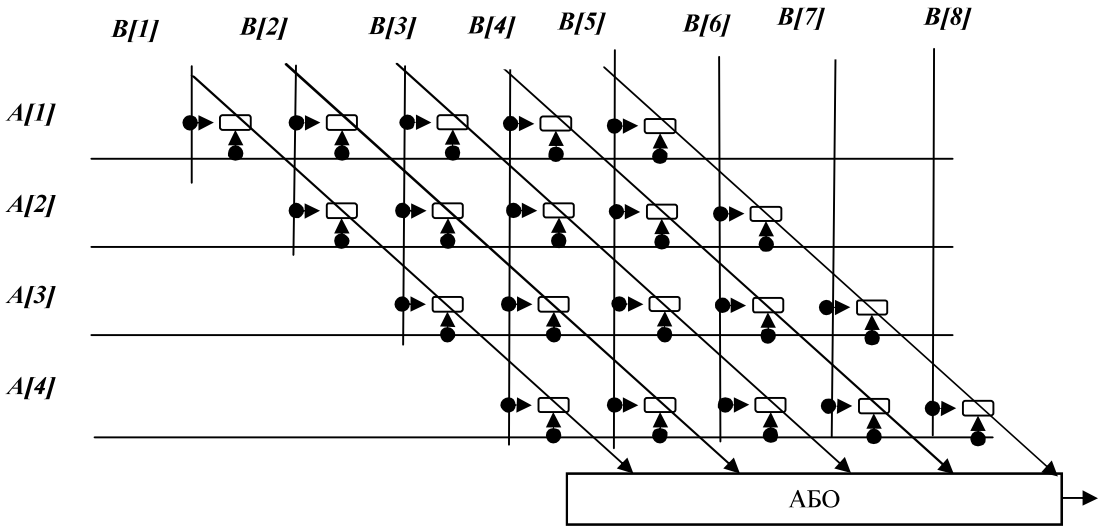

Рис. 2. Приклад матричної схеми порівняння

Команда використовує лише стандартні компоненти soft процесора MicroBlaze. Шляхом ретельного проектування та оптимізацій, час на виконання циклу порівняння команди *Double_Scan* можна наблизити до відповідного часу для команди *Simple_Scan*.

Fast_Scan

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

Реалізація команди ґрунтується на принципі побудови матричної схеми порівняння підпослідовності з A, довжиною c символів із підпослідовністю B довжиною c* символів. Приклад такої схеми при c=4 та c*=8 подано на рис.2, де елементарні побайтові порівняння в діагоналі об'єднуються по схемі «І» , а виходи діагоналей – по схемі «АБО».

Коментар 4. Використання команди *Fast_Scan* дозволить використовувати значення $y > 1$.

Як видно із цього прикладу, створення параметричного пристрою для реалізації команди в ПЛІС не є складним та є дуже корисним із точки зору ефективності. 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. В даному випадку під багаторозрядним регістром мається на увазі або набір стандартних регістрів загального призначення мікропроцесорного ядра «зчеплених» послідовно, або регістр, що реалізований на базі тригерів ПЛІС.

Висновки

Оптимізація алгоритму порівняння інформаційних об'єктів за рахунок введення спеціалізованих інструкцій дозволяє досягти показників швидкодії обробки символьних послідовностей, що перевищують показники програмної моделі алгоритму на базі стандартних інструкцій. Це створює умови для реалізації низки гібридних (програмно-апаратних) реалізацій інформаційно-аналітичних систем, призначених для оперативної і надоперативної інтелектуальної обробки великих масивів даних різноманітного типу (електронний текст, числові дані, графіка, мультимедіа тощо).

Подальші дослідження можуть бути спрямовані зокрема на визначення виграшу в швидкодії, наприклад шляхом порівняння часу обробки (в тактах) алгоритмом тестових послідовностей за допомогою стандартних інструкцій та спеціалізованих.

Література

1. Черников А. Авансы революции / А. Черников // Компьютерное обозрение. – 2007. – №12. – С. 22-25.

2. Функціональність спеціалізованих інформаційно-аналітичних систем для підтримки інформаційно-навчальної діяльності/ В.П. Тарасенко, А.Ю. Михайлюк, М.В. Сніжко, Л.М. Бігун // Проблеми інформатизації та управління: зб. наук. праць. – К.: НАУ, 2009. – № 3 (27). – С.123-130.

3. Функціональні профілі спеціалізованих інформаційно-аналітичних систем / О.С. Кебкало, А.Ю. Михайлюк, В.П. Тарасенко // Науковий вісник Чернівецького університету: Збірник наук. праць. Вип. 423: Фізика. Електроніка.: Тематичний випуск «Комп'ютерні системи та компоненти». Ч. І. – Чернівці: ЧНУ, 2009. – С. 117-123.

4. 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. – С. 23-29.

5. Тарасенко В.П. Швидкісне адаптивне порівняння / В.П. Тарасенко, О.К. Тесленко, Я.М. Клятченко // Сучасні комп'ютерні системи та мережі: розробка та використання. Матеріали 4-ї Міжнародної науково-технічної конференції ACSN-2009. – Львів: НВФ "Українські технології", 2009. – С. 253-255.

Надійшла в редакцію 9.02.2010

Рецензент: д-р техн. наук, проф., проф. кафедри Є.Т.Володарський, Національний технічний університет України «Київський політехнічний інститут», Україна.

КОМАНДЫ СПЕЦИАЛИЗИРОВАННОГО ПРОЦЕССОРА НА ПЛИС ДЛЯ АДАПТИВНОГО СРАВНЕНИЯ ИНФОРМАЦИОННЫХ ОБЪЕКТОВ

Я.М. Клятченко, В.П. Тарасенко, А.К. Тесленко, А.Ю. Михайлюк

Рассмотрены подходы для создания специализированных команд для soft процессора на базе ПЛИС для реализации алгоритма скоростного адаптивного сравнения. Оптимизация алгоритма сравнения информационных объектов за счет введения специализированных инструкций позволяет достичь показателей быстродействия обработки символьных последовательностей, превышающие показатели программной модели алгоритма на базе стандартных инструкций. Это создает условия для реализации ряда гибридных (программно-аппаратных) реализаций информационно-аналитические систем, предназначенных для оперативной и сверхоперативной интеллектуальной обработки больших массивов данных различного типа (электронный текст, числовые данные, графика, мультимедиа и т.д.).

Ключевые слова: ПЛИС, адаптивное сравнение.

INSTRUCTIONS IMPLEMENTATION OF SPECIALIZED PLD-BASED PROCESSOR FOR ADAPTIVE COMPARISON OF INFORMATION OBJECTS

Y.M. Klyatchenko, V.P. Tarasenko, O.K. Teslenko, A.Y. Mykhailyuk

The approaches are considered for PLD-based soft processor specialized instruction creating for the implementation an algorithm of high-speed adaptive comparison. Optimization in comparisons between algorithms of information objects by introducing specialized instructions can achieve better performance speed in the process of character sequences, exceeding the performance of a software model of the algorithm that is based on standard instructions. This creates conditions for a number of hybrid (software and hardware) implementations of information-analytical systems to operate and scratch-pad intelligent process of large data sets of different types (electronic text, numeric data, graphics, multimedia, etc.).

Keywords: FPGA, adaptive comparison.

Клятченко Ярослав Михайлович – асистент кафедри спеціалізованих комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут», Україна, e-mail: k_yaroslav@scs.ntu-kpi.kiev.ua.

Тарасенко Володимир Петрович – д-р техн. наук, проф., зав. кафедри спеціалізованих комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут»e-mail: vtarasen@scs.ntu-kpi.kiev.ua.

Тесленко Олександр Кирилович – канд. техн. наук, доц. кафедри спеціалізованих комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут», e-mail: teslenko@scs.ntu-kpi.kiev.ua.

Михайлюк Антон Юрійович – канд. техн. наук, ст. наук. співр., завідувач НДЛ прикладної інформатики, Київський університет ім. Бориса Грінченка, Україна.