УДК 519.681.5

М.С. ЯДЖАК, М.И. ТЮТЮННИК

ОПТИМАЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ ЦИФРОВОЙ ФИЛЬТРАЦИИ С ИСПОЛЬЗОВАНИЕМ АДАПТИВНОГО СГЛАЖИВАНИЯ

Ключевые слова: *оптимальный параллельно-конвейерный алгоритм, задача цифровой фильтрации, адаптивное сглаживание, квазисистолическая структура, структурно-процедурная организация вычислений.*

ВВЕДЕНИЕ

Задачи цифровой фильтрации (ЗЦФ) в различных формулировках рассматривались многими авторами [1–4]. Для их решения были предложены численные алгоритмы, ориентированные на разные типы архитектур вычислительных систем [5–7]. В большинстве случаев ЗЦФ необходимо решать в режиме реального времени при условии ограничения объема вычислительных средств, поэтому в данной ситуации отдается предпочтение разработке эффективных параллельных алгоритмов, ориентированных на реализацию на специализированных вычислительных системах. В работе [4] для решения одномерной ЗЦФ предложен оптимальный по быстродействию и использованию памяти параллельно-конвейерный алгоритм (ПКА), ориентированный на реализацию на квазисистолической вычислительной структуре. В этом случае задача цифровой фильтрации — это выполнение некоторого числа пересчетов сглаживания для массива значений переменных через плавающее окно заданного размера. Затем был предложен квазисистолический метод построения оптимальных по быстродействию алгоритмов решения задач фильтрации произвольной размерности [8]. Следует отметить, что оптимальность была доказана в классе алгоритмов, эквивалентных по информационному графу с точностью до выполнения соотношений ассоциативности и коммутативности для операции сложения. При этом полученные результаты верны и в случае, когда вместо операций сложения (+) и умножения (*) рассматриваются любые другие две операции, например, логические — $\vee$ и $\wedge$, теоретико-множественные — $\cup$ и $\cap$, а также отыскание максимального (max) и минимального (min) значений. В данном случае квазисистолическая [9] обработка отличается от чисто систолической [10] тем, что при ней допускается передача данных из одной инстанции сразу в несколько точек приема. Технически это легко и эффективно реализуется с помощью оптоэлектронных элементов [11].

В настоящей работе рассматривается задача фильтрации, согласно которой пересчет значения каждой переменной осуществляется через окно размера, определенного для этой переменной, т.е. фактически используется процедура адаптивного сглаживания [12]. Для многократного применения этой процедуры к массиву значений некоторого входного сигнала предложен оптимальный по быстродействию ПКА.

ФОРМУЛИРОВКА ПРОБЛЕМЫ

Рассматриваемая ЗЦФ заключается в выполнении C пересчетов сглаживания массива значений переменных x_i $(i=\overline{1, n})$, причем каждое x_j пересчитывается

через плавающее окно размера M_j. При этом пересчет значений осуществляется по формуле

$$x_i = \sum_{s=-m_i}^{m_i} x_{i+s} \cdot f_s^i; \quad i = \overline{1, n}. \tag{1}$$

Здесь $M_i = 2m_i + 1$, а весовые коэффициенты f_s^i ($s = \overline{-m_i, m_i}$; $i = \overline{1, n}$) и «заграничные» значения $x_{m_0}, x_{m_0+1}, \ldots, x_0$; $x_{n+1}, x_{n+2}, \ldots, x_{m^0}$ — известные константы, к тому же $m_0 = \min\limits_{1 \le i \le n} \{i - m_i\}$, $m^0 = \max\limits_{1 \le i \le n} \{i + m_i\}$.

Последовательный алгоритм решения этой задачи фильтрации имеет вид

```
FOR  t = 1, C  DO
  FOR  i = 1, n  DO
    p = 0
    FOR  s = −m_i , m_i  DO
      p = p + x_{i+s} * f_s^i ;
    x_i = p.
```
(2)

В приведенной конструкции точка с запятой указывает на конец тела цикла по переменной *s*.

Согласно алгоритму (2) для пересчета значений переменной x_i на *t*-м шаге используются $x_{i-m_i}, x_{i-m_i+1}, \ldots, x_{i-1}$, пересчитанные уже на этом шаге. Граф информационной зависимости между итерациями цикла, определенного (2) для $C = 3, n = 7, m_1 = m_3 = m_6 = m_7 = 1, m_2 = m_5 = 2, m_4 = 3,$ изображен на рис. 1.

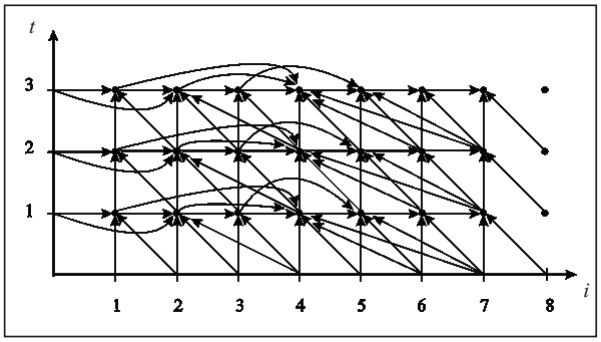

Рис. 1

Данная публикация посвящена разработке оптимального по быстродействию ПКА численного решения сформулированной ЗЦФ с ориентацией на для эффективную реализацию на вычислительных системах специального назначения.

Будем считать, что время выполнения операций сложения и умножения одинаково и равно одному такту, а единственным условием, требуемым для операций, является выполнение законов ассоциативности и коммутативности для операции сложения.

ПОСТРОЕНИЕ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА

Используя метод гиперплоскостей Лемпорта [13] для распараллеливания циклов, получаем параллельную версию алгоритма (2):

```
FOR  ν = 1, (1 + m_max)(C − 1) + n  DO
  FOR  ALL  (i, t) ∈ {(i, t): t = (ν − i)/(1 + m_max) +
                + 1 ∧ 1 ≤ i ≤ n ∧ 1 ≤ t ≤ C}  DO  CONC
    p_i = 0
    FOR  s = −m_i , m_i  DO
      p_i = p_i + x_{i+s} * f_s^i ;
    x_i = p_i .
```
(3)

В приведенной конструкции $m_{\max} = \max\limits_{1 \le i \le n} \{m_i\}$, а тип параллельности *CONC* [14] определяет независимое (параллельное) выполнение итераций, лежащих в одной гиперплоскости. Здесь общее количество гиперплоскостей составляет $(1+m_{\max})(C-1)+n$. При этом граф информационной зависимости алгоритма (3) полностью совпадает с соответствующим графом последовательного алгоритма решения задачи. В данном случае распараллеливание осуществляется на уровне итераций двойного цикла (переменные t, i) алгоритма (2), а каждая такая итерация — это один пересчет значения некоторой переменной.

Предложенный в [8] квазисистолический метод организации вычислений позволяет повышать степень параллелизма за счет конвейеризации, обеспечивая при этом функциональную эквивалентность с исходным (последовательным) алгоритмом. На основании этого метода построим параллельно-конвейерную версию алгоритма (2). Сначала рассмотрим случай, когда весовые коэффициенты, используемые для пересчета значений i-й переменной, равны между собой, т.е.

$$f^i_{-m_i} = f^i_{1-m_i} = \ldots = f^i_0 = f^i_1 = \ldots = f^i_{m_i} = f^i. \qquad (4)$$

Отметим, что (4) имеет место для всех $i = \overline{1, n}$.

Для решения сформулированной ЗЦФ с учетом (4) предложен ПКА, определяемый следующей конструкцией:

```
FOR  i = 1, n  DO  SYNCH
    DELAY (2(m_{i^0} − m_i + i − i^0))
        FOR  j = 1, C  DO
BEGIN
        FOR  s = 1, 2m_i  DO                                   (5)
        x_i = x_i + IF ((s/2) = [s/2]) THEN x_{i+s/2−m_i−1}
            ELSE x_{i+(s+1)/2}
        x_i = x_i * f^i;
        DELAY (2(m_max − m_i + 1))
END.
```

Здесь *SYNCH* — тип параллельности [14], обозначающий синхронизацию сдвинутых между собой n параллельных ветвей алгоритма после выполнения в них каждой операции; *DELAY* (p) — оператор задержки на p тактов; i^0 — номер переменной, для которой имеет место равенство $2m_{i^0} + 1 + 2n - 2i^0 = \max\limits_{1 \le i \le n} \{2m_i + 1 + 2n - 2i\}$, а $[\cdot]$ здесь и далее обозначает целую часть.

Отметим, что алгоритм (5) и соответствующий ПКА из [4] имеют одинаковую структуру. Фрагмент потактовой схемы предложенного ПКА для указанных в предыдущем разделе значений параметров задачи фильтрации C, n, m_i $(i = \overline{1, n})$ приведен на рис. 2. Чтобы не загромождать рисунок, передача данных стрелками изображена лишь для ветви, в которой перевычисляются значения переменной x_4. Задержки между перевычислениями значений переменных на каждом такте обозначены маленькой окружностью.

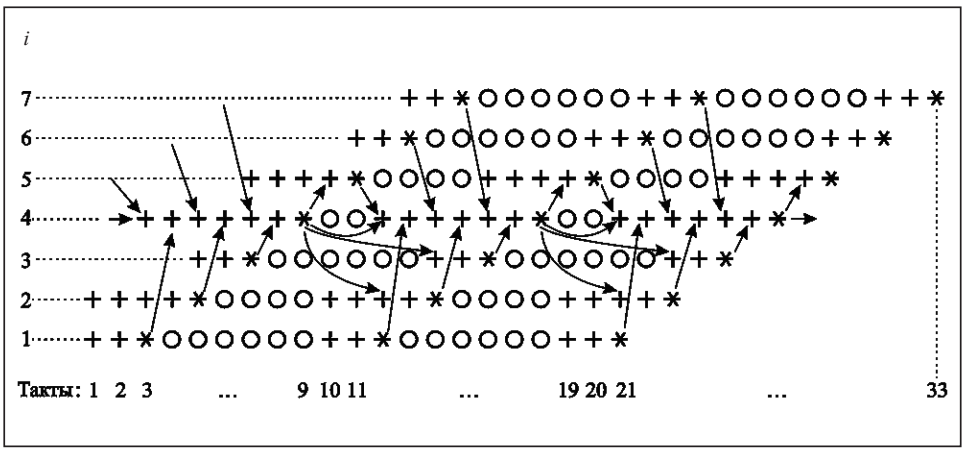

Рис. 2

В случае, когда весовые коэффициенты являются разными, ПКА (5), как и соответствующий алгоритм из [4], несколько изменится, поскольку в каждой из параллельных ветвей одновременно с суммированием происходит подготовка одного из последующих слагаемых, т.е. умножение на соответствующий весовой коэффициент. При этом для сохранения суммируемых величин для каждой пары ветвей необходима хотя бы одна дополнительная переменная. Соответствующая конструкция, определяющая ПКА решения ЗЦФ в данном случае, выглядит следующим образом:

$$
\begin{aligned}
&FOR \quad i=1,\ n \quad DO \quad SYNCH\\
&\quad DELAY\ (2(m_{i^0}-m_i+i-i^0))\\
&\quad\quad FOR \quad j\ =\ 1,\ C \quad DO\\
&BEGIN\\
&\quad\quad FOR \quad s=\ 1, M_i \quad DO \qquad\qquad (6)\\
&\quad BEGIN\\
&\quad\quad\quad x_i = IF\ (s=1)\ THEN\ \ f_0^i * x_i\ \ ELSE\ \ x_i + z_i\ ,\\
&\quad\quad\quad z_i = f^i_{h(s,\,m_i)} * x_{i+h(s,\,m_i)}\\
&\quad END;\\
&\quad\quad\quad DELAY\ (2(m_{\max}-m_i+1))\\
&END.
\end{aligned}
$$

Предполагается, что операторы $x_i = \ldots$ и $z_i = \ldots$, разделенные запятой, выполняются синхронно. Целочисленная функция $h(s, m_i)$ — это следующий условный оператор:

$$IF\ ((s/2)=[s/2])\quad THEN\quad s/2-m_i-1\quad ELSE\quad (s+1)/2\,.$$

Фрагмент потактовой схемы ПКА (6) для использованых значений параметров ЗЦФ C, n, m_i $(i=\overline{1,\,n})$ изображен на рис. 3. Стрелками показана передача данных для пары ветвей, в которой перевычисляются значения переменной x_4.

Алгоритм (6) по сравнению с (5) задает вычисления в виде n пар параллельных ветвей, которые синхронизуются после выполнения каждой операции. Из потактовых схем (см. рис. 2, 3) следует, что время работы этих алгоритмов одинаковое.

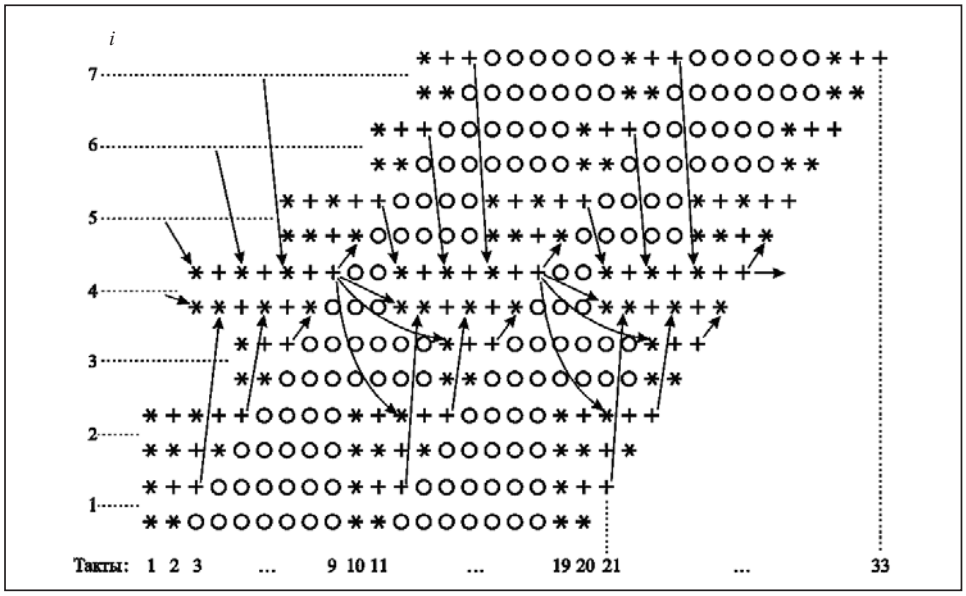

Рис. 3

ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА

Оценим время выполнения алгоритмов (2), (3), (6) с одинаковым информационным графом. Для выполнения последовательного алгоритма (2) требуется

$$2C\left(n+2\sum_{i=1}^{n} m_i\right) \tag{7}$$

тактов. При использовании параллельного алгоритма (3) количество необходимых тактов несколько уменьшится:

$$2\left(n+(1+m_{\max})(C-1)+2\sum_{\nu=1}^{(1+m_{\max})(C-1)+n} m^{\nu}\right), \tag{8}$$

где $(2m^{\nu}+1)$ — максимальный размер окна для переменных, пересчитываемых на итерациях, лежащих в ν-й гиперплоскости. Для реализации ПКА (6) (как и ПКА (5)) требуется

$$2(n+(1+m_{\max})(C-1))+2m_{i_0}+C-2i^0 \tag{9}$$

тактов.

Легко установить, что если $m_i \ll n \ \ \forall\, i\colon\ i=\overline{1,n}$, то величина, вычисляемая по формуле (9), намного меньше величины, которая вычисляется по (8).

Для рассмотренных выше значений параметров ЗЦФ C, n, m_i $(i=\overline{1,n})$ (см. рис. 1, 3) оценки (7)–(9) принимают соответственно такие значения: 174, 138, 33. Следовательно, в данном случае получаем ускорение ПКА (6) по сравнению с (2) приблизительно в 5.27 раза.

Рассмотрим Ω — класс алгоритмов, эквивалентных алгоритму (2) с точностью до выполнения законов ассоциативности и коммутативности и таких, которые используют переменные из набора $x_{m_0}, x_{m_0+1}, \ldots, x_0, x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{m^0}; z_1, z_2, \ldots, z_n$. Имеет место следующее утверждение.

Теорема. Если среди x_i $(i=\overline{1,n})$ существует переменная с номером i_0, для которой $m_{i_0}=m_{\max}$ и $m_{i_0}\le n-i_0$, то алгоритм (6) принадлежит классу Ω и имеет минимальное время выполнения среди всех алгоритмов этого класса.

Для доказательства этой теоремы необходимо сформулировать и доказать некоторые дополнительные утверждения.

Лемма 1. Если на некотором такте работы алгоритма (6) для некоторого i в i-й паре ветвей осуществляются присваивания $x_i=x_i+(*)\ldots$ и $z_i=f_h^i*\ldots$, то:

а) в правых частях этих присваиваний, соответственно кроме x_i и f_h^i, присутствует только один аргумент;

б) в любой другой i'-й паре ветвей отсутствует присваивание $z_{i'}=f_h^{i'}*\ldots$, в котором x_i было бы аргументом.

Доказательство этой леммы осуществляется по схеме, аналогичной доказательству соответствующих лемм об отсутствии конкуренции над памятью для алгоритмов, исследуемых в [4, 15]. При этом выполнение п. а) обусловлено однозначностью выбора второго аргумента с помощью функции *IF … THEN … ELSE* и синхронным выполнением операторов $x_i=\ldots$ и $z_i=\ldots$

Доказательство п. б) можно провести прямым подсчетом тактов считывания и записи. Поскольку результат r-го пересчета значения переменной x_i генерируется на такте с номером $w=(2m_{\max}+3)(r-1)+2(m_{i^0}+i-i^0)+1$, то легко установить, что считывание этого значения как аргумента осуществляется для всех j' таких, что $j'-m_{j'}=i$, в парах ветвей $i+m_{j'}$ на следующем же такте, а для всех j таких, что $j+m_j=i$, — в парах ветвей $i-m_j$ на такте с номером $w+2(m_{\max}-m_j)+1=(2m_{\max}+3)r+2(m_{i^0}+i-i^0)-2m_j-1$.

Таким образом, лемма 1 говорит об отсутствии конкуренции над памятью при выполнении алгоритма (6).

Для дальнейшего изложения введем некоторые обозначения. Пусть x_i^k и $\widetilde{x}_i^k$ — значения x_i, получаемые как результаты k-го перевычисления при применении соответственно алгоритмов (2) и (6); t_i^k и τ_i^k — терм-истории (деревья) вычисления соответственно x_i^k и $\widetilde{x}_i^k$; $h(t)$ — высота дерева t. Вводя эти обозначения, предполагаем выполнение условий сформулированной выше теоремы о существовании переменной с номером i_0.

Из потактовой схемы алгоритма (6) следует, что $h(\tau_n^k)$ — это фактически высота дерева выполнения всех вычислений, связанных с k-м пересчетом сглаживания массива значений переменных x_i $(i=\overline{1,n})$.

Лемма 2. Для любых i,k справедливо следующее:

а) τ_i^k эквивалентно t_i^k с точностью до применения законов ассоциативности и коммутативности;

б) $h(\tau_i^k)=(2m_{\max}+3)k+2i-2(m_{\max}-m_{i^0}+i^0+1)$.

Доказательство этой леммы проводится аналогично доказательству соответствующих пунктов лемм из [4, 16]. На основании леммы 1 легко построить рекуррентную схему формирования дерева τ_i^k. При этом нужно учесть, что результат деревьев $\tau_{i+1}^{k-1}, \tau_{i+2}^{k-1}, \ldots, \tau_{i+m_i}^{k-1}$ используется в τ_i^k не на следующем такте, а на $2(m_{\max}-m_i)+1$ такте после вычисления. Из этой рекуррентной схемы следует,

что t_i^k и τ_i^k с точностью до порядка суммирования вычисляют одно и то же выражение. Это и доказывает п. а) леммы 2.

Пункт б) доказывается одновременной индукцией по i, k, как это было сделано в [4, 16].

Лемма 3. Для любого k высота $h(\tau_n^k)$ минимальна в классе всех t, эквивалентных τ_n^k и использующих только переменные из набора: $x_{m_0}, x_{m_0+1}, \ldots, x_0, x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{m^0}$; $z_1, z_2, \ldots, z_n$.

Доказательство этого утверждения следует из того, что, учитывая условия сформулированной теоремы, существует хотя бы один критический путь, определяющий максимальную высоту дерева τ_n^k, и по которому осуществляется чередование операций $*$ и +, причем между двумя соседними операциями $*$ находится не больше двух операций +. Следовательно, уменьшить высоту дерева τ_n^k невозможно. Исключением является первый пересчет сглаживания, где суммирование можно было бы осуществить по схеме, близкой к двоичному дереву, но для этого необходимы дополнительные переменные для запоминания промежуточных результатов, что противоречит предположению леммы.

Таким образом, лемма 3 доказана.

Доказательство теоремы. То, что алгоритм (6) принадлежит классу Ω, следует из п. а) леммы 2 и из описания этого алгоритма. Покажем, что время выполнения его минимально среди алгоритмов рассматриваемого класса. Для этого подсчитаем, на каком такте вычисляется результирующее значение x_i^k. Согласно алгоритму (6) сначала пропускается $2(m_{i^0} - m_i + i - i^0)$ тактов задержки. Затем выполняется k итераций, каждая из которых содержит $2m_i + 1$ такт суммирования–умножения и $2(m_{\max} - m_i + 1)$ тактов задержки, при этом на последней итерации тактов задержки не требуется. Таким образом, получаем

$$2(m_{i^0} - m_i + i - i^0) + (k-1)(2m_i + 1 + 2m_{\max} - 2m_i + 2) + 2m_i + 1 =$$

$$= (2m_{\max} + 3)(k-1) + 2(m_{i^0} + i - i^0) + 1$$

тактов, что при $i = n$ в точности совпадает с $h(\tau_n^k)$. Согласно лемме 3 высота $h(\tau_n^k)$ минимальна среди всех эквивалентных деревьев над тем же множеством переменных.

Теорема доказана, поскольку $h(\tau_n^C)$ — высота дерева вычисления всего алгоритма (6).

СРЕДСТВА ВОЗМОЖНОЙ РЕАЛИЗАЦИИ ПРЕДЛОЖЕННЫХ ПКА

Построенные ПКА (5) и (6) ориентированы на реализацию на соответствующих специализированных вычислительных системах — квазисистолических структурах (КСС) [9], «копирующих» потактовые схемы этих алгоритмов и состоящих из элементарных функциональных элементов (ФЭ), выполняющих одну из операций: сложение, умножение, транспортировку данных. Следует отметить, что ФЭ, реализующие арифметические операции, могут осуществлять и передачу данных. Нетрудно заметить, что объем аппаратуры в этих КСС главным образом зависит от количества пересчитываемых значений переменных, числа необходимых пересчетов сглаживания, а также величины $m_{\max}$. При этом энергопотребление вы-

числительных систем в данном случае уменьшается в результате уменьшения числа ФЭ, реализующих арифметические операции. Для разработки архитектуры таких вычислительных средств используются те же подходы, что и в работах [9, 15], а их загрузка может быть значительно увеличена путем решения на них целых серий соответствующих задач фильтрации.

Для уменьшения объема аппаратуры ПКА (5), (6) предлагается реализовать на КСС, соответствующих потактовой схеме не всего алгоритма, а лишь одного пересчета сглаживания. Очевидно, что в этом случае необходимо предвидеть средства синхронизации при передаче результатов выполнения отдельных пересчетов. Пример такой двумерной КСС для реализации потактовой схемы ПКА (5) из рис. 2 приведен на рис. 4. Эта вычислительная структура состоит фактически из ФЭ четырех типов, выполняющих операции сложения, умножения и транспортировки данных (два типа элементов).

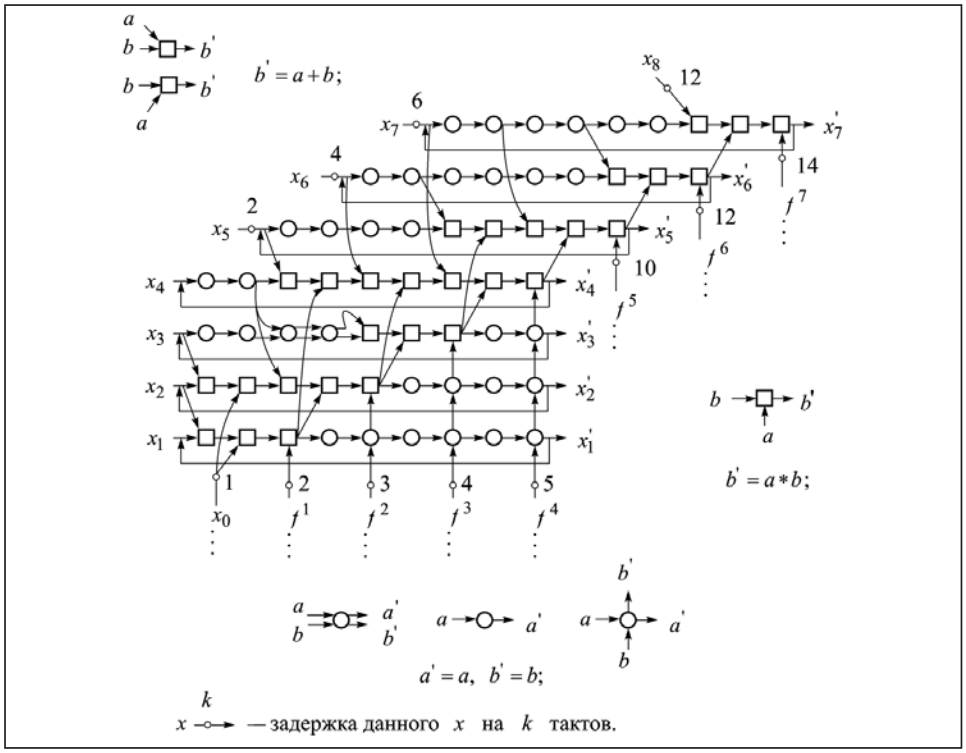

Рис. 4

Еще одним потенциальным средством реализации рассмотренных в работе ПКА могут быть параллельные вычислительные системы со структурно-процедурной организацией вычислений [17], состоящие из множества элементарных процессоров (ЭП), множества секторов памяти (СП) и коммутатора. Коммутатор из ЭП может собирать конвейер любой конфигурации или множество конвейеров. При этом через коммутатор каждый из процессоров соединяется с необходимым СП. Примеры конфигураций таких систем для выполнения алгоритмов с ограниченным параллелизмом решения одномерной задачи фильтрации более детально рассмотрены в [18]. Важной проблемой для вычислительных систем со структурно-процедурной организацией вычислений является проблема безконфликтного размещения данных в СП [19]. Она состоит в том, чтобы данные, которые окажутся в одном секторе, не нужно было обрабатывать одновременно. Так, рассмотренный выше пример потактовой схемы выполнения ПКА (5) (см. рис. 2)

можно реализовать на вычислительной системе такого типа, состоящей из семи конвейеров различной длины (один конвейер для пересчета значений одной переменной, а длина конвейера равна размеру соответствующего плавающего окна). Для хранения значений переменных в данном случае используем четыре СП:

$$\begin{aligned}
&\text{1-й СП:} \quad x_1,\ x_5;\\
&\text{2-й СП:} \quad x_3,\ x_7;\\
&\text{3-й СП:} \quad x_0,\ x_2,\ x_4,\ x_6,\ x_8;\\
&\text{4-й СП:} \quad f^1,\ f^2,\ f^3, \ldots, f^7.
\end{aligned}$$

С помощью универсального коммутатора [19] осуществляются необходимые связи между СП и ЭП. Между стартами вычислительных конвейеров, которые собирает универсальный коммутатор, выполняются задержки, определенные в алгоритме (5) для каждой параллельной ветви.

ЗАКЛЮЧЕНИЕ

В настоящей работе рассмотрена одномерная ЗЦФ, использующая процедуру адаптивного сглаживания. Для численного решения этой задачи предложен оптимальный по быстродействию ПКА. Оптимальность доказана в классе алгоритмов, эквивалентных по информационному графу с точностью до выполнения законов ассоциативности и коммутативности. Обсуждается возможность реализации предложенного алгоритма фильтрации на квазисистолических вычислительных структурах и параллельных системах со структурно-процедурной организацией вычислений.

Разработанные варианты параллельного алгоритма могут применяться как для высокоскоростной цифровой обработки сигналов, так и для исследования архитектуры специализированных вычислительных систем (КСС) и систем со структурно-процедурной организацией вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. Ковальчук Л.В. Способы повышения быстродействия в процессах медианной фильтрации // Упр. системы и машины. — 1998. — № 2. — С. 13–15.
2. Кучинський Т., Яцимірський М. Скорочений алгоритм швидкого біортогонального (9/7) хвильового перетворення // Вісн. ДУ «Львівська політехніка». Комп'ютерна інженерія та інформаційні технології. — 2000. — № 392. — С. 159–162.
3. Мозговой Д.К., Волошин В.И., Бушуев Е.И. Фильтрация радиометрических помех с пространственно-периодичной структурой // Проблемы управления и информатики. — 2004. — № 3. — С. 97–106.
4. Val'kovskii V.A. An optimal algorithm for solving the problem of digital filtering // Pattern recognition and image analysis. — 1994. — **4**, N 3. — P. 241–247.
5. Иванов В.Г. Параллельные и последовательные структуры Хаара для цифровой обработки сигналов // Электронное моделирование. — 2005. — **27**, № 3. — С. 55–66.
6. Иванов С.М., Тропченко А.Ю. Конвейерные разрядно-срезовые алгоритмы ранговой фильтрации и их реализация на ПЛИС // Proc. of Fifth Intern. Conf. «Pattern Recognition and Information Processing». — Minsk, 1999. — **2**. — P. 239–243.
7. Мархівка В.С., Яцимірський М.М. Розпаралелювання програм швидких ортогональних перетворень у мультипроцесорних системах // Вісн. ДУ «Львівська політехніка». Комп'ютерна інженерія та інформаційні технології. — 1998. — № 349. — С. 21–26.
8. Анисимов А.В., Яджак М.С. Построение оптимальных алгоритмов массовых вычислений в задачах цифровой фильтрации // Кибернетика и системный анализ. — 2008. — № 4. — С. 3–14.

9. Яджак М.С. Квазісистолічні обчислювальні структури та їх застосування // Академический вестник. — 2007. — № 20. — С. 53–57.
10. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. — Новосибирск: Наука. Сиб. отд.-ние, 1988. — 129 с.
11. Анисимов А.В., Завадский И.А., Марченко А.А. Исследование эффективности использования оптической элементной базы в синхронной арифметике // Упр. системы и машины. — 2006. — № 1. — С. 49–58, 66.
12. Файнзильберг Л.С. Адаптивное сглаживание шумов в информационных технологиях обработки физиологических сигналов // Математичні машини і системи. — 2002. — № 3. — С. 96–104.
13. Lamport L. The parallel execution of DO loops // Comm. ACM. — 1974. — **17**, N 2. — P. 83–93.
14. Вальковский В.А. Распараллеливание алгоритмов и программ. Структурный подход. — М.: Радио и связь, 1989. — 176 с.
15. Яджак М.С. О численном алгоритме решения пространственной задачи каскадной цифровой фильтрации // Проблемы управления и информатики. — 2004. — № 3. — С. 107–120.
16. Яджак М.С. Об оптимальном в одном классе алгоритме решения трехмерной задачи цифровой фильтрации // Там же. — 2000. — № 6. — С. 66–81.
17. Штейнберг Р.Б. Вычисление задержки в стартах конвейеров для суперкомпьютеров со структурно-процедурной организацией вычислений // Искусственный интеллект. — 2003. — № 4. — С. 105–112.
18. Яджак М.С. Вирішення проблеми реалізації деяких паралельних алгоритмів цифрової фільтрації даних // Відбір і обробка інформації. — 2011. — Вип. 35 (111). — С. 116–121.
19. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. — Ростов н/Д: Изд-во Ростов. ун-та, 2004. — 192 с.

Поступила 03.05.2012