УДК 004.415.2:004.272.43:681.3.07

Дробик О.В., к.т.н.; **Лобанов Л.П.,** к.т.н.; **Яскевич В.О.**
(Государственный университет телекоммуникаций)

РАСПАРАЛЛЕЛИВАНИЕ ПОТОКА КОМАНД В МУЛЬТИЯДЕРНЫХ МИКРОПРОЦЕССОРАХ

Дробик О.В., Лобанов Л.П., Яскевич В.О. Розпаралелювання потоку команд у мультиядерних мікропроцесорах. Пропонується якість розпаралелювання потоку команд у мультиядерних мікропроцесорах оцінювати наступними параметрами: коефіцієнт розпаралелювання та коефіцієнт простою кожного ядра. Пропонуються шляхи поліпшення цих параметрів за рахунок використання табличних методів реалізації операцій множення і ділення.

Ключові слова: мультиядерний мікропроцесор, паралельні обчислення, табличний метод, множення, ділення, потік команд, якість розпаралелювання

Дробик А.В., Лобанов Л.П., Яскевич В.А. Распараллеливание потока команд в мультиядерных микропроцессорах. Предлагается качество распараллеливания потока команд в мультиядерных микропроцессорах оценивать параметрами: коэффициент распараллеливания и коэффициент простоя каждого ядра. Предлагаются пути улучшения этих параметров за счет использования табличных методов реализации операций умножения и деления.

Ключевые слова: мультиядерный микропроцессор, параллельные вычисления, табличный метод, умножение, деление, поток команд, качество распараллеливания

Drobyk O.V., Lobanov L.P., Yaskevych V.O. Construction digital circuits using multiplexer. Offer quality instruction stream parallelism in multicore microprocessors estimated parameters: the coefficient of parallelism and idle ratio of each core. The ways to improve these parameters through the use of tabular implementation methods of multiplication and division.

Keywords: multicore microprocessor, parallelism, tabular method, multiplication, division, instruction stream, quality of the parallelism

Проблема параллельных вычислений продолжает оставаться актуальной и с появлением мультиядерных микропроцессоров. Одна из главных задач при распараллеливании на конечном этапе – распределение потока команд произвольной программы по ядрам микропроцессора [1…3].

Качество процесса распределения команд для микропроцессора с n ядрами можно оценить такими параметрами:

– коэффициент распараллеливания ядра:

$$k_r(i) = \frac{m_i}{M}, \qquad \textbf{(1)}$$

где M – общее число команд в программе,

m_i – число команд, которые выполняются i-м ядром;

k_p – коэффициент простоя ядра, определяется как:

$$k_p = \frac{t_p(i)}{T}, \qquad \textbf{(2)}$$

где T – общее время выполнения программы, $t_p(i)$ – время простоя i-го ядра.

В идеальном случае коэффициент распараллеливания i-го ядра должен иметь значение близкое к значению

$$k_r(i) = \frac{1}{n}. \qquad \textbf{(3)}$$

Реально значения коэффициентов $k_r(i)$ находятся в пределах

$$0 < k_r(i) < 1. \qquad \textbf{(4)}$$

Близость коэффициента к единице характеризует большую зависимость каждой команды от предыдущих, что затрудняет процесс распараллеливания. Близость коэффициента к нулю свидельствует, что ядро практически работает вхолостую.

Для коэффициента простоя идеальным значением является значения близкое к нулю.

Если не принимать никаких мер, указанные параметры принимают значения отличные от идеальных. С большой долей уверенности можно предположить, что для получения значений параметров близких к оптимальным, является создание на этапе проектирования алгоритмов, которые поддаются распараллеливанию.

Для примера рассмотрим алгоритм поиска в массиве максимального (или минимального) по значению элемента для двуядерного микропроцессора. Для классического варианта алгоритма значения коэффициентов распараллеливания имеют значения

$$k_r(1) = 0,8, \quad k_r(2) = 0,2. \tag{5}$$

При использовании приема, при котором один процесс осуществляет поиск среди четных элементов массива, а другой – среди нечетных, значения коэффициентов примут значения

$$k_r(1) = 0,55, \quad k_r(2) = 0,45. \tag{6}$$

Аналогичным образом можно организовать поиск и для мультиядерного микропроцессора. При таком построении исходного алгоритма естественно, что и коэффициенты простоя принимают значения, близкие к оптимальным.

В общем случае, на процесс распараллеливания и на значения коэффициентов простоя, влияет длительность выполнения команд.

Следующий пример подтверждает этот факт. Рассмотрим фрагмент программы для двуядерного микропроцессора:

Ядро 1	**Ядро 2**
mov al, const	–
mov cl, al	imuld
mov b, ax	–

На Рис. 1 показана временная диаграмма выполнения данного фрагмента программы. Вследствие того, что каждая команда имеет свою длительность выполнения, возникает ситуация, когда одно ядро вынуждено ожидать завершения команды другого ядра. Наиболее часто эта ситуация возникает при появлении в программе команд умножения или деления, длительность выполнения которых значительно превышает длительность выполнения других команд.

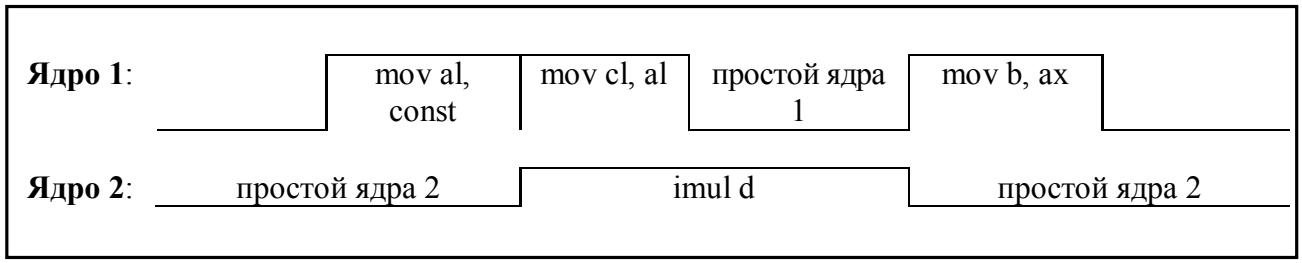

Рис. 1. Временная диаграмма выполнения фрагмента программы

Сокращение времени выполнения команд умножения и деления – реальный способ улучшения параметров распараллеливания. Для этого подходят табличные методы реализации операций умножения и деления.

Табличные методы в чистом виде по времени самые быстрые, но требуют большого объема памяти. Как показано на Рис. 2 для двух n-разрядных операндов объем табличной памяти составляет 2^{2n} слов, разрядность которых определятся требуемой точностью представления результата.

В связи с этим возникает задача минимизации табличной памяти. Предлагаются два способа аппаратной реализации операции умножения с использованием табличной памяти меньшего объема.

Первый способ основан на использовании известной формулы

$$R = a \cdot b = \frac{1}{4}[(a+b)^2 - (a-b)^2], \qquad \textbf{(7)}$$

где a, b – сомножители разрядности n, R – результат умножения.

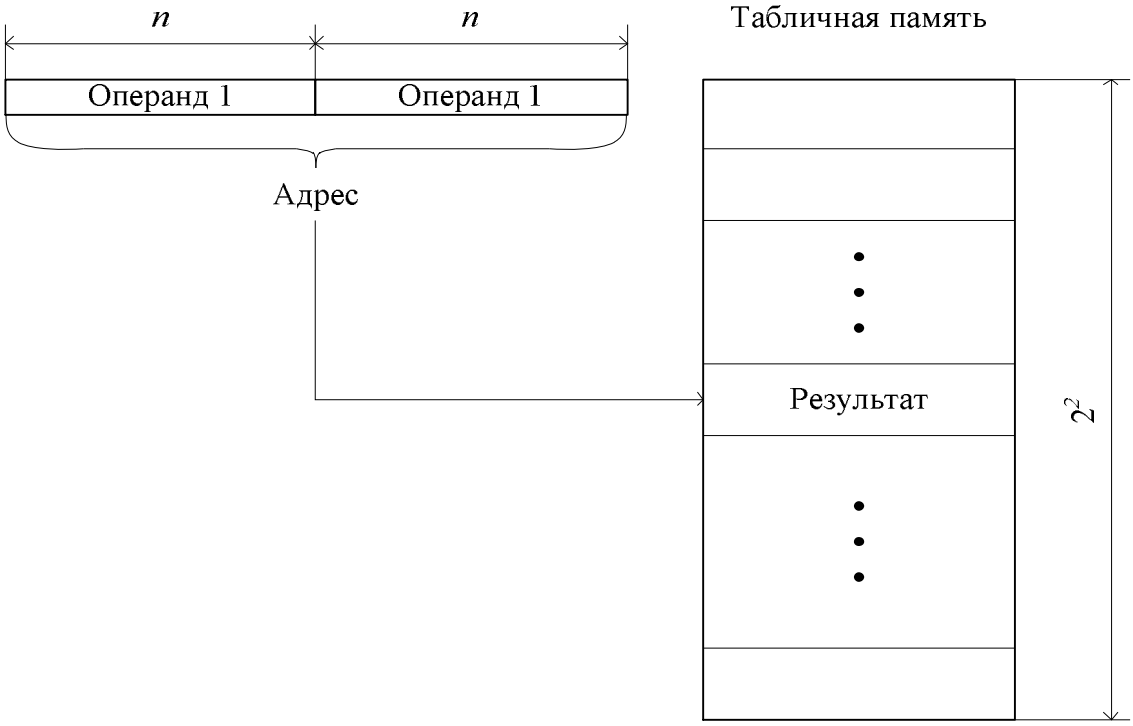

Рис. 2. Принцип работы табличной памяти

При этом способе используется таблица квадратов чисел разрядности $n+1$, что дает уменьшение объема по сравнению с предыдущим (чисто табличным) оцениваемое значением

$$\frac{2^{2n}}{2^{n+1}} = 2^{n-1}. \qquad \textbf{(8)}$$

Если при этом способе использовать для сложения и вычитания обратные коды, то выражение (7) преобразуется в следующее:

$$D = \frac{1}{4}\left[\overline{\overline{(a+b)^2} + \overline{(a+\bar{b})^2}}\right], \qquad \textbf{(9)}$$

где черта сверху означает, что используется обратный код.

Необходимо отметить, что при выполнении условия

$$a < b \qquad \textbf{(10)}$$

операция сложения выполняется без появления единицы циклического переноса, что упрощает аппаратную реализацию умножителя. Если условие (10) не выполняется, сомножители меняются местами, что можно выполнить легко.

На Рис. 3 представлена схема умножителя. Деление на 4 можно осуществить сдвигом результата на 2 разряда вправо за один такт [4].

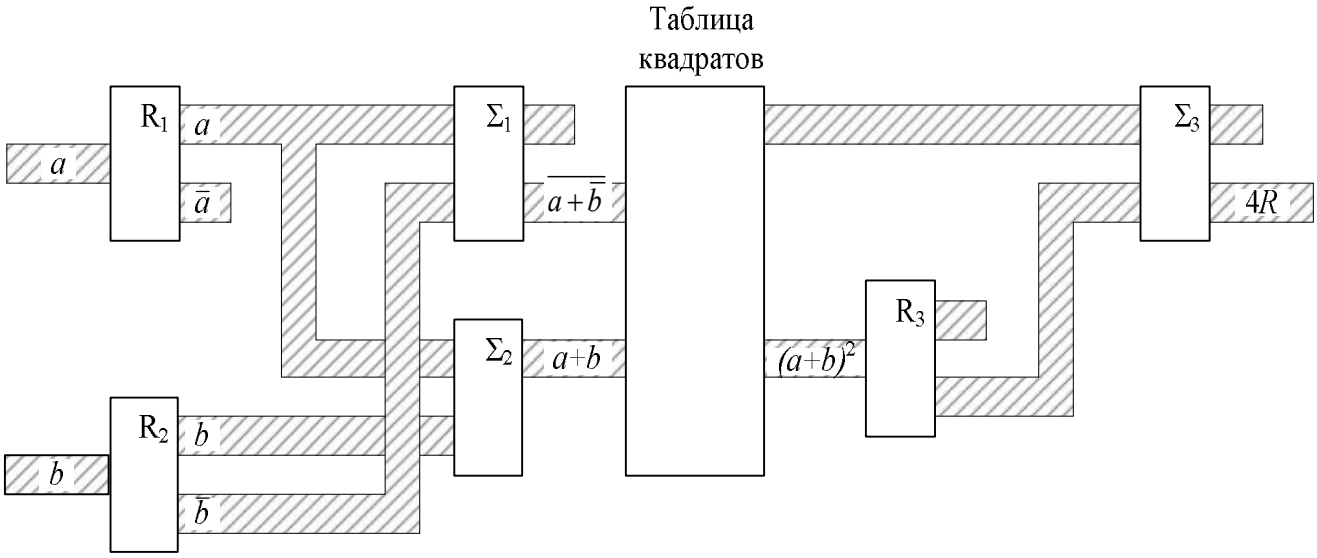

Рис. 3. Умножитель чисел по формуле (9)

Второй способ основан на использовании табличной памяти объемом $2^{n/2}$ при умножении чисел разрядности n. Можно показать, что только при таком объеме табличной памяти, количество суммирований частичных произведений и сдвигов будет минимальным, а именно:

– осуществляется одно сложное суммирование четырех частичных произведений;

– три сдвига частичных произведений (Рис. 4).

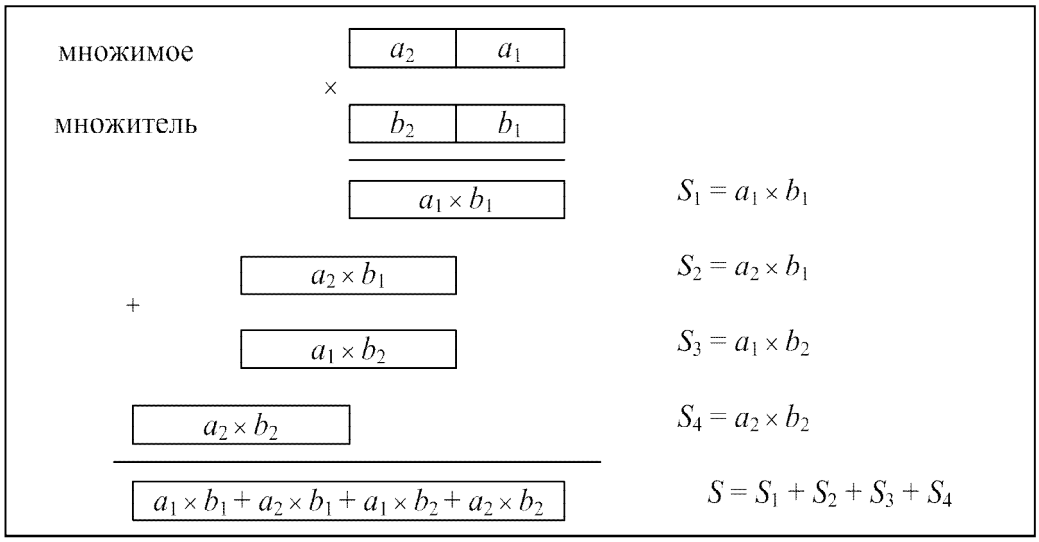

Рис. 4. Умножение чисел разрядности n (n-четное)

Сложность суммирования заключается в том, что одновременно необходимо складывать три двоичных цифры. Частичные произведение $S_i (i = 1, 2, 3, 4)$ извлекаются из табличной памяти по адресам, получаемых комбинацией всех возможных половин сомножителей

$$a_i \cdot b_j, \quad i, j = 1, 2.$$

Структуру сложного сумматора получить несложно, если использовать нетрадиционный логический базис [5, 6].

Выводы

1. Распараллеливание потока команд по ядрам микропроцессора является непростой задачей в основном из-за того, что в большинстве программ имеется большая зависимость каждой очередной команды от одной или нескольких предыдущих команд.

2. Применение ускоренных методов реализации операций умножения и деления позволяет улучшить параметры распараллеливания потока команд.

3. Чисто табличные методы реализации длинных операций требует больших объемов памяти. Предложенные методы реализации операций умножения и деления позволяют уменьшить объем памяти при незначительном увеличении времени выполнения.

Литература

1. Корнеев В.В. Современные микропроцессоры / В.В. Корнеев, А.В. Киселев. – [3-е изд.]. – СПб.: БХВ-Петербург, 2003. – 448 с.

2. Таненбаум Э. Архитектура компьютера / Э. Таненбаум. – [5-е изд.]. – СПб.: Питер, 2007. – 844 с.

3. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации : учебн. для ВУЗ-ов / В.Л. Бройдо. – [2-е изд.]. – СПб.: Питер, 2007. – 844 с.

4. Потемкин Н.С. Функциональные узлы цифровой автоматики / Н.С. Потемкин. – М.: Энергоатомиздат, 1988.

5. Дробик О.В. Побудова цифрових схем на мультиплексорах / [О.В. Дробик, Л.П. Лобанов, В.О. Яскевич] // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2012. – №8. – С.16-21.

6. Лобанов Л.П. Функциональные построения в EMS-базисе / Л.П. Лобанов, В.О. Яскевич // Вісник Державного університету інформаційно-комунікаційних технологій. – 2007. – Т.5, №2. – С. 185-188.