ОПРЕДЕЛЕНИЕ ВЕРОЯТНОСТИ КЭШ-ПОПАДАНИЙ ПО ГРАФ-СХЕМЕ АЛГОРИТМА В КОМПОЗИЦИОННЫХ МИКРОПРОГРАММНЫХ УСТРОЙСТВАХ УПРАВЛЕНИЯ

Ковалев С.А., Бабаков Р.М.

Кафедра ЭВМ, ДонГТУ
kovalev@pop.dgtu.donetsk.ua

Abstract

Kovalev S.A., Babakov R.M. Determination of probability of cache-hits by a flow-chart in compositional microprogram control devices. The method of determination of probability of cache-hits by a flow-chart is considered. It can be used for determination of effectn>eness of using of cache memory with direct-mapping organization in compositional microprogram control devices.

Введение

Одним из центральных блоков цифровых устройств является устройство управления, в качестве которого может использоваться композиционное микропрограммное устройство управления, обладающее лучшими характеристиками по сравнению с устройствами управления, использующими операционно-адресную структуру управляющих слов [1]. Одной из важных задач, возникающих при синтезе устройств управления, является повышение быстродействия работы схемы, что возможно за счет использования кэш-памяти для кэширования микрокоманд [2]. При оценке эффективности использования кэш-памяти важной характеристикой является вероятность кэш-попаданий, зависящая, главным образом, от реализуемого алгоритма. В настоящей работе предлагается методика аналитического определения вероятности кэш-попаданий по заданной граф-схеме алгоритма для кэш-памяти с прямым отображением в композиционных микропрограммных устройствах управления.

Организация композиционного микропрограммного устройства управления с кэш-памятью

В структуре композиционных микропрограммных устройств управления @<МУУ) наибольшее время срабатывания имеет управляющая память микрокоманд (УП) [2, 3]. Это обусловлено тем, что при синтезе схем КМУУ УП обычно реализуется на ПЗУ [3, 4]. Если считать, что УП реализована на ПЗУ и время ее срабатывания tyrf=200 нс [5], а время срабатывания счетчика адреса микрокоманд (СЧАМК) tc4AMK=20 нс, то становится очевидным, что общее время срабатывания цепочки <С4АМК, УП> составляет 220 нс, причем время срабатывания УП здесь составляет 90%. Следовательно, важной задачей, возникающей при синтезе КМУУ, является снижение времени доступа к УП.

Один из способов увеличения быстродействия КМУУ заключается в использовании в структуре КМУУ кэш-памяти, предназначенной для хранения часто используемых микрокоманд (МК) и выполненной на быстродействующих микросхемах типа SRAM [2]. При этом вредя доступа к кэш-памяти во много раз меньше, чем к УП, а размер кэш-памяти обычно составляет несколько процентов от общего размера УП. Это позволяет многократно повысить быстродействие схемы при незначительном увеличении аппаратурных затрат и стоимости.

В [2] предлагается встраивать кэш-память так, как показано на рис. 1. Помимо самой кэш-памяти микрокоманд КПМК в схеме присутствует кэш-контроллер К, управляющий процессом выборки микрокоманды из памяти.

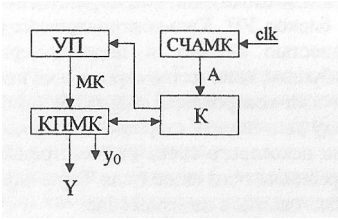

Рис. 1. Расположение кэш-памяти
в структуре КМУУ

Управляющая и кэш-память организованы следующим образом. Известно, что в КМУУ управляющая память содержит только операционные микрокоманды. Пусть разрядность адреса R =]log2M[, где M - число операторных вершин в граф-схеме алгоритма, а число разрядов в микрокоманде равно K. Значит, УП может содержать до 2^K слов (далее под словом будем понимать микрокоманду разрядности K). Управляющую память можно представить как набор последовательно расположенных блоков по S слов в каждом. Всего таких блоков будет $2^R/S$. Количество слов в каждом блоке будем считать кратным степени двойки. Тогда кэш-память можно представить как некоторое множество строк, каждая из которых имеет длину, равную длине одного блока УП. В каждой из N строк может быть размещен блок из S слов, причем число строк много меньше количества блоков в управляющей памяти: N « $2^R/S$.

Схема (рис. 1) работает следующим образом.

Получив из СЧАМК адрес очередной МК, кэш-контроллер проверяет наличие этой МК в кэш-памяти. Если МК обнаружена (кэш-попадание), то осуществляется «короткий» цикл чтения микрокоманды из кэш-памяти. Если же микрокоманда не обнаружена (кэш-промах), то выполняется «длинный» цикл чтения из УП блока микрокоманд, к которому принадлежит текущая МК, и записи блока в КПМК с последующим чтением микрокоманды из КПМК. В любой момент времени в кэш-памяти содержится некоторое подмножество блоков из УП.

Организация КМУУ с кэш-памятью с прямым отображением

Наиболее простой организацией кэш-памяти является кэш-память с прямым отображением [6], при которой один или несколько блоков УП строго соответствуют

некоторой строке кэш-памяти. Поскольку в каждый момент времени в строке кэш-памяти может быть размещен только один блок, для каждой строки кэш-памяти вводится специальный признак ~ тэг, однозначно идентифицирующий блок УП.

Прямое отображение описывается формулой

$$i = j \bmod N, \tag{1}$$

где i - номер строки кэш-памяти; j - номер блока УП; N - число строк в кэш-памяти.

Функция прямого отображения реализуется путем анализа адреса управляющей памяти, который состоит из трех полей. Первое поле Word разрядностью w определяет порядковый номер слова в блоке УП и в строке кэш-памяти. Следующие s бит определяют один из 2^s блоков УП. Кэш-контроллер интерпретирует эти разряды как поле тэга Tag разрядностью s-r бит и поле номера строки кэш-памяти Line разрядностью r. Таким образом, количество строк в кэш-памяти $N=2^r$.

Принцип работы кэш-контроллера с прямым отображением следующий. При записи блока УП в строку кэш-памяти старшие s-r разрядов записываются в поле тэга этой строки. При чтении некоторого слова кэш-контроллер выделяет из адреса слова поле Line, после чего сравнивает значение поля Tag в адресе запрашиваемого слова и значение тэга в строке кэш-памяти с номером Line.

Если значения тэгов совпадают, значит, блок УП, к которому принадлежит запрашиваемое слово, размещен в кэш-памяти. В этом случае из строки кэш-памяти с номером Line выбирается слово с номером, равным значению поля Word в адресе запрашиваемого слова. В случае же несовпадения тэгов кэш-контроллер вьщает адрес запрашиваемого слова на входы кэш-памятн, выполняет цикл чтения блока из УП, после чего размещает прочитанный блок в соответствующей строке кэш-памяти.

Оценка эффективности использования кэш-памяти в композиционных микропрограммных устройствах управления

В процессе выполнения произвольной микрокоманды возможно одно из двух событий: произошло кэш-попадание либо кэш-промах. Поскольку эти два события являются несовместными и могут произойти с некоторой вероятностью, то среднее время ожидания композиционного микропрограммного устройства управления, включающего в себя кэш-память, можно определить следующим образом:

$$T_E = P_h \cdot T_C + (1 - P_h) \cdot T_m, \tag{2}$$

где Ph - вероятность кэш-попаданий при выполнении реализуемого алгоритма;

Tc - время поиска в кэш-памяти и выбор данных в случае кэш-попадания;

T_m - время доступа к данным в случае кэш-промаха, при этом **T_m=Tyn+Ti,** где **Tyn** - время доступа к управляющей памяти в системе без кэш-памяти, **Ti** - так называемое время потерь при кэш-промахе, затрачиваемое на поиск запрашиваемых данных в кэш-памяти и помещение данных из управляющей памяти в кэш-память.

Очевидно, что для достижения высокой скорости доступа к управляющей памяти требуется высокая вероятность кэш-попаданий.

Эффективность использования кэш-памяти определяется тем, насколько кэш-память повышает скорость доступа к системе памяти. Полагая, что в общем случае Tc < Tyn < T_m, выигрыш в скорости доступа к данным в системе с кэш-памятью по сравнению с системой без кэш-памяти можно определить как отношение Tyn к T^:

$$E = \frac{T_{УП}}{T_E} = \frac{T_{УП}}{P_h \cdot T_C + (1 - P_h) \cdot T_m}. \quad (3)$$

Методика определения вероятности кэш-попаданий по заданному алгоритму

В формуле (3) величины Tc, T_m и T_{yn} определяются элементной базой и функциональной организацией аппаратуры. Конкретные значения этих величин можно легко определить, зная длительность и порядок срабатывания электронных компонентов схем кэш-памяти и кэш-контроллера.

Наиболее сложно определить значение вероятности кэш-попаданий Ph, поскольку эта величина зависит как от выбранных характеристик кэш-памяти, так и от реализуемого алгоритма. При этом величина Ph может изменяться в пределах от 0 до 1 и тем самым фундаментально влиять на эффективность использования кэш-памяти.

Если для величин Tc и T_m можно легко получить точные значения (например, по временной диаграмме работы устройства), то даже приблизительное определение значения Ph является затруднительным, поскольку ситуации кэш-попаданий и кэш-промахов не являются очевидными из заданной граф-схемы алгоритма и характеристик кэш-памяти.

Исходными данными для определения вероятности кэш-попаданий Ph будем считать:

- граф-схему реализуемого алгоритма;
- вероятности условных переходов для логических условий, анализируемых в алгоритме;
- адресацию микрокоманд;
- число строк и слов кэш-памяти;
- организацию кэш-памяти (в данной работе рассматривается кэш-память с прямым отображением).

Граф-схему алгоритма будем считать преобразованной для реализации в устройстве управления с естественной адресацией микрокоманд [3].

Содержимое операционной микрокоманды на этапе определения вероятности кэш-попаданий не имеет значения. Действия, выполняемые в операционных микрокомандах, либо никак не влияют на процесс выполнения алгоритма, либо влияют на вероятности переходов для логических условий, которые уже считаются заданными. Поэтому алгоритм, анализируемый с применением методики, разработанной в данной работе, принимает абстрактный характер.

Для аналитического определения вероятности кэш-попаданий для заданного алгоритма предлагается методика, включающая следующие этапы:

- определение вероятности выполнения i-й микрокоманды;
- определение вероятности кэш-попадания i-й микрокоманды и алгоритма в целом.

Определение вероятности выполнения i-й микрокоманды

На протяжении выполнения алгоритма каждая микрокоманда выполняется некоторое число раз, зависящее от вероятностей условных переходов и в общем случае изменяющееся при каждом повторном выполнении алгоритма. Следовательно, можно говорить о вероятности выполнения отдельной микрокоманды p(a;), определяемой как отаошение среднего количества выполнений этой микрокоманды q(aO к среднему количеству микрокоманд Q, выполняющихся за один проход алгоритма:

$$p(a_i) = \frac{q(a_i)}{Q}. \qquad (4)$$

Здесь среднее количество микрокоманд Q равно алгебраической сумме значений q(a,) всех микрокоманд:

$$Q = \sum_{i=1}^{M} q(a_i). \qquad (5)$$

Значения q(a,) можно определить, как сумму вероятностей вьшолнения каждой операторной вершины граф-схемы алгоритма, умноженных на вероятности перехода из соответствующей вершины в вершину a,. Для этого составим систему уравнений вида

$$\begin{aligned} q(a_i) &= p(a_1 \to a_i)\cdot q(a_1) + p(a_2 \to a_i)\cdot q(a_2) + \ldots + p(a_{i-1} \to a_i)\cdot q(a_{i-1}) + \\ &+ p(a_{i+1} \to a_i)\cdot q(a_{i+1}) + \ldots + p(a_M \to a_i)\cdot q(a_M) = \\ &= \sum_j p(a_j \to a_i)\cdot q(a_j), \quad j \in 1..M;\ j \neq i. \end{aligned} \qquad (6)$$

Здесь p(aj -> aO - вероятность непосредственного перехода из вершины aj в вершину ai. Значение этой вероятности может быть равно:

- нулю, если непосредственный переход из a, в a; не возможен;
- единице, если обе вершины расположены последовательно в одной операторной линейной цепи;
- вероятности перехода для соответствующей ветви условной вершины, если переход из щ в a; осуществляется через одну условную вершину;
- произведению вероятностей соответствующих ветвей условных вершин, если переход ИЗ aj в a; осуществляется через несколько условных вершин.

Полученная таким образом система представляет собой полную систему из M линейных алгебраических уравнений, имеющую единственное решение. Такая система может быть решена любым известным методом решения систем линейных уравнений (например, методом последовательной подстановки) или с помощью специальных пакетов математических расчетов типа MathCad.

После решения системы и получения значений q(a;) определяются значение Q по формуле (5) и значения p(a;) по формуле (4).

Определение вероятности кэш-попадания i-й микрокоманды

Поскольку при каждом выполнении микрокоманды a; могут произойти либо кэш-попадание, либо кэш-промах, можно утверждать, что полное количество выполнений q(a,) состоит из двух частей: количества выполнений qt(aO, при которых произошло кэш-попадание, и количества выполнений $я_{ш}(a_{:})$, при которых произошел кэш-промах:

$$q(a_i) = q_h(a_i) + q_m(a_i). \quad (7)$$

Преобразуя формулу (4), можно записать, что

$$p(a_i) = \frac{q(a_i)}{Q} = \frac{q_h(a_i) + q_m(a_i)}{Q} = \frac{q_h(a_i)}{Q} + \frac{q_m(a_i)}{Q} = p_h(a_i) + p_m(a_i), \quad (8)$$

то есть вероятность выполнения i-й микрокоманды также складывается из двух частей: из вероятности кэш-попаданий рь(aO и вероятности кэш-промахов $р_{„}(a_{,})$.

Для случая кэш-памяти с прямым отображением кэш-промах возможен как для микрокоманд, являющихся первыми в блоке управляющей памяти (соответствующий блок может отсутствовать в управляющей памяти на момент выполнения микрокоманды), так и для команд, являющихся входами операторных линейных цепей (первая в цепи и предыдущая микрокоманды могут принадлежать разным блокам управляющей памяти). Для остальных же микрокоманд кэш-промах невозможен ни при каких обстоятельствах.

Сформируем подмножество микрокоманд V, для которых кэш-промах возможен.

Определим вероятность кэш-промаха для каждой микрокоманды из множества V. Пусть кэш-память содержит несколько строк. В случае организации кэш-памяти с прямым отображением каждая микрокоманда «претендует» на место в строго определенной строке, определяемой соответствующим битом (битами) в адресе микрокоманды. Тогда для кэш-памяти с прямым отображением справедливы три правила:

1) Если в адресах микрокоманд a_3 и ai номера строк кэш-памяти различны, то очевидно, что при их последовательном выполнении микрокоманда щ не может заменить микрокоманду a; в строке кэш-памяти и подготовить тем самым ситуацию кэш-промаха при выполнении $a_{„}$, то есть при последовательном выполнении таких микрокоманд кэш-промах невозможен.
2) Если же a, и a, имеют в своем адресе одинаковые значения номера строки, но принадлежат разным блокам управляющей памяти, то при выполнении щ блок, к которому принадлежит aj, заменит в кэш-памяти блок, к которому принадлежит ai, и при последующем выполнении a, возникнет ситуация кэш-промаха.
3) Если a, и ai принадлежат одному блоку (и, как следствие, одной строке кэш-памяти), то при их последовательном выполнении произойдет кэш-попадание, поскольку на момент выполнения a; соответствующий блок уже будет находиться в кэш-памяти (будет загружен туда при выполнении щ или раньше).

Поскольку микрокоманды, относящиеся к одной строке кэш-памяти, не могут создать ситуации кэш-промаха для микрокоманд, относящихся к другой строке кэш-памяти (правило 1), мы можем разбить множество V на N подмножеств V_0, …, V_{1w}, в

каждом из которых будут находиться микрокоманды, относящиеся к соответствующей строке кэш-памяти.

Построим для каждого подмножества Vi подграф Г; исходной граф-схемы алгоритма, в котором микрокоманды, не принадлежащие подмножеству V, заменяются безусловными переходами.

Рассмотрим произвольную микрокоманду a;eVk. В общем случае в подграфе $Г_ь$ возможны следующие типы переходов в микрокоманду а,:

1. Переход из начальной вершины;
2. Переход из ajeVk, причем а; и щ находятся в разных блоках управляющей памяти;
3. Переход из ajeVk, причем а; и aj находятся в одном блоке управляющей памяти;
4. Переход из а; (цикл).

В первом случае всегда будет иметь место кэш-промах, поскольку в начале выполнения алгоритма строки кэш-памяти считаются незаполненными, и микрокоманда ai заведомо отсутствует в кэш-памяти. Во втором случае кэш-промах также будет иметь место в силу правила 2. В третьем случае всегда будет происходить кэш-попадание в силу правила 3. В четвертом случае также всегда будет иметь место кэш-попадание.

Рассматривая случаи 3 и 4, можно определить вероятность кэш-попаданий для каждой микрокоманды.

Отметим, что для микрокоманд aj и а; (ajeV^, a;eVk) существует некоторое множество маршрутов W(a^ai), задающих непосредственные переходы из aj в а;. Если из aj в а; можно попасть лишь единственным способом, то множество $Wla_j\text{^}a_1$) будет включать единственный маршрут.

Каждый маршрут проходит через некоторое количество условных вершин, и, следовательно, может характеризоваться некоторой вероятностью. Для нескольких маршрутов суммарная вероятность равна сумме вероятностей каждого маршрута:

$$p(W(a_j \to a_i)) = \sum_{n=1}^{N} p(W_n(a_j \to a_i)). \qquad \textbf{(9)}$$

Здесь p(W(a, H>a,)) - вероятность перехода из вершины щ в вершину а; с учетом всех возможных маршрутов; $p(W_n(aj\text{^}ai))$ - вероятность перехода из вершины а; в вершину aj для η-го маршрута; N - количество маршрутов между вершинами aj и а;.

Если переход из вершины в вершину невозможен, то вероятность такого перехода равна нулю; для безусловного перехода вероятность равна единице.

Для каждой микрокоманды a,eV сформируем множество микрокоманд R_b переход из которых в а, не приведет к кэш-промаху. С учетом правила 3, в это множество могут входить собственно микрокоманда а; (если существуют переходы из а; в а;), а также микрокоманды, находящиеся с а; в одном блоке управляющей памяти (если существуют соответствующие переходы).

Теперь мы можем определить вероятность кэш-попадания для і-й микрокоманды, используя следующую формулу:

$$p_h(a_i) = \sum_{j \in R_i} p(M(a_j \to a_i)) \cdot p(a_j).$$

Здесь Σ - сумма вероятностей маршрутов из каждой микрокоманды, принадлежащей Ri, в aj; p(aj) - вероятность выполнения микрокоманды aj.

Зная значение Pi,(a,), можно определить вероятность кэш-промаха каждой микрокоманды Pm(aO по формуле (8):

$$p_m(a_i) = p(a_i) - p_h(a_i). \qquad (H)$$

Вероятность кэш-промаха для всего алгоритма, в соответствии с правилом сложения вероятностей, будет равна

$$P_m = \sum p_m(a_i), \quad a_i \in V. \qquad (12)$$

Тогда вероятность кэш-попаданий алгоритма будет равна

$$P_h = 1 - P_m. \qquad (13)$$

Заключение

Эффективность использования кэш-памяти в композиционных микропрограммных устройствах управления в значительной степени определяется вероятностью кэш-попаданий. Эта вероятность зависит от реализуемого алгоритма и параметров кэш-памяти (длины строки и количества строк). Предложенная методика определения вероятности кэш-попаданий для кэш-памяти с прямым отображением позволяет аналитически определить эту характеристику для заданного алгоритма управления. Это дает возможность подобрать для реализуемого алгоритма параметры кэш-памяти, увеличивающие производительность устройства управления без значительного увеличения стоимости.

Литература

1. Баранов С.И. Синтез микропрограммных автоматов. - Д : Энергия, 1979. - 232 с.
2. Ковалев С.А., Бабаков Р.М., Баркалов А.А. (ВТ-95д). Применение методов кэширования в композиционных микропрограммных устройствах управления. - В кн.: Наукові праці Донецького державного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка». Випуск 15. - Донецьк, ДонДТУ, 2000. - с. 118-123.
3. Баркалов А.А., Палагин А.В. Синтез микропрограммных устройств управления. - Киев: ИК НАН Украины, 1997. - 135 с.
4. Баркалов А.А. Микропрограммное устройство управления как композиция автоматов с программируемой и жесткой логикой // Автоматика и вычислительная техника. - 1983. - № 4. - с. 42-50.
5. Интегральные микросхемы: Справочник / Под ред. Б. В. Тарабрина. - М.: Радио и связь, 1984 - 528 с.
6. Mark D. Hill. Aspects of Cache Memory and Instruction Buffer Performance. - Computer Science Division, University of California, Berkeley, 1987 - 187 p.