УДК 621.3.049.771.14

Гибридный метод аллокации массивов памяти в аппаратных платформах с разветвленной структурой памяти на базе процессора NeuroMatrix® DSP

С.В. Мушкаев, А.В. Андрианов

ЗАО НТЦ «Модуль», mushkaev@module.ru, andrianov@module.ru

***Аннотация* — Рассматриваются особенности построения эффективного прикладного программного обеспечения в аппаратных платформах с разветвленной организацией системы памяти на примере DSP процессора с архитектурой NeuroMatrix®, разработки ЗАО «НТЦ Модуль». Предлагается гибридный метод выделения памяти из нескольких неравнозначных банков памяти, позволяющий значительно повысить производительность программ, ускорить этап разработки и отладки ПО, упростить используемые интерфейсы, повысить переносимость программ.**

***Ключевые слова* — аллокатор, аллокация памяти, управление памятью, malloc, memory allocation, NeuroMatrix, NMC, system-on-a-chip, SoC, DSP, МВ77.07, МЦ51.03, NM6406, К1879ВМ5, К1879ХБ1Я.**

I. Введение

В современных вычислительных системах DSP (СБИС, микрокомпьютеры) часто используется системы со сложной разветвленной организацией системы памяти. Применяются независимые и соответственно физически неравнозначные банки памяти, они могут быть как внешними по отношению к процессору, так и внутренними, с разным объемом и временем доступа. Большие и сложные программы, рассчитанные на единую область памяти, не всегда помещаются в пространстве быстрой внутренней памяти процессора как по размеру кода, так и по объему обрабатываемых данных. В доступной же внешней памяти скорость обмена может падать в несколько раз. В таких случаях требуется адаптация или даже значительное изменение алгоритма, чтобы перераспределить потоки данных на максимальное использование внутренней памяти и минимизацию внешних обращений. Кроме того, если система аппаратно поддерживает одновременный доступ к различным банкам памяти, то это предполагает возможность параллелизма вычислений, что также необходимо учитывать при разработке алгоритма. В виду многообразия аппаратных платформ и конфигураций систем памяти перенос приложения на другие платформы становится также крайне затруднительным.

Приведенные архитектурные особенности показывают насколько важной и сложной может быть задача управления памятью при построении высокоэффективных DSP-приложений. Наличие удобных программных инструментов менеджмента памяти может существенно облегчить данную задачу и ускорить процесс разработки ПО. Стандартные средства аллокации памяти такие как, malloc/new не позволяют это сделать в полной мере.

В данной статье предлагается метод управления памятью, который позволяет программисту частично абстрагироваться от особенностей доступной памяти, автоматизировать процесс оптимизации и таким образом получить максимум производительности из данной архитектуры.

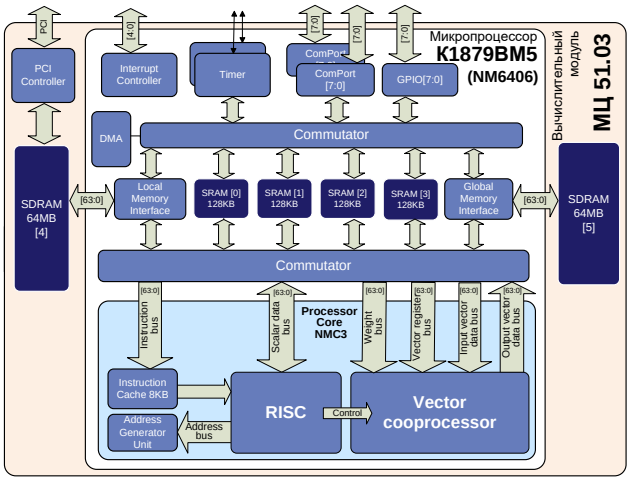

Рис. 1. Функциональная схема СБИС К1879ВМ5 [6] (NM6406), конфигурация накристальной и внешней памяти в составе вычислительного модуля МЦ 51.03 [2]

II. Предпосылки и особенности процессора NeuroMatrix (NMC)

Предлагаемый метод основывается на следующих особенностях архитектуры NMC (NeuroMatrix Core) [1], которые необходимо учитывать при разработке эффективных DSP программ:

1) DSP процессор NeuroMatrix с ядром NMC3 [1] имеет собственную внутреннюю 4-х банковую память объемом 4х128кБ (SRAM[0]..SRAM[3] для СБИС К1879ВМ5[6] см. рис. 1, или IM1,IM3 для СБИС К1879ХБ1Я [7] см. рис. 2), работающую на частоте процессора 324 МГц. Данная память не является

кэшем и программа сама должна максимально задействовать эту память.

2) Векторный сопроцессор включает в себя векторно-матричный умножитель и имеет SIMD структуру команд [1]. Соотношение между скалярными подготовительными операциями (вызов функции, передача параметров, настройка регистров) и количеством непосредственно векторных вычислительных инструкций определяют КПД сопроцессора. Чем больший объем данных поступает на вход функции, тем меньше процент накладных расходов и тем выше ее производительность. На рис. 3 представлена зависимость среднего времени обработки одного байта от размера входных данных на примере 4-х функций: КИХ-фильтрации с размерами окна 9 и 19, умножения вектора на константу и сложения двух векторов. Как видно из графика функции достигают 90% от предельно возможной производительности при объемах данных от 1-2 kB.

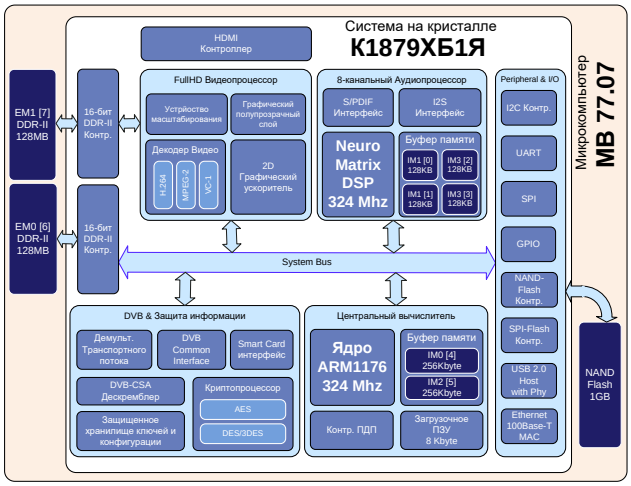

Рис. 2. Функциональная схема СБИС К1879ХБ1Я [7], конфигурация накристальной и внешней памяти в составе микрокомпьютера МВ 77.07 [3]

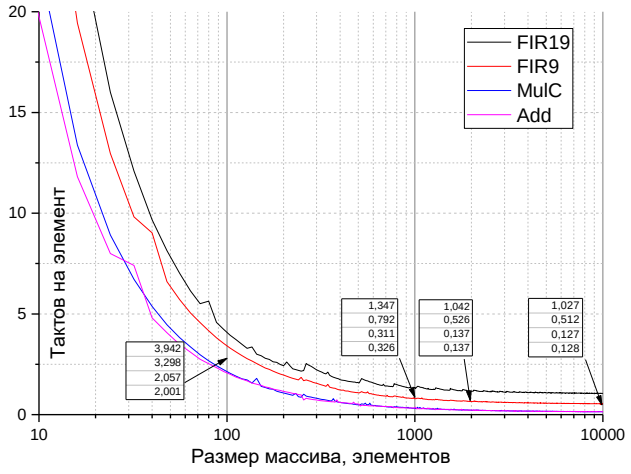

Рис. 3. Зависимость среднего времени обработки элемента массива от его размера

3) Пункт 1 и 2 приводят к тому, что алгоритм будет наиболее оптимальным в том случае, когда обработка данных с объемом, превосходящим внутреннюю память, будет вестись фрагментарно, поочередно обрабатывая каждый фрагмент во всей внутренней памяти. При этом обработка самого фрагмента должна по возможности вестись в терминах векторов и матриц на векторном сопроцессоре.

4) Внутренняя память DSP-процессора аппаратно разбита на 4 банка - SRAM[0]..SRAM[3] (см рис. 1) с независимым доступом к каждому. Также имеется две шины данных к внешней памяти. Если адреса операндов векторных инструкций ссылаются на разные банки, то доступ к ним осуществляется одновременно. Так на рис. 1 показан коммутатор, связывающий 6 шин к памяти с шиной инструкций RISC ядра и 4-мя шинами данных к векторному узлу: входной шине данных, шине векторного регистра, шине весовых коэффициентов и выходной шине. Таким образом, если адреса в этих шинах не пересекаются по банкам памяти, то максимально можно инициировать 5 параллельных потоков данных, включая код. Только уже за счет такого распараллеливания производительность на некоторых инструкциях может возрастать до 4 раз.

A. Пример

На рис. 4 показана зависимость среднего времени обработки одного элемента для функции умножения вектора на константу при различных размещениях массивов по банкам памяти. Видно, что за счет оптимального использования разных банков одной только внутренней памяти скорость может отличаться более чем в два раза.

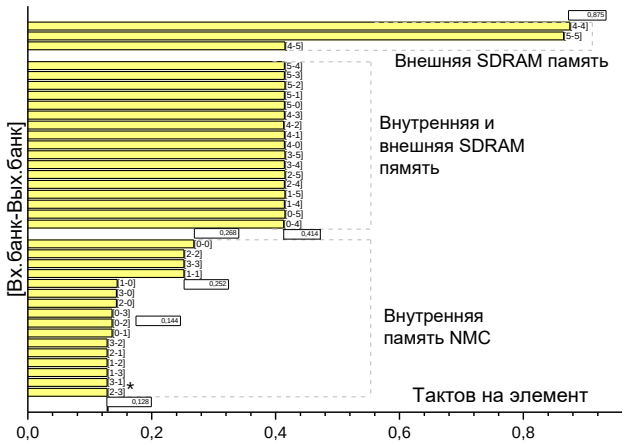

Рис. 4. Среднее время умножения 8-разрядного вектора на константу при разных вариантах размещения вх. и вых. массивов в банках памяти вычислительного модуля МЦ51.03.

**В квадратных скобках указаны номера используемых банков памяти для хранения вх. и вых. данных, соответственно*

B. Пример

На рис. 5 приведены данные производительности той же функции, но на другой платформе – микрокомпьютере МВ77.07 [3]. На внутренней памяти имеем такую же скорость, но на внешней памяти результаты уже сильно разнятся из-за различных типов памяти.

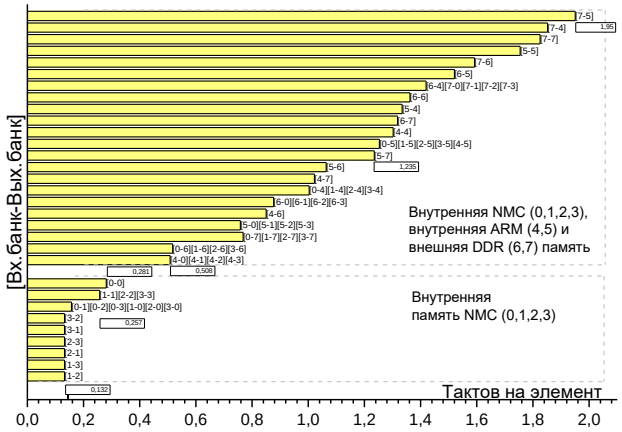

Рис. 5. Среднее время умножения 8-разрядного вектора на константу при разных вариантах размещения вх. и вых. массивов в банках памяти микрокомпьютера МВ77.07

C. Пример

На рис. 6 приведен случай использования функции сложения двух векторов, где производительность на разных конфигурациях внутренней памяти может отличаться уже в три раза. Так, распределяя шину входных данных (для первого вектора), шину векторного регистра (для данных второго вектора), выходную шину по банкам: 1-2-3 соответственно и шину инструкций на банк-0, достигаем максимальной производительности 0,127 такта на сложение двух байтов. В худшем варианте она бы составляла ~0.4 такта на внутренней памяти и 1.3 такта на внешней.

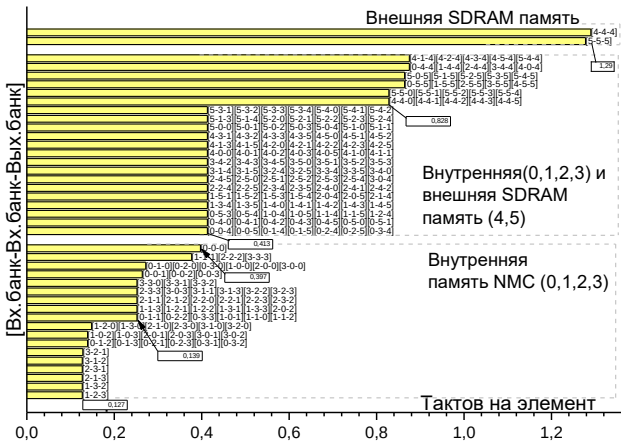

Рис. 6. Производительность функции сложения двух байтовых вектора на константу при разных вариантах размещения вх. и вых. массивов в банках памяти вычислительного модуля МЦ51.03

Приведенные результаты демонстрируют как важно при разработке программы, помимо математической составляющей алгоритма, также учитывать и особенности управления памятью. Этот процесс во многом зависит от доступных инструментов работы с памятью.

III. Существующие методы управления памятью

Для выделения участков памяти в программе существует два стандартных механизма:

1) Динамическое выделение памяти с помощью malloc/new. Метод прост и удобен в использовании, однако, в задачах реального режима времени требует существенных затрат времени, кроме того, в условиях предельного использования всей имеющейся внутренней памяти может быть небезопасным.

2) Статическое распределение. Из некоторого доступного статического пула в программе вручную размечаются области памяти. Способ дает максимальную производительность, однако, в сложных алгоритмах такой способ бывает весьма неудобным, так как необходимо выделять статические массивы максимального объема и жестко привязываться к целевой платформе и параметрам обрабатываемых данных.

Кроме того, в обоих случаях на пользователе также лежит задача оптимального распределения входных, выходных и промежуточных массивов по банкам памяти. Цепочка вычислений и передачи данных в алгоритме может быть весьма длинной и с учетом большого количества доступных банков памяти задача оптимального размещения массивов может быть довольно сложной в виду большого числа комбинаций. Полученная программа лишается гибкости - в случае модификации кода потребуется новая раскладка по банкам. Разработка переносимой программы на другие платформы делает задачу еще более сложной.

IV. Гибридный подход

В некоторых случаях, когда алгоритм является линейным (безусловным с точки зрения вызова аллокаторов) и цикличным, что свойственно задачам цифровой обработки сигналов, решением может служить гибридный подход, сочетающий преимущества динамического и статического механизма управления памятью.

Принцип гибридного механизма состоит в первоначальном использовании динамического выделения на этапе обучения и в статическом выделении в рабочем режиме.

A. Процесс работы

1) Изначально программа пишется так, как если бы она работала с обычным динамическим выделением памяти, но вместо стандартных функций malloc и free используются специальные функции аллокатор/деаллокатор, работающие в двух режимах.

2) На предварительном этапе (этапе обучения) аллокатор запускается в режиме динамического выделения памяти. В этом режиме алгоритм функционирует в своем обычном рабочем порядке, но на каждой итерации цикла меняется порядок банков, из которых выделяются массивы. Путем последовательного перебора всех возможных вариантов выборок (маршрутов) и замера времени каждой итерации производится автоматический поиск оптимального по скорости маршрута.

3) Наилучший маршрут (история аллокаций) сохраняется в виде последовательности выделенных адресов массивов.

4) В рабочем режиме функция-аллокатор переключается на воспроизведение адресов из сохраненной истории в той же последовательности, а деаллокатор отключается вовсе, таким образом, и достигается высокое быстродействие.

5) При последующих запусках программы можно не запускать обучение каждый раз заново, а инициализировать историю адресов с помощью соответствующего маршрута индексов банков при первой итерации цикла программы.

B. Преимущества

1) С помощью полного перебора производится автоматический поиск наилучшего маршрута экспериментальным путем на целевой платформе. Решается зависимость от типа и скорости внешней памяти. Модификация кода не влечет за собой задачу поиска новой оптимальной раскладки вручную.

2) Повышается читаемость кода. Ускоряется цикл разработки ПО. Предоставляются дополнительные отладочные возможности, как например: контроль выхода за пределы массива и утечки памяти.

3) Скорость аллокации в рабочем режиме алгоритма близка к скорости работы при статическом распределении.

4) Ротация массивов в памяти в режиме обучения попутно выполняет функцию тестирования программы.

5) Возможна запись маршрута как в виде последовательности физических адресов, так и в виде цепочки с номерами банков памяти.

6) Упрощается интерфейс и работа с библиотечными функциями, требующих временных массивов, оптимальное размещение которых зависит от конечной программы и поэтому заранее неизвестно.

C. Недостатки

1) Алгоритм должен быть линейный. Присутствие функций-аллокаторов в разных ветках недопустимо. Также в многопоточных системах требуется изоляция кучи от внешних аллокаций.

2) Управление маршрутами и аллокаторами с вложенными функциями требует дополнительной осторожности.

3) В случае больших и сложных алгоритмов полный перебор по всем доступным банкам может занять много времени.

4) Для инициализации маршрута физических адресов требуется один полный проход цикла по маршруту с индексами банков. Соответствующее динамические выделение памяти приведет к более длительному прохождению первого цикла, чем все последующие.

V. Результаты и перспективы

Описанный механизм реализован в виде с-функций, входящих в состав библиотеки векторно-матричных функций NMPP[4], был опробован на архитектуре NMC при построении фильтра Собеля[5], показал хорошие результаты и оказался удобным в применении. В частности, замеры производительности гибридных функций аллокации-деаллокации показали ускорение более чем в 10 раз по сравнению со стандартными malloc и free. При оптимизации функции на ассемблере NMC можно ожидать ускорение еще в несколько раз. Приведенный механизм может быть использован и на других аппаратных платформах. Так на ARM-1176 выигрыш составил около 5 раз, а на PC – уже более 13. В дальнейшем проект предполагается развивать в сторону эвристических методов поиска длинных маршрутов, внедрения отладочных средств и ассемблерной оптимизации кода.

Литература

[1] URL: http://www.module.ru/catalog/micro/sfblok_yadra_processora_neuromatrix_nmc_3/ (дата обращения 01.04.2016).

[2] URL: http://www.module.ru/catalog/micro/instrumental_niy_modul__ms_513_na_baze_processora_1879vm5ya/ (дата обращения 01.04.2016).

[3] Микрокомпьютер МВ77.07 на базе отечественной СБИС К1879ХБ1Я // Компоненты и технологии № 9’2014 (in Russian). URL: http://www.module.ru/interesting/page-1/mb777_news1 (дата обращения 01.04.2016).

[4] URL: https://github.com/RC-MODULE/nmpp (дата обращения 01.04.2016).

[5] URL: https://github.com/RC-MODULE/sobel-steps (дата обращения 01.04.2016).

[6] URL: http://www.module.ru/catalog/micro/processor_1879vm5ya

[7] «Отечественная СБИС декодера цифрового телевизионного сигнала К1879ХБ1Я (PDF)», Шевченко П.А., Цифровая обработка сигналов и её применение — DSPA'2011, Москва 2011 (in Russian). URL: http://www.module.ru/upload/images/1354523271art_dspa2011_1.pdf (дата обращения 01.04.2016).

Hybrid method of memory allocation in multibank platforms based on the DSP NeuroMatrix architecture

S.V. Mushkaev, A.V. Andrianov

Moscow JSC RC “Module”, mushkaev@module.ru, andrianov@module.ru

***Keywords* — malloc, memory allocation, NeuroMatrix, NMC, system-on-a-chip, SoC, DSP.**

ABSTRACT

The peculiarities of effective software design for hardware platforms with multibank memory architecture are considered. Two target multibank DSP platforms based on NeuroMatrix® architecture designed by JSC RC “Module” are used in experiments. The hybrid method of allocation of memory from several different type memory banks is proposed. The hybrid approach allows achieving significant performance improvement, accelerating software design, simplifying interface and improving portability of programs.

REFERENCES

[1] URL: http://www.module.ru/catalog/micro/sfblok_yadra_processora_neuromatrix_nmc_3/ (accessed 01.04.2016)

[2] URL: http://www.module.ru/catalog/micro/instrumental_niy_modul__ms_513_na_baze_processora_1879vm5ya/ (accessed 01.04.2016)

[3] Mikrokomp'juter MV77.07 na baze otechestvennoj SBIS K1879HB1Ja // Komponenty i tehnologii No. 9’2014 URL: http://www.module.ru/interesting/page-1/mb777_news1 (accessed 01.04.2016)

[4] URL: https://github.com/RC-MODULE/nmpp (accessed 01.04.2016)

[5] URL: https://github.com/RC-MODULE/sobel-steps (accessed 01.04.2016)

[6] URL: http://www.module.ru/catalog/micro/processor_1879vm5ya

[7] URL: http://www.module.ru/catalog/micro/mikroshema_dekodera_cifrovogo_televizionnogo_signala_sbis_k1879hb1ya/ (accessed 01.04.2016)