УДК 621.396

ДЕМОДУЛЯЦИЯ ЧМн-СИГНАЛОВ С ПРИМЕНЕНИЕМ ГРАФИЧЕСКОГО ПРОЦЕССОРА

Д.С. Марычев, О.А. Морозов, М.М. Сорохтин

Нижегородский госуниверситет им. Н.И. Лобачевского

dsmarychev@nifti.unn.ru

Поступила в редакцию 01.02.2013

Представлен обзор существующих подходов к демодуляции ЧМн-сигналов и к повышению производительности операции дискретной свертки. Рассмотрен алгоритм, повышающий производительность дискретной свертки применительно к задаче демодуляции ЧМн-сигналов. Представлены результаты исследований производительности предложенного алгоритма, проведен сравнительный анализ с существующими параллельными реализациями дискретной свертки.

Ключевые слова: алгоритм, демодуляция, цифровая обработка сигналов, фильтрация, параллельные вычисления.

Демодуляция ЧМн-сигналов

Во многих современных системах цифровой связи, предполагающих распространение сигналов в каналах со значительными помехами, используется частотная манипуляция (ЧМн). Главным достоинством частотной манипуляции является высокая устойчивость по отношению к аддитивным шумам ввиду того, что аддитивные помехи слабо влияют на спектральный состав сигнала и точность детектирования бит приемным устройством: приемники сигналов ЧМн успешно выполняют детектирование при отношениях сигнал/шум до 2 дБ.

Методы демодуляции частотно-манипулированных сигналов можно условно разделить на три группы: корреляционные демодуляторы, непараметрические алгоритмы и селективные фильтры.

К корреляционным методам относятся когерентный и некогерентный детекторы. К недостаткам когерентного детектора можно отнести необходимость знания начальной фазы сигнала на каждом обрабатываемом участке. Практическая реализация требует алгоритмически сложного устройства определения фазы сигнала. Особенностью ее является независимость в работе от значения начальной фазы сигнала.

Демодуляция непараметрическими методами заключается в использовании какого-либо метода непараметрической оценки текущей частоты сигнала, например с использованием статистик нуль-пересечений. Эти методы значительно проще в реализации и требуют на порядки меньший объем вычислений, однако имеют более низкую устойчивость к помехам аддитивного характера.

Общий вид схемы демодуляции на основе селективных фильтров, предназначенной для модуляции с использованием двух частот, показан на рис 1. В данной схеме выполняется параллельная обработка отсчетов двумя линейными фильтрами с частотными характеристиками, синтезированными для получения максимального отклика на частоте передачи соответствующего символа. Цифровая реализация данной схемы демодуляции основана на операции дискретной свертки отсчетов сигнала с импульсной характеристикой фильтра. Алгоритм позволяет обрабатывать как отсчеты сигнала на несущей частоте, так и в виде IQ-компонент. При этом метод синтеза характеристики фильтров определяется условиями приема и соотношением частоты дискретизации и частоты передачи данных. Так, наилучшим с точки зрения помехоустойчивости является использование в демодуляторе согласованных фильтров.

Для демодуляции двухпозиционного ЧМн-сигнала, заданного в общем виде выражением

$$s_j[n] = A\exp\left(i2\pi\left(f_c + (-1)^j\frac{\Delta f}{2}\right)\frac{n}{f_s} + \varphi_c\right), \quad (1)$$
$$j = \{0,1\},$$

где A – амплитуда, f_c – несущая (промежуточная) частота, f_s – частота дискретизации, n – номер временного отсчета, φ_c – начальная фаза, Δf – девиация частоты, согласованный фильтр будет определяться следующим выражением:

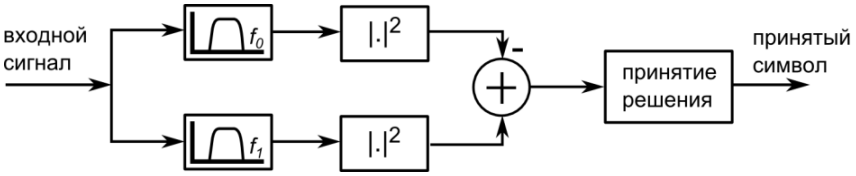

Рис. 1. Схема демодуляции на основе селективных фильтров

$$h_j[k] = A\exp\left(-i2\pi\left(f_c + (-1)^j \frac{\Delta f}{2}\right)k\right), \quad j = \{0,1\}. \qquad (2)$$

Отметим, что построение согласованного фильтра в общем случае требует полной априорной информации о сигнале, которая, как правило, недоступна вследствие действия различного рода шумов и помех в канале связи. Для решения данной проблемы вместо согласованных фильтров могут использоваться информационно-оптимальные фильтры, построенные на основе обобщения подхода Кейпона [1].

В целом с точки зрения вычислительной реализации все описанные алгоритмы используют сходный набор вычислительных операций – вычисление тригонометрических функций, линейная фильтрация, операция умножения с накоплением. Следует отметить, что эти операции являются основными при проектировании цифровых систем для высокопроизводительных вычислений. Важной особенностью перечисленных алгоритмов является также возможность параллелизации вычислений, то есть возможность разделения вычислений на нити, исполняющиеся на отдельных вычислительных устройствах.

В существующих приложениях, связанных с цифровой обработкой сигналов, широкое применение находят цифровые сигнальные процессоры (DSP). Процессоры данного класса являются достаточно узкоспециализированными, а их архитектура оптимизирована для выполнения свертки и быстрого преобразования Фурье в режиме потоковой обработки данных. С другой стороны, их применение в системах цифровой обработки сигналов предполагает разработку соответствующих устройств сопряжения с центральным процессором (CPU), что требует существенных временных и материальных затрат. В многоканальных системах цифровой обработки сигналов, предназначенных для работы в реальном масштабе времени, представляется перспективным использование графических процессоров (GPU) вместо DSP по следующим причинам: графический процессор имеет стандартный интерфейс сопряжения с CPU и достаточно низкую стоимость, присутствует в большинстве компьютерных систем и обладает высокой производительностью. Вместе с этим задачи, решаемые графическим процессором (матричное умножение, наложение текстур), в значительной степени сходны с задачами, решаемыми DSP.

Особенности вычислений общего назначения на GPU

В настоящее время существует большое число публикаций, связанных с задачей цифровой фильтрации общего назначения с использованием графического процессора. Известно два способа применения GPU для вычислений общего назначения: использование графического конвейера [2–5] и использование аппаратных расширений для вычислений общего назначения, таких как NVIDIA CUDA (Compute Unified Device Architecture) [6].

Как показывают исследования [2], использование графического конвейера GPU позволяет повысить производительность операции свертки до 50 раз по сравнению с последовательной реализацией на CPU, сравнимом по величине тактовой частоты. Существующие реализации свертки [2, 3] интенсивно используют текстуры для размещения в них отсчетов последовательностей, подлежащих свертке. Для вычислений общего назначения текстуры полезны тем, что предоставляют произвольный доступ к своим элементам, кроме того, доступ к элементам буферизуется в кэш-памяти, что повышает производительность. Вместе с тем использование графического конвейера для вычислений общего назначения накладывает достаточно жесткие ограничения на алгоритмы, которые могут быть реализованы таким способом.

Выпуск компанией NVIDIA архитектуры CUDA [6], а также соответствующего прикладного инструментария для разработки снял основное ограничение, препятствующее использованию GPU для реализации сложных алгоритмов, – невозможность взаимодействия между отдельными участками программы, выполняющимися параллельно, и предоставил удобный интерфейс для разработки неграфических программ.

Графический процессор с поддержкой CUDA содержит большое число скалярных процессорных ядер, объединенных в группы (мультипроцессоры), и способен выполнять большое количество потоков инструкций параллельно. В отличие от исполнительных эле-

ментов графического конвейера потоки CUDA поддерживают механизм барьерной синхронизации и могут обмениваться данными через память. Память CUDA-совместимого GPU устроена существенно сложнее, чем у CPU. Аппаратно поддерживаются 6 классов памяти, отличающиеся областью действия, доступным объемом и скоростью доступа: глобальная, разделяемая, память констант, память текстур, локальная память, регистры общего назначения. Модель доступа к глобальной памяти из потоков CUDA является критическим фактором, определяющим производительность программ, т.к. объем памяти этого класса на GPU самый большой, а скорость доступа самая низкая. У графического процессора практически отсутствует кэш данных, однако шина данных у него очень широкая (до 512 бит); это значит, что наибольшая производительность программы будет наблюдаться в случае, когда потоки CUDA одновременно обращаются к смежным ячейкам глобальной памяти, максимально загружая шину данных.

В противоположность глобальной памяти, скорость доступа к разделяемой памяти является самой высокой среди всех остальных за исключением регистров, однако ее объем очень мал по сравнению с объемом глобальной памяти, а область действия физически ограничена одним мультипроцессором. Архитектура CUDA предполагает объединение потоков в группы (блоки). Потоки каждого блока выполняются от начала и до конца на одном мультипроцессоре и могут совместно использовать разделяемую память.

Реализация цифрового КИХ-фильтра

Предлагаемый в данной работе подход к увеличению производительности цифрового ЧМн-демодулятора с использованием графического процессора основан на высокопроизводительной реализации линейной свертки. Применительно к задаче демодуляции ЧМн-сигналов представляется наиболее перспективным прямое вычисление свертки вместо применения алгоритма на основе быстрого преобразования Фурье (БПФ) [7]. Это обусловлено двумя факторами: во-первых, как показано ранее, демодуляция ЧМн-сигналов не требует фильтров с большим числом коэффициентов. Так, демодуляция сигнала с девиацией частоты 16 кГц, индексом модуляции 1 при частоте дискретизации 192 кГц требует построения пары фильтров с числом коэффициентов, равным 12. Вместе с тем, как показано в [7], при длине фильтра до 128 отсчетов количество операций, необходимое для непосредственного вычисления свертки, оказывается меньше, чем для быстрого алгоритма. Во-вторых, обладая специализированной архитектурой, GPU реализует инструкции, позволяющие ускорять операцию свертки по сравнению с реализацией для процессоров общего назначения (CPU). Так, современные GPU подобно процессорам DSP реализуют операцию умножения с накоплением (mad), которая является основной при непосредственном вычислении свертки.

Линейная свертка описывается следующим выражением:

$$r[n] = \sum_{m=0}^{M} h[m]s[n-m], \qquad (3)$$

где $h[m]$ – отсчеты фильтра, определенные формулой (2), $s[n]$ – отсчеты исходного сигнала, определенные формулой (1), M – число отсчетов фильтра. При числе отсчетов сигнала, равном N, максимальное число отсчетов свертки равно $N-M+1$. Принимая во внимание особенности демодуляции ЧМн-сигналов, а именно сравнительно небольшую длину фильтра и большую длину сигнала $N >> M$, можно сделать вывод, что приемлемый способ ускорения фильтрации состоит в параллельном вычислении отсчетов свертки множеством потоков по формуле (3). Вместе с этим преимущества использования GPU в этом случае также очевидны, поскольку для параллельной фильтрации сигнала, содержащего $N = 10000$ отсчетов, таким способом потребуется приблизительно такое же количество потоков. Как правило, при реализации расчетов на CPU не имеет смысла использовать число потоков, более чем вдвое превышающее число процессорных ядер; так, для процессора Intel Core i7 это число равно 16, а для Intel Xeon X5650 – 48. В то же время видеокарта NVIDIA GeForce GTS 250 аппаратно поддерживает одновременное выполнение до 12288 потоков. Общий вид схемы прямого вычисления свертки на GPU представлен на рис. 2.

Как уже упоминалось в предыдущем разделе, фактором, определяющим производительность вычислений на GPU, является модель доступа к памяти. Как видно из рис. 2, доступ к отсчетам сигнала и фильтра производится поразному. Так, на каждой итерации доступ к отсчетам сигнала производится последовательно относительно номеров потоков, а из массива отсчетов фильтра извлекается единственное значение всеми потоками. Принимая во внимание этот факт, имеет смысл использовать для хранения отсчетов фильтра тип памяти, доступ к которой кэшируется, – текстурную память. Заметим также, что на каждой следующей

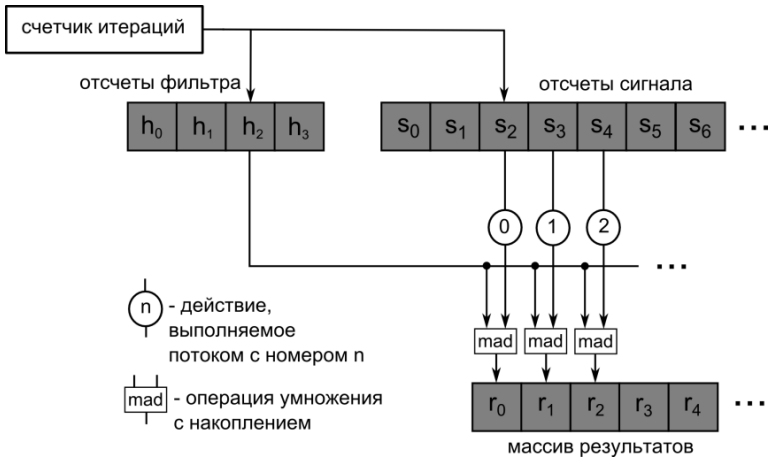

Рис. 2. Общий вид схемы прямого вычисления свертки на GPU

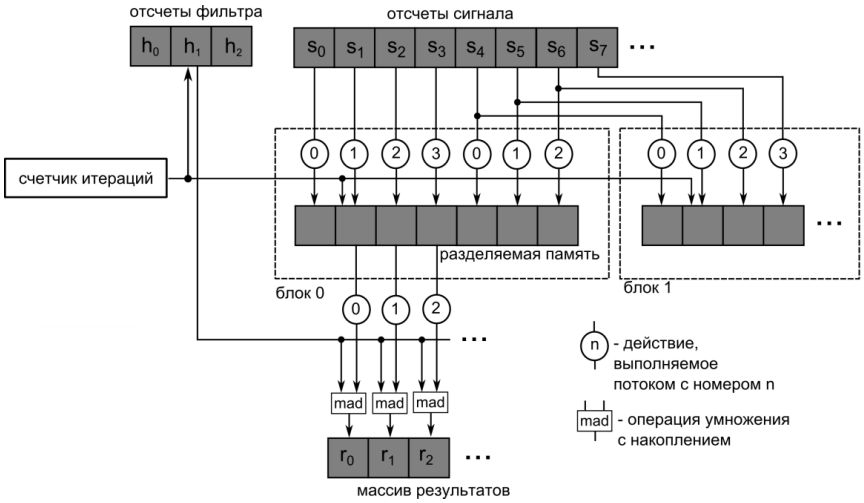

Рис. 3. Модифицированная схема прямого вычисления свертки на GPU

итерации i-й поток будет использовать отсчет сигнала, который уже был загружен потоком с номером $i-1$ на предыдущей итерации. Таким образом, имеет смысл также сохранять во временной памяти отсчеты сигнала, загруженные потоками на каждой итерации. Для хранения этих данных целесообразно использовать разделяемую память GPU. Модифицированная схема вычисления свертки представлена на рис. 3.

В представленной схеме потоки CUDA сгруппированы в блоки численностью N_b и совместно используют $N_b + M$ отсчетов сигнала, хранящихся в разделяемой памяти. Перед началом вычислений потоки производят параллельную загрузку буферов в разделяемой памяти. После того как данные загружены, каждый из потоков производит выборку отсчета фильтра из текстуры в соответствии с номером итерации и отсчета сигнала из разделяемой памяти в соответствии со значением задержки и номера итерации, вычисляет их произведение и добавляет к соответствующему элементу массива результатов. После того как в соответствии со счетчиком итераций будут выбраны все отсчеты фильтра, в массиве результатов будут находиться $N - M + 1$ значений свертки.

Результаты численного моделирования

Целью численного моделирования являлась оценка производительности параллельной реализации алгоритма вычисления линейной свертки применительно к задаче демодуляции ЧМн-сигнала. Сравнительный анализ результатов, полученных для предложенного подхода и существующих аналогов, позволил сделать вывод о возможностях его применения в практических приложениях. Для моделирования использовался ЧМн-сигнал с девиацией частоты 16 кГц, индексом модуляции 1 и частотой дискретизации 192 кГц, представленный отсчетами синфазной и квадратурной компонент. Для де-

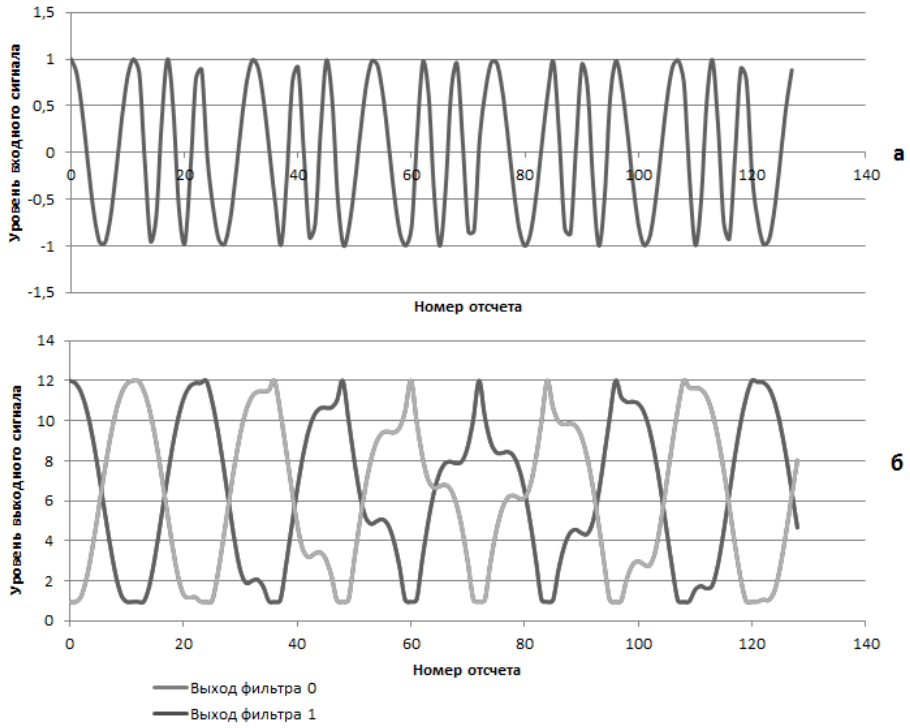

Рис. 4. Выходной сигнал демодулятора ЧМн-сигнала

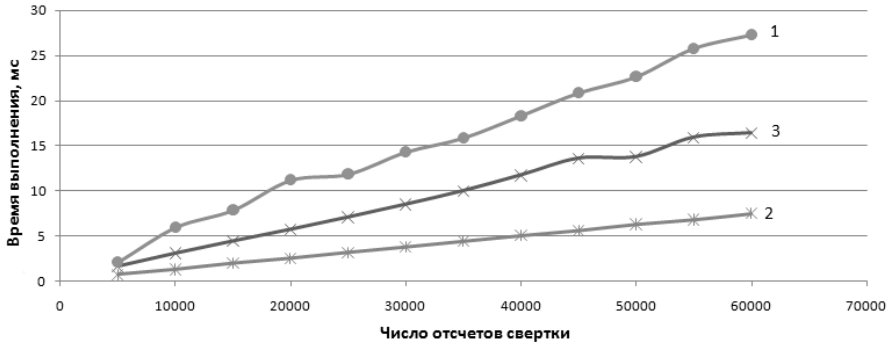

Рис. 5. Производительность алгоритмов при вычислении на CPU и GPU при различных объемах результата

модуляции применялась пара согласованных фильтров по 12 отсчетов каждый.

Результат демодуляции ЧМн-сигнала (рис. 4а) представлен на рис. 4б в виде выходных сигналов фильтров, настроенных на передачу символов 0 и 1. Отметим, что представленный результат идентичен результату, полученному при аналогичных условиях с использованием последовательной реализации ЧМн-демодулятора на CPU.

На рис. 5 представлены графики зависимости времени выполнения операции линейной свертки от длины обрабатываемого сигнала для параллельных реализаций на CPU и GPU. Для моделирования использовался GPU NVIDIA GeForce GTS 250 и CPU Intel Core i7. Все представленные в данной работе результаты получены путем усреднения 10^3 измерений.

Линия 1 на рис. 5 соответствует реализации алгоритма на GPU по схеме без кэширования данных в разделяемой памяти (рис. 2), линия 2 получена для реализации на GPU по схеме с кэшированием в разделяемой памяти (рис. 3). Линия 3 получена для параллельной реализации алгоритма прямого вычисления линейной свертки на CPU с использованием библиотеки Intel Threading Building Blocks (TBB) [**Ошибка! Источник ссылки не найден.**].

Каждый из представленных графиков характеризуется линейным ростом времени выполнения в зависимости от длины обрабатываемого сигнала, это связано с нехваткой вычислительных ядер для одновременного вычисления всего массива результатов и постановкой в очередь соответствующих заданий. Важным результатом здесь является тот факт, что реализация

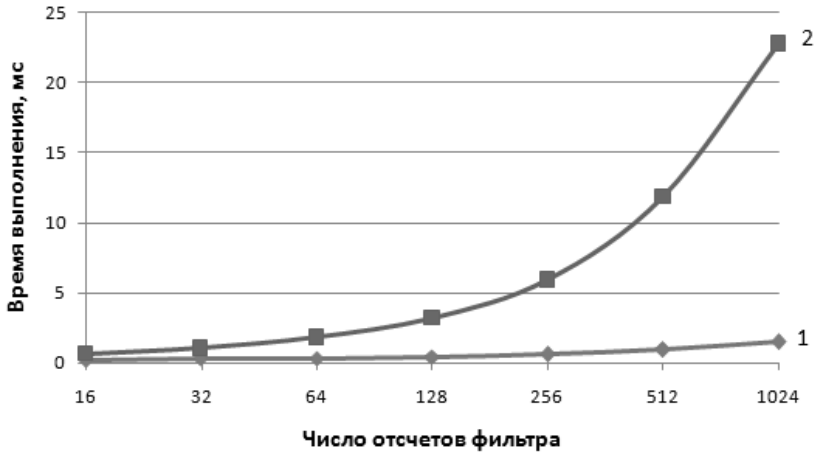

Рис. 6. Производительность алгоритмов при вычислении на CPU и GPU при различной длине фильтра

алгоритма на GPU по схеме, представленной на рис. 2 (линия 1), уступает в производительности аналогичной реализации с применением CPU примерно в 2 раза, несмотря на 16-кратное превосходство GPU над CPU по числу вычислительных ядер. Причиной является неоптимальная модель доступа к глобальной памяти GPU, отмечавшаяся ранее.

На рис. 6 представлены графики зависимости времени выполнения свертки на GPU (линия 1) и CPU (линия 2) от числа отсчетов фильтра. Число отсчетов результата было фиксировано и равно 16384. При увеличении длины фильтра наблюдается прирост производительности от 2 до 20 раз при увеличении числа отсчетов фильтра.

Заключение

В данной работе рассмотрена задача повышения производительности цифрового ЧМн-демодулятора. Основу предложенного подхода составляет высокопроизводительная реализация линейной свертки с использованием графического процессора. Особенностью рассматриваемой задачи является наличие длинных выборок сигнала для демодуляции и использование фильтров с небольшим числом отсчетов. Представлена реализация алгоритма с использованием технологии NVIDIA CUDA.

Исследования производительности алгоритма показали, что модель доступа к глобальной памяти является ключевым фактором, определяющим его производительность. Показано также, что параллельная реализация алгоритма, использующая GPU, позволяет достичь повышения производительности от 2 до 20 раз при увеличении длины фильтра по сравнению с параллельной реализацией, использующей многоядерный CPU.

Таким образом, представленная в данной работе реализация алгоритма вычисления линейной свертки позволяет добиться существенного повышения производительности демодулятора ЧМн-сигнала без применения специализированного аппаратного обеспечения.

Список литературы

1. Логинов А.А., Морозов О.А, Семенова М.Ю., Хмелев С.Л. Синтез субоптимальных цифровых фильтров на основе обобщения подхода Кейпона // Вестник Нижегородского университета им. Н.И. Лобачевского. 2008. № 2. С. 39–45.

2. Cowan B., Kapralos B. GPU-based one-dimensional convolution for real-time spatial sound generation // Loading. 2009. V. 3. № 5.

3. Cowan B., Kapralos B. Real-time GPU-based convolution: A follow-up // Proceedings of the 2009 Conference on Future Play on @ GDC Canada.

4. Wefers F., Berg J. High-performance real-time fir-filtering using fast convolution on graphics hardware // Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10), Graz, Austria, September 6–10, 2010.

5. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics [Электронный ресурс]. Режим доступа: http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter01.html

6. NVIDIA CUDA ZONE [Электронный ресурс]. Режим доступа: http://www.nvidia.ru/object/cuda_home_new_ru.html

7. Айфичер Э.У., Джервис Б.У. Цифровая обработка сигналов: практический подход: Пер. с англ. 2-е изд. М.: Издательский дом «Вильямс», 2004. 992 с.

8. Intel® Threading Building Blocks [Электронный ресурс]. Режим доступа: http://threadingbuildingblocks.org/

9. Madisetti V.K., Williams D.B. The digital signal processing Handbook. CRC Press, 2001.

FSK SIGNAL DEMODULATION ON A GPU

D.S. Marychev, O.A. Morozov, M.M. Sorokhtin

An overview of existing approaches to FSK signal demodulation and to improving discrete convolution operation efficiency is given. An algorithm is presented which increases the discrete convolution operation efficiency as applied to the problem of the FSK signal demodulation. The results on the proposed algorithm performance and its comparison with existing parallel implementations of discrete convolution are presented.

Keywords: algorithm, demodulation, digital signal processing, filtering, parallel computing.