ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 621.391

Е. П. Овсянников, С. Е. Петров, К. В. Юрков

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СЛОЖНОСТИ РЕАЛИЗАЦИИ БЫСТРЫХ ЦИФРОВЫХ ПРЕОБРАЗОВАНИЙ НА RISC-ПРОЦЕССОРАХ

Рассматриваются некоторые известные алгоритмы быстрого преобразования Фурье, перенесенные на RISC-платформу. Показано, что оптимизированный алгоритм radix-2 преобразования Фурье обладает наименьшей вычислительной сложностью, измеренной в процессорных тактах.

Ключевые слова: *быстрое преобразование Фурье, RISC-платформа, сложность алгоритма.*

Введение. Дискретное преобразование Фурье (ДПФ) — одна из основных операций при цифровой обработке сигналов и в алгоритмах сжатия видео- и аудиоданных. Широко известные методы быстрого преобразования Фурье (БПФ) позволяют существенно снизить сложность вычисления ДПФ. Для задач, связанных с обработкой звука, как правило, используются ДПФ длиной от 64 до 16 384. Цель настоящей статьи — сравнительный анализ вычислительной сложности известных методов БПФ применительно к RISC-платформам.

Исторически сложилось так, что для оценивания сложности различных реализаций БПФ используются так называемые флопы — операции над вещественными числами. Сложность БПФ для последовательности длиной $N = 2^m$ пропорциональна $N \log_2 N$ флопам [1—3]. Разные алгоритмы имеют разные коэффициенты пропорциональности. Самыми быстрыми из известных методов являются алгоритм split-radix БПФ [2], а также его более поздняя модификация [3], число флопов в которой уменьшено приблизительно на 3 % за счет усложнения алгоритма.

С развитием вычислительной техники приобрел актуальность вопрос, насколько корректной в настоящее время является оценка сложности, измеренная во флопах. При появлении первых алгоритмов БПФ выполнение одной операции с вещественными числами занимало в десятки, а то и в сотни раз больше процессорного времени, чем операция с целочисленными данными. Разработка сопроцессоров, выполняющих операции с плавающей точкой за один цикл, привела к тому, что при оценивании сложности алгоритма следует учитывать и такие факторы, как затраты на реализацию циклов и ветвлений, а также время доступа к памяти и т.д.

Особо следует отметить проблему, связанную с переносом алгоритмов на RISC-платформы. Благодаря низкой стоимости, малому энергопотреблению и способности оперировать с 32-разрядными числами они являются основными процессорами в таких портативных устройствах, как мобильные телефоны, коммуникаторы и карманные компьютеры. Перенос алгоритмов на RISC-процессоры подразумевает переход к целочисленным операциям с фиксированной точкой, поскольку эмуляция операций с плавающей точкой приводит к катастрофическому замедлению вычислений. В этих условиях правомерность оценивания сложности во флопах нецелесообразна.

В настоящей статье рассматриваются результаты сравнительного анализа сложности реализации различных алгоритмов БПФ на RISC-процессоре ARM9E. Естественно, для других процессоров статистика может быть иной, но можно ожидать, что общие соотношения сложности алгоритмов сохранятся для всего семейства ARM, а также для близкого к ним по архитектуре семейства MARVEL PXA27x и PXA3xx.

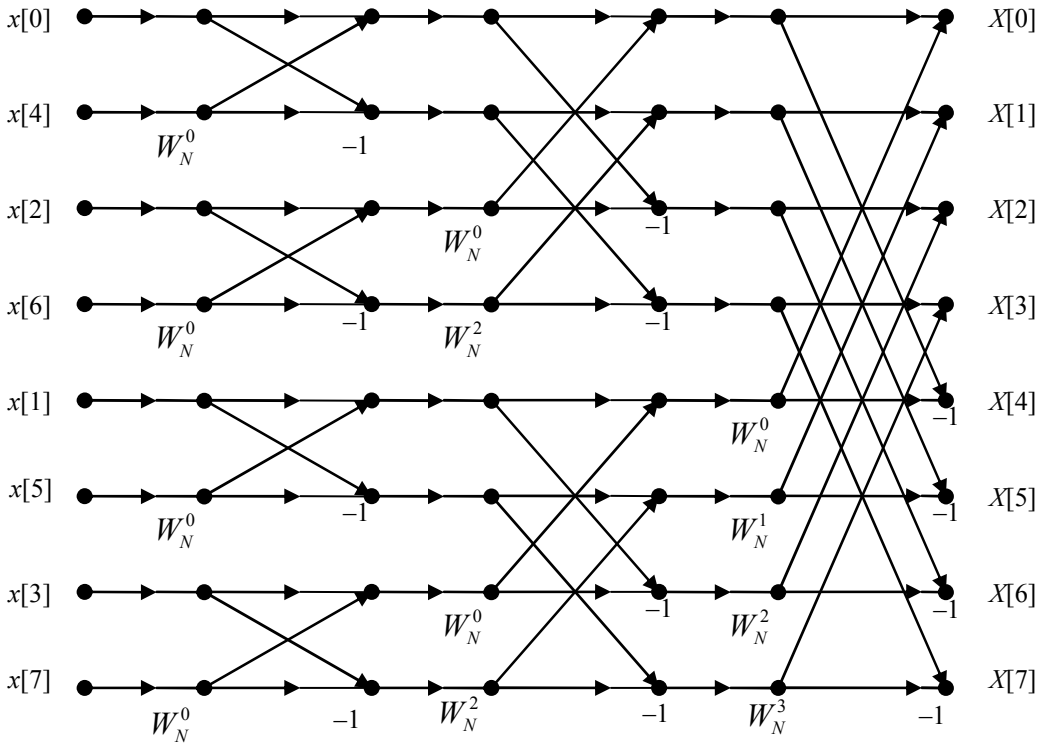
Условия сравнения алгоритмов БПФ. Выберем для сравнения три алгоритма БПФ:

1 — алгоритм radix-2 БПФ с прореживанием по времени [1];

2 — алгоритм split-radix [2];

3 — алгоритм Кули — Тьюки (Cooley — Tukey) radix-N [4].

Алгоритм 1 реализует классическую схему вычислений, пример которой для восьми точек представлен на рисунке. Множители W_n^m являются комплексными числами.



Алгоритм 2 — нерекурсивная реализация алгоритма, первоначально предложенного в работе [2] в виде рекурсивной функции `splitfft`:

$$\texttt{function } y_{k=0..N-1} \leftarrow \texttt{splitfft}_N\,(x_n)\,\texttt{:}$$

$$u_{k_2=0..N/2-1} \leftarrow \texttt{splitfft}_{N/2}\left(x_{2n_2}\right)$$

$$z_{k_4=0..N/4-1} \leftarrow \texttt{splitfft}_{N/4}\left(x_{4n_4+1}\right)$$

$$z'_{k_4=0..N/4-1} \leftarrow \texttt{splitfft}_{N/4}\left(x_{4n_4-1}\right)$$

`for k=0 to N/4 – 1 do`

$$y_k \leftarrow u_k + \left(\omega_N^k z_k + \omega_N^{-k} z'_k\right)$$

$$y_{k+N/2} \leftarrow u_k + \left(\omega_N^k z_k + \omega_N^{-k} z'_k\right)$$

$$y_{k+N/4} \leftarrow u_{k+N/4} - i\left(\omega_N^k z_k + \omega_N^{-k} z'_k\right)$$

$$y_{k+3N/4} \leftarrow u_{k+N/4} + i\left(\omega_N^k z_k + \omega_N^{-k} z'_k\right)$$

`end for`

где $\omega_N = e^{\frac{-2\pi i}{N}}$ — поворачивающий множитель, $x_{-k} = x_{N-k}$ — способ пересчета отрицательных индексов; в теле функции использованы следующие обозначения индексов: $2n_2$ — все четные индексы, $4n_4$ — все индексы, кратные 4.

Как показал сравнительный анализ, рекурсивная форма алгоритма split-radix существенно превосходит нерекурсивную форму по быстродействию, притом что количество вещественных операций у них одинаково. Поэтому в дальнейшем будем рассматривать именно нерекурсивную реализацию алгоритма.

Алгоритм 3 является реализацией общей идеи Кули — Тьюки о том, что если длина L преобразования может быть представлена в виде произведения $L = MN$, то вычисление ДПФ может быть сведено к M преобразованиям длиной N и N преобразованиям длиной M [1, 4]. Суть метода заключается в том, что последовательность представляется в виде прямоугольной матрицы M на N. К каждому столбцу применяется короткое преобразование Фурье длиной N. Затем элементы матрицы умножаются на соответствующие поворачивающие множители, и короткое преобразование Фурье длиной M применяется к каждой строке. В качестве короткого преобразования Фурье используется алгоритм split-radix. По количеству вещественных операций алгоритм 3 должен проигрывать алгоритму 2, но его преимуществом является то, что циклы коротких преобразований могут быть расписаны в виде линейного алгоритма для фиксированного набора длин. В этом случае ожидается выигрыш по быстродействию, во-первых, за счет отказа от организации циклов и, во-вторых, за счет того, что параметры алгоритма будут вычислены на этапе компиляции.

Поскольку в статье рассматриваются преобразования длиной от 64 до 16 384, то для реализации алгоритма 3 потребуется набор коротких преобразований длиной 8, 16, 32, 64 и 128.

Все тестируемые алгоритмы были реализованы в виде вычислений с фиксированной точкой. Точность представления данных и тригонометрических констант была выбрана таким образом, чтобы результат обратного преобразования целочисленных алгоритмов отличался от результата обратного преобразования вещественных алгоритмов не более чем по двум младшим знакам.

В качестве меры сложности тестируемых алгоритмов выбрано количество тактов процессора, затраченных для вычисления преобразования, при условии, что доступ к памяти требует 0 тактов.

Все тестируемые алгоритмы реализованы на языке С и оптимизированы по быстродействию для процессора ARM9E. Оптимизация представляла собой итеративную процедуру, при которой фрагменты функций переписывались, после чего анализировался построенный ассемблерный код. Основная стратегия состояла в том, чтобы минимизировать число переменных, одновременно вовлеченных в вычисления. Это позволяло компилятору расположить данные в регистрах, что обеспечивало наиболее быстрый доступ.

Анализ результатов. Оценим сложность тестируемых алгоритмов. В табл. 1 приведено общее число вещественных операций N_{total}, а также число сложений N_{add} и умножений N_{mul} в зависимости от длины L преобразования. Очевидно, что алгоритм 2 (split-radix БПФ) имеет существенно меньшую сложность, чем алгоритм 1 (radix-2 БПФ), выигрывая до 13 % по общему числу операций и до 30 % по числу умножений. Последнее обстоятельство особенно важно, поскольку, как правило, именно операция умножения является наиболее медленной в RISC-процессорах. Алгоритм 3 (radix-N БПФ) уступает менее 1 % алгоритму 2, при этом предполагается, что вычисления коротких БПФ для него будут выполнены в виде

линейного алгоритма с заранее вычисленными его параметрами. Таким образом, алгоритм 3 можно априори считать наиболее предпочтительным среди тестируемых алгоритмов БПФ.

Таблица 1

L	Алгоритм 1			Алгоритм 2			Алгоритм 3		
	N_{add}	N_{mul}	N_{total}	N_{add}	N_{mul}	N_{total}	N_{add}	N_{mul}	N_{total}
64	844	344	1188	912	248	1160	930	260	1190
128	2060	920	2980	2164	660	2824	2194	676	2870
256	4876	2328	7204	5008	1656	6664	5058	1668	6726
512	11276	5656	16932	11380	3988	15368	11490	3972	15462
1024	25612	13336	38968	25488	9336	34824	25730	9220	34950
2048	57356	30744	88100	56436	21396	77832	56898	21124	78022
4096	126988	69656	196644	123792	48248	172040	124674	47620	172294
8192	278540	155672	434212	269428	107412	376840	271234	105988	377222
16384	606220	344088	950308	582544	236664	819208	586242	233476	819718

После адаптации к математическим операциям с фиксированной точкой и оптимизации по быстродействию сложность алгоритмов была исследована для вычислительного комплекса на базе RISC-процессора ARM9E. Результаты оценивания сложности алгоритмов, вычисленной в процессорных тактах, представлены в табл. 2.

Таблица 2

L	Алгоритм 1	Алгоритм 2	Алгоритм 3
64	6107	7954	7089
128	14393	18070	16272
256	33334	41232	36752
512	76020	92433	82256
1024	171058	205911	182128
2048	380528	453093	396400
4096	838318	990790	857200
8192	1831660	2148765	1866608
16384	3973930	4635694	4037744

Как оказалось, вопреки априорным представлениям о сложности, наилучшим быстродействием обладает алгоритм 1 (radix-2 БПФ), при реализации которого требуется на 14 % меньше процессорных тактов, чем при реализации алгоритма 2 (split-radix БПФ). Это объясняется тем, что radix-2 БПФ имеет существенно более регулярную структуру, чем split-radix БПФ. Несмотря на то, что операций умножения выполняется намного больше, загрузка процессора данными осуществляется более эффективно. Ниже приведено отношение среднего числа процессорных тактов к числу вещественных операций для $L = 16\,384$.

Алгоритм 1	Алгоритм 2	Алгоритм 3
4,1	5,6	4,9

Сравнение алгоритмов 2 и 3 демонстрирует предсказуемое преимущество последнего за счет того, что при вычислении коротких БПФ, реализованных в виде линейных алгоритмов, на организацию циклов и вычисление параметров алгоритмов процессорное время не затрачивается.

Поскольку основной областью применения RISC-процессоров являются мобильные устройства, то наряду с быстродействием алгоритма важной характеристикой является размер исполняемого кода. Данные о длине кода преобразования (в килобайтах) для процессора ARM9E приведены ниже.

Алгоритм 1	Алгоритм 2	Алгоритм 3
1,9	5,2	64,5

Таким образом, можно утверждать, что наиболее пригодным для реализации на RISC-платформе является традиционный алгоритм radix-2 БПФ, который обеспечивает наилучшее быстродействие при наименьшей длине кода.

СПИСОК ЛИТЕРАТУРЫ

1. *Cooley J. W., Tukey J. W.* An algorithm for the machine computation of the complex Fourier series // Math. Computation. 1965. Vol. 19. P.297—301.

2. *Yavne R.* An economical method for calculating the discrete Fourier transform // Proc. AFIPS Fall Joint Computer Conf. 1968. Vol. 33. P. 115—125.

3. *Johnson S. G., Frigo M.* A modified split-radix FFT with fewer arithmetic operations // IEEE Trans. Signal Processing. 2007. Vol. 55. P. 111—119.

4. *Blahut R. E.* Fast Algorithms for Digital Signal Processing. Reading, MA: Addison Wesley, 1985.

Сведения об авторах

Евгений Порфирьевич Овсянников — канд. техн. наук, доцент; Санкт-Петербургский государственный университет авиационного приборостроения, кафедра информационных систем; E-mail: eovs@mail.ru

Сергей Евгеньевич Петров — Санкт-Петербургский государственный университет информационных технологий, механики и оптики, НИИ наукоемких компьютерных технологий; мл. науч. сотрудник; E-mail: petrovse@mail.ru

Кирилл Валерьевич Юрков — канд. техн. наук; ЗАО „Интел“, Санкт-Петербург; науч. сотрудник; E-mail: yourkovkirill@mail.ru

Рекомендована кафедрой информационных систем СПбГУАП

Поступила в редакцию 28.09.10 г.