Программа быстрого преобразования Фурье для устройств автоматизации на базе процессора TMS320

Александр Агапиев, Виктор Милашенко

Сегодня большинство современных устройств автоматизации, способных обрабатывать сигналы различной природы в режиме реального времени, разрабатываются на базе высокоскоростных процессоров цифровой обработки сигналов (ЦОС). При этом преобразование Фурье является важным инструментом цифрового спектрального анализа, который наиболее часто используется в системах ЦОС. В этом случае для систем реального времени (СРВ) становится актуальной проблема разработки максимально эффективного программного кода данного алгоритма, требующего минимального времени выполнения при фиксированном объеме памяти. В общем случае универсальная программа для любых приложений СРВ вряд ли существует, и основная цель разработчика — решить задачу оптимального выбора вида программирования и основания алгоритма в зависимости от длины преобразования, доступного размера памяти и требуемого времени выполнения.

Как известно, элементарная процедура быстрого преобразования Фурье (БПФ) по произвольному основанию состоит в многократном выполнении базовой операции «бабочка» над разными частями входных данных. При этом данная операция может быть оформлена как макрорасширение или как подпрограмма. С одной стороны, необходимость расчета текущих адресов данных и поворачивающих множителей увеличивает время выполнения всего алгоритма, что делает целесообразным оформить процедуру «бабочка» как макрорасширение. При таком программировании однотипные процедуры с использованием непосредственной адресации записываются друг за другом, как этого требует алгоритм преобразования, образуя линейную структуру (линейное программирование). С другой стороны, использование макрорасширений неизбежно приводит к значительному увеличению требуемого объема памяти для хранения программного кода.

Использование же подпрограмм позволяет получить компактный программный код, размер которого не зависит от длины преобразования. В этом случае текущие адреса операндов и коэффициенты преобразования рассчитываются на этапе работы программы, что приводит к некоторому увеличению времени реализации всего алгоритма. При комбинированном варианте программирования целесообразным является расчет поворачивающих множителей на этапе инициализации или определение их константами в доступной памяти данных.

Во врезке статьи приведен листинг программной реализации алгоритма БПФ длиной 512 точек для процессора TMS320C25. В данном случае базовая операция оформлена как подпрограмма, а коэффициенты преобразования определены константами в виде четырех выделенных блоков данных (листинг, метка MET1:). Для оценки эффективности разработанной программы в таблице 1 представлены требуемый объем памяти и время выполнения программы при различных длинах преобразования и видах программирования.

Из таблицы 1 видно, что если длина преобразования БПФ не превышает 128 точек, использование макрорасширений позволяет получить некоторый выигрыш за счет меньшего времени выполнения. Однако при длинах преобразования, превышающих 256 точек, линейное программирование становится неэффективным, а в некоторых приложениях даже неосуществимым из-за больших объемов требуемой памяти. Что касается разработанной программы, то для неё затраты памяти в 50 раз меньше, чем при использовании линейного программирования. При этом время выполнения разработанной программы увеличилось лишь в 1,6 раза при длине преобразования 512 и всего в 1,25 раза при длине преобразования 1024. Рост требуемого объема памяти от длины преобразования для случая программирования с подпрограммами обусловлен только увеличением количества хранимых поворачивающих множителей. Необходимо также отметить, что подавляющее большинство существующих стандартных библиотек функций, предлагаемых фирмой-разработчиком Texas Instruments и имеющих в своем составе процедуру БПФ, основываются именно на программировании с использованием макрорасширений.

Продолжая разговор об эффективной реализации алгоритма БПФ, необходимо отметить ряд важных особенностей, связанных со структурой алгоритма и комплексным характером преобразования. Известно, что в случае, когда длина дискретного преобразования Фурье (ДПФ) $N = n_1 n_2 ... n_k ... n_K$, в частном случае $N = n^K$ (именно такое ДПФ называют быстрым преобразованием Фурье по основанию n), возможно значительное сокращение объема вычислений ДПФ, а сам алгоритм распадается на две части: выполнение Kn^{K-1} операций n-точечного ДПФ («бабочка») и выполнение операции перестановки, так как порядок входных и выходных индексов меняется. Последовательность выполнения операций может быть произвольной. При этом, если выполнить вначале операцию «бабочка», то выходные отсчеты БПФ будут располагаться в поразрядно-обратном порядке при представлении индексов в системе счисления по основанию n, то есть входному отсчету с порядковым номером $i_{вх} = p_m n^{m-1} + p_{m-1} n^{m-2} + ... + p_2 n^1 + p_1 n^0$ будет соответствовать выходной отсчет $i_{вых} = p_1 n^{m-1} + p_2 n^{m-2} + ... + p_{m-1} n^1 + p_m n^0$.

Кроме конвейерной обработки команд процессором, возможности аппаратного умножения 16-битовых слов (16x16) или выполнения операции свертки (A*B+C) за один такт,

Таблица 1. Объем памяти и время выполнения программы БПФ при различных длинах преобразования и видах программирования

Длина преобразования	Время выполнения, включая команду TBLR, мкс	Чистое время преобразования, мкс	Длина программы, включая таблицы коэффициентов, 16-разрядных слов	Вид программирования
8	28,8	28,8	264	Линейное
8	317,6	317,0	65	Подпрограммы
16	76,8	76,8	704	Линейное
16	466,7	465,9	81	Подпрограммы
32	192,0	192,0	1760	Линейное
32	670,0	668,5	113	Подпрограммы
64	460,8	460,8	4224	Линейное
64	1176,1	1169,7	177	Подпрограммы
128	1075,2	1075,2	9856	Линейное
128	2194,4	2168,8	305	Подпрограммы
256	2457,6	2457,6	22526	Линейное
256	4268,5	4217,3	561	Подпрограммы
512	5529,6	5529,6	50688	Линейное
512	9183,4	8974,8	1073	Подпрограммы
1024	12158,9	12158,9	112640	Линейное
1024	15253,7	15048,9	2096	Подпрограммы

TMS320C25 располагает дополнительными способами адресации, которые обеспечивают быстрое выполнение операции перестановки для алгоритма БПФ по основанию 2.

Имеющаяся в TMS320C25 возможность использования обратного распространения переноса позволяет переставлять входные или выходные данные одновременно с выполнением процедуры ввода-вывода данных. Этот режим адресации является частью косвенной адресации, выполняемой с помощью дополнительных регистров и связанного с ними арифметического устройства. В таком режиме (режим формирования индексного адреса) значение (индекс), находящееся во вспомогательном регистре AR0, или складывается, или вычитается из дополнительного регистра, на который указывает ARP (указатель вспомогательного регистра). Однако разряд переноса распространяется не в прямом, а в обратном направлении, в результа-

те чего выполняется перестановка имеющихся адресов (листинг, MET2:).

Процедура создания зеркально-инверсной адресной последовательности заключается в загрузке AR0 значением, которое соответствует половине длины обрабатываемого блока (листинг, MET3:) и загрузке другого дополнительного регистра (например AR3) базовым адресом области данных (листинг, MET4:).

Кроме того, выполнение БПФ включает в себя операции с комплексной арифметикой, поэтому для представления каждого отсчета данных требуются две ячейки памяти (для действительной и мнимой части). Таким образом, отчеты хранятся в памяти парами, при этом действительная часть находится в ячейке с четным адресом, а мнимая — в ячейке с нечетным адресом. Это значит, что смещение относительно базового адреса для любого отчета равно удвоенному значению индекса отчета. Чтобы действительные входные данные передавались в память данных и хранились там в переставленном виде вместе с ячейками памяти данных, представляющих мнимую часть, необходимо загрузить регистр AR0 значением, равным длине обрабатываемого блока (листинг, MET3:).

Необходимо также затронуть вопрос об оптимальной конфигурации памяти процессора для выполнения программы. При одном и том же программном коде алгоритм может выполняться в течение разного количества тактов процессора в зависимости от адресов расположения программы и данных в доступной памяти. Каждая команда принадлежит одному из 15 классов, которые определяют количество тактов, занимаемых командами. В таблице 2 приведены данные о первых пяти классах команд.

Большинство команд процессора TMS320C25 принадлежит первым четырем классам. Как видно из таблицы 2, эти команды выполняются за 1 такт работы процессора, если данные, к которым они обращаются, расположены во внутренней памяти данных. Если данные размещены во внешней памяти, то команды, принадлежащие первым трем классам, выполняются за 2 такта. К сожалению, в процессоре TMS320C25 на кристалле размещены только три блока данных B0, B1 и B2 размером 256, 256 и 32 16–разрядных слов соответственно. Для коротких длин преобразования (до 256 точек) данного размера внутренней памяти достаточно. В случаях же большего количества точек преобразования данные будут располагаться во внешней памяти, что приводит к дополнительному увеличению времени выполнения программы. В таблице 1 представлено время выполнения программы для случая, когда данные располагаются во внешней памяти для всех длин БПФ.

В этом смысле цифровой процессор обработки сигналов TMS320C50 выгодно отличается от своего предшественника размером памяти данных на кристалле (10 килослов). Представленная программа может быть использована на данном процессоре с учетом незначительных поправок, учитывающих его структуру и карту памяти.

Отдельно необходимо сказать о третьем классе команд. Если команды данного класса размещены во внутренней памяти программ, то они выполняются за 1 такт работы процессора даже при обращении к данным, которые расположены во внешней памяти. Как уже было отмечено, это становится полезным, если длина преобразования превышает 256 точек. В процессоре TMS320C25 можно переопределить блок B0 как память программ на кристалле, пе-

Таблица 2. Количество тактов для выполнения команд разных классов

Класс команд	Количество тактов					
	PI/DI	PI/DE	PE/DI	PE/DE	PR/DI	PR/DE
1	1	2+d	1+p	2+d+p	1	2+d
2	1	2+d	1+p	2+d+p	1	2+d
3	1	1+d	1+p	2+d+p	1	1+d
4	1	1	1+p	1+p	1	1
5	2	2	2+2p	2+2p	2	2

Условные обозначения:
PI — команда выполняется из внутренней памяти программ (RAM);
PR — команда выполняется из внутренней памяти программ (ROM);
PE — команда выполняется из внешней памяти программ;
DI — команда выполняется, используя внутреннюю память данных;
DE — команда выполняется, используя внешнюю память данных;
d — время, требуемое для доступа к внешней памяти данных;
p — время, требуемое для доступа к внешней памяти программ.

реместив предварительно в данный блок программный код алгоритма БПФ. Данный механизм использован авторами в описываемой программе (листинг, MET5:).

Необходимо отметить, что размер блока B0 ограничен, и при использовании линейного программирования обратиться к переопределению блока даже для коротких преобразований БПФ будет невозможно. А разработанная подпрограмма БПФ имеет фиксированный размер и состоит из 86 слов, что позволяет использовать блок B0 как память программ для любых длин преобразования.

В заключение необходимо сказать несколько слов о недостатках разработанной программы. Они связаны с выполнением процедуры БПФ в арифметике с фиксированной точкой. В этом случае ощутимо проявляются эффекты квантования в БПФ, и ошибка округления существенно влияет на конечную точность преобразования. При 16-битовом смещении таблицы косинусов и синусов удается представить поворачивающие множители с точностью только до четвертого знака после запятой. Для исключения эффектов арифметического переполнения аккумулятора при выполнении базовой операции «бабочка» необходимо ограничить динамический диапазон входного сигнала. Для разработанной программы он составляет 60 дБ. Авторы программы сознательно не стали переходить на 32- или 24-разрядный формат с плавающей точкой, так как это приводит к значительному увеличению времени выполнения и объема занимаемой памяти. Однако, если в конкретном приложении требуется высокая точность преобразования и разработчик не ограничен временем выполнения программы, такой переход может быть оправдан.

Листинг описанной в статье программы с блоками констант (P0-P3) можно загрузить с Web-сайта www.cta.ru.

Авторы работают в Военном институте правительственной связи
302034, г. Орел, ВИПС
Телефон: (086) 41-9933 E-mail: bor@vips.icn.gov.ru

Листинг программной реализации алгоритма БПФ для процессора TMS320C25

```
.title " 512 point Radix-2, Complex FFT "
                .globl FFT                                  ; Глобальные метки
                .globl BAB,BPER,LOOP,KL                     ; * * * * * * * * * * * * *
  DATA          .set  0400h
  PM            .set  0800h
  TIKET         .set  0060h
  TIKETEND      .set  0068h
  PR            .set  0070h
  RL            .set  0071h
  MN            .set  0072h
  DWR           .set  0073h
  DWI           .set  0074h
  LOADDATA      .set  0075h
  AD_BEG .set  0076h
                .text
  RESET         B     BEG

  ************************************************
  *      Таблица векторов прерывания             *
  ************************************************

                .space  (18h-($-RESET))*10h

  ************************************************
  *      Таблица векторов прерывания             *
  ************************************************

                .space  (100h-($-RESET))*10h
  COEFF         .label C_START
                .word 255,127,63,31,15,7,3,1,0          ; Таблица значений счетчика циклов
  COEFFE        .label C_END
  COEFFL        .equ COEFFE-COEFF
  BEG           DINT                                    ; Запрет прерывания
                LDPK    0000h                           ; Страница памяти данных - 0
                ZAC
                SACL    00,0                            ; Обнуление регистров приема/передачи
                SACL    01,0                            ; последовательного порта
                LACK    0000h                           ; Маскирование всех возможных прерываний
                SACL    04,0                            ; * * * * * * * * * * * * * * * * *
                ROVM                                    ; Инициализация флагов арифметических
                SSXM                                    ; операций
                SPM     0                               ; * * * * * * * * * * * * * * * * *
                LRLK    AR0,DATA                        ; Размещение адреса первой ячейки памяти данных
                SAR     AR0,LOADDATA                    ; в LOADDATA
  *-----------------------------------------------
```

```
                LALK    C_START                         ; Размещение таблицы значений счетчика
                LARP    AR1                             ; циклов с 60h ячейки памяти данных
                LRLK    AR1,TIKET                       ; * * * * * * * * * * * * * * * * * *
                RPTK    COEFFL-1                        ; * * * * * * * * * * * * * * * * * *
                TBLR    *+                              ; * * * * * * * * * * * * * * * * * *
*-----------------------------------------------
MET1:           LALK    PM0                             ; Перемещение блока PM0 из памяти
                LARP    AR1                             ; программ в память данных по адресу
                LRLK    AR1,PM                          ; 0800h
                RPTK    255
                TBLR    *+
                LALK    PM1                             ; Перемещение блока PM1
                LARP    AR1
                RPTK    255
                TBLR    *+
                LALK    PM2                             ; Перемещение блока PM2
                LARP    AR1
                RPTK    255
                TBLR    *+
                LALK    PM3                             ; Перемещение блока PM3
                LARP    AR1
                RPTK    255
                TBLR    *+

***********************************************
*      Размещение входных данных - 1000h-1200h  *
***********************************************

                CALL    FFT                             ; Вызов процедуры FFT
                RET                                     ; Конец

***********************************************
*                                FUNCTIONS      *
***********************************************

FFT:            LARP    AR1                             ; Перемещение подпрограммы БПФ
                LRLK    AR1,200h                        ; на кристалл (блок B0)
                RPTK    PROGL-1                         ; * * * * * * * * * * * * * * * * *
MET5:           BLKP    P_START,*+                      ; * * * * * * * * * * * * * * * * *
                CNFP                                    ; Определение блока B0 как памяти программ
                CALL    PROG                            ; Вызов процедуры PROG
                CNFD                                    ; Определение блока B0 как памяти данных
                RET
******* FAST FOURIER TRANSFORMATION *************
                .asect "on-chip",0FF00h
PROG            .label P_START
                LARP    AR1                             ; Подготовка памяти для расчета БПФ
                ZAC                                     ;
                LRLK    AR1,DATA+1                      ; Обнуление каждой нечетной ячейки
                LARK    AR0,0002h                       ; памяти данных (обнуление мнимой
                RPTK    0FFh                            ; части, так как входной сигнал
                SACL    *0+                             ; действительный)
                RPTK    0FFh                            ; * * * * * * * * * * * * * * * *
                SACL    *0+                             ; * * * * * * * * * * * * * * * *
***********************************************
                LARP    AR3                     ; Перемещение входных данных
MET3:           LRLK    AR0,200h                ; в ячейку с адресом 0400h
MET4:           LRLK    AR3,DATA                ; с использованием обратного
                RPTK    255                     ; распространения переноса
MET2:           BLKD    1000h,*br0+             ; (перестановка входных данных)
                RPTK    255
                BLKD    1100h,*br0+
***********************************************
                LARK    AR0,2                   ; Инициализация вспомогательных
                LARK    AR2,TIKET               ; регистров начальными значениями
                LARK    AR4,TIKETEND            ; счетчиков циклов
                LRLK    AR6,PM                  ; Указатель на начало блока коэфф.
                LARK    AR7,8                   ; * * * * * * * * * * * * * * * *
                LARP    AR2                     ; Активный регистр AR2
*-----------------------------------------------
LOOP:           LAR     AR1,LOADDATA            ; Указатель на начало блока данных
                SAR     AR1,AD_BEG              ; Сохранение текущего указателя
                LAR     AR3,*,AR4               ; Инициализация счетчика циклов 1
                LAR     AR5,*-,AR6              ; Инициализация счетчика циклов 2
                ZALH    *+                      ; Загрузка мнимой и реальной частей
                SACH    RL                      ; активного поворачивающего множителя
                ZALH    *+,AR1                  ; из таблицы коэффициентов
                SACH    MN                      ; * * * * * * * * * * * * * * * *
*----------------BUTTERFLY-----------------------
BAB:            MAR     *0+                     ; Инкремент AR1 на значение AR0
                LT      *+                      ; Re2
                MPY     RL                      ; Re2 * RL
                LTP     *-                      ; Im2
                MPY     MN                      ; Im2 * MN
                MPYA    RL                      ; Im2 * RL, Re2 * RL - Im2 * MN
                SACH    DWR,1                   ; Промежуточный результат в DWR
                LTP     *0-                     ; Re2
                MPY     MN                      ; Re2 * MN
                SPAC                            ; Re2 * MN + Im2 * RL
                SACH    DWI,1                   ; Промежуточный результат в DWI
*-----------------------------------------------
                LAC     *0+,0                   ; Re1
                ADD     DWR,0                   ; Re1 - DWR
                SACL    *0-,0                   ; Промежуточный результат в Re2
                LAC     *,0                     ; Re1
                SUB     DWR,0                   ; Re1 + DWR
                SACL    *+,0                    ; Промежуточный результат в Re1
*-----------------------------------------------
```

```
                LAC     *0+,0                   ; Im1
                ADD     DWI,0                   ; Im1 - DWI
                SACL    *0-,0                   ; Промежуточный результат в Im2
                LAC     *,0                     ; Im2
                SUB     DWI                     ; Im1 + DWI
                SACL    *-,0                    ; Промежуточный результат в Im1
*---------------END OF BUTTERFLY-----------------
                MAR     *0+                     ; Проверка значений счетчика циклов
                MAR     *0+,AR3                 ; * * * * * * * * * * * * * * * * *
                BANZ    BAB,AR1                 ; * * * * * * * * * * * * * * * * *
                LARP    AR5                     ; * * * * * * * * * * * * * * * * *
                BANZ    BPER,AR6                ; * * * * * * * * * * * * * * * * *
*-----------------------------------------------
                LARP    AR0                     ; Активный регистр AR0
                MAR     *0+,AR2                 ; Проверка значений счетчика циклов
                MAR     *+,AR7                  ; если = 0 выход, иначе следующий этап
                BANZ    LOOP,AR2                ; * * * * * * * * * * * * * * * * *
                B       ENDB                    ; * * * * * * * * * * * * * * * * *
BPER:           ZALH    *+                      ; Загрузка мнимой и реальной частей
                SACH    RL                      ; активного поворачивающего множителя
                ZALH    *+,AR2                  ; из таблицы коэффициентов
                SACH    MN                      ; * * * * * * * * * * * * * * * * *
                LAR     AR3,*,AR1               ; Перед каждым этапом БПФ
                LAR     AR1,AD_BEG              ;восстановление указателя на начало данных
                ADRK    2h                      ; Увеличение значения указателя на 2
                SAR     AR1,AD_BEG              ; Сохранение результата
                B       BAB,AR1                 ;Безусловный переход в начало "бабочки"
***********************************************
ENDB:           LAR     AR2,LOADDATA            ; Указатель на начало блока данных
                LRLK    AR1,DATA
                LRLK    AR3,255-1               ; Количество циклов повторения
                SQRA    *+                      ; Расчет спектральной плотности
                ZAC                             ; мощности анализируемого сигнала
                SQRA    *+                      ;  U = Re2 + Im2
KL:             SQRA    *+,AR1                  ; * * * * * * * * * * * * * * * * *
                SACH    *+,AR2                  ; * * * * * * * * * * * * * * * * *
                ZAC                             ; * * * * * * * * * * * * * * * * *
                SQRA    *+,AR3                  ; * * * * * * * * * * * * * * * * *
                BANZ    KL,*-,AR2               ; * * * * * * * * * * * * * * * * *
                APAC                            ; * * * * * * * * * * * * * * * * *
                LARP    AR1                     ; * * * * * * * * * * * * * * * * *
                SACH    *                       ; * * * * * * * * * * * * * * * * *
                RET                             ; Конец процедуры БПФ
PROGE           .label  P_END
PROGL           .equ    PROGE-PROG
```