Ускорение быстрого преобразования Фурье для многомерных массивов комплексных векторов на основе технологии OpenCL

А.А. Бурцев[1]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, burtsev@niisi.msk.ru

Аннотация. Статья посвящена применению технологии OpenCL, позволяющей использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Рассматриваются варианты разработки в среде OpenCL эффективных параллельных программ, позволяющих ускорить операции быстрого преобразования Фурье для многомерных массивов комплексных векторов.

Ключевые слова: параллельное программирование, технология OpenCL, гетерогенные системы, операция Быстрого Преобразования Фурье (БПФ).

1. Введение

В настоящее время на современных компьютерных установках наряду с широко известными технологиями OpenMP [1, п. 5.1] и MPI [1, п. 5.2] для разработки параллельных программ можно применять также и технологию OpenCL [2], позволяющую использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Благодаря этой технологии сегодня можно ускорять вычисления даже на одном компьютере, если имеющаяся в его составе видеокарта поддерживает технологию OpenCL.

В серии статей [3-6] автором было предложено знакомство с технологией OpenCL с демонстрацией приёмов разработки программ для решения ряда вычислительных задач. В частности, рассматривались варианты построения по технологии OpenCL программ, ускоряющих перемножение больших матриц [4], вычисление интегралов [6], а также программ, ускоряющих в среде OpenCL выполнение операции быстрого преобразования Фурье (БПФ) для комплексных векторов большой длины [3,5]. Полученный опыт разработки подобных OpenCL-программ позволяет утверждать, что с помощью технологии OpenCL действительно можно существенно ускорить выполнение такого рода задач.

Но чтобы добиться требуемого ускорения, необходимо уметь распараллеливать используемую вычислительную программу, т.е. перестраивать её, разбивая на такие части, которые могли бы исполняться одновременно разными обработчиками, призванными совместно решать вычислительную задачу. И чем лучше удастся такую программу распараллелить, тем в большей степени получится ускорить её выполнение в среде OpenCL.

Программы обработки массивов данных, в которых над каждым отдельным элементом требуется совершать однотипные вычислительные действия независимо от значений других элементов, безусловно, можно отнести к классу тех вычислительных программ, которые хорошо поддаются распараллеливанию. Поэтому можно ожидать, что выполнение одних и тех же вычислительных операций параллельно над каждым элементом большого множества позволит в результате существенно ускорить решение всей вычислительной задачи в целом.

Во многих задачах цифровой обработки сигналов операцию дискретного преобразования Фурье требуется применять не просто к одному одиночному вектору комплексных чисел, а сразу к множеству комплексных векторов, уложенных в многомерный массив. Операцию Фурье, производимую над многими векторами, будем образно называть множественной операцией Фурье. Заметим, что такую множественную операцию можно зачислить в обозначенный класс задач, легко поддающихся распараллеливанию. А значит, можно ожидать, что и эту операцию над массивами удастся значительно ускорить, применяя технологию OpenCL.

Представим сначала алгоритмы и формулы, по которым осуществляется операция множественного Фурье в обычных программах, рассчитанных на последовательное их исполнение одним процессором. А затем рассмотрим разнообразные варианты программ, позволяющих ускорить исполнение такой операции в среде OpenCL. И проанализируем показатели их производительности, сравнивая результаты их прогонов для одних и тех же наборов данных.

2. Базовая программа для БПФ над комплексными векторами многомерного массива

Операция дискретного преобразования Фурье (ДПФ) над одиночным вектором комплексных чисел X_N заключается в получении результирующего вектора Y_N комплексных чисел, элементы которого Y_k вычисляются на основе исходного вектора X по правилу:

$$Y_k = \sum_{n=0}^{N-1} \{X_l \times W_N^{k \cdot n}\}$$

Здесь используются так называемые весовые коэффициенты вида W^q_N, которые вычисляются как комплексные величины по формуле:

$$W_N^q = e^{-i \cdot 2\pi \cdot q/N},$$

которую можно выразить иначе:

$$W_N^q = \cos\frac{2\pi q}{N} - i \cdot \sin\frac{2\pi q}{N},$$

применяя формулу Эйлера:

$$e^{i \cdot \varphi} = \cos\varphi + i \cdot \sin\varphi,$$

где i – мнимая комплексная единица.

Вычисление одиночной ДПФ непосредственно по этому правилу приводит к алгоритму сложности $O(N^2)$, который требует слишком длительного времени исполнения и потому редко применяется на практике, особенно для преобразований комплексных векторов большого размера. Существует, однако, особый класс алгоритмов под общим названием «Быстрое Преобразование Фурье» (БПФ), которые характеризуются вычислительной сложностью $O(N\log_2 N)$ и позволяют выполнить ДПФ гораздо быстрее.

Один из таких алгоритмов для исполнения операции Фурье над одиночным вектором был подробно описан в [3, п.2.2] и представлен там в виде функции **FFT** (**F**ast **F**ourier **T**ransform) на языке Си с таким заголовком:

```
void FFT(int N, complex *Y,
 complex *X, complex *W)
```

Такая функция совершает операцию БПФ по заданному комплексному вектору X длины N, получая в качестве результата комплексный вектор Y той же длины. Далее будем обозначать такую операцию в виде:

$$Y = FFT^N(X)$$

Предполагается, что весовые коэффициенты, используемые этой функцией для исполнения операции БПФ, вычисляются заранее и передаются ей в виде в виде таблицы – массива комплексных чисел W (размером от 0 до N).

Используя эту функцию, выразим операцию множественного БПФ для двумерных массивов (матриц) комплексных векторов:

$$Y_{M \times J} = MFFT^N(X_{M \times J})$$

с помощью Си-функции **MFFT**:

```
void MFFT(int M, int J, int N,
complex *X,complex *Y,complex *W){
int m,j,mj;
for (m=0; m<M; m++)
 for (j=0;j<J;j++){ mj=(m*J+j)*N;
  FFT(N,&Y[mj],&X[mj],W);
 }//for j,m
}//MFFT
```

В ней в теле внутреннего цикла (по j) над векторами X[m,j] и Y[m,j], являющимися элементами двумерных массивов X и Y, осуществляется одиночная операция БПФ:

$$Y_{m,j} = FFT^N\left(X_{m,j}\right)$$

Путём перебора циклов по m и j функция MFFT в результате исполнит операции БПФ над всеми парами векторов заданных матриц.

Такой последовательный алгоритм выполнения операции множественного БПФ примем в качестве базового. Далее будем рассматривать различные варианты исполнения той же операции множественного БПФ в среде параллельных исполнителей. А также будем сравнивать каждый из них (по производительности) с этим базовым, чтобы выявить, какой вариант даёт максимальное ускорение в среде OpenCL.

3. Первоначальный вариант (A) OpenCL-программы для множественной операции БПФ

Ранее (см. [3, п.3]) был предложен вариант OpenCL-программы, позволяющий ускорить операцию БПФ для одиночного вектора комплексных чисел. Приняв его за основу, попытаемся модифицировать его OpenCL-программу так, чтобы она осуществляла операцию БПФ для всех пар векторов заданных матриц X и Y, т.е. в итоге стала бы исполнять множественную операцию БПФ над матрицами векторов.

3.1. Простая модификация основной OpenCL-программы

Сначала попробуем просто переделать основную программу, оставив прежними её процедуры ядра. Осуществим такую переделку тем же естественным приёмом, как и в случае с последовательной программой. Для выполнения множественной операции БПФ в среде OpenCL составим отдельную функцию (см. далее **clMFFT**). В ней разместим два вложенных цикла с перебором по m и по j. А в теле внутреннего цикла (по j) для осуществления одиночной операции БПФ

с очередной парой векторов X[m,j] и Y[m,j] поставим вызов функции **clFFT**, которая была подробно описана ранее (она была представлена в п.3.2 в [3]).

```
void clMFFT(int M,int J,int N,
complex *Y,complex *X,complex *W){
int m,j,mj;
for (m=0; m<M; m++) {
 for (j=0;j<J;j++){ mj=(m*J+j)*N;
   clFFT(N,&Y[mj],&X[mj]);
 }//for j,m
}//clMFFT
```

Заметим, что перед вызовом функции **clFFT** в переменной **mj** по формуле **mj=(m*J+j)*N** предварительно вычисляется индекс, начиная с которого в многомерных массивах X[M][J][N] и Y[M][J][N] располагаются соответственно вектора X[m,j] и Y[m,j]. Именно для этих комплексных векторов, как элементов матриц X[M][J] и Y[M][J], на очередном шаге внутреннего цикла и вызывается функция **clFFT**, чтобы исполнить одиночную операцию БПФ. В последующих алгоритмах будем неоднократно использовать такую формулу вычисления значения mj для определения индекса месторасположения вектора в многомерном массиве.

Чтобы оценить, насколько удаётся ускорить операцию множественного БПФ с помощью такой переделанной OpenCL-программы, сравним время исполнения ($\mathbf{T_{CL}}$) функции **clMFFT** (в среде OpenCL с множеством параллельных исполнителей) со временем исполнения ($\mathbf{T_{CPU}}$) функции **MFFT** (на одном процессоре) с теми же параметрами и вычислим коэффициент ускорения **K** как отношение $\mathbf{T_{CPU}/T_{CL}}$.

Такая программа с вызовами этих функций (варианта A1) была разработана (на языке Си) как консольное приложение в MS Visual Studio. Для представления комплексного числа парой вещественных значений одинарной точности в Си-программе тип complex определялся так:

```
#define complex cl_float2
```

А в программе, содержащей процедуры ядра на языке OpenCL, – немного иначе:

```
#define complex float2
```

Программа была скомпонована в MS Visual Studio под ОС Windows-7 для исполнения на OpenCL-платформе, содержащей процессор Intel i3-2100 (3.1 Ггц) и специализированную видеокарту NVidia GeForce 1050ti (1392 Мгц).

На этой платформе (будем называть её платформа «NVidia») скомпонованная OpenCL-программа многократно прогонялась для матриц комплексных векторов различных размеров (M×J) и длин (N). Усреднённые показатели коэффициентов ускорения, полученные в результате таких прогонов, представлены в таблице 1.

Таблица 1. Ускорение множественного БПФ OpenCL-программой варианта A1 на платформе NVidia

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.023	0.024	0.0175	0.020	0.014	0.0032	0.0012
64	0.046	0.051	0.0475	0.0325	0.019	0.0079	0.0036
128	0.110	0.1166	0.1033	0.0815	0.0812	0.0085	0.0034
256	0.254	0.2240	0.2247	0.2316	0.0879	0.0181	0.0091
512	0.444	0.4332	0.4163	0.2780	0.1030	0.0348	---
1024	0.898	1.0071	0.9260	0.6293	0.2368	0.0535	---
2048	2.165	1.9904	2.0805	1.5418	0.4658	0.0864	---
4096	4.449	3.7327	3.7107	2.9872	0.6236	---	---
8192	7.461	6.2371	5.6467	4.5101	0.8492	---	---
16384	12.79	12.214	9.1541	9.7322	---	---	---
32768	26.40	21.397	22.617	21.582	---	---	---
65536	**42.35**	40.909	36.044	36.285	---	---	---

Замечание: символы --- означают, что для таких значений параметров программа не может быть корректно исполнена (например, из-за недостатка памяти)

Сравнивая показатели ускорения операции множественной БПФ (*MFFT*) из таблицы 1 с аналогичными показателями ускорения одиночной операции БПФ (*FFT*), представленными в таблице 2 в [3], можно заметить, что они примерно одинаковы для одной и той же длины векторов. Причём с ростом M и J итоговые коэффициенты ускорения мало изменяются.

3.2. Модификация процедур ядра

Теперь попробуем исходную OpenCL-программу, рассмотренную ранее (в п.3 в [3]) для одиночной операции БПФ, переделать немного по-другому. Внедрим циклы по m и j для перебора векторов не в основную программу, а в её процедуры OpenCL-ядра.

Вставим такие циклы в процедуру ядра, чтобы осуществлять бит-реверсное копирование векторов из матрицы X в матрицу Y:

```
__kernel void fft_brvprstMJ
( uint N,const uint L,
 __global complex *X,
 __global const complex *Y,
 __global const uint *BRT,
 const uint M, const uint J){
uint m,j,mj,k,i; i=get_global_id(0);
k=BRT[i]; // k=бит-реверсное значение для i
for (m=0; m<M; m++)
for (j=0; j<J; j++)
{ mj=(m*J+j)*N; Y[mj+k]=X[mj+i]; }
}// fft_brvprstMJ
```

В результате вызова такой процедуры исполнителем под номером **i** в среде OpenCL элементы с индексом **i** всех векторов, содержащихся в матрице **X**, будут скопированы в соответствующие им векторы матрицы **Y** на позицию с индексом **k**, вычисленную как бит-реверсное значение для индекса **i**.

Аналогичную модификацию сделаем и для другой процедуры ядра, которая вызывается каждым исполнителем OpenCL-среды на очередной стадии группового исполнения параллельных операций «бабочек Фурье» (БФ) над векторами (см. 5-й пункт в списке действий, описанных в п.3.2 в [3]). В результате получим такую процедуру ядра:

```
__kernel void fft_btflyMJ
(int t, int N,__global complex *Y,
__global complex *W,uint M,uint J)
{uint G,R,k,f,s,h,a,b,u,m,j,mj;
 uint i= get_global_id(0);
 G=1<<(t-1); // кол-во групп G= 2^(t-1)
 R=N>>t; // кол-во пар в группе R= 2^(P-t);
 k=i&(G-1); // номер группы = i % G
 f=i>>(t-1);// номер пары в группе= i / G
 s=G; h=2*s;
 a=k+f*h; b=a+s; u=R*k;
 complex V= W[u];
 for (m=0; m<M; m++)
 for (j=0; j<J; j++) {
   mj=(m*J+j)*N;
   BTF(Y[mj+a],Y[mj+b],V);
 }// for j,m
}//fft_btflyMJ
```

Конечно же, и в основную OpenCL-программу тоже придётся внести добавления, чтобы эти процедуры ядра могли запускаться уже с расширенным набором параметров:

```
kn1=clCreateKernel(prgrm,"fft_brvprstMJ",0);
kn2=clCreateKernel(prgrm,"fft_btflyMJ",0);
clSetKernelArg(kn1,5,szI,&M);
clSetKernelArg(kn1,6,szI,&J);
clSetKernelArg(kn2,4,szI,&M);
clSetKernelArg(kn2,5,szI,&J);
```

Коэффициенты ускорения, которые обеспечивает полученный в результате такой переделки вариант (A2) OpenCL-программы, представим в таблице 2.

Таблица 2. Ускорение множественного БПФ OpenCL-программой варианта A2 на платформе NVidia

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.091	0.600	1.667	2.667	6.25	5.10	5.25
64	0.111	0.857	4.00	6.333	8.714	8.571	9.238
128	0.364	2.00	7.00	11.750	15.556	17.625	17.308
256	0.923	3.875	12.60	34.333	29.273	31.950	31.152
512	1.438	9.650	17.875	42.999	59.150	66.073	63.240
1024	2.826	17.083	46.089	72.489	88.283	101.18	102.16
2048	6.088	33.075	79.185	109.39	128.61	134.75	---
4096	12.89	51.199	115.08	150.09	166.12	157.44	---
8192	30.88	79.759	138.40	159.42	165.28	---	---
16384	52.80	128.08	175.53	149.61	166.70	---	---
32768	80.28	136.18	166.71	182.04	---	---	---
65536	106.9	172.85	**187.84**	184.12	---	---	---

Сравнивая его с предыдущим вариантом (A1), можно заметить, что показатели ускорения заметно подросли. Чем же можно объяснить такой их рост?

Дело в том, что в процедурах ядра индексы (i,k) элементов для бит-реверсной перестановки и индексы (a,b,u) элементов, участвующих в операциях БФ («бабочки Фурье»), теперь вычисляются всего лишь один раз. А затем они многократно применяются сразу для обработки элементов (с такими индексами) у всех векторов, перебираемых далее в циклах (по m и по j). Вот поэтому процедуры ядра и стали выполняться немного быстрее.

Заметим также, что таблице 2 (в отличие от таблицы 1 предыдущего варианта) уже становится заметно, что с ростом размеров (M и J) матриц векторов коэффициенты ускорения тоже подрастают. Правда, увы, не в такой же степени, как сами эти размеры.

Чтобы добиться ещё больших показателей ускорения, попробуем перестроить OpenCL-программу так, чтобы при прогоне её в среде OpenCL использовать как можно больше параллельных исполнителей.

4. Альтернативный вариант (B) OpenCL-программы для множественной операции БПФ

Ранее уже неоднократно демонстрировалось (например, в [4]), что любую программу, которая над каждым элементом массива осуществляет вычислительные действия, независимые от значения других элементов, можно легко распараллелить с тем, чтобы ускорить её исполнение в среде OpenCL. Для этого надо лишь составить OpenCL-программу так, чтобы вычислительная обработка каждого элемента массива выполнялась отдельным исполнителем OpenCL-среды, функционирующим параллельно с другими.

Следуя этому правилу, составим OpenCL-программу выполнения операции множественного Фурье $Y_{M\times J}=MFFT(X_{M\times J})$ для двумерных массивов $X_{M\times J}$ и $Y_{M\times J}$ комплексных векторов. Будем задействовать ансамбль из M×J параллельных исполнителей OpenCL-среды, сгруппированных в двумерный массив [M][J], назначив каждому из них в качестве задания исполнить операцию $Y_{m,j}=FFT(X_{m,j})$ над выделенной ему парой векторов (с индексами m,j).

Для исполнения одиночной операции БПФ над парой векторов оформим функцию **cl_FFT**, заимствовав при её оформлении алгоритм функции FFT, подробно описанный в п. 2.2 в [3], и подкорректировав её заголовок с учётом особенностей языка OpenCL:

```
void cl_FFT ( uint N,
```

```
 __global complex *VY,
 __global complex *VX,
 const __global complex *VW) {
uint P,t,s,h,G,R,d,k,u,j,a,b,i;
complex W; complex XP[NP];
P=iLog2(N);       // so that N = 2^P
// ч1. Бит-реверсное копирование XP <=VX
 copy_brvprst(N, P, VX, XP);
s=1;h=2;G=1; R=N/2; d=N/2;
//ч2.Основной цикл исполнения бабочек Фурье:
for (t=1; t<=P; t++) {
 for (k=0,u=0; k<G; k++,u=u+d) {
  W= VW[u];
  for (j=0,a=k; j<R; j++,a=a+h)
  { b=a+s; BTF(XP[a],XP[b],W); }
 }//for k
 h=h*2;s=s*2; G=G*2; R=R/2; d=d/2;
}//for t
// ч3. Копирование XP => VY
 for (i=0; i<N; i++) VY[i]= XP[i];
}//cl_FFT
```

В этой функции для улучшения быстродействия вектор VX из глобальной памяти сначала копируется при бит-реверсной перестановке в вектор XP, размещаемый в приватной памяти исполнителя. Далее все операции БФ («бабочек Фурье») с помощью макрооперации **BTF**, подробно описанной в п.3.1 в [3], исполняются над вектором XP, который затем записывается как результат в вектор VY глобальной памяти.

Используя функцию **cl_FFT** в качестве вспомогательной, составим теперь процедуру OpenCL-ядра, которая будет запускаться на каждом из параллельных исполнителей OpenCL-среды, собранных в двумерный массив [M][J]:

```
__kernel void cl_kern_MFFT
( uint M, uint J, uint N,
 __global complex *Y,
 const __global complex *X,
 const __global complex *W ) {
 int m, j, mj;
 m= get_global_id(0);
 j= get_global_id(1);
 mj=(m*J+j)*N;
// БПФ для вектора Y[m,j] <= FFT(X[m,j])
 cl_FFT(N, &Y[mj], &X[mj], W);
}//cl_kern_MFFT
```

В этой процедуре с помощью применения функции get_global_id каждый исполнитель сам определяет значения индексов **m** и **j**, указывающие месторасположение в матрицах X и Y той пары векторов, для которых ему предстоит выполнить операцию БПФ.

Для запуска такой процедуры ядра в OpenCL-среде в основную OpenCL-программу добавим следующие действия.

1. Процедуре ядра cl_kern_MFFT назначим дескриптор **knM**:

```
cl_kernel knM=
 clCreateKernel(prgrm,"cl_kern_MFFT",NULL);
```

2. Выделим в памяти OpenCL-устройства буферы для данных, участвующих в операции:

```
cl_mem objX,objY;
cl_uint szC= sizeof(complex);
ObjX=clCreateBuffer(cntxt,
 CL_MEM_READ_ONLY,szC*M*J*N,0,0);
ObjY=clCreateBuffer(cntxt,
CL_MEM_READ_WRITE,szC*M*J*N,0,0);
```

3. Загрузим матрицу комплексных векторов X[M][J][N] в буфер объекта **objX**:

```
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,objX
   CL_TRUE,0,szC*N*M*J,X,0,0,0);
```

4. Назначим процедуре ядра cl_kern_MFFT 6 фактических параметров:

```
cl_uint szI= sizeof(cl_uint);
cl_uint szM= sizeof(cl_mem);
clSetKernelArg(knM,0,szI,&M);
clSetKernelArg(knM,1,szI,&J);
clSetKernelArg(knM,2,szI,&N);
clSetKernelArg(knM,3,szM,&objY);
clSetKernelArg(knM,4,szM,&objX);
clSetKernelArg(knM,5,szM,&objW);
```

5. Запустим в OpenCL-среде процедуру ядра cl_kern_MFFT на множестве из M×J исполнителей и дождёмся её завершения всеми исполнителями:

```
size_t gWS[2]= {M,J};
cl_event evM;
clEnqueueNDRangeKernel(cmndQ,
 knM,2,NULL,gWS,NULL,0,NULL,&evM);
clFinish(cmndQ);
```

6. Полученную в буфере объекта **objY** матрицу векторов запишем в Y[M][J][N]:

```
clEnqueueReadBuffer(cmndQ,objY,
CL_TRUE,0,szC*N*M*J,Y,0,0,0);
```

Представим теперь, какие коэффициенты ускорения обеспечивает полученный таким способом вариант (B) OpenCL-программы (см. таблицу 3).

Сравнивая их с показателями предыдущих вариантов A1 и A2 (в таблицах 1 и 2), можно заметить, что при малой длине обрабатываемых векторов (N<1024), вариант B обеспечивает лучшие коэффициенты ускорения для больших размеров матриц (M×J>=20×20). Но при больших значениях N (N≥1024) вариант B, увы, уступает варианту A2 почти на всех размерах матриц M×J (и больших, и малых).

Таблица 3. Ускорение множественного БПФ OpenCL-программой варианта B на платформе NVidia

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.40	1.50	5.00	25.00	49.73	51.00	82.00
64	0.40	2.00	5.50	19.00	59.00	61.67	48.50
128	0.67	4.67	14.00	22.00	72.50	74.75	63.57
256	0.91	5.33	21.00	35.67	66.84	78.88	71.79
512	1.53	7.89	23.50	45.60	78.22	82.76	78.18
1024	1.79	7.00	22.29	50.71	86.78	87.78	82.44
2048	1.90	7.91	24.03	58.42	91.50	96.72	---
4096	1.61	8.75	28.15	64.95	98.73	107.80	---
8192	2.03	11.69	36.28	81.03	125.21	---	---
16384	2.08	11.61	37.60	88.17	**131.26**	---	---
32768	2.24	12.44	40.37	92.67	---	---	
65536	---	---	---	---	---	---	

Получается, что OpenCL-программы вариантов A1,A2 проявляют свои достоинства с ростом длины векторов N, а OpenCL-программа варианта B – с ростом размеров матриц M×J. А нельзя ли совместить оба этих достоинства в одной OpenCL-программе?

5. Комбинированный вариант (C) OpenCL-программы для множественной операции БПФ

Создадим теперь такую OpenCL-программу для множественной операции БПФ, которая задействует N×M×J параллельных исполнителей OpenCL-среды. И распределяет вычисления между ними так, что операцией БПФ для каждой пары векторов ($X_{m,j}$,$Y_{m,j}$) совместно занимаются N исполнителей (как в варианте A1), но при этом вычисления БПФ для всех пар векторов матриц $X_{M\times J}$ и $Y_{M\times J}$ осуществляются не последовательно, а параллельно (одновременно).

5.1. Модификация процедур ядра

Возьмём процедуры ядра, рассмотренные в п. 3.2 (варианта A2), и заменим в их теле циклы по m и по j (выделенные жирным шрифтом) на одиночные действия с парой элементов обрабатываемых векторов. А для вычисления месторасположения **mj** тех векторов в матрицах X и Y, которые назначены для обработки конкретному исполнителю, определим значения индексов для **m** и **j** с помощью функции get_global_id.

В результате заменим обозначенные циклы в процедуре ядра fft_brvprstMJ на фрагмент:

```
m= get_global_id(1);
j= get_global_id(2);
{ mj=(m*J+j)*N; Y[mj+k]=X[mj+i]; }
```

А в процедуре cl_fft_btflyMJ – на фрагмент:

```
m= get_global_id(1);
j= get_global_id(2);
mj=(m*J+j)*N;
BTF(Y[mj+a],Y[mj+b],V);
```

5.2. Модификация основной OpenCL-программы варианта C

Для запуска таких процедур ядра в OpenCL-среде в основной OpenCL-программе необходимо изменить прежнюю последовательность действий, описанных в п.3.2 в [3] и заимствованных для вариантов A1 и A2.

1. Во-первых, внесём добавления для определения дескрипторов kn1 и kn2 новых процедур ядра и назначения для них новых параметров, которые были сделаны в разделе 3.2.

2. Во-вторых, добавим фрагменты для выделения памяти объектам objX, objY, а также загрузки матрицы X в память OpenCL-устройства (в буфер objX), описанные в пунктах 2,3 списка действий в разделе 4.

3. Запустим в OpenCL-среде процедуру ядра fft_brvprstMJ на множестве из N×M×J исполнителей, сгруппированных в трёхмерный массив [N][M][J], для бит-реверсного параллельного копирования всех M×J векторов из буфера objX, представляющего матрицу $X_{M\times J}$, в соответствующие им вектора буфера objY, представляющего матрицу $Y_{M\times J}$. И дождёмся окончания её завершения всеми исполнителями:

```
size_t gWS[3]= { N,M,J };
cl_event ev1;
clEnqueueNDRangeKernel(cmndQ,
 kn1,3,NULL,gWS,NULL,0,NULL,&ev1);
clFinish(cmndQ);
```

4. Теперь поставим основной цикл (по t), в котором на каждом шаге (t) будем запускать процедуру ядра cl_fft_btflyMJ на множестве из (N/2)×M×J исполнителей для совершения всех операций БФ t-го этапа (t=1,..,P) параллельно для всех N/2 пар элементов всех M×J векторов, содержащихся в буфере матрицы Y:

```
cl_uint t; gWS[0]=N/2;
cl_event ev2;
for (t=1; t<=P; t++) {
 clSetKernelArg(kn2,0,szI,&t);
 clEnqueueNDRangeKernel(cmndQ,
 kn2,3,NULL,gWS,NULL,0,NULL,&ev2);
 clWaitForEvents(1,&ev2);
}//for t
```

В начале каждого шага цикла требуется назначить 0-му параметру процедуры ядра значение номера (t) текущего этапа, а перед его окончанием следует дождаться завершения этой процедуры ядра всеми исполнителями с помощью вызова функции clWaitForEvents.

5. Наконец, полученную в буфере objY матрицу векторов нужно скопировать из памяти OpenCL-устройства в итоговую матрицу Y, для чего добавим в основную программу фрагмент, описанный в п. 6 списка действий в разделе 4.

Представим теперь, какие коэффициенты ускорения обеспечивает полученный в результате проведённой модификации новый вариант (C) OpenCL-программы (см. таблицу 4).

Таблица 4. Ускорение множественного БПФ OpenCL-программой варианта C на платформе NVidia

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.061	0.375	1.667	14.50	36.00	26.07	124.40
64	0.312	0.680	5.567	10.067	45.15	82.72	76.45
128	0.241	3.025	8.840	34.10	66.82	112.60	94.80
256	0.778	3.281	20.36	71.30	81.60	119.45	121.77
512	1.498	11.267	29.415	56.087	129.45	120.34	119.58
1024	4.578	20.417	47.680	93.176	116.61	140.66	133.59
2048	8.276	33.489	89.667	120.26	151.15	160.42	---
4096	15.32	59.788	115.75	147.54	164.95	162.56	---
8192	27.81	86.733	143.77	172.49	184.86	---	---
16384	41.17	115.56	170.44	195.11	176.31	---	---
32768	67.48	147.76	171.20	209.55	---	---	---
65536	117.0	179.57	**222.73**	207.10	---	---	---

Из приведённой таблицы хорошо видно, что при таком варианте действительно удаётся совместить достоинства рассмотренных ранее вариантов A1, A2 и B. Во-первых, анализируя показатели в каждой строке этой таблицы, можно заметить, что при любой фиксированной длине (N) коэффициент ускорения значительно увеличивается с ростом размеров матриц (M×J). И во-вторых, анализируя показатели каждого столбца таблицы, можно отметить и другую закономерность: для матриц любого фиксированного размера (M×J) коэффициент ускорения существенно вырастает при росте длины векторов (N). И это позволяет утверждать, что в этом комбинированном варианте удаётся совместить достоинства всех прежних вариантов.

Сравнивая показатели коэффициентов ускорения таблицы 4 с аналогичными показателями из приведённых ранее таблиц 1-3, можно сделать вывод, что такой комбинированный вариант (C) обеспечивает наилучшие результаты ускорения множественной операции Фурье $Y_{M\times J}=MFFT^{N}(X_{M\times J})$ почти для всех параметров: длинах векторов N и размеров матриц M×J.

А максимального значения (**222.73**) коэффициент ускорения этой операции (на платформе «NVidia») достигает (как это и отмечено в таблице 4) при значениях параметров: N=65536 (2^{16}) и M×J=10×10. Заметим при этом, что для одиночной операции FFT^{N} на векторах той же длины ($N=2^{16}$) при запуске программы в OpenCL-среде можно добиться ускорения лишь в **45.67** раза, применив первоначальный вариант OpenCL-программы, представленной в [3], и только в **69** раз, если применить улучшенный вариант этой же OpenCL-программы, описанный в [5]. (Эти показатели приведены соответственно в таблице 2 в [3] и таблице 10 в [5]).

6. Множественные операции БПФ для платформы «Intel»

Рассмотренные варианты программ, разработанные для ускорения множественной операции БПФ (МБПФ) в среде OpenCL, были опробованы также и на других OpenCL-платформах. В частности, они компоновались на обычном настольном компьютере (в MS Visual Studio под ОС Windows-10) и запускались на аппаратной платформе, оснащённой процессором CPU Intel-i59400 (2.9 Ггц) и встроенным графическим процессором GPU UHD 630 (350 МГц).

На этой платформе (будем называть её далее платформа «Intel») скомпонованные OpenCL-программы всех рассмотренных вариантов (A1, A2, B, C) многократно прогонялись для матриц векторов комплексных чисел одинарной точности с теми же параметрами, что и для платформы «NVidia». Коэффициенты ускорения, полученные в результате произведённых прогонов с матрицами таких же размеров (M×J) и такими же длинами векторов (N), представлены соответственно в таблицах 5-8.

Таблица 5. Ускорение множественного БПФ OpenCL-программой варианта A1 на платформе Intel

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.012	0.011	0.011	0.012	0.012	0.012	0.012
64	0.024	0.024	0.021	0.026	0.024	0.025	0.025
128	0.048	0.045	0.049	0.049	0.047	0.032	0.049
256	0.096	0.093	0.094	0.097	0.097	0.096	0.112
512	0.196	0.197	0.188	0.192	0.195	0.193	0.195
1024	0.376	0.385	0.389	0.378	0.390	0.402	0.402
2048	0.766	0.768	0.795	0.784	0.763	0.745	---
4096	1.505	1.576	1.564	1.568	1.402	1.624	---
8192	3.116	3.296	2.658	2.995	2.954	---	---
16384	5.864	5.893	6.063	5.502	6.566	---	---
32768	11.02	10.797	10.994	11.307	---	---	---
65536	19.92	20.507	20.275	**21.023**	---	---	---

Таблица 6. Ускорение множественного БПФ OpenCL-программой варианта A2 на платформе Intel

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.045	0.260	0.836	1.785	2.439	2.548	1.987
64	0.094	0.519	1.638	3.350	4.508	4.223	3.980
128	0.184	0.954	3.239	5.888	9.838	7.771	7.575
256	0.379	2.276	6.149	12.707	14.681	14.563	13.625
512	0.763	4.088	11.470	23.291	25.159	24.262	22.689
1024	1.485	8.004	22.506	36.549	33.480	32.379	---
2048	3.138	15.043	37.352	44.731	33.296	29.169	---
4096	5.745	25.840	43.223	38.666	32.489	---	---
8192	10.74	37.401	44.659	31.583	27.576	---	---
16384	18.51	40.257	30.305	27.803	---	---	---
32768	29.29	40.174	26.196	26.154	---	---	---
65536	**41.23**	32.990	28.406	---	---	---	---

Таблица 7. Ускорение множественного БПФ OpenCL-программой варианта B на платформе Intel

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.175	1.006	3.867	12.976	15.648	12.423	11.666
64	0.273	1.713	6.069	17.946	14.600	8.963	9.770
128	0.368	2.012	8.131	**21.097**	8.490	6.131	7.176
256	0.460	2.487	10.206	18.855	6.061	5.127	5.018
512	0.526	2.900	9.506	15.522	5.099	4.565	3.793
1024	0.580	3.126	10.544	11.370	4.424	4.197	---
2048	0.605	3.004	10.053	9.557	4.319	4.006	---
4096	0.617	3.067	8.567	7.896	4.092	---	---
8192	0.650	2.927	8.399	6.741	3.929	---	---
16384	0.660	2.828	6.567	5.883	---	---	---
32768	0.635	2.613	5.382	5.753	---	---	---
65536	0.637	2.218	6.285	---	---	---	---

Таблица 8. Ускорение множественной операции БПФ OpenCL-программой варианта C на платформе Intel

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.046	0.292	1.084	4.321	18.408	29.340	22.365
64	0.093	0.595	2.302	7.844	29.483	29.211	21.656
128	0.187	1.190	4.067	13.439	31.170	29.595	27.637
256	0.384	2.237	7.895	22.462	36.091	28.672	29.091
512	0.757	4.706	13.512	33.059	34.963	30.628	28.058
1024	1.496	8.116	23.629	40.344	32.374	32.791	---
2048	3.010	14.688	34.601	42.086	35.342	34.897	---
4096	5.731	25.160	42.610	37.504	37.322	---	---
8192	10.77	37.797	46.239	38.721	38.265	---	---
16384	21.92	**47.783**	41.826	40.246	---	---	---
32768	31.03	43.424	39.520	37.231	---	---	---
65536	42.23	41.645	41.793	---	---	---	---

Сравнивая показатели ускорения, записанные в этих таблицах, можно сделать вывод, что и на платформе «Intel» наилучшие коэффициенты ускорения множественной операции БПФ почти для всех параметров обеспечивает комбинированный вариант C (см. таблицу 8).

В строках и столбцах таблицы результатов, полученных для этого варианта на OpenCL-платформе «Intel», также можно увидеть соблюдение отмеченных ранее закономерностей. При фиксированном значении длины векторов N коэффициент ускорения заметно подрастает при увеличении размеров матриц M×J. А для фиксированных размеров матриц M×J коэффициент ускорения вырастает с ростом длины векторов N.

Правда, эти закономерности перестают соблюдаться для самых крайних показателей каждой строки и каждого столбца, где параметры N и M×J достигают предельно высоких значений. При таких больших значениях параметров OpenCL-программа либо вообще не может корректно вычислить требуемый результат из-за нехватки памяти (тогда вместо коэффициента в таблице проставляется знак ---), либо её вычисления в OpenCL-среде оказываются недостаточно эффективными по причине реального отсутствия столь необходимого количества вычислительных ядер на используемой OpenCL-платформе.

Как видно из приведённой таблицы 5, для множественной операции БПФ на платформе «Intel» удаётся достичь максимального значения коэффициента ускорения **47.783** при значениях параметров: N=16384 (2^{14}) и M×J=5×5.

Заметим, что на этой же платформе ранее для одиночной операции БПФ той же длины N=2^{14} были получены такие значения коэффициентов ускорения: **5.333** для первоначального варианта OpenCL-программы, представленного в [3]; и **11.38** для оптимизированного варианта OpenCL-программы, описанного в [5]. (Эти значения приведены соответственно в таблице 1 в [3] и в таблице 9 в [5]).

7. Заключение

Таким образом, технологию OpenCL можно успешно применять для повышения быстродействия таких вычислительных программ, в которых требуется выполнять однотипные операции над многочисленными объектами, сгруппированными в массивы данных.

Но для этого требуется уметь распараллеливать вычислительную задачу так, чтобы каждый элемент огромного массива объектов данных мог обрабатываться отдельным исполнителем OpenCL-среды независимо от других. В таком случае можно добиться значительного ускорения выполнения такой задачи в OpenCL-среде.

Это наглядно демонстрируют представленные примеры OpenCL-программ, позволяющих существенно ускорить выполнение множественной операции быстрого преобразования Фурье для многомерных массивов комплексных векторов большого размера. Следует отметить также, что в среде OpenCL для множественной операции Фурье над массивом векторов удаётся добиться ускорения в значительно большей степени, чем для такой же операции Фурье над одиночным вектором.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0003 «Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов»

.

Acceleration of Fast Fourier Transform for Multidimensional Arrays of Complex Vectors Based on OpenCL Technology

A.A. Burtsev

Abstract. The article is devoted to the use of OpenCL technology, which allows using the powerful resources of graphic processors to increase the speed of computing programs. Development variants of efficient parallel programs in the OpenCL environment to accelerate the fast Fourier transform operation for multidimensional arrays of complex vectors are considered.

Keywords: parallel programming, OpenCL technology, heterogeneous systems, operation of Fast Fourier Transform (FFT)

Литература

1. В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. Спб., БХВ-Петербург, 2004.
2. Официальный OpenCL–сайт организации Khronos Group, http://www.khronos.org/opencl/
3. А.А. Бурцев. Ускорение быстрого преобразования Фурье на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 11 (2021), № 4, 27–37.
4. А.А. Бурцев. Оптимизация операции перемножения матриц на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, 100–112.
5. А.А. Бурцев. Оптимизация операции быстрого преобразования Фурье в среде OpenCL. // «Труды НИИСИ РАН», Т.12 (2022), №1-2, 11-27.
6. А.А. Бурцев. Применение технологии OpenCL для ускорения вычисления интегралов. // «Труды НИИСИ РАН», Т.13 (2023), №1-2, 19-24.