Katedra Wytrzymałości Materiałów i Metod Komputerowych Mechaniki, Wydział Mechaniczny Technologiczny, Politechnika Śląska
44-100 Gliwice, ul. Konarskiego 18a, tel. +48 32 2371204, fax +48 32 2371282
www.kwmimkm.polsl.pl

Obliczenia z użyciem języka OpenCL

Wykonał: **Krzysztof Ryczek**
Kierunek: **Edukacja Techniczno-Informatyczna**
Promotor: **Prof. dr hab. inż. Tadeusz Burczyński**
Specjalność: **Techniki informacyjne**

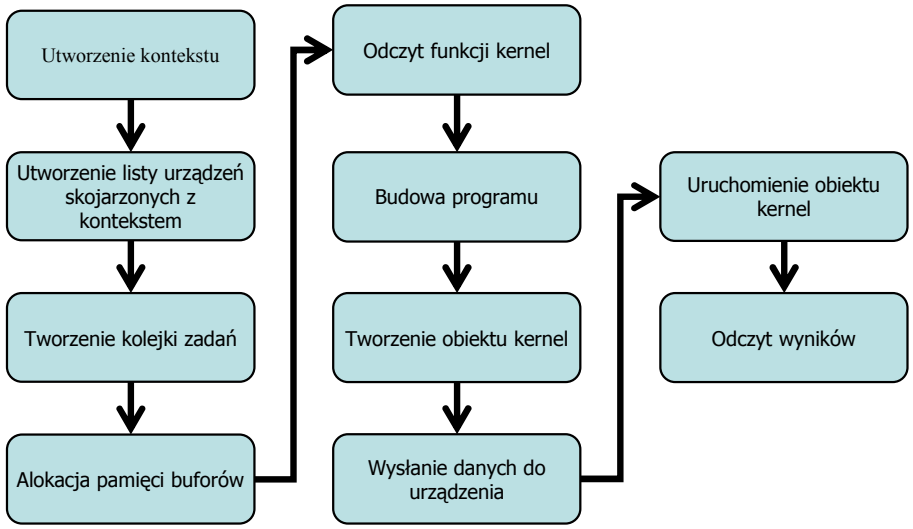
Celem pracy magisterskiej pracy było opracowanie programów realizujących obliczenia algebraiczne wykorzystując do tego język OpenCL oraz zbadanie przyspieszenia tych obliczeń. OpenCL wykorzystuje architekturę CUDA firmy NVIDIA, co pozwala na użycie do obliczeń układów GPU (Graphics Processing Unit – jednostka przetwarzania grafiki).
Opracowane programy realizowały:
- iloczyn dwóch wektorów
- iloczyn macierzy i wektora
- iloczyn macierzy rzadkiej i wektora
- rozwiązywanie układów równań metodą Jacobiego

Implementacja algorytmu rozwiązywania układów równań metodą Jacobiego w OpenCL

```
Wyznacz zerowe przybliżenie x0 rozwiązania x
dla k = 1, 2, …
    {
    dla i = 1, 2, …, n
            {
            x_i = 0
            dla j = 1, 2, …, i - 1, i + 1, …, n
                    {
                    x_i = x_i + a_i,j * x_j^(k-1)
                    }
                    xi = (b_i - x_i)/a_i,i
            }
            Sprawdź warunek zakończenia, kontynuuj jeśli to
            konieczne
    }
```

```
kernel void Jacobi (const global int* EW, const global float* W_P,
                    const global float* W, const global int* C,
                    const global float* b,  global float* x, global float* x_p,
                    int wys, int iter, global float error, global float* e,
                    global float E[], global int I[])
{
uint i1=0, i2=0, j=0,y =0, iteracja=1;
uint i=get_global_id(0);
float suma=0;
i-=1;
if (i<wys)
{
do {
   iteracja++;
   x_p[i] = x[i];
   barrier(CLK_GLOBAL_MEM_FENCE);
   suma = 0;
   if (i == 0) i1 = 0;
   else i1 = EW[i-1];
   i2 = EW[i];
   for (j = i1; j < i2; j++){
      if (i! = C[j]) suma += W[j]*x_p[C[j]];}
   x[i]=(b[i]-suma) / W_P[i];
   e[i]=(x[i]-x_p[i])*(x[i]-x_p[i]);
   barrier(CLK_GLOBAL_MEM_FENCE);
   if (get_global_id(0) == 1)
      {
                              E[0] = 0;
                              I[0] = iteracja-1;
                              for (y=0; y<wys; y++) E[0]+=e[y];
                              barrier(CLK_GLOBAL_MEM_FENCE);
      }
while ((iter >= iteracja)&&(E[0] > error));
}}
```

Ogólny schemat działania programów



Wyniki testów numerycznych

Rozmiar macierzy	Ilość elementów niezerowych[%]	Czas GPU [s]	Czas CPU [s]	Przyspieszenie obliczeń
3000x3000	2	0.0497	0.0178	0.35
	5	0.0650	0.0422	0.65
	10	0.1165	0.0987	0.85
	15	0.1669	0.5713	3.4
	20	0.2381	1.1380	4.7
5000x5000	2	0.0988	0.0492	0.5
	5	0.1679	0.1183	0.7
	10	0.2015	1.1171	5.5
	15	0.6607	3.1602	4.8
	20	1.0286	4.3731	4.3
8000x8000	2	0.1998	0.1210	0.6
	5	0.4559	1.3658	3
	10	1.2246	5.8045	4.7
	15	2.3015	10.724	4.6
	20	2.8761	15.623	5.4
10000x10000	2	0.2934	0.1874	0.64
	5	0.8523	3.1366	3.6
	10	1.9278	10.545	5.5
	15	3.2704	16.877	5.2
	20	4.7365	26.379	5.7

Czasy obliczeń implementacji metody Jacobiego dla CPU i GPU