PaP

Projet : rapport4

4TIN804U

GOEDEFROIT Charles

2021-2022

Table des matières

4.7 OpenCL Implementation

4.7.3 Detecting termination

Pour implémenter la terminaison, j'ai commencé par ajouter un nouveau kernel ocl qui prend en paramètre un buffer pour la terminaison. Je remplis ce buffer avec 1 s'il y a eu un changement et 0 sinon. C'est la ligne 13 du code ci-dessous :

```
1  __kernel void ssandPile_ocl_term(__global unsigned *in, __global unsigned *out, __global unsigned *buffer)
2  {
3    int myPos = y * DIM + x;
4
5    if (!(x == 0 || y == 0 || x == DIM - 1 || y == DIM - 1))
6    {
7      unsigned result = in[myPos] % 4;
8      result += in[myPos + DIM] / 4;
9      result += in[myPos - DIM] / 4;
10     result += in[myPos + 1] / 4;
11     result += in[myPos - 1] / 4;
12     out[myPos] = result;
13     buffer[myPos] = result != in[myPos];
14   }
15 }
```

Puis j'ai ajouté la fonction *ssandPile_invoke_ocl_term()* en m'inspirant de la fonction *ocl_invoke_kernel_generic()*. Les changements que j'ai apporté sont :

- un buffer nommé *term_buffer* que je passe en paramètre au kernel ocl :

```
err |= clSetKernelArg(compute_kernel, 2, sizeof(cl_mem), &term_buffer);
```

- 2 variables pour contrôler l'intervalle d'itérations où on vérifie si l'états du tas de sable est stable.

```
static uint countIter = 0;
static uint checkTermIterm = 20;
```

- la copie du buffer en ram avec la vérification que le buffer soit remplie de 0. Qu'il n'y ait pas eu de changement.

```
if (countIter > checkTermIterm) {
  err = clEnqueueReadBuffer(queue, term_buffer, CL_TRUE, 0, DIM * DIM * sizeof(TYPE), tmpTab, 0, NULL, NULL);
```

```
  check(err, "Failed to read buffer from GPU");

  bool notChange = true;
  for (size_t i = 0; i < DIM * DIM; i++)
    if (tmpTab[i] != 0) {
      notChange = false;
      break;
    }

  if (notChange) {
    ret = 1;
    break;
  }
  countIter = 1;
} else countIter++;
```

La terminaison fonctionne bien, j'obtiens presque le même nombre d'itérations que la version séquentielle (+20 itérations).

4.7.4 OpenCL + OpenMP Implementation

J'ai essayé d'implémenter la version *GPU + OpenMP* mais je n'ai pas réussi à l'implémenter à temps. Ma version a des problèmes avec la bordure du calcul entre le GPU et le CPU. Pour corriger, il faudrait que je partage quelque lignes de calcul entre les 2. Sur l'image suivante on peut voir le problème de partage :

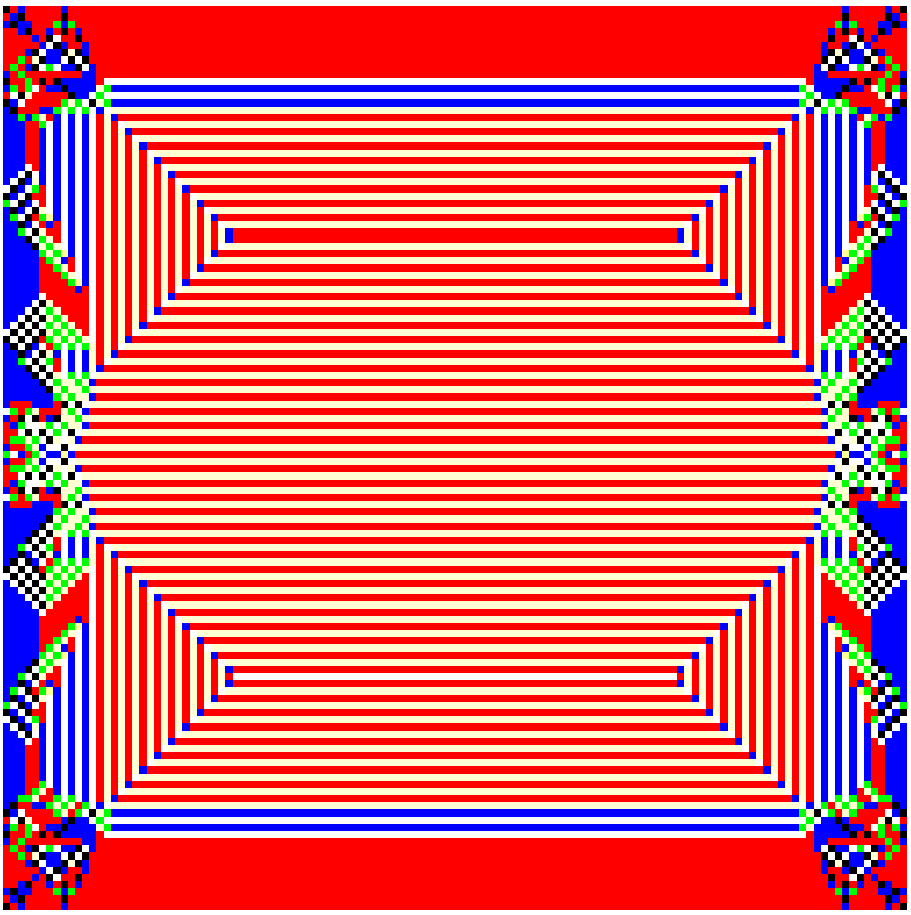

Figure 1 – ssandPile ocl_omp size 128, iterations 200