

PaP

# **Projet : rapport4**

4TIN804U

GOEDEFROIT Charles

2021-2022

## Table des matières

<b>4.7 OpenCL Implementation</b>	<b>2</b>
4.7.3 Detecting termination . . . . .	2
4.7.4 OpenCL + OpenMP Implementation . . . . .	3

## 4.7 OpenCL Implementation

### 4.7.3 Detecting termination

Pour implemter la terminaison j'ai commencer par ajouter un nouveau kernel ocl qui prend en parametre un buffer pour la terminaison. Je remplie ce buffer avec 1 si 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 ajouter la la fonction `ssandPile_invoke_ocl_term()` en m'inspirant de la fonction `ocl_invoke_kernel_generic()`. Les changement que j'ai apporter sont :

- un buffer nommer `term_buffer` que je passe en parametre au kernel ocl :

---

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

---

- 2 variables pour controller l'intervale d'iterations ou on verifie si l'états du tas de sable est stable.

---

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

---

- la copie du beffeur en ram avec la verification que le buffer soit remplie de 0. Qu'il n'y est 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 d'itération que la version séquentielle ( +20 iterations).

#### 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 quelques lignes de calcul entre les 2. Sur l'image suivante on peut voir le problème de partage :

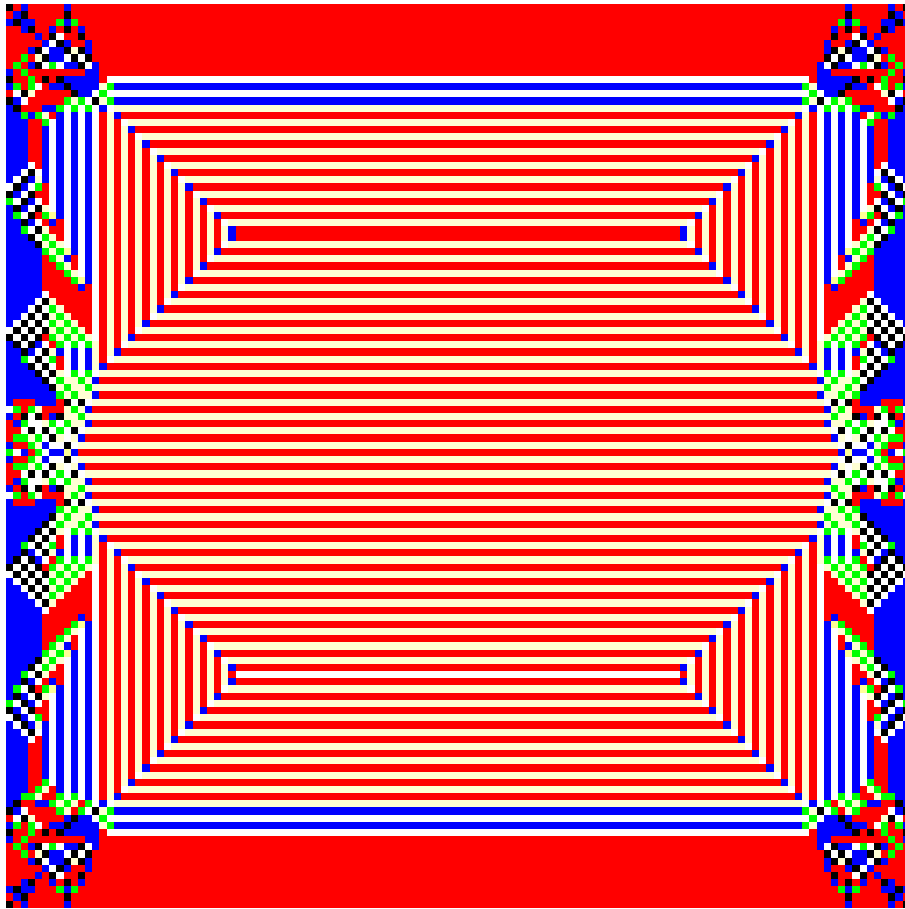


Figure 1 – ssandPile ocl\_omp size 128, iterations 200