



# C-Stolic: Algorithmes numériques

Projet Architectures Parallèles

Yvette Le Bras, Jean-Marie Le Yaouanc

Brest, 7 avril 2005

Enseignant responsable : Erwan Fabiani

**Département informatique**

UFR Sciences et Techniques 6 avenue Le Gorgeu 29200 Brest

Master 1 informatique

# Table des matières

<b>1</b>	<b>Problème 1 : PGCD</b>	<b>1</b>
1.1	Explications . . . . .	1
1.2	Influence de la taille du réseau . . . . .	1
1.3	Problèmes rencontrés . . . . .	2
1.4	Trace du programme . . . . .	2
<b>2</b>	<b>Problème 2 : nombre de diviseurs</b>	<b>4</b>
2.1	Explications . . . . .	4
2.2	Trace du programme . . . . .	4
<b>3</b>	<b>Problème 3 : Suite de Fibonacci avec origines quelconques</b>	<b>6</b>
3.1	Explication du choix de conception . . . . .	6
3.2	Trace du programme . . . . .	6
<b>A</b>	<b>Mise en oeuvre du PGCD</b>	<b>8</b>
A.1	Code du fichier pgcd.c . . . . .	8
A.2	Code du fichier pgcd.cs . . . . .	9
<b>B</b>	<b>Mise en oeuvre du nombre de diviseurs</b>	<b>11</b>
B.1	Code du fichier nbDiv.c . . . . .	11
B.2	Code du fichier nbDiv.cs . . . . .	12
<b>C</b>	<b>Mise en oeuvre de la suite de Fibonacci</b>	<b>14</b>
C.1	Code du fichier fibo.c . . . . .	14
C.2	Code du fichier fibo.cs . . . . .	15

# Table des figures

1.1	Représentation d'une cellule . . . . .	2
2.1	Représentation d'un réseau de 5 cellules . . . . .	5
3.1	Représentation d'une cellule . . . . .	7

# Chapitre 1

## Problème 1 : PGCD

### 1.1 Explications

Une cellule de base du réseau est constituée du composant **COMP**, défini dans le sujet, ainsi que de **OP4** qui permet d'affecter à  $e$  la plus grande valeur de  $x$  ou  $y$ , et à  $f$  la plus petite.

Le nombre nécessaire de cellules pour trouver le PGCD est ainsi réduit : en effet, si nous nous contentons de relier les composants **COMP** avec la sortie  $y$  allant à l'entrée  $x$ , la sortie  $b$  à l'entrée  $y$ , et que  $y > x$  alors la cellule n'effectue aucun calcul. Elle retourne  $b = x$  et  $y$ .

**OP4** permet donc selon  $x$  et  $y$  de “gagner” une cellule.

```
#Algorithme du composant OP4
entrées (x,y)
si (x>y)
    e=x ;
    f=y ;
sinon
    e=y ;
    f=x ;
sorties(e,f)
```

### 1.2 Influence de la taille du réseau

Lorsque le nombre de cellules du réseau est inférieur à 4, les résultats donnés sont exacts s'ils sont trouvés en une seule étape, sinon le résultat est zéro. Le nombre de cellules n'est alors pas suffisant pour déterminer le PGCD. Lorsque le réseau est composé de cinq cellules ou plus, les résultats

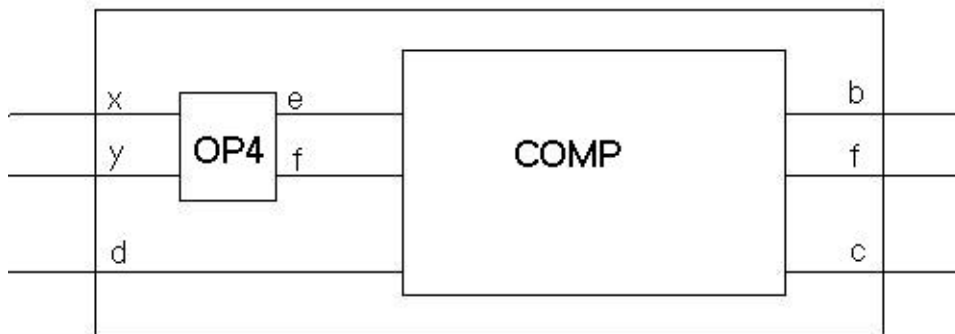


FIG. 1.1 – Représentation d'une cellule

sont toujours exacts.

### 1.3 Problèmes rencontrés

Deux problèmes (désormais résolus) sont apparus au niveau du vidage :

- Nous ne pouvons placer uniquement le tableau **tabR** suivi du résultat à lui affecter, car cela génère une erreur de compilation. Une variable *poubelle* est introduite, permettant de recueillir les résultats inutiles, mais surtout d'éviter l'erreur précédemment citée.
- La première valeur du tableau était totalement aléatoire. Pour remédier à ce problème, nous utilisons un registre *cBis*, qui permet d'envoyer la valeur du résultat de l'étape précédente.

### 1.4 Trace du programme

Notre programme fonctionne correctement. Son code est situé à l'annexe [A](#) page 8.

Number of cells = 10

Les tableaux d'entiers

Ta : 842 1271 144 91 40 841 65 84 85 400

Tb : 15 20 84 267 8 137 15 144 657 100

En systolic :

Tr : 1 1 12 1 8 1 5 12 1 100

Verification

Tr : 1 1 12 1 8 1 5 12 1 100

## Chapitre 2

# Problème 2 : nombre de diviseurs

### 2.1 Explications

La cellule de base est identique à celle explicitée dans le sujet. Comme précédemment une variable *poubelle* est utilisée, ainsi qu'un registre *dOut-Bis* permettant d'envoyer le résultat de l'étape précédente. Contrairement au problème 1, ce programme systolique utilise deux boucles.

Le réseau doit contenir un nombre de cellules égal au nombre d'éléments de l'ensemble  $E$ . En effet, chaque élément de  $E$  est placé dans une cellule lors d'une phase de remplissage. Puis les éléments restent figés dans leurs cellules durant les différents calculs.

### 2.2 Trace du programme

Notre programme fonctionne correctement. Son code est situé à l'annexe B page 11.

```
Number of cells = 5
```

```
L'ensemble des nombres premiers:  
E : 2 3 5 7 11
```

```
En systolic :  
f0 : 15 => 2  
f1 : 42 => 3
```

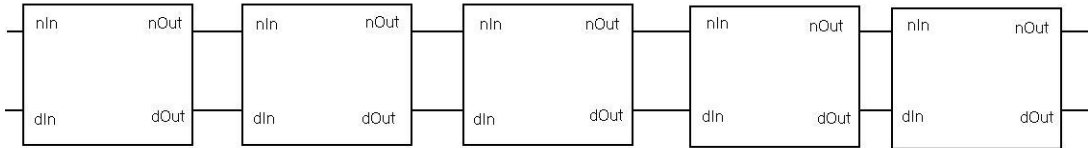


FIG. 2.1 – Représentation d'un réseau de 5 cellules

f2 : 9 => 1  
f3 : 17 => 0  
f4 : 2310 => 5

Verification :  
f0 : 15 => 2  
f1 : 42 => 3  
f2 : 9 => 1  
f3 : 17 => 0  
f4 : 2310 => 5

## Chapitre 3

# Problème 3 : Suite de Fibonacci avec origines quelconques

### 3.1 Explication du choix de conception

Une cellule de base a 2 entrées et 2 sorties. Pour la cellule d'indice  $i$ , les entrées sont  $f(i-1)$  et  $f(i)$ , les sorties sont  $f(i)$  et  $f(i+1)$ . Dans chaque cellule, une addition est suffisante pour obtenir la valeur de  $f(i+1)$ ,  $f(i-1)+f(i)$ .

Comme précédemment, une variable *poubelle* est utilisée pour permettre de remplir le tableau de résultats. Cette conception est choisie car elle nous paraît la plus simple et la plus évidente à mettre en oeuvre.

### 3.2 Trace du programme

Notre programme fonctionne correctement. Son code est situé à l'annexe **C** page 14.

Number of cells = 5

```
En systolic :  
f(0)=1 ; f(1)=2 ; f(6)=21  
f(0)=3 ; f(1)=5 ; f(6)=55  
f(0)=8 ; f(1)=9 ; f(6)=112  
f(0)=10 ; f(1)=15 ; f(6)=170  
f(0)=42 ; f(1)=72 ; f(6)=786
```

CHAPITRE 3. PROBLÈME 3 : SUITE DE FIBONACCI AVEC ORIGINES QUELCONQUES7

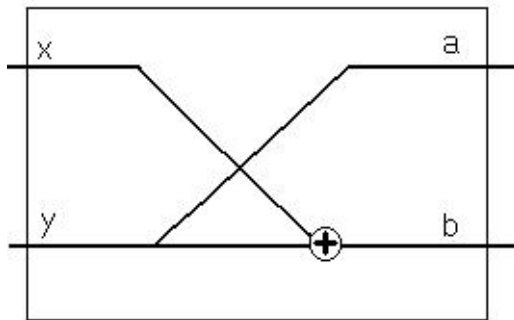


FIG. 3.1 – Représentation d'une cellule

Verification :

$$f(0)=1 ; f(1)=2 ; f(6)=21$$

$$f(0)=3 ; f(1)=5 ; f(6)=55$$

$$f(0)=8 ; f(1)=9 ; f(6)=112$$

$$f(0)=10 ; f(1)=15 ; f(6)=170$$

$$f(0)=42 ; f(1)=72 ; f(6)=786$$

## Annexe A

# Mise en oeuvre du PGCD

### A.1 Code du fichier pgcd.c

```
main(int argc, char * argv[])
{
    int i,m,n,tmp;
    // tableaux d'entiers (10)
    int Ta[NBCELL] = {842,1271,144,91,40,841,65,84,85,400}; // premiere entree
    int Tb[NBCELL] = {15,20,84,267,8,137,15,144,657,100}; // deuxieme entree
    int Tr[NBCELL]; // pour les resultats

    printf ("\nNumber of cells = %d\n\n",NBCELL);

    /* affiche les tableaux d'entree */
    printf("\n Les tableaux d'entiers \n");
    printf ("\n Ta : ");
    for (i=0;i<NBCELL;i++)
        printf ("%d ",Ta[i]);
    printf ("\n Tb : ");
    for (i=0;i<NBCELL;i++)
        printf ("%d ",Tb[i]);

    /* Fait appel au programme systolic*/
    CsOn();

    Pgcd(Ta,Tb,Tr);

    CsOff();
    /* affiche le resultat systolic */
```

```

printf ("\n\n En systolic : ");
printf ("\n Tr : ");
for (i=0;i<NBCELL;i++)
    printf ("%d ",Tr[i]);
printf ("\n");

/* affiche le résultat séquentiel */
printf ("\n Verification\n Tr : ");
for (i=0; i<NBCELL;i++)
{
    m=Ta[i]; // m stock la premiere entree
    n=Tb[i]; // n stock la deuxieme entree
    while (n!=0) // calcul le pgcd
    {
        tmp=n;
        n=m%n;
        m=tmp;
    }
    printf ("%d ",m);
}
printf ("\n");
}

```

## A.2 Code du fichier pgcd.cs

```

void Pgcd(int tabX[NBCELL],int tabY[NBCELL],int tabR[NBCELL] )
{
    int i;
    systolic int x,y,d,b,c,a,e,f,poubelle,cBis;

    for (i=0;i<NBCELL*2; i++)
    {
        x => b : tabX[i] ;
        y => f : tabY[i] ;
        d => c : 0;
        e = (x<y) ? y : x ; // OP4
        f = (x<y) ? x : y ; // OP4
        a = (f==0) ? 1 : f ; // OP2
        b = (e % a) ; // OP1
        c = ((d==0) && (b==0)) ? f : d ; // OP3
        poubelle : tabR[i-NBCELL] => cBis;
    }
}

```

```
        cBis = c;  
    }  
}
```

## Annexe B

# Mise en oeuvre du nombre de diviseurs

### B.1 Code du fichier nbDiv.c

```
main(int argc, char * argv[])
{
    int i,j,n;
    int E[NBCELL] = {2,3,5,7,11}; // ensemble E
    int F[NBCELL]={15,42,9,17,2310}; // les nombres
    int res[NBCELL]; // les resultats

    printf ("\nNumber of cells = %d\n\n",NBCELL);

    printf("\nL'ensemble des nombres premiers ");
    printf ("\n E : ");
    for (i=0;i<NBCELL;i++)
        printf ("%d ",E[i]);
    printf ("\n ");

    // appel au programme systolic
    CsOn();

    Nbdiv(E,F,res);

    CsOff();

    // affiche resultat en systolic
    printf ("\n En systolic : ");
    for (i=0;i<NBCELL;i++)
        printf ("\n f%d : %d => %d ",i,F[i],res[i]);
}
```

```

printf ("\n");

// affiche resultat en sequentiel
printf ("\n Verification : ");
for (i=0; i<NBCELL;i++)
{
    n=0;
    // calcul le nombre de diviseur
    for (j=0; j<NBCELL;j++)
    {
        if ((F[i]%E[j])==0) n++;
    }
    printf ("\n f%d : %d => %d ",i,F[i],n);
}
printf ("\n");
}

```

## B.2 Code du fichier nbDiv.cs

```

void Nbdiv(int tabE[NBCELL],int tabF[NBCELL],int tabR[NBCELL] )
{
    int i;
    systolic int e,nIn,dIn,dOut,a,f,poubelle,dOutBis,nOut;

    //remplissage des nb premiers
    for (i=0;i<NBCELL;i++)
        e => e : tabE[i];

    for (i=0;i<NBCELL;i++)
    {
        nIn => nOut : tabF[i];
        dIn => dOut : 0;
        a = (((nIn%e) == 0) ? 1 : 0); //OP1
        dOut = ((a == 1) ? (dIn + 1) : dIn); //OP2
        nOut = nIn;
        dOutBis = dOut;
    }
    for (i=0;i<NBCELL;i++)
    {
        nIn => nOut;
        dIn => dOut;
    }
}

```

```
    a = (((nIn%e) == 0) ? 1 : 0); //OP1
    dOut = ((a == 1) ? (dIn + 1) : dIn); //OP2
    poubelle : tabR[i] => dOutBis;
    dOutBis = dOut;
    nOut = nIn;
  }
}
```

## Annexe C

# Mise en oeuvre de la suite de Fibonacci

### C.1 Code du fichier fibo.c

```
main(int argc, char * argv[])
{
    int i,j,u,z,tmp;
    int Zero[NBCELL] = {1,3,8,10,42}; // valeur de f(0)
    int Un[NBCELL]= {2,5,9,15,72}; // valeur de f(1)
    int res[NBCELL]; // resultats

    printf ("\nNumber of cells = %d\n\n",NBCELL);

    /* appel au programme systolic */
    CsOn();

    Fibonacci(Zero,Un,res);

    CsOff();

    /* Resultat en systolic */
    printf ("\n En systolic : ");
    for (i=0;i<NBCELL;i++)
        printf ("\n f(0)=%d ; f(1)=%d ; f(%d)=%d ",Zero[i],Un[i],NBCELL+1,res[i]);
    printf ("\n");

    /* Resultat en sequentiel */
    printf ("\n Verification : ");
    for (i=0; i<NBCELL;i++)
```

```

    {
        z=Zero[i]; // f(0)
        u=Un[i]; // f(1)
        for (j=0; j<NBCELL;j++)
            {
                tmp=z+u; // f(j+1)
                z=u;
                u=tmp;
            }
        printf ("\n f(0)=%d ; f(1)=%d ; f(%d)=%d ",Zero[i],Un[i],NBCELL+1,u);
    }
    printf ("\n");
}

```

## C.2 Code du fichier fibo.cs

```

void Fibonacci(int tabX[NBCELL],int tabY[NBCELL],int tabR[NBCELL] )
{
    int i;
    systolic int x,y,a,b,poubelle;

    for (i=0;i<NBCELL*2; i++)
        {
            x => y : tabX[i] ;
            y => b : tabY[i] ;
            b = x+y ;
            poubelle : tabR[i-(NBCELL-1)] => b ;
        }
}

```