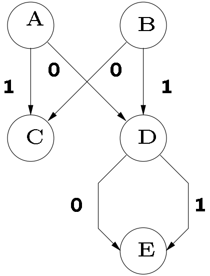
Outline

1. Introduction au cours
   - Compilation et optimisations de codes
   - Des p'tites boucles, toujours des p'tites boucles
   - Exemples de spécificités architecturales
2. Pipeline logiciel
   - Sans contraintes de ressources
   - Compaction de boucles et retiming
   - Optimisations des durées de vie
3. Fusion de boucles
   - Intérêts et problèmes
   - Fusion de boucles simple : variantes
   - Fusion avec décalage

Fusion with loop shifting

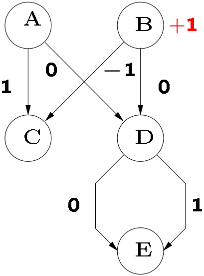
```
DO i=2, n
   a(i) = f(i)
   b(i) = g(i)
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
   e(i) = d(i-1) + d(i)
ENDDO
```




```
DOPAR i=2, n
   a(i) = f(i)
   b(i) = g(i)
ENDDOPAR
DOPAR i=2, n
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
ENDDOPAR
DOPAR i=2, n
   e(i) = d(i-1) + d(i)
ENDDO
```

Fusion with loop shifting

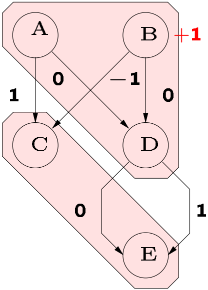
```
DO i=2, n
   a(i) = f(i)
   b(i) = g(i)
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
   e(i) = d(i-1) + d(i)
ENDDO
```




```
a(2) = f(2)
DOPAR i=3, n
   a(i) = f(i)
   b(i-1) = g(i-1)
ENDDOPAR
b(n) = g(n)
DOPAR i=2, n
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
ENDDOPAR
DOPAR i=2, n
   e(i) = d(i-1) + d(i)
ENDDO
```

Fusion with loop shifting

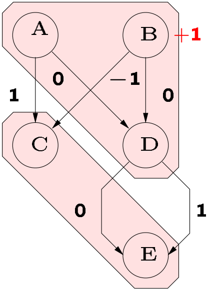
```
DO i=2, n
   a(i) = f(i)
   b(i) = g(i)
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
   e(i) = d(i-1) + d(i)
ENDDO
```




```
a(2) = f(2)
d(2) = a(2) + b(1)
DOPAR i=3, n
   a(i) = f(i)
   b(i-1) = g(i-1)
   d(i) = a(i) + b(i-1)
ENDDOPAR
b(n) = g(n)
DOPAR i=2, n
   c(i) = a(i-1) + b(i)
   e(i) = d(i-1) + d(i)
ENDDOPAR
```

Fusion with loop shifting

```
DO i=2, n
   a(i) = f(i)
   b(i) = g(i)
   c(i) = a(i-1) + b(i)
   d(i) = a(i) + b(i-1)
   e(i) = d(i-1) + d(i)
ENDDO
```




```
a(2) = f(2)
d(2) = a(2) + b(1)
DOPAR i=3, n
   a(i) = f(i)
   b(i-1) = g(i-1)
   d(i) = a(i) + b(i-1)
ENDDOPAR
b(n) = g(n)
DOPAR i=2, n
   c(i) = a(i-1) + b(i)
   e(i) = d(i-1) + d(i)
ENDDOPAR
```

➽ Yet another retiming problem.
But strongly NP-complete (∼ scheduling with communications).

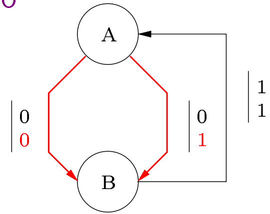
Multi-dimensional loop shifting and parallel loops

In dimension one :

- Minimal number of parallel loops with shifting : NP-complete.
- Complete fusion possible ⇔ any (undirected) cycle weight is 0.

In 2D, find an outer shift that enables the complete inner fusion.

```
DO i=1, n-1
  DO j=1, n-1
    a(i,j) = b(i-1,j-1)
    b(i,j) = a(i,j) + a(i,j-1)
  ENDDO
ENDDO
```




```
DO i=1, n-1
  DOPAR j=1, n-1
    a(i,j) = b(i-1,j-1)
  ENDDOPAR
  DOPAR j=1, n-1
    b(i,j) = a(i,j) + a(i,j-1)
  ENDDOPAR
ENDDO
```

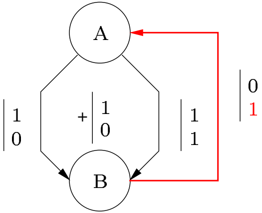
Multi-dimensional loop shifting and parallel loops

In dimension one :

- Minimal number of parallel loops with shifting : NP-complete.
- Complete fusion possible ⇔ any (undirected) cycle weight is 0.

In 2D, find an outer shift that enables the complete inner fusion.

```
DO i=1, n-1
   DO j=1, n-1
      a(i,j) = b(i-1,j-1)
      b(i,j) = a(i,j) + a(i,j-1)
   ENDDO
ENDDO
```




```
DO i=1, n
   DOPAR j=1, n-1
      IF (i>1) THEN
         b(i-1,j) = a(i-1,j) + a(i-1,j-1)
   ENDDOPAR
   DOPAR j=1, n-1
      IF (i<n) THEN
         a(i,j) = b(i-1,j-1)
   ENDDOPAR
ENDDO
```

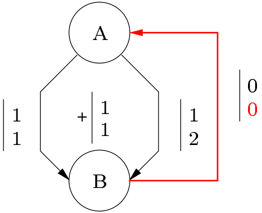
Multi-dimensional loop shifting and parallel loops

In dimension one :

- Minimal number of parallel loops with shifting : NP-complete.
- Complete fusion possible ⇔ any (undirected) cycle weight is 0.

In 2D, find an outer shift that enables the complete inner fusion.

```
DO i=1, n-1
   DO j=1, n-1
      a(i,j) = b(i-1,j-1)
      b(i,j) = a(i,j) + a(i,j-1)
   ENDDO
ENDDO
```




```
DO i=1, n
   DOPAR j=1, n
      IF (i>1) and (j>1) THEN
         b(i-1,j-1) = a(i-1,j-1) + a(i-1,j-2)
      IF (i<n) and (j<n) THEN
         a(i,j) = b(i-1,j-1)
   ENDDOPAR
ENDDO
```

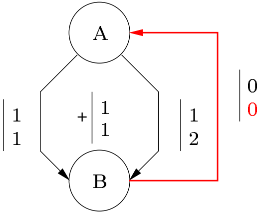
Multi-dimensional loop shifting and parallel loops

In dimension one :

- Minimal number of parallel loops with shifting : NP-complete.
- Complete fusion possible ⇔ any (undirected) cycle weight is 0.

In 2D, find an outer shift that enables the complete inner fusion.

```
DO i=1, n-1
   DO j=1, n-1
      a(i,j) = b(i-1,j-1)
      b(i,j) = a(i,j) + a(i,j-1)
   ENDDO
ENDDO
```




```
DO i=1, n
   DOPAR j=1, n
      IF (i>1) and (j>1) THEN
         b(i-1,j-1) = a(i-1,j-1) + a(i-1,j-2)
      IF (i<n) and (j<n) THEN
         a(i,j) = b(i-1,j-1)
   ENDDOPAR
ENDDO
```

➽ Strongly NP-complete.

Parallelism detection : more loop transformations needed

Is there some loop parallelism (i.e., parallel loop iterations) in the following two codes ? What is their degree of parallelism ?

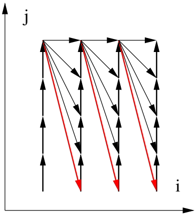
```
DO i=1, N
  DO j=1, N
    a(i,j) = c(i,j-1)
    c(i,j) = a(i,j) + a(i-1,N)
  ENDDO
ENDDO
```

```
DO i=1, N
  DO j=1, N
    a(i,j) = c(i,j-1)
    c(i,j) = a(i,j) + a(i-1,j)
  ENDDO
ENDDO
```

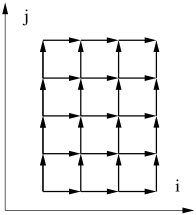
Parallelism detection : more loop transformations needed

Is there some loop parallelism (i.e., parallel loop iterations) in the following two codes ? What is their degree of parallelism ?

```
DO i=1, N
  DO j=1, N
    a(i,j) = c(i,j-1)
    c(i,j) = a(i,j) + a(i-1,N)
  ENDDO
ENDDO
```

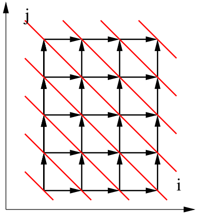



```
DO i=1, N
  DO j=1, N
    a(i,j) = c(i,j-1)
    c(i,j) = a(i,j) + a(i-1,j)
  ENDDO
ENDDO
```

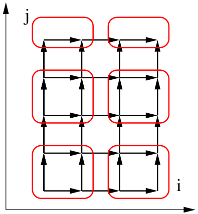


Loop tiling, blocked algorithms

```
DO t=2, 2N
  DOPAR j=max(1,t-N), min(N,t-1)
    a(t-j,j) = c(t-j,j-1)
    c(t-j,j) = a(t-j,j) + a(t-j-1,j)
  ENDDO
ENDDO
```




```
DO I=1, N, B
  DO J=1, N, B
    DO i=I, min(I+B-1,N)
      DO j=J, min(J+B-1,N)
        a(i,j) = c(i,j-1)
        c(i,j) = a(i,j) + a(i-1,j)
      ENDDO
    ENDDO
  ENDDO
ENDDO
```



➽ Tiling and parallelism detection : similar problem.