

État de l'art du calcul sur support reconfigurable dynamiquement

Quelques perspectives et éléments de R&D

Lörinc Antoni

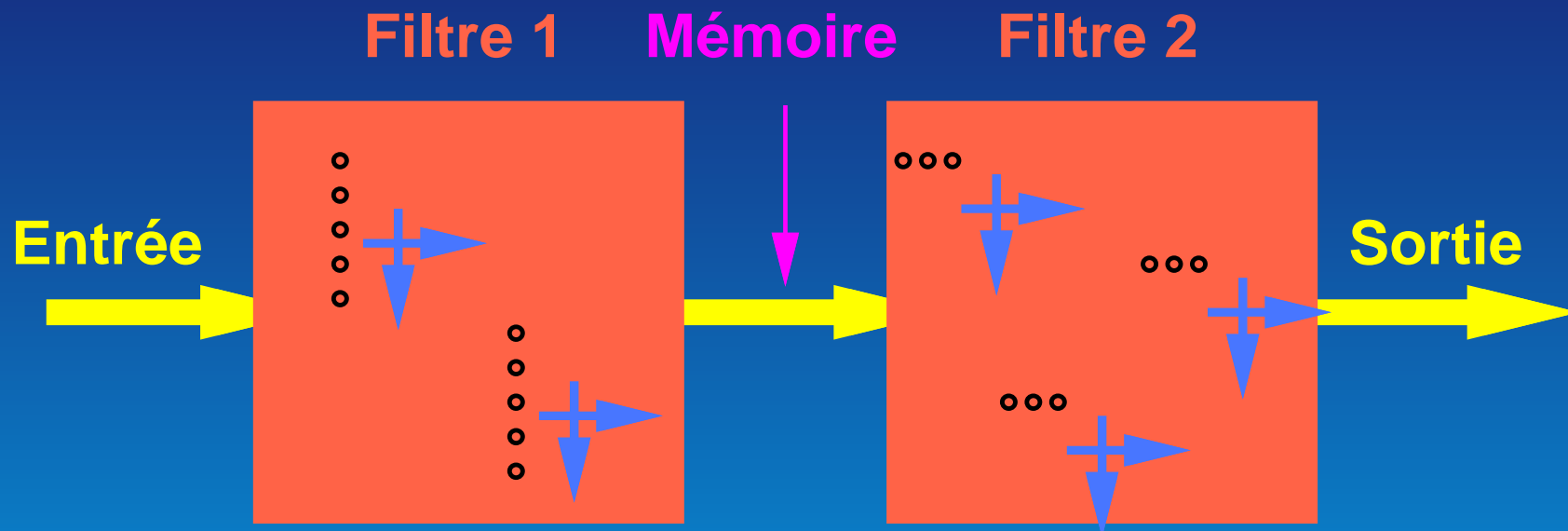
Ronan Keryell

Gérald Ouvradou

Département Informatique de
l'École Nationale Supérieure des Télécommunications de Bretagne

4 juin 1999

- Projeter une (grande) architecture virtuelle sur du (petit) matériel
- Compromis coût-performance
- \rightsquigarrow Trouver l'« application qui tue »



- Composition de différents filtres de taille et/ou forme différentes
- Calculs et communication en $\mathcal{O}(N^2)$
- Évaluation partielle : « tabulation » des opérateurs
- Cas général des traitements multirésolution

- Localisation en fréquence *et* en espace
- Opérateurs de différentes tailles
- Compromis entre taille et nombre des opérateurs utilisables en parallèle
- Reconfiguration de la machine pour chaque taille d'opérateur

- Changer les coefficients des filtres avec évaluation partielle
- Changer la configuration du filtre

- Beaucoup de calculs simples
- Multiplexage temporel de différentes couches
- Apprentissage :
 - ▶ Changement *dynamique* des coefficients
 - ▶ Changement *dynamique* du nombre de neurones

- Si algorithme asymétrique, 2 configurations distinctes :
 1. Chiffrement
 2. Déchiffrement
- Spécialisation du matériel en fonction de chaque clé


- Multiplexage temporel de fonctionnalités indépendantes ou pas
- Simulation d'un gros système ou circuit
- Chaîne de traitement d'image
- Coupleur ATM + cryptage
- Couplé au contexte Unix pour gérer différents utilisateurs
 ~> intérêt des FPGAs multi-contexte

En FortranHDL :

```
program double_filtre
integer i,j,n,t
integer longtemps,tx,ty
parameter (tx = 512)
parameter (ty = 512)
integer op1x,op1y,op2x,op2y
parameter (op1x = 1)
parameter (op1y = 5)
parameter (op2x = 3)
parameter (op2y = 1)
integer image(1:tx,1:ty), image2(1:tx,1:ty), image3(1:tx,1:ty)
integer coeff1(-op1y/2:op1y/2), coeff2(-op2x/2:op2x/2)
do t = 1, longtemps
  do i = 1, tx
```

```
        do j = 1, ty
            call entree(image(i,j))
        enddo
    enddo
do i = 1, tx
    do j = 1, ty
        image2(i,j) = 0
        if (j .ge. 1 + op1y/2 .and. j .le. ty - op1y/2) then
            do n = -op1y/2, op1y/2
                image2(i,j) = image2(i,j) + image(i,j+n)*coeff1
            enddo
        endif
    enddo
enddo
do i = 1, tx
```

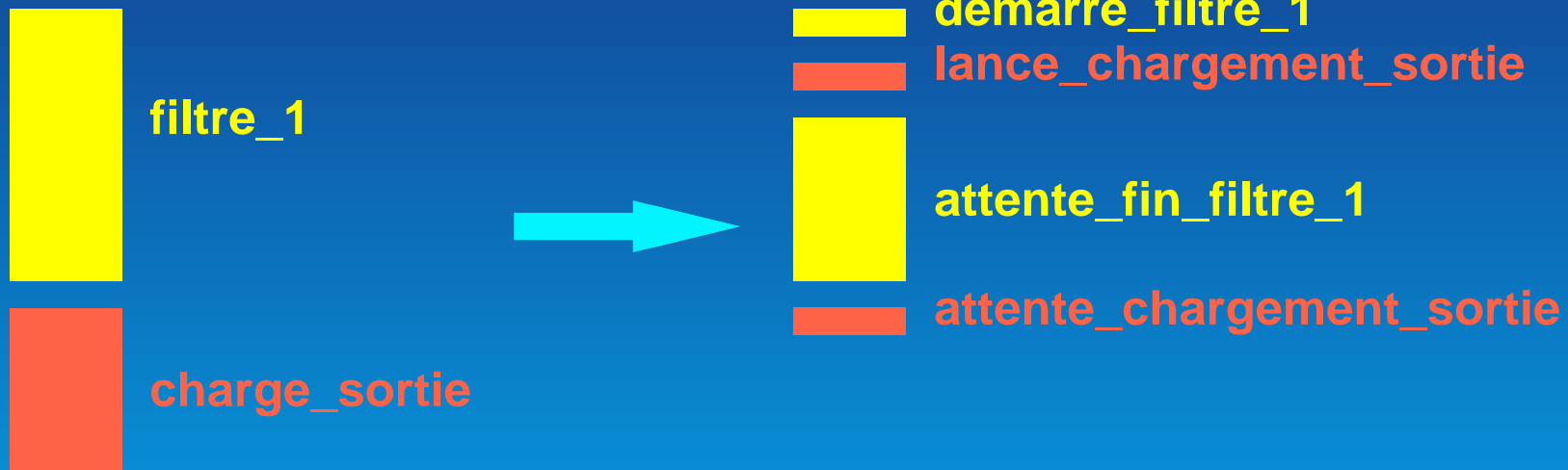
```
do j = 1, ty
  image3(i,j) = 0
  if (i.ge.1 + op2x/2 .and. i.le.tx - op2x/2) then
    do n = -op2x/2, op2x/2
      image3(i,j) = image3(i,j) + image2(i+n,j)*coeff
    enddo
  endif
enddo
do i = 1, tx
  do j = 1,ty
    call sortie(image3(i,j))
  enddo
enddo
end
```


- But : aider le travail manuel
- Cible : Ardoise (action ARP-ISIS, DSP+FPGA)
- Transformer & paralléliser le code (boucles,...)
- Simplification du code (évaluation partielle,...)
- Extraire les macro-fonctions et sous-traiter à une chaîne de compilation FPGA
- Planifier l'exécution
- Préchargement des données et des reconfigurations, pavage multidimensionnel avec pipeline logiciel,...
- Générer le code DSP d'infrastructure
-  Comment choisir parmi toutes ces techniques ?

Code tournant sur le DSP d'Ardoise avec 3 zones programmables

```
program double_filtre_DSP
integer t, longtemps
! Charge les configurations statiques :
call charge_entree
call charge_sortie
! Le temps qui passe :
do t = 1, longtemps
    call entree
    call charge_filtre_1
    call filtre_1
    call charge_filtre_2
    call filtre_2
    call sortie
enddo
end
```

Ardoise avec 2 zones programmables



→ Masquage des reconfigurations filtre_1 et sortie

Code tournant sur le DSP d'Ardoise avec 2 zones programmables
(E/S sur une zone et filtres sur une autre)

```
program double_filtre_DSP
integer t
integer longtemps
!   Prologue du pipeline :
↓   call lance_chargement_filtre_1
```



```
!      Le temps qui passe avec pipeline :
do t = 1, longtemps
  call charge_entree
  call entree
  ↑      call attente_chargement_filtre_1
  ↓
  ↓      call démarre_filtre_1
  ↓
  ↑      call lance_chargement_sortie
  ↑      call attente_fin_filtre_1
  call charge_filtre_2
  call filtre_2
  ↑      call attente_chargement_sortie
  ↓      call démarre_sortie
  ↓      call lance_chargement_filtre_1
  ↑      call attente_sortie
enddo
end
```

List of Slides

- 1 Intérêt du reconfigurable dynamique
- 2 Filtrage
- 3 Transformée en ondelettes
- 4 Filtrage adaptatif
- 5 Réseaux de neurones
- 6 Cryptographie
- 7 Multitâche en général
- 8 Exemple — filtrage
- 12 Techniques de compilation utilisables
- 13 Exemple de code à générer
- 14 Version pipelinée — principe
- 15 Version pipelinée — le code
- 17 Table des matières