Implantation optimisée d'algorithmes sur architecture configurable

Implantation optimisée d'algorithmes sur architecture configurable

Mots-clefs:
Méthodologie AAA,
Caractérisation, Optimisation,
Traitement bas niveau d'images,
Composant programmable,
SynDEx

Implantation optimisée d'algorithmes sur architecture configurable

- Modèle AAA
- Caractérisation
- Optimisation
- Implantation dans *SynDEx*
- Exemple d'application

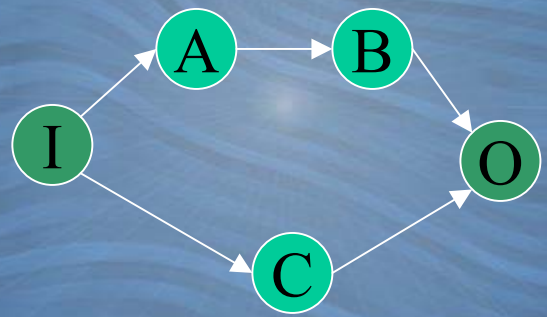
Modèle AAA

Objectif

- Spécifier un algorithme
- Exhiber le parallélisme potentiel
- Exploiter la régularité algorithmique

Modèle AAA

Graphe de dépendance de données



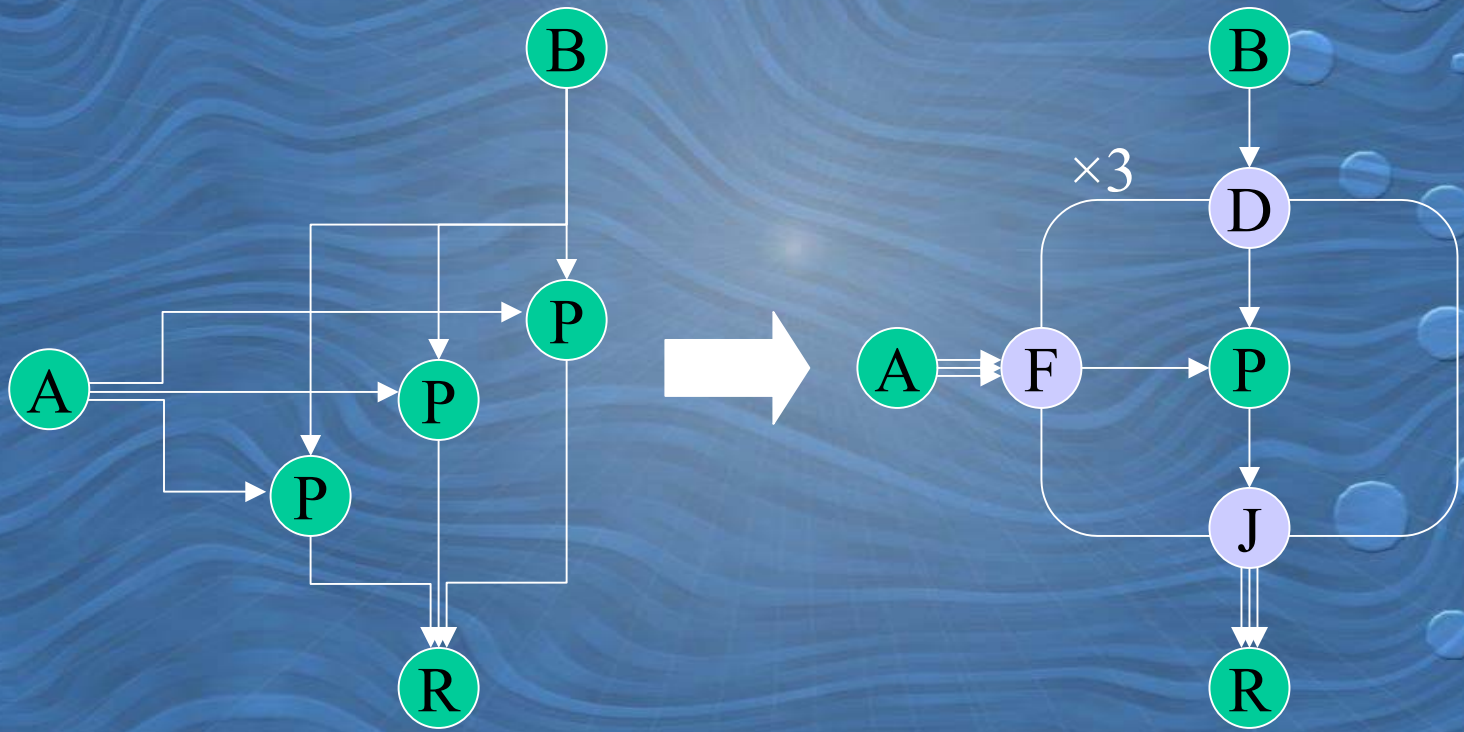
Modèle AAA

Graphe factorisé

$$\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix} \cdot \begin{pmatrix} b1 \\ b2 \\ b3 \end{pmatrix} = \begin{pmatrix} a11.b1+a12.b2+a13.b3 \\ a21.b1+a22.b2+a23.b3 \\ a31.b1+a32.b2+a33.b3 \end{pmatrix}$$

Modèle AAA

Fork, Diffuse & Join



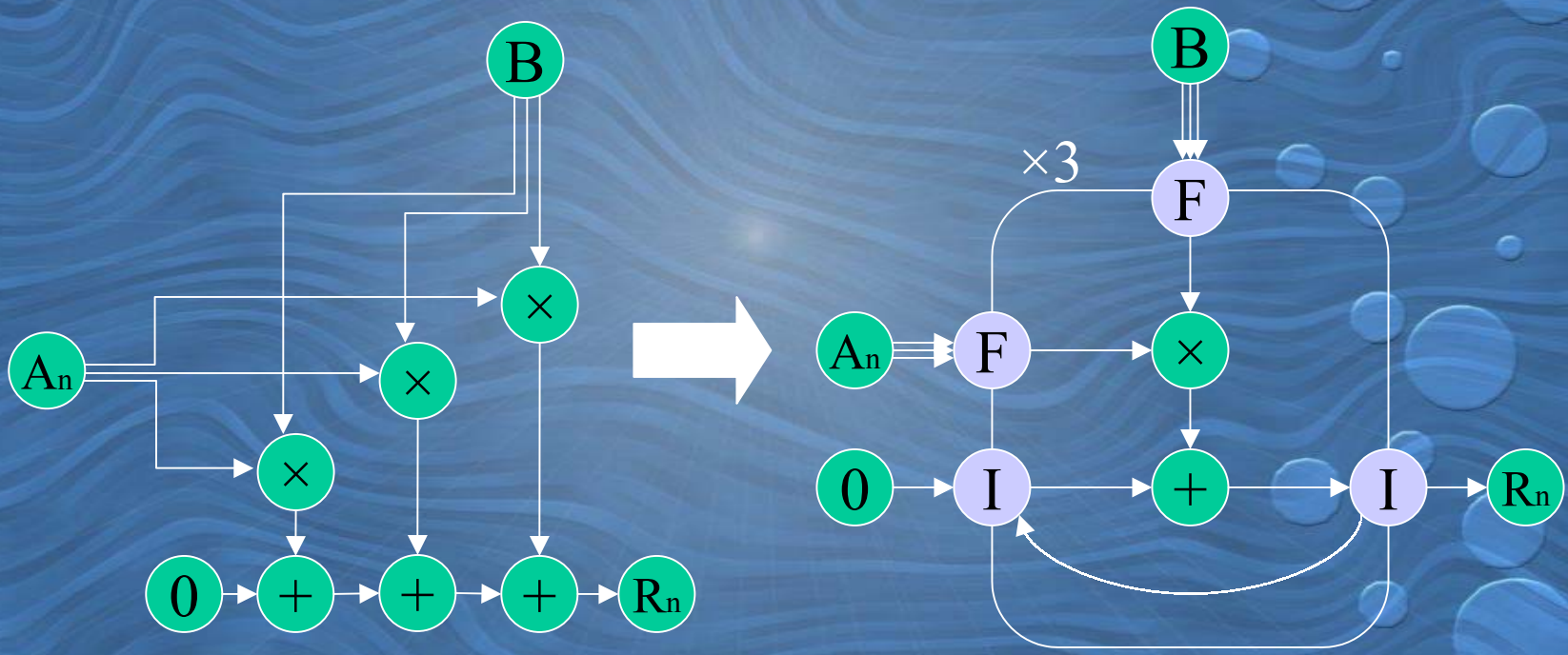
Modèle AAA

Iterate

$$\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a333 \end{pmatrix} \bullet \begin{pmatrix} b1 \\ b2 \\ b3 \end{pmatrix} = \begin{pmatrix} a11.b1+a12.b2+a13.b3 \\ a21.b1+a22.b2+a23.b3 \\ a31.b1+a32.b2+a33.b3 \end{pmatrix}$$

Modèle AAA

Iterate



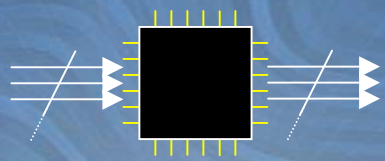
Modèle AAA

Frontière:
- Compacité d'expression d'un algorithme
- Exhiber les parties réutilisables de l'implémentation

Port:
- Répartir les données entre les répétitions d'une frontière.

Modèle AAA

Entrées-sorties

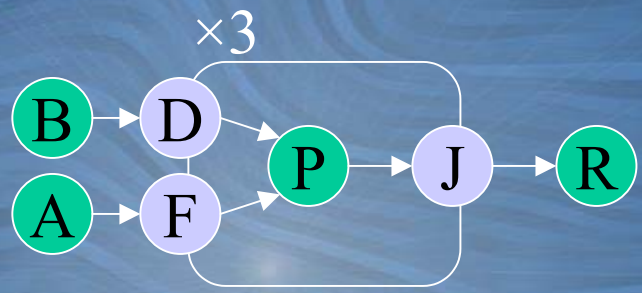


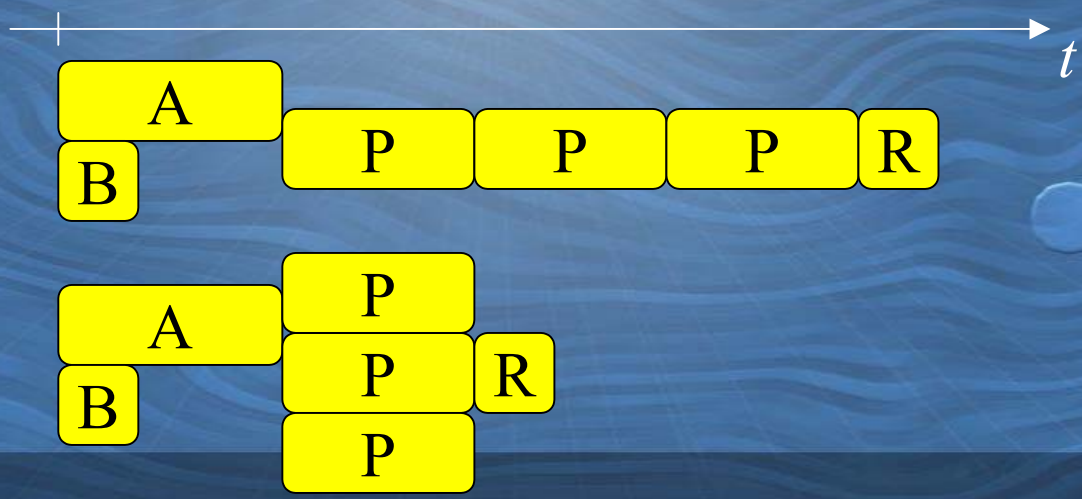
Port infini:

- Répartir les données entre les répétitions d'une frontière.
  ➡ Frontière infinie.

Modèle AAA

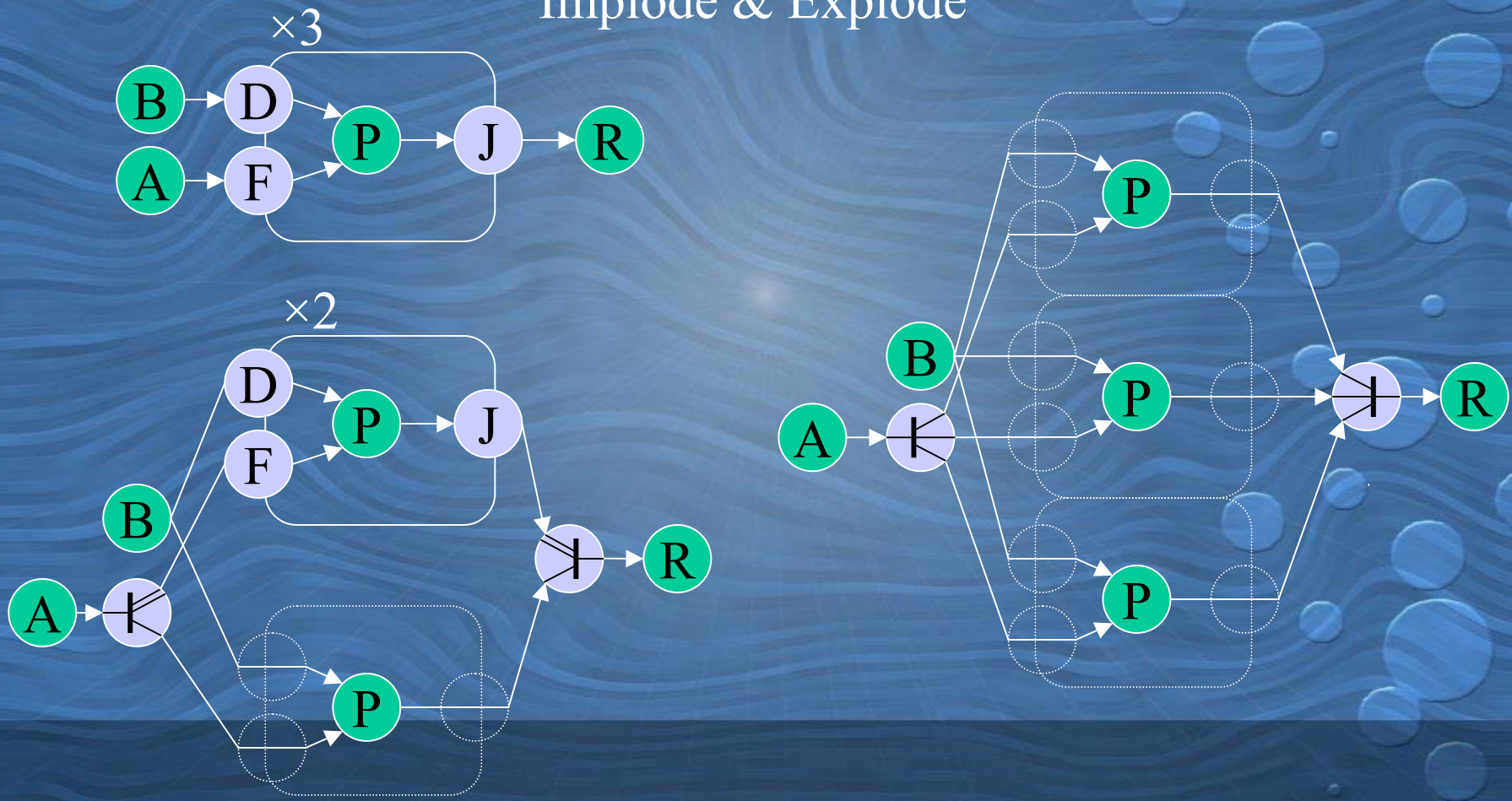
Factorisation et performances





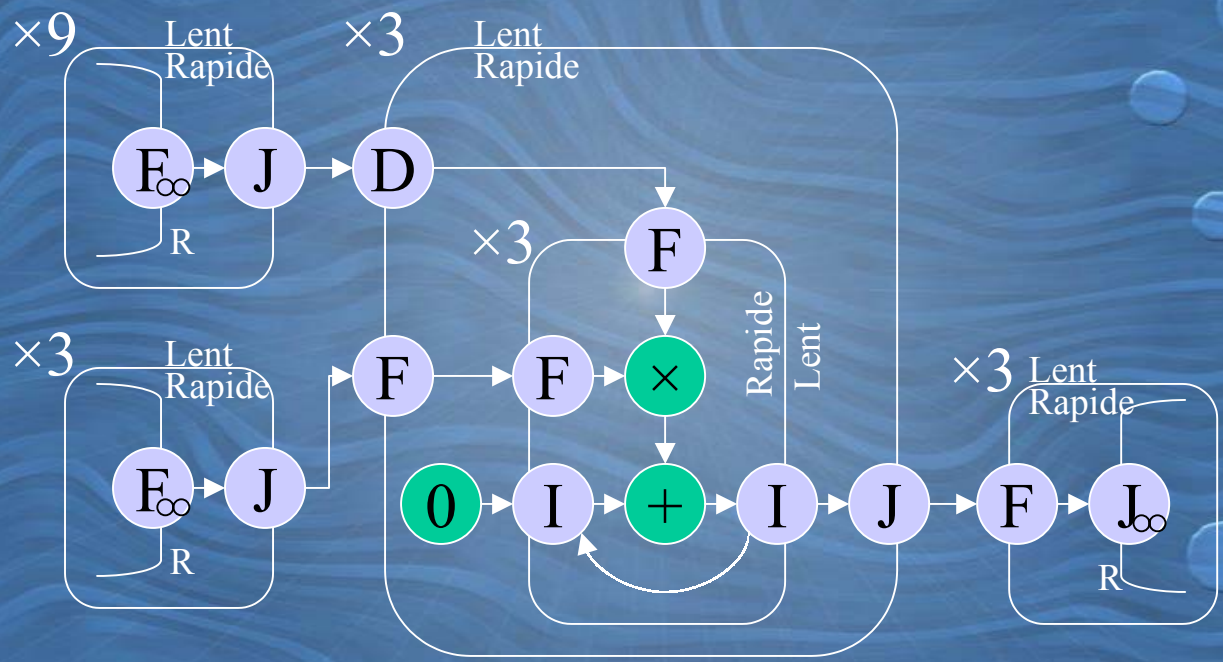
Modèle AAA

Implode & Explode



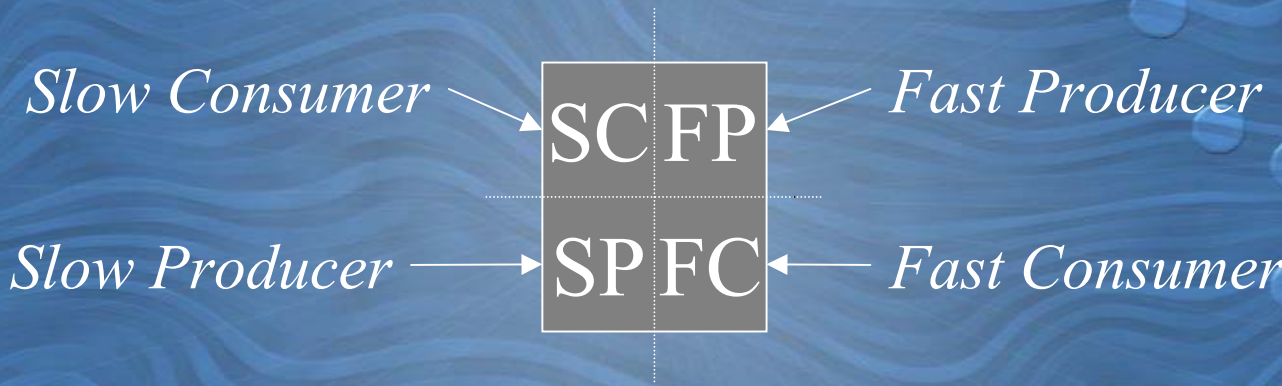
Modèle AAA

Graphe de voisinage de frontières



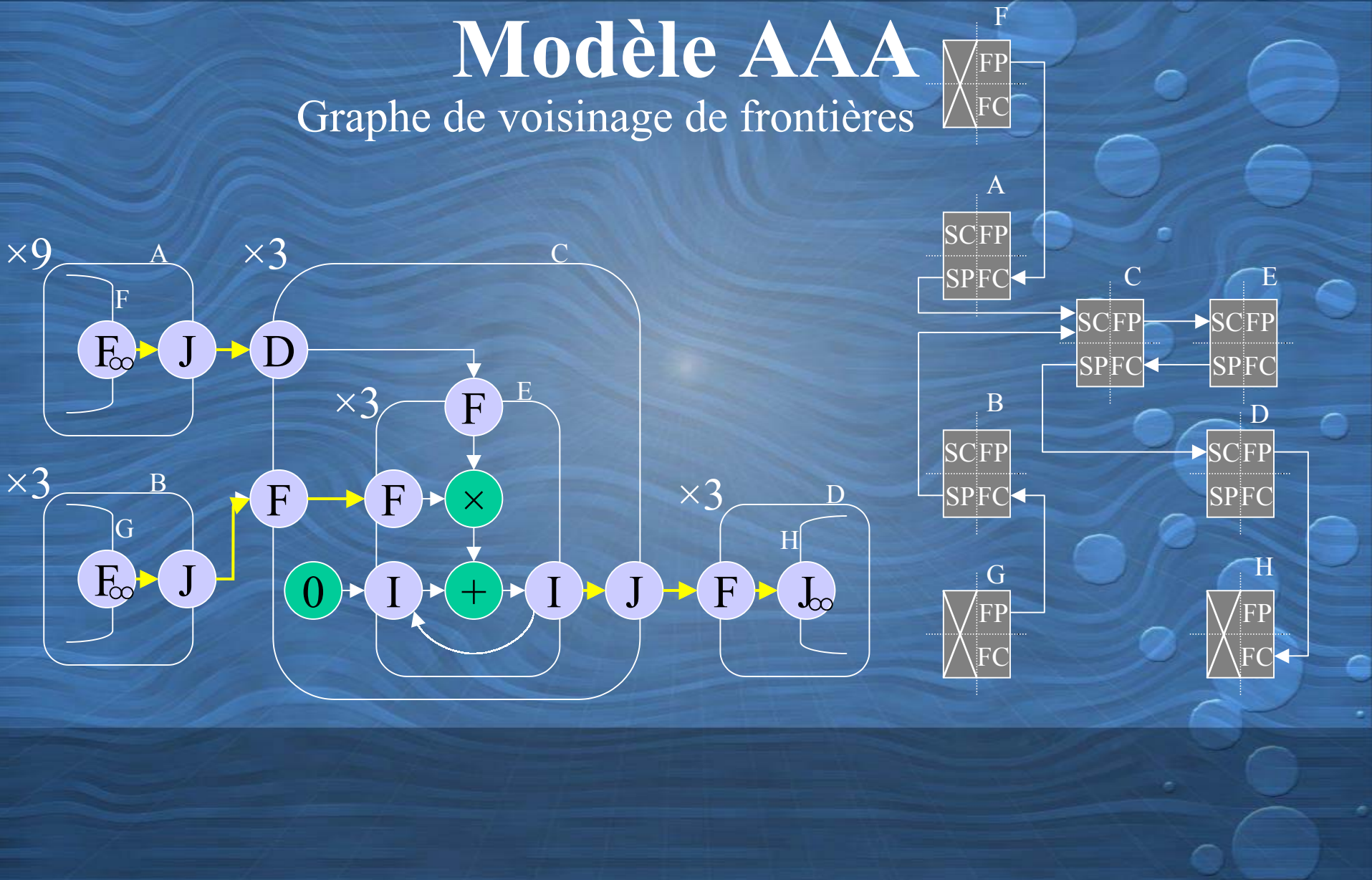
Modèle AAA

Graphe de voisinage de frontières



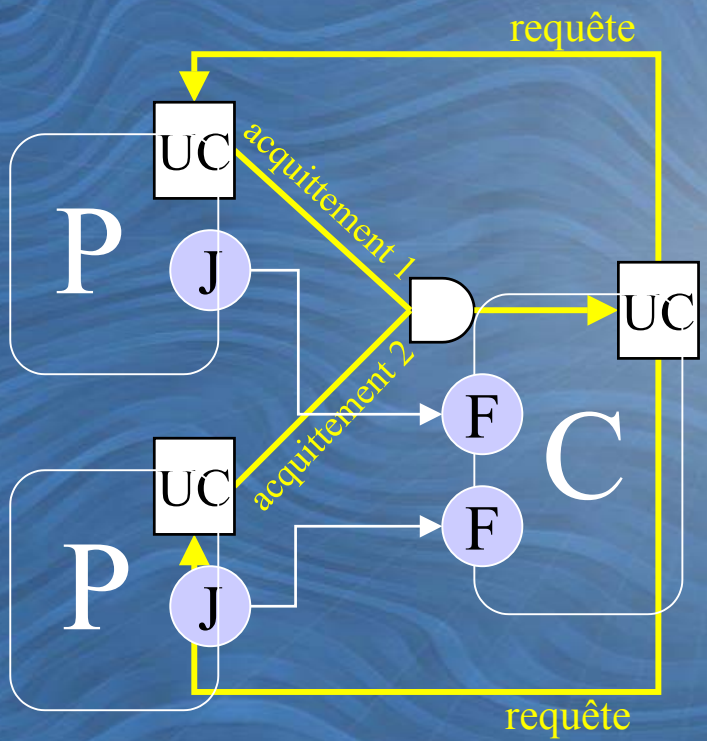
Modèle AAA

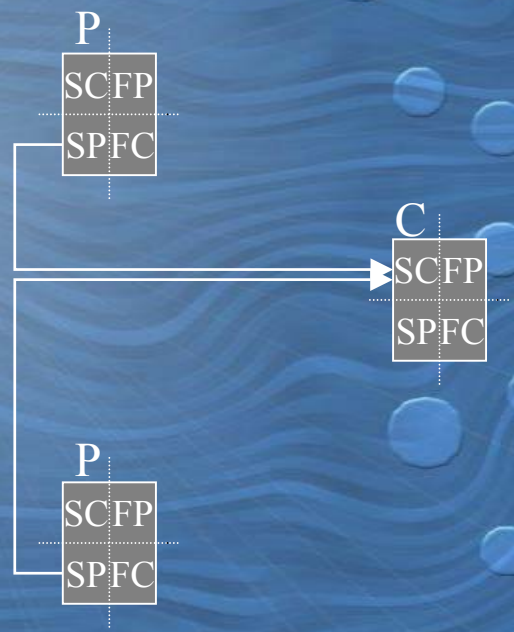
Graphe de voisinage de frontières



Modèle AAA

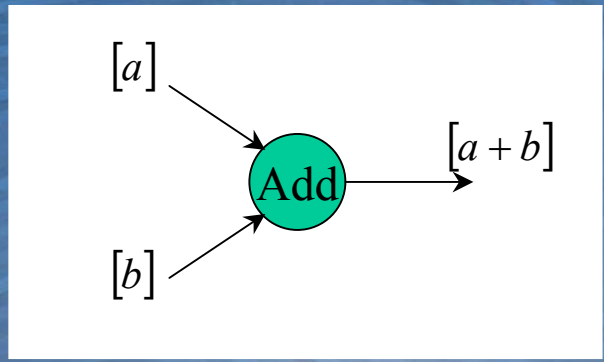
Système de contrôle des frontières.





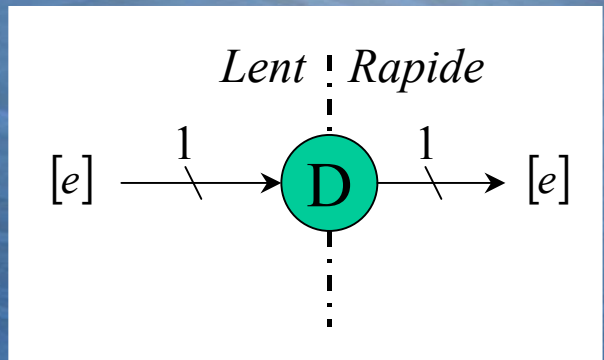
Implantation optimisée d'algorithmes sur architecture configurable

Calcul



Logique

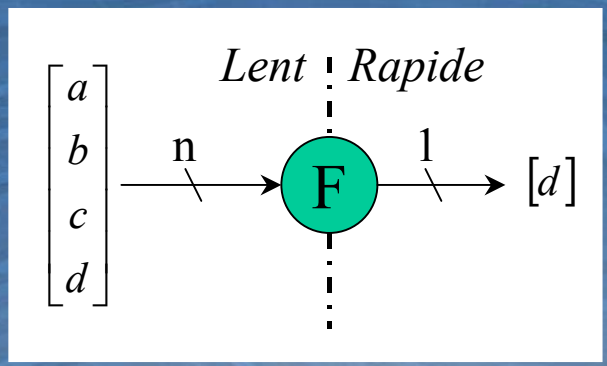
Diffusion



Câblage

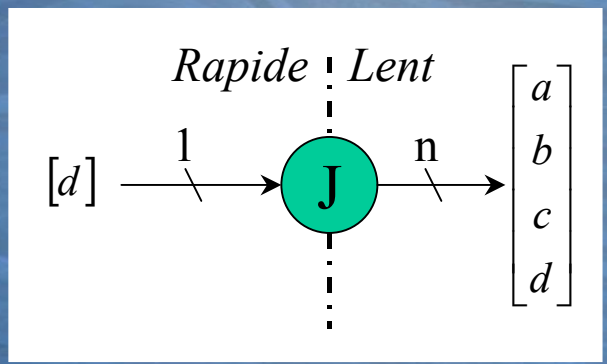
Implantation optimisée d'algorithmes sur architecture configurable

Fork



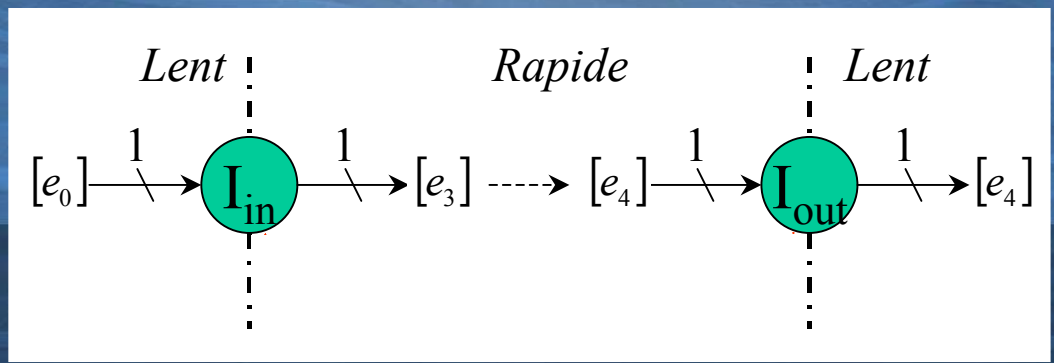
Multiplexeurs

Join



Banc de Registres

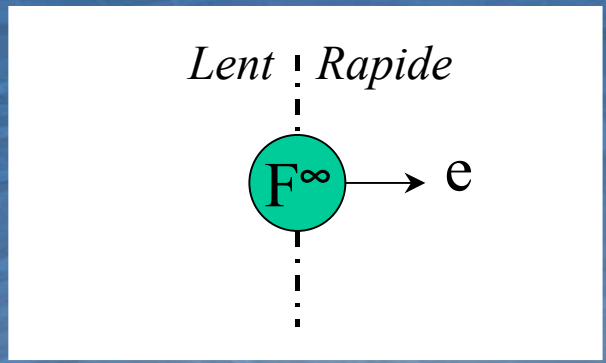
Iterate



1 registre

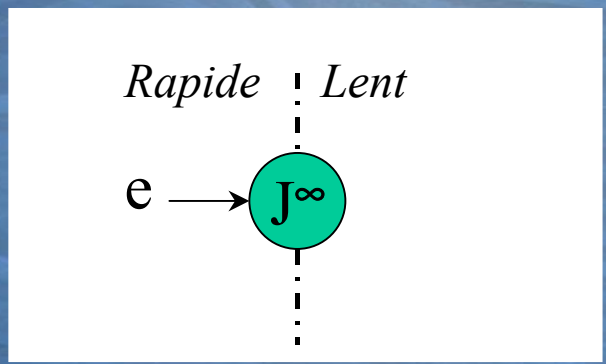
Implantation optimisée d'algorithmes sur architecture configurable

Fork∞



Registre

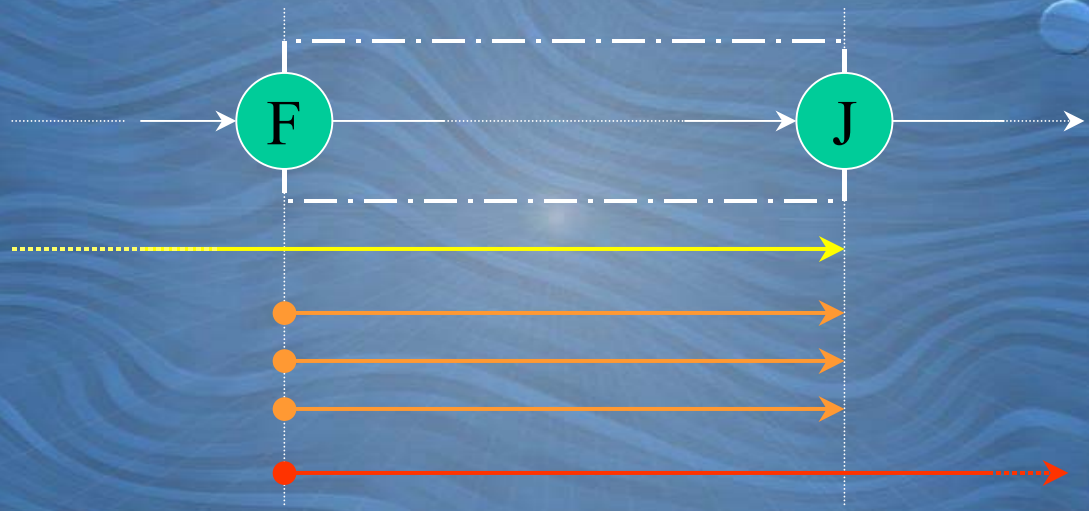
Join∞



Registre

Modèle AAA

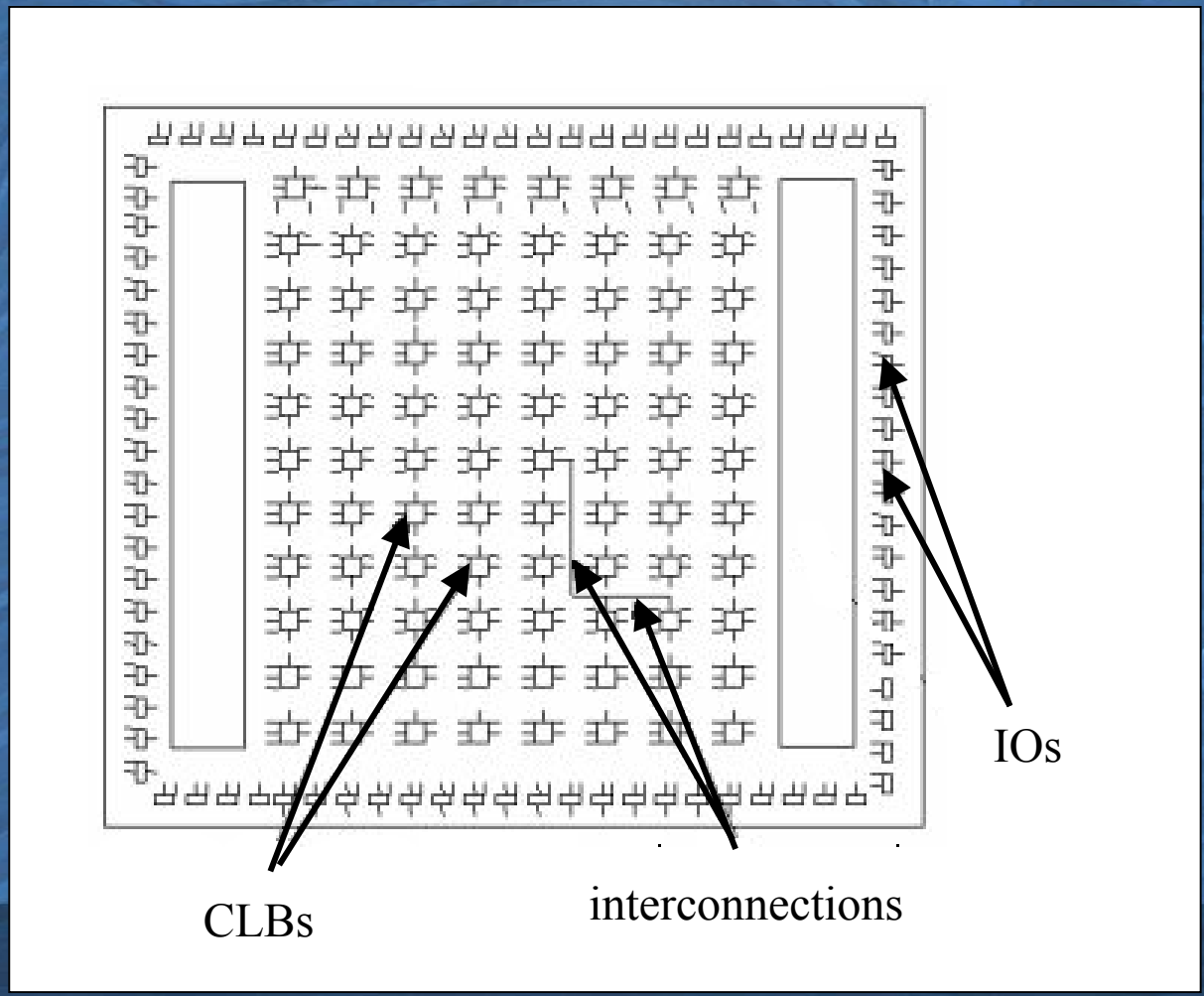
Exécution du contenu d'une frontière



Caractérisation

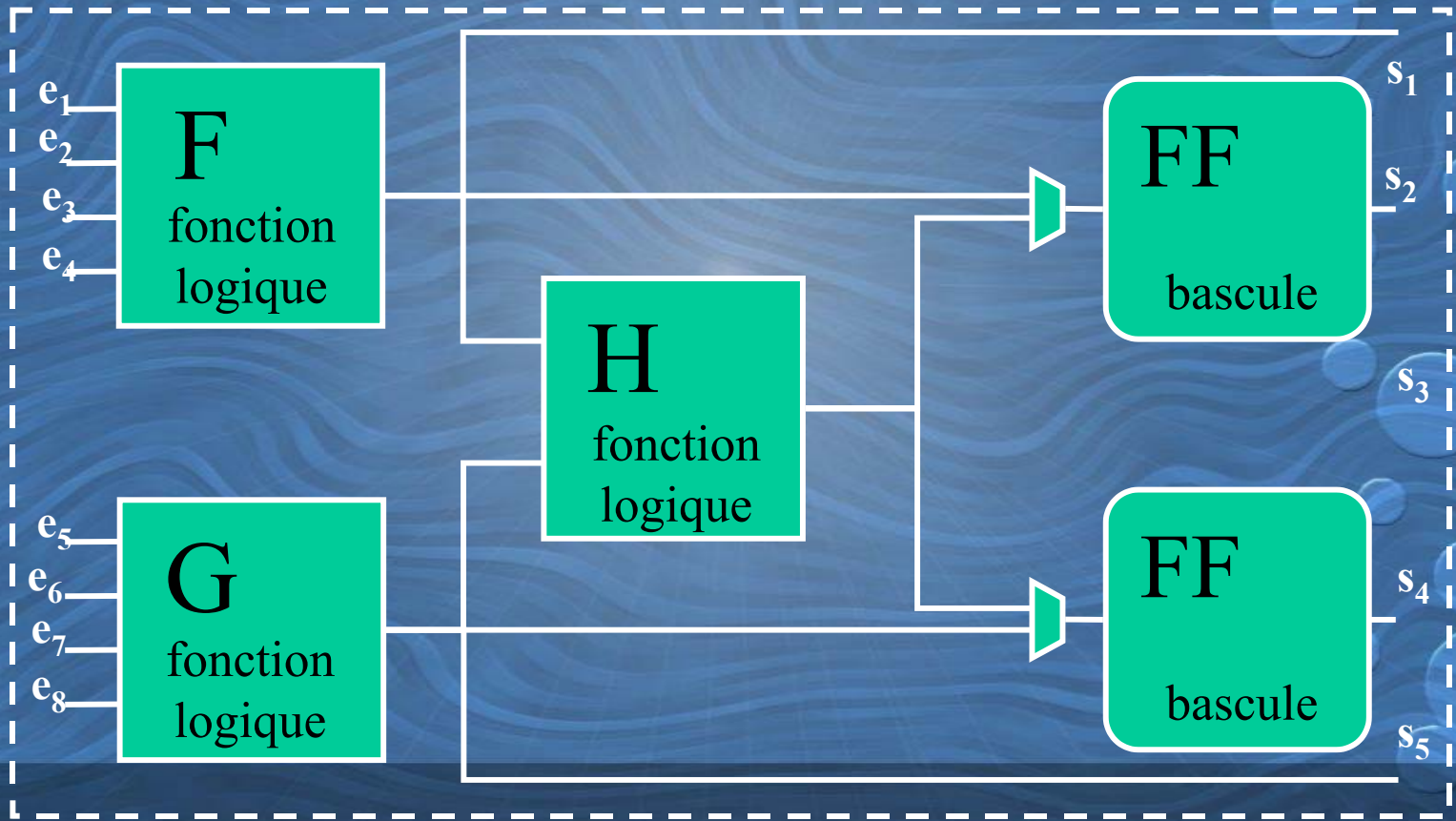
- Présentation du Matériel : Le FPGA
- Caractérisation Temporel
- Caractérisation en Surface

FPGA



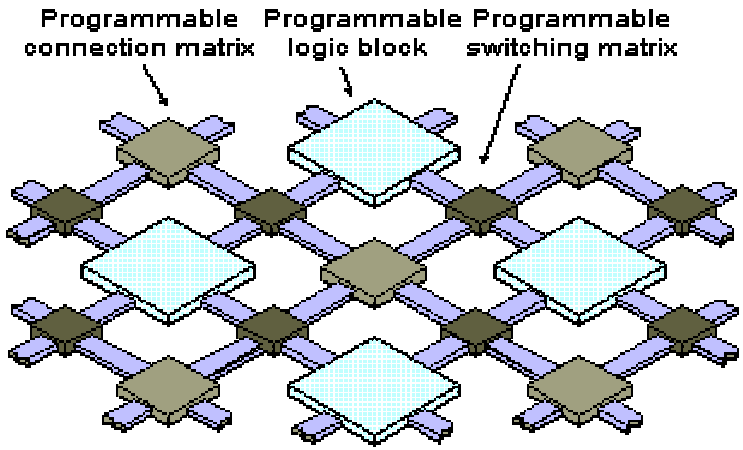
Caractérisation

Description du FPGA



Caractérisation

Description du FPGA



Generic coarse-grained FPGA architecture

Caractérisation

Caractérisation en temps

- **Calcul de la latence.**

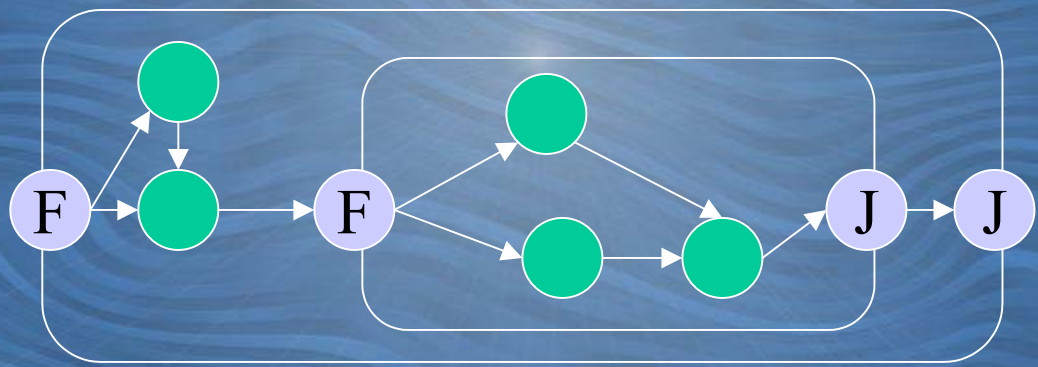
Calcul du nombre de cycles.

Calcul du temps d'un cycle.

Caractérisation

Nombre de cycles.

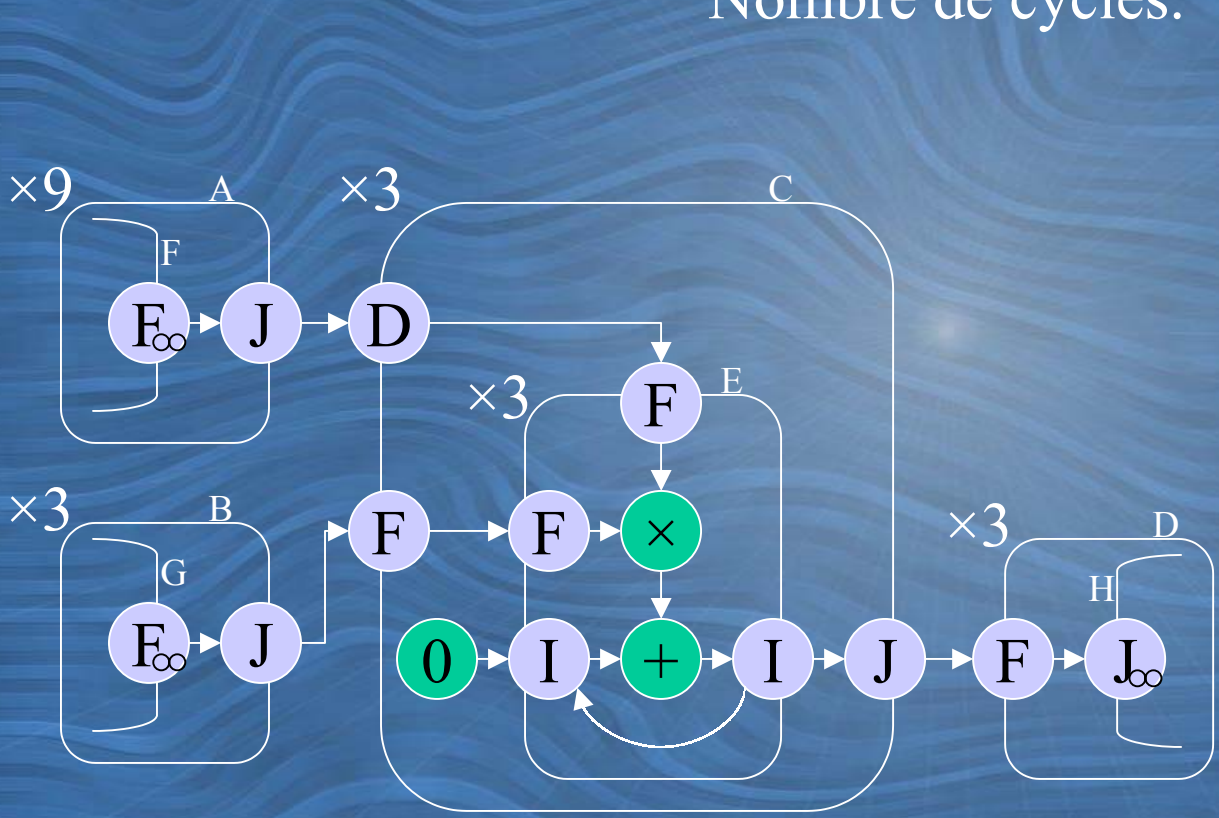
- Principe: 1 cycle pour une frontière « vide ».

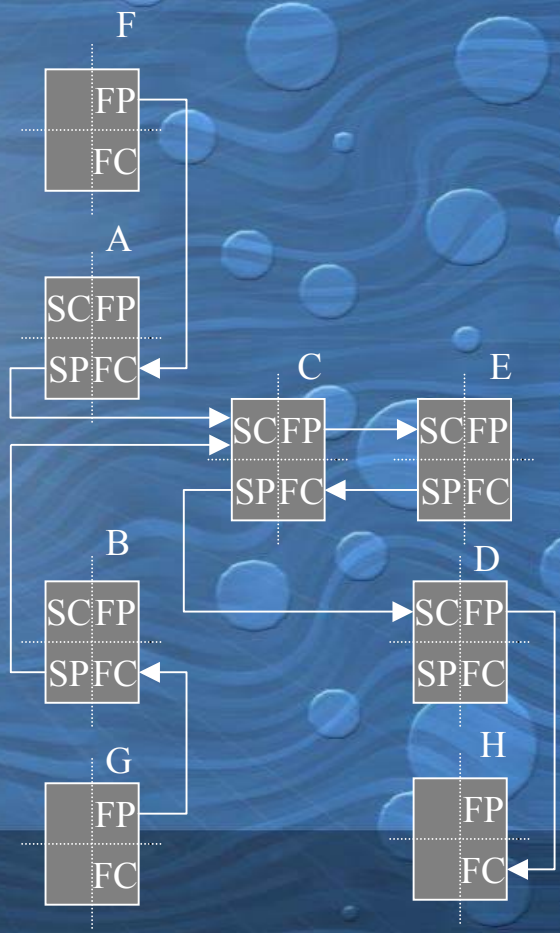


⇒ on ne s'intéresse qu'aux relations de voisinage de frontières

Caractérisation

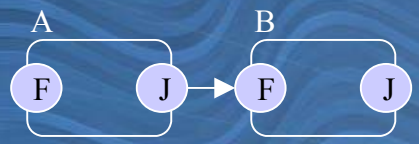
Nombre de cycles.



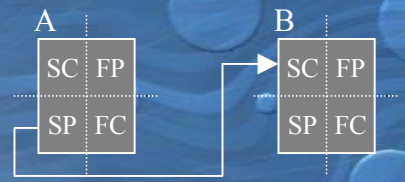


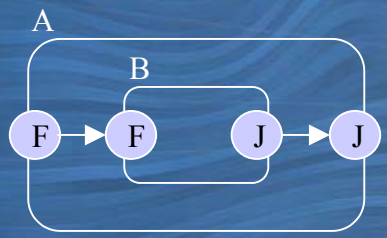
Caractérisation

Nombre de cycles.

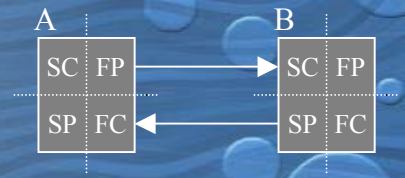


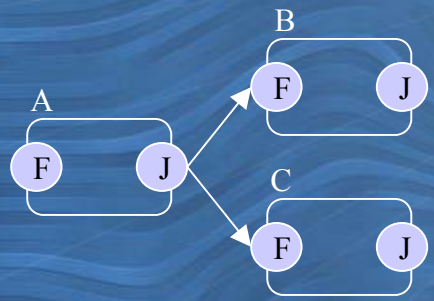
Série(A,B)=C(A)+C(B)-1



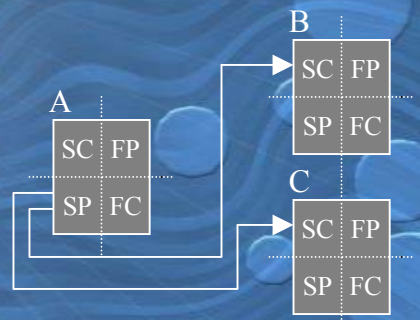


Inclus(A,B)=Fact(A)*C(B)



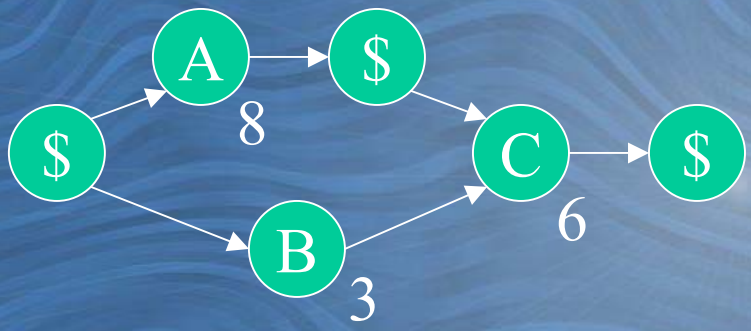


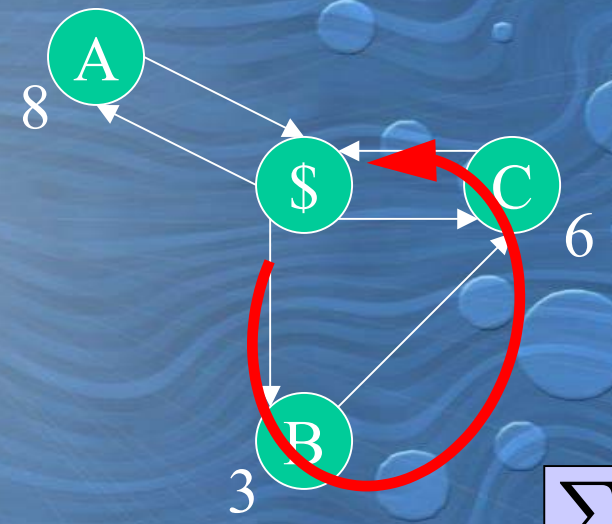
Parallèle(A,B,C)=Max(Série(A,B),Série(A,C))



Caractérisation

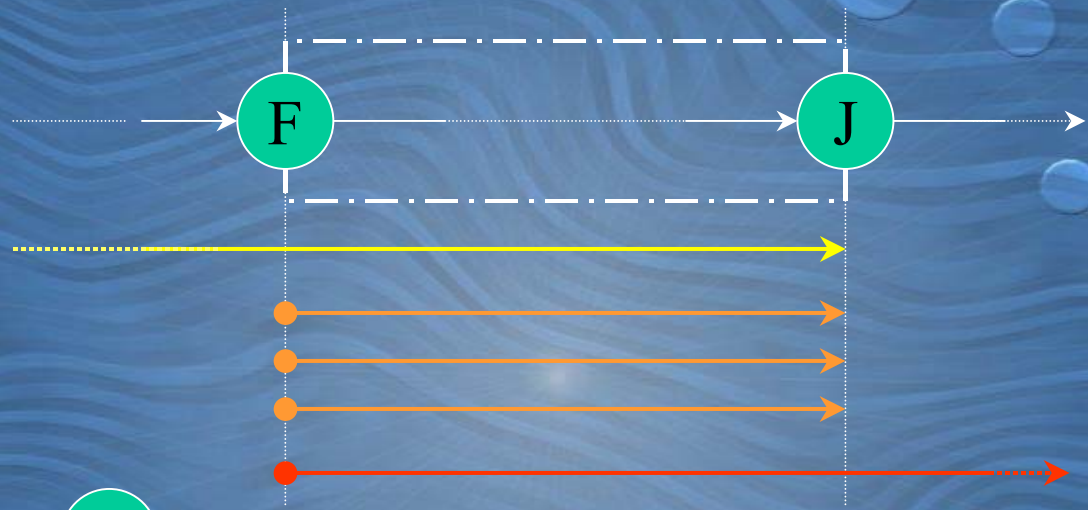
Temps de Cycle





$\Sigma = 9$

Caractérisation



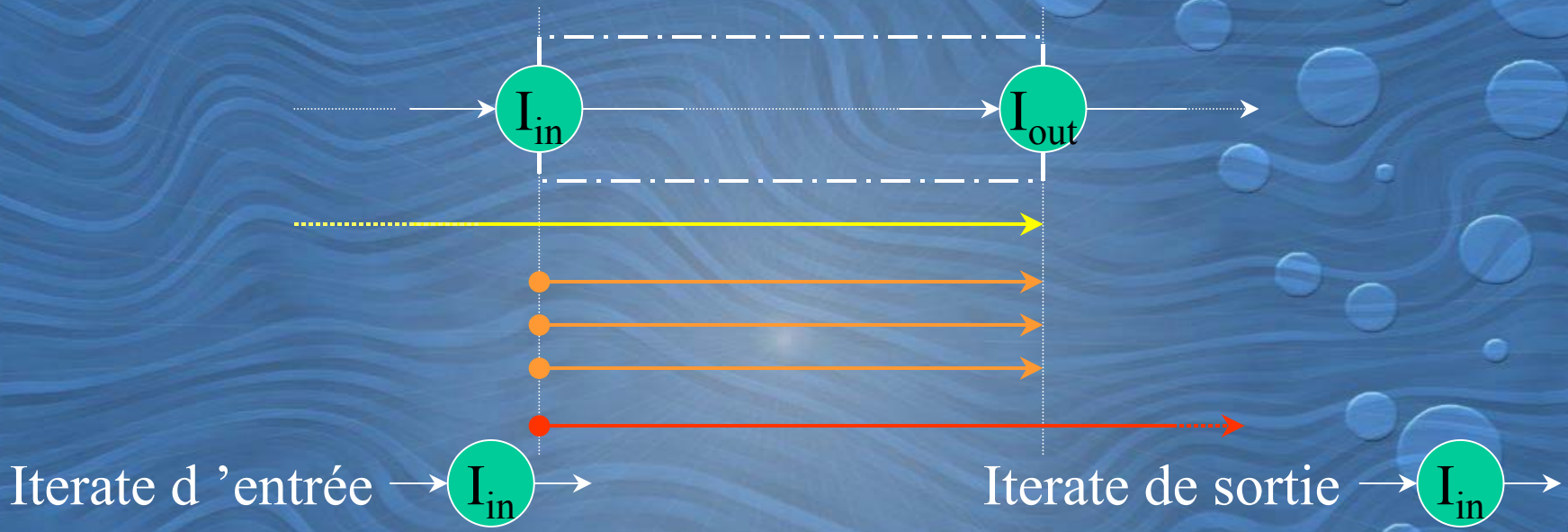
Fork → F → Join → J

E → T_F → S — cycle initial — E → T_w → #

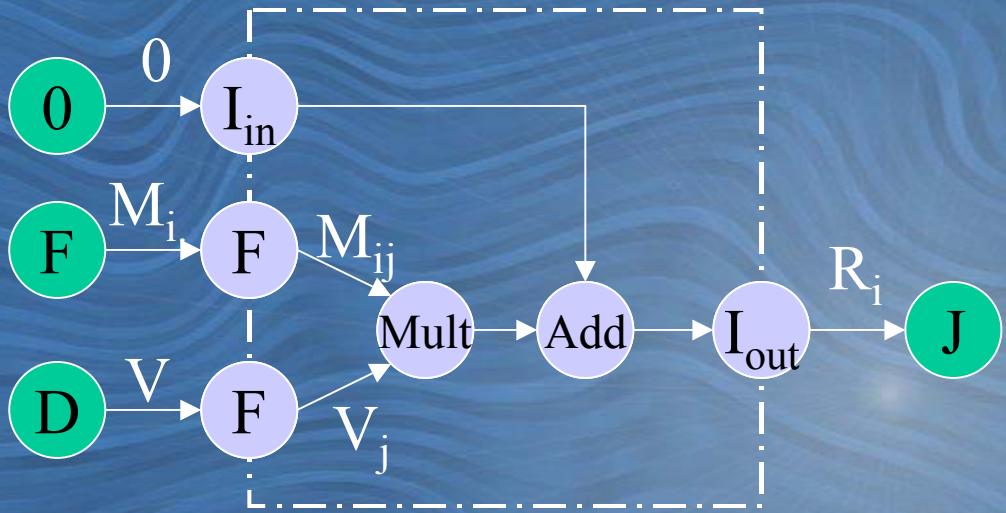
→ T_F → S — cycles intermédiaires — E → T_w →

→ T_F → S — cycle final — E → 0 → S

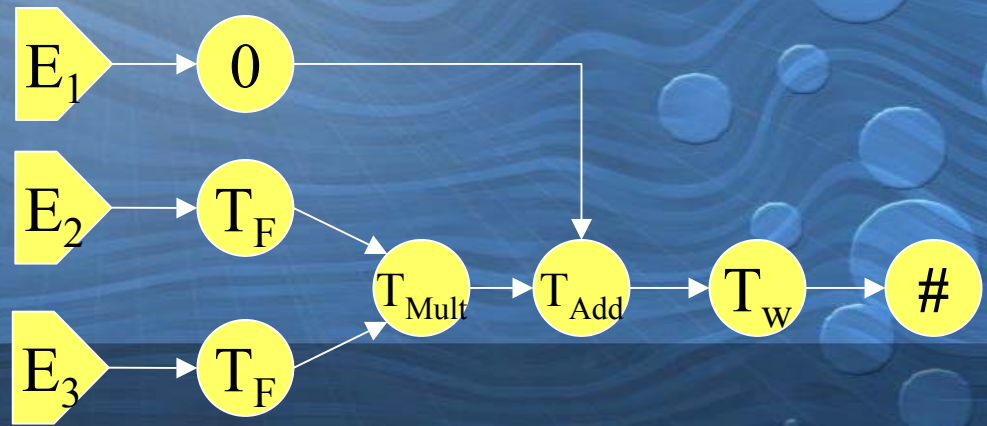
Caractérisation



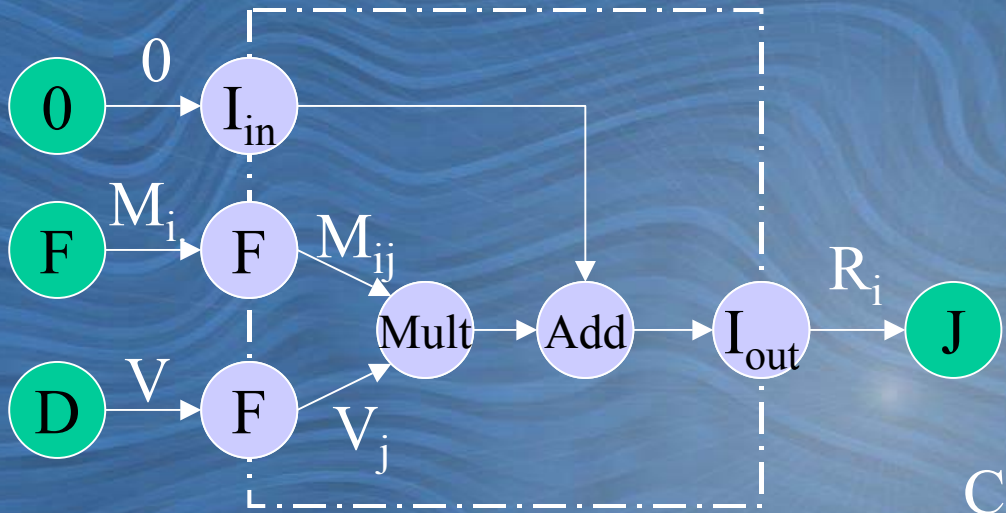
Caractérisation



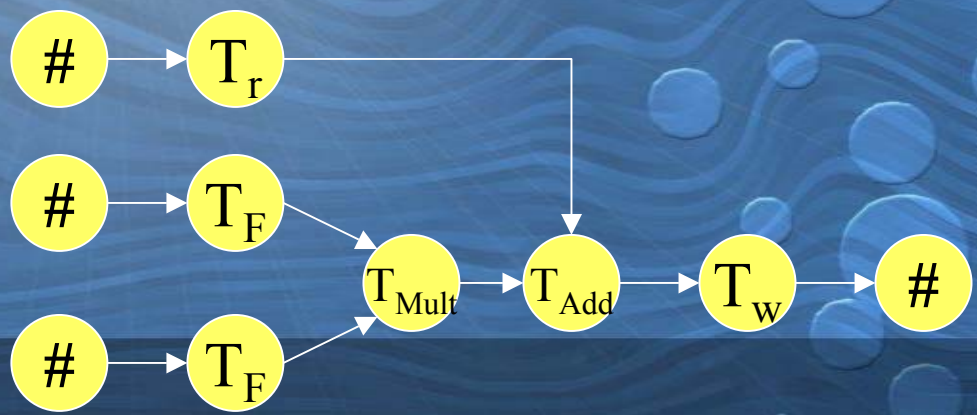
Cycle Initial



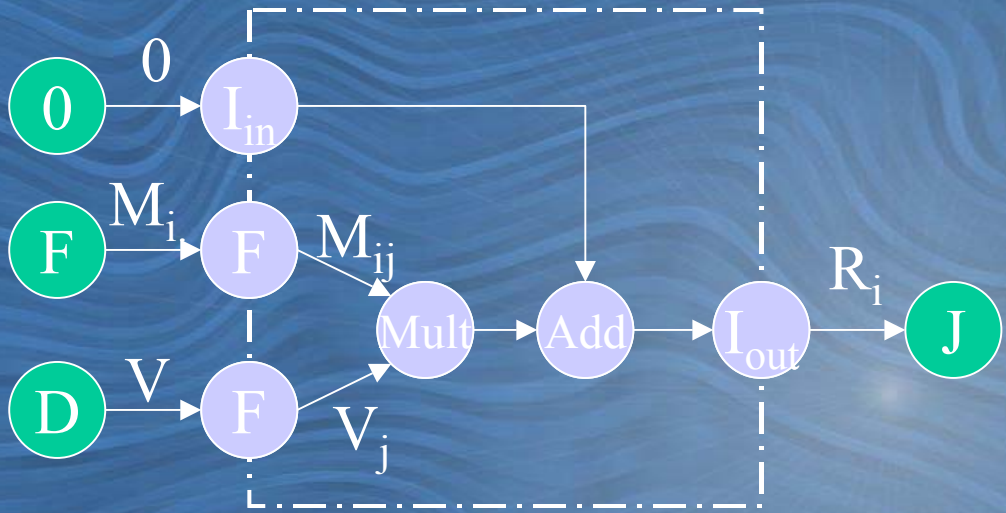
Caractérisation

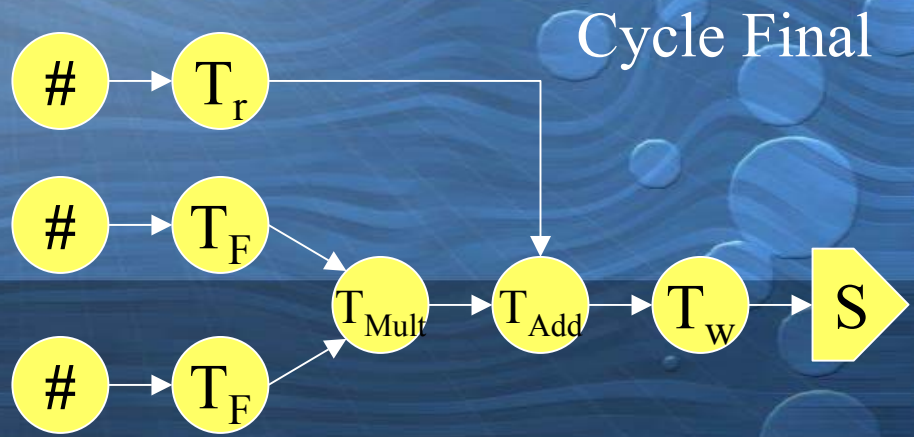


Cycles Intermédiaires

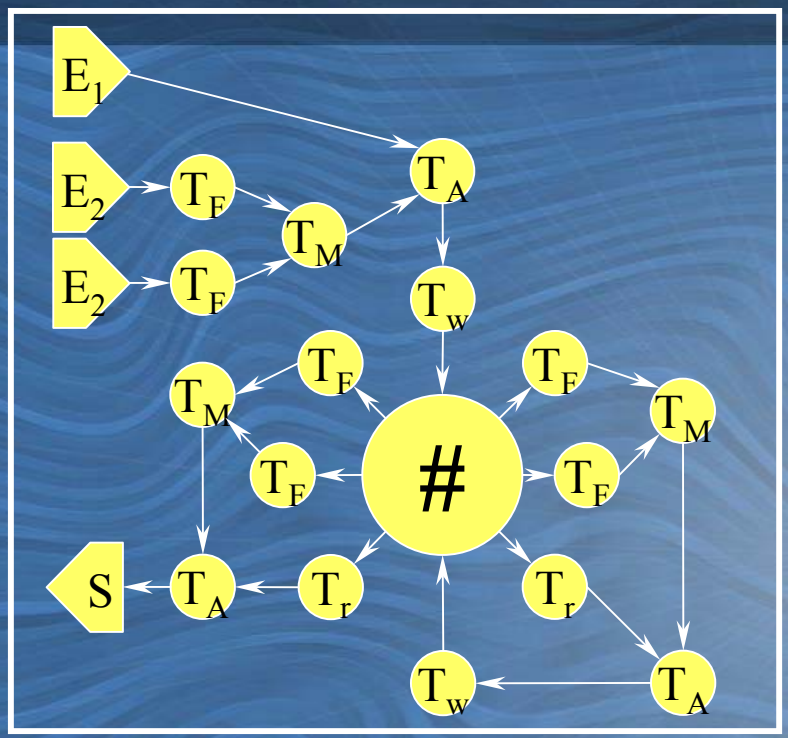


Caractérisation

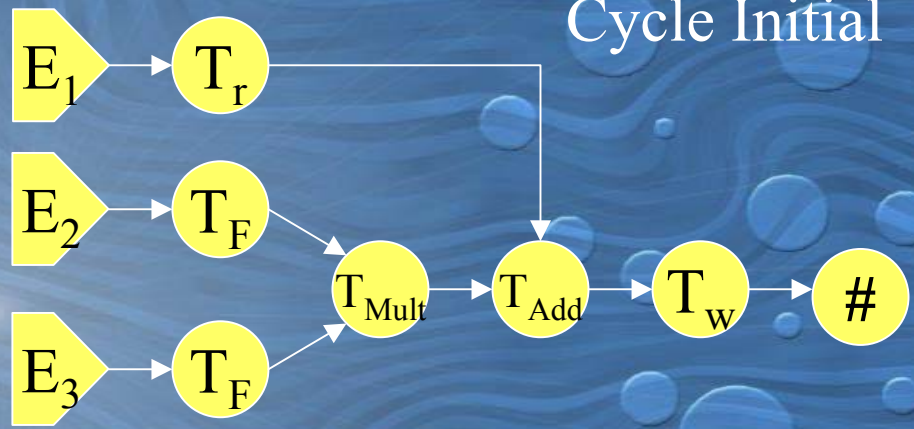




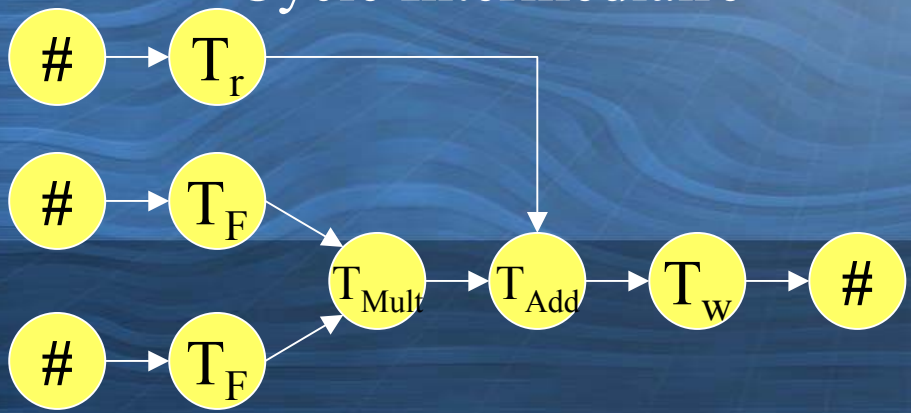
Caractérisation



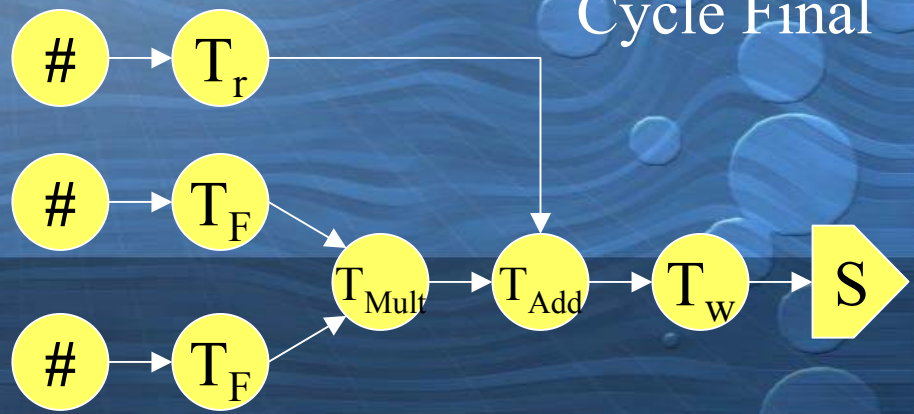
Cycle Initial

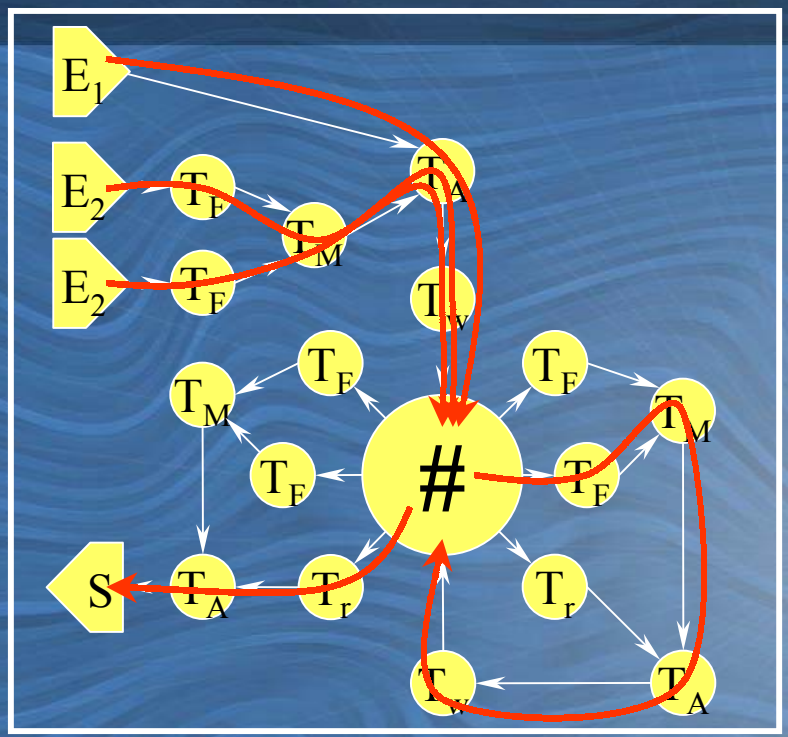


Cycle Intermédiaire



Cycle Final

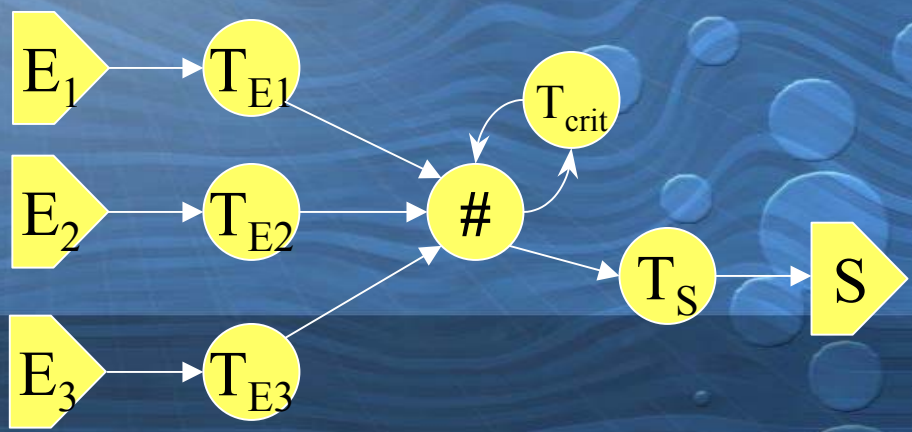




Caractérisation

Pour simplifier on calcule :

- Temps d'entrée
- Temps interne
- Temps de sortie



Caractérisation

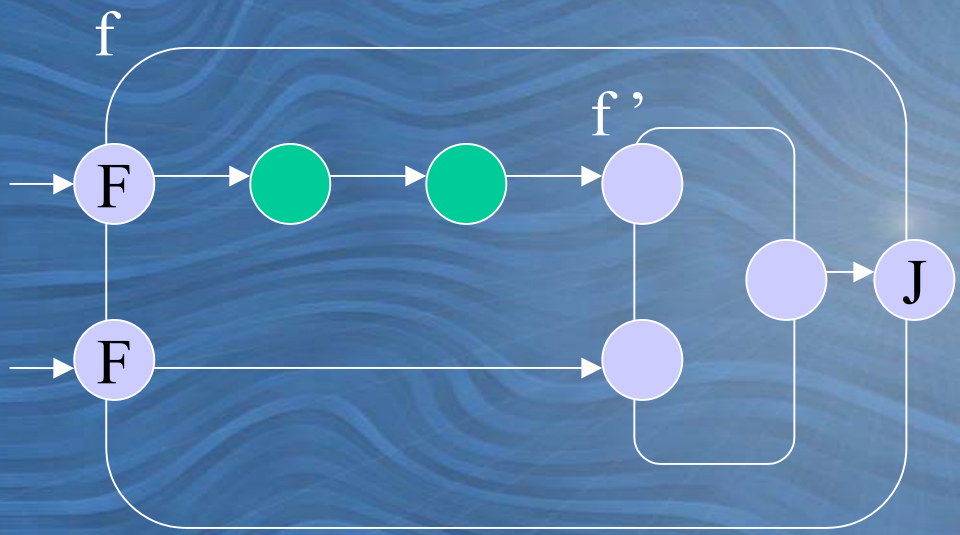
Surface

Data-path : surface de calcul

Control-path : surface de factorisation

Caractérisation

Datapath : surface de calcul



- **SDP(f)** : surface de calcul d'une frontière f
- **S(x)** : surface d 'un sommet
- **ports(f)** : ensemble des sommets ports sur la frontière f
- **sommets(f)** : ensemble des sommets de calcul directement inclus dans f

$$SDP(f) = \sum_{p \in ports(f)} S(p) + \sum_{v \in sommets(f)} S(v) + \sum_{f' \subset f} SDP(f')$$

Caractérisation

Surface du Datapath

- Bibliothèque de valeurs pour les nœuds de calcul
- Formule générale pour les Ports

$$S(I) = (1;d)$$

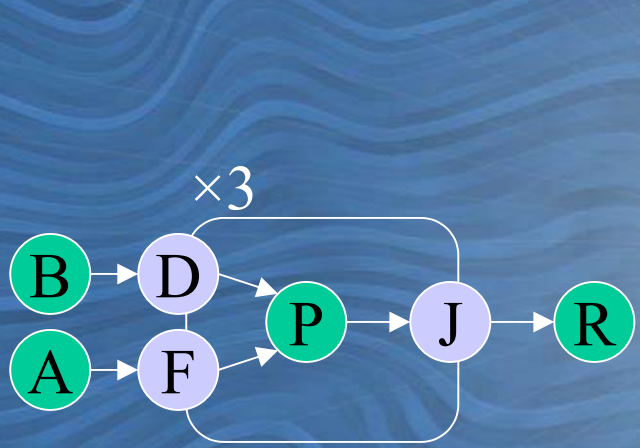
$$S(J) = (f-1;d.(f-1))$$

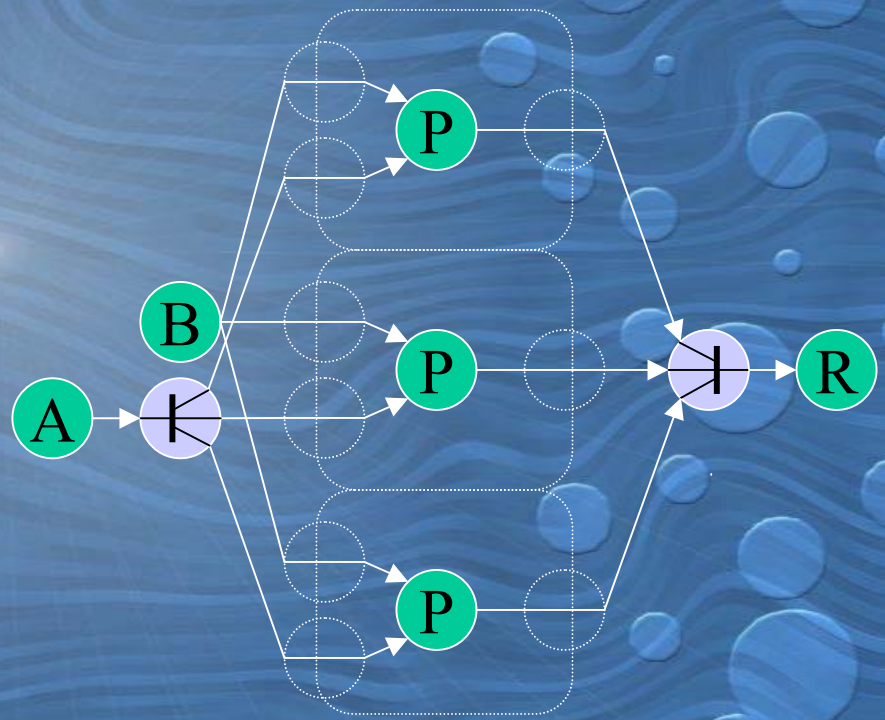
$$S(F) = \left(2d \sum_{i=0}^{\log_4 2f} (f / 2i) ;0 \right)$$

f : facteur de factorisation *d* : taille des données

Caractérisation

Défactorisation totale : disparition des ports





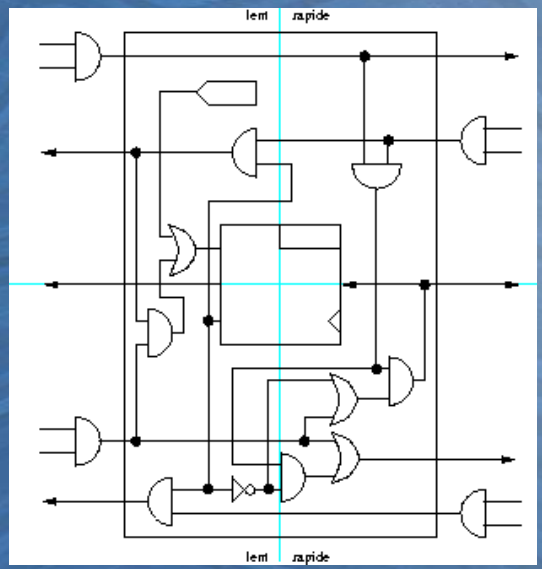
Caractérisation

Surface du Datapath

```
Surface Datapath(f):
Pour tout S ∈ sommets(f) :
| ajouter surface(S);
Si f non totalement défactorisé :
| Pour tout V ∈ ports(f) :
| | ajouter surface(V);
Si ∃ f '⊂ f :
| Pour tout f ' ⊂ f :
| | ajouter Surface Datapath(f ');
```

Caractérisation

Controlpath : surface de factorisation



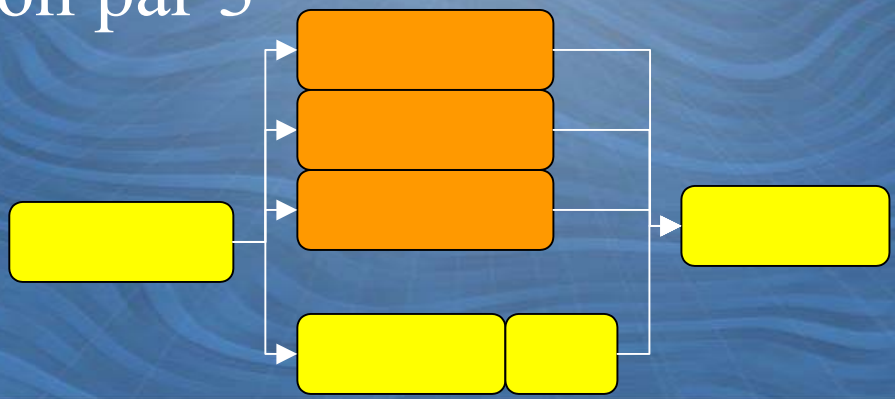
SCP(f) : surface de gestion de la synchronisation de f
Scpt(f) : surface du compteur
Sli(f) : surface logique interne
Sle(f) : surface logique externe, surface de conjonction des R/A

$$SCP(f) = Scpt(f) + Sli(f) + Sle(f) + \sum_{f' \subset f} SCP(f')$$

Caractérisation

Surface du Controlpath

Défactorisation par 3



Caractérisation

Surface du Controlpath

```
Surface Controlpath(f):
Si f non totalement défactorisé :
|  ajouter surface du compteur;
|  ajouter surface de la logique interne ;
|  ajouter surface conjonction R/A;
Si ∃ f '⊂ f :
|  Pour tout f ' ⊂ f :
|  |  ajouter Surface Controlpath(f ');
```

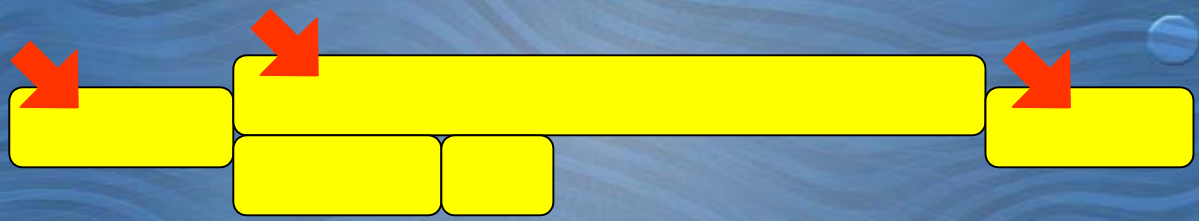
Optimisation

Heuristique

- Quelle frontière défactoriser ?
- De combien la défactoriser ?

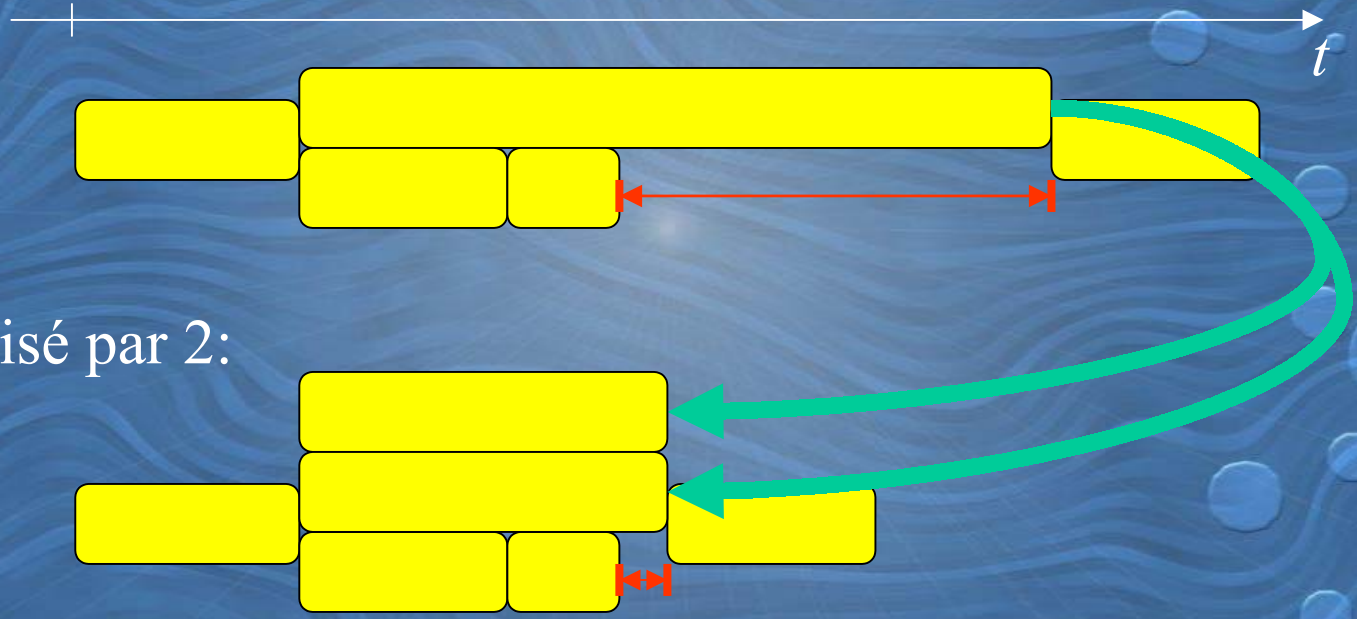
Optimisation

Quelles frontières défactoriser ?



Optimisation

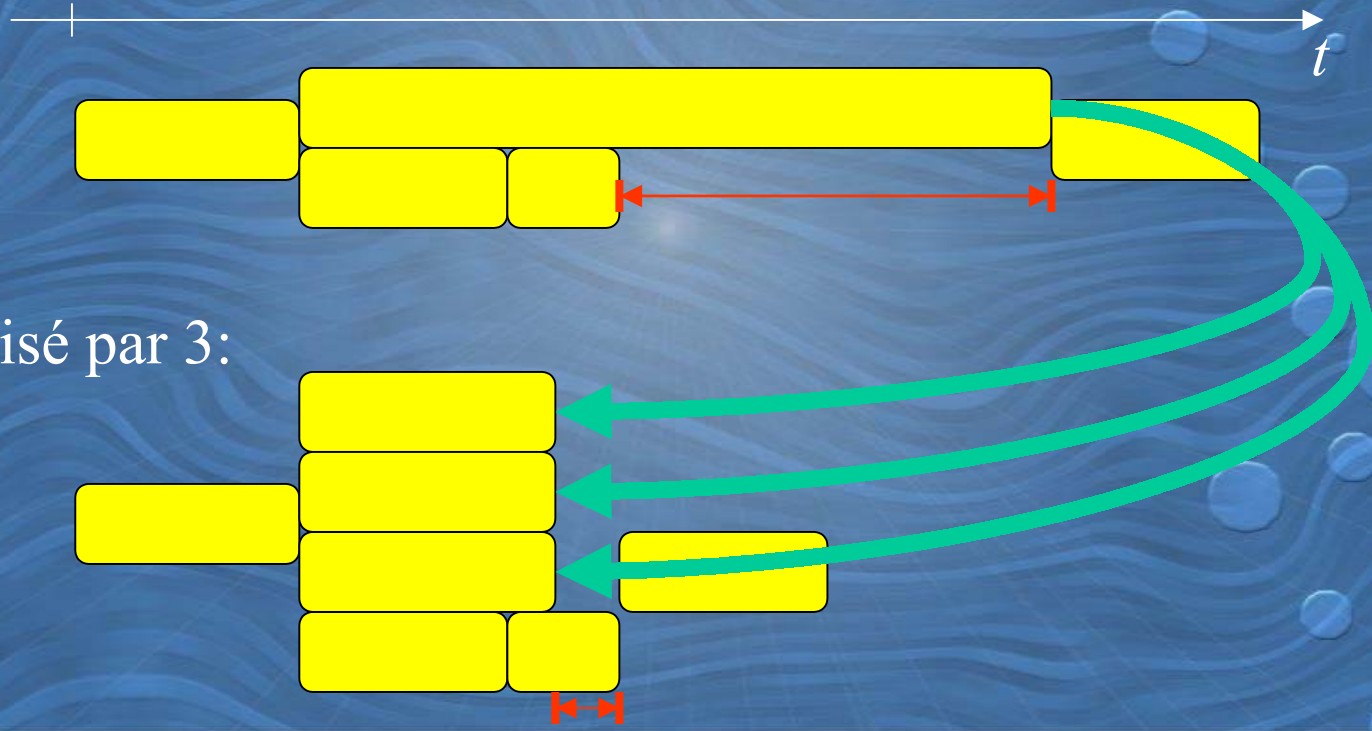
De combien la défactoriser ?



Défactorisé par 2:

Optimisation

De combien la défactoriser ?



Défactorisé par 3:

Optimisation

Quelle frontière défactoriser ?

Notion de rentabilité:

- Maximiser le gain de temps
- Minimiser le coût en surface

$$\text{Rentabilité} = \frac{T - \max(T', C)}{\Delta\text{Surface}}$$

T: latence avant défactorisation

T': latence après défactorisation

C: contrainte de temps

ΔSurface : variation de surface

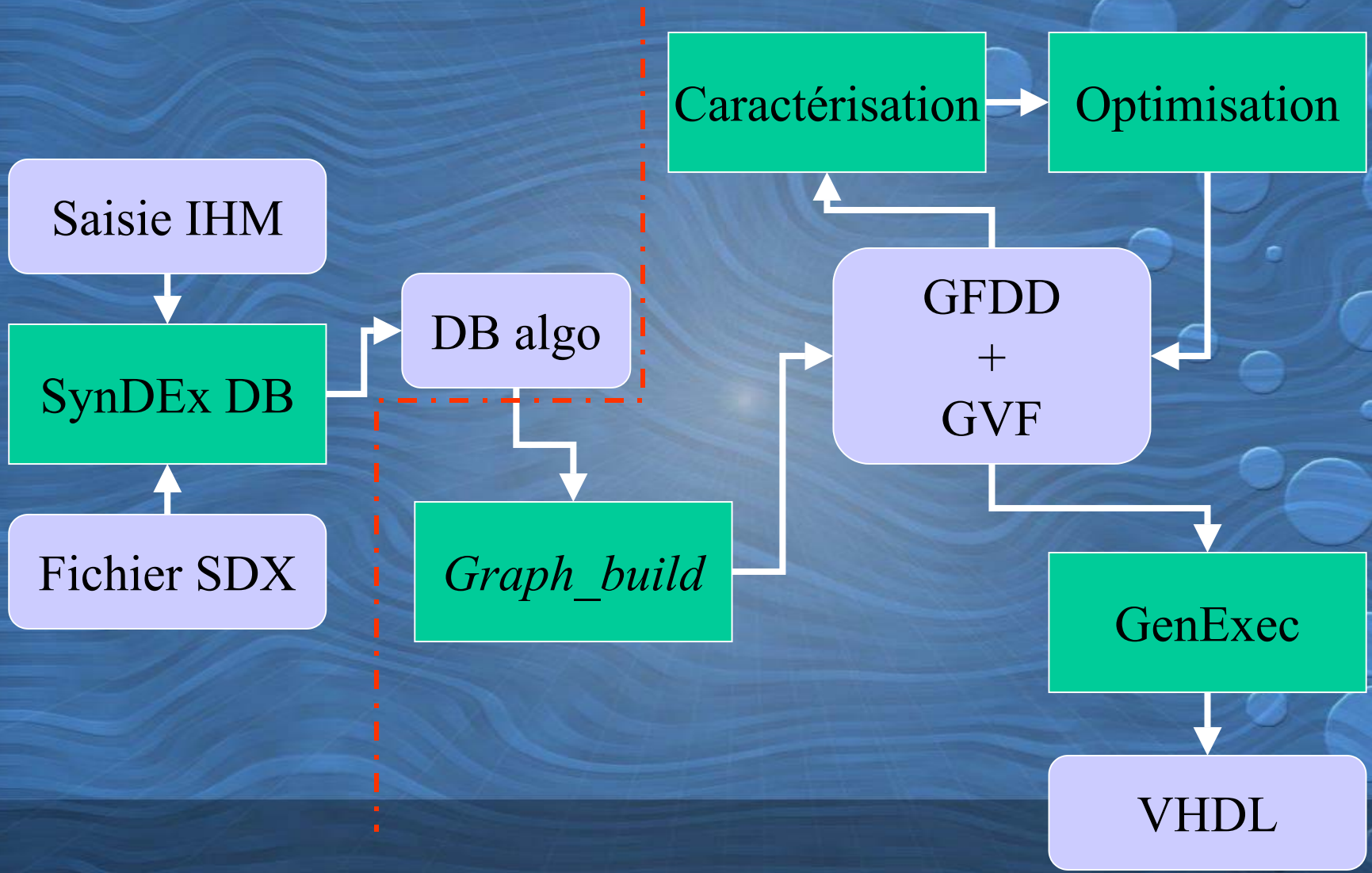
Optimisation

Algorithmes

```
Optimisation():
Tant que T<C:
    Pour chaque f ∈ chemin critique:
        Déterminer défactorisation optimale;
        Déterminer rentabilité;
    Défactoriser la frontière la plus rentable;
```

```
Défactorisation optimale(f):
défac(f):=1;
Tant que f ∈ chemin critique & T>C:
    défac(f) := défac(f)+1;
```

Implantation optimisée d'algorithmes sur architecture configurable



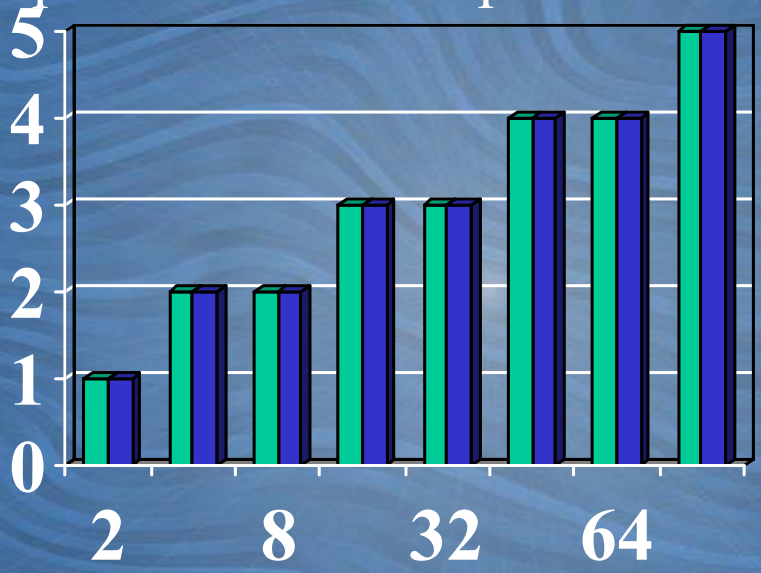
Application

Validation du modèle théorique

- Taille/temps pour les ports de base (F, J…)
- logiciel de simulation (Leonardo)

Application

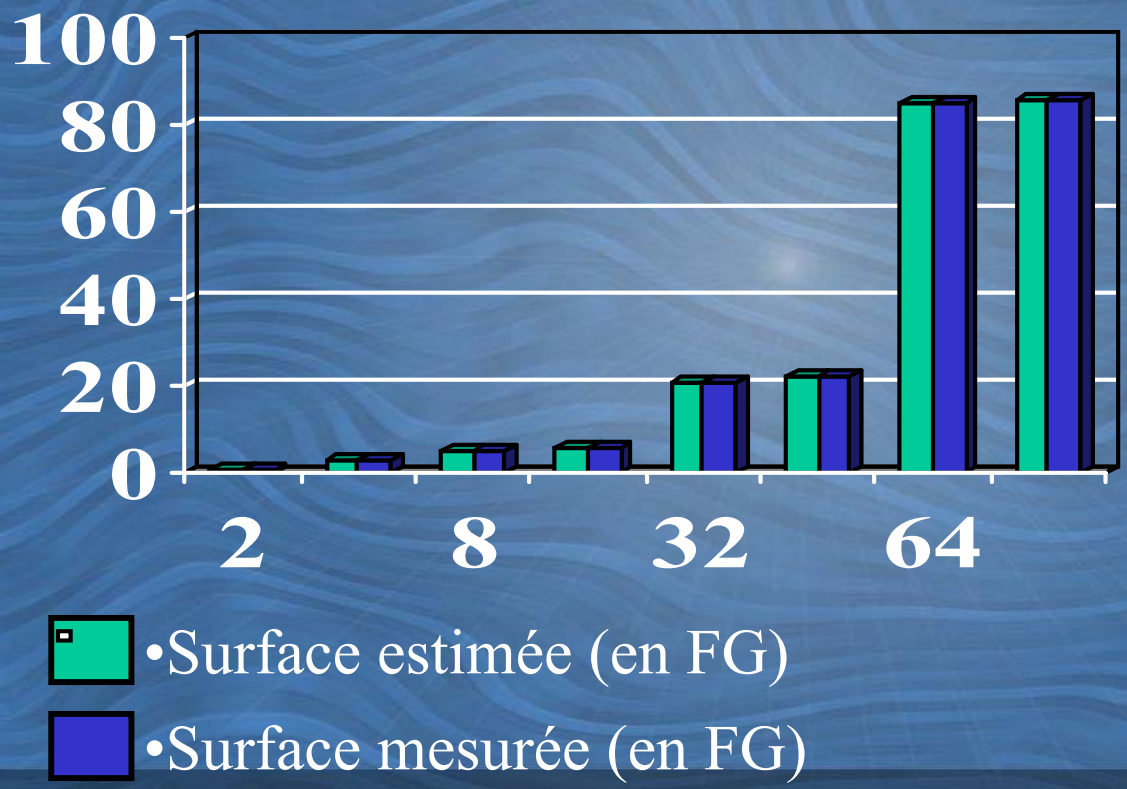
Exemple d 'estimation pour des multiplexeurs n->1



- Temps estimé (en u de temps)
- Temps mesuré (en u de temps)

Application

Exemple d 'estimation pour des multiplexeurs n->1



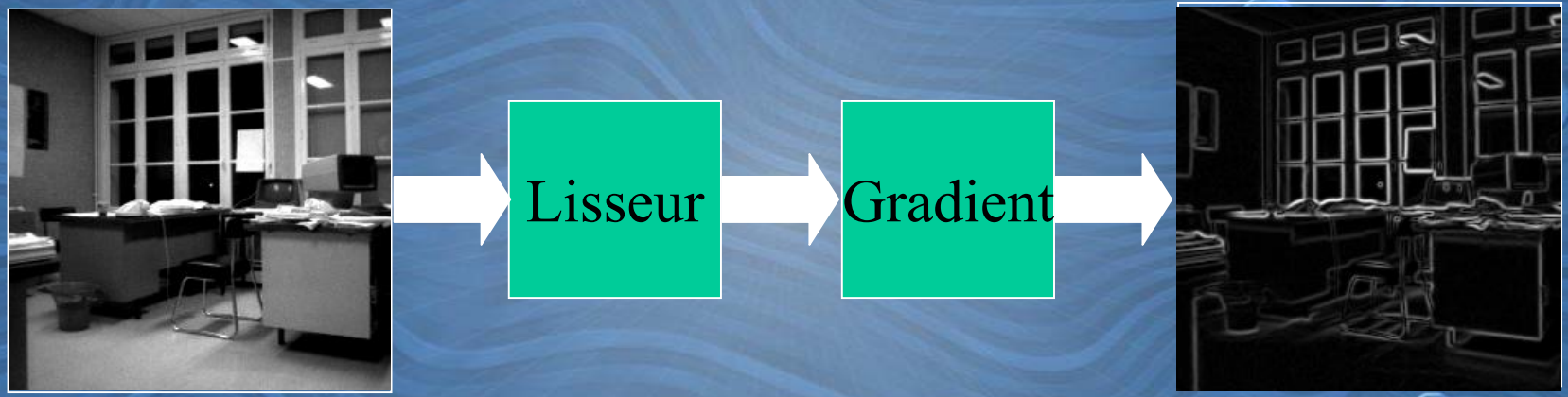
Application

Validation du modèle théorique.

- Taille/temps pour les ports de base (F,J..)
  - logiciel de simulation (Leonardo)
- Opérateurs simples (mul,add…)
  - documentation Xilinx
- Graphes de traitement d'image
  - –>produit matrice/vecteur
  - –>filtre de Deriche

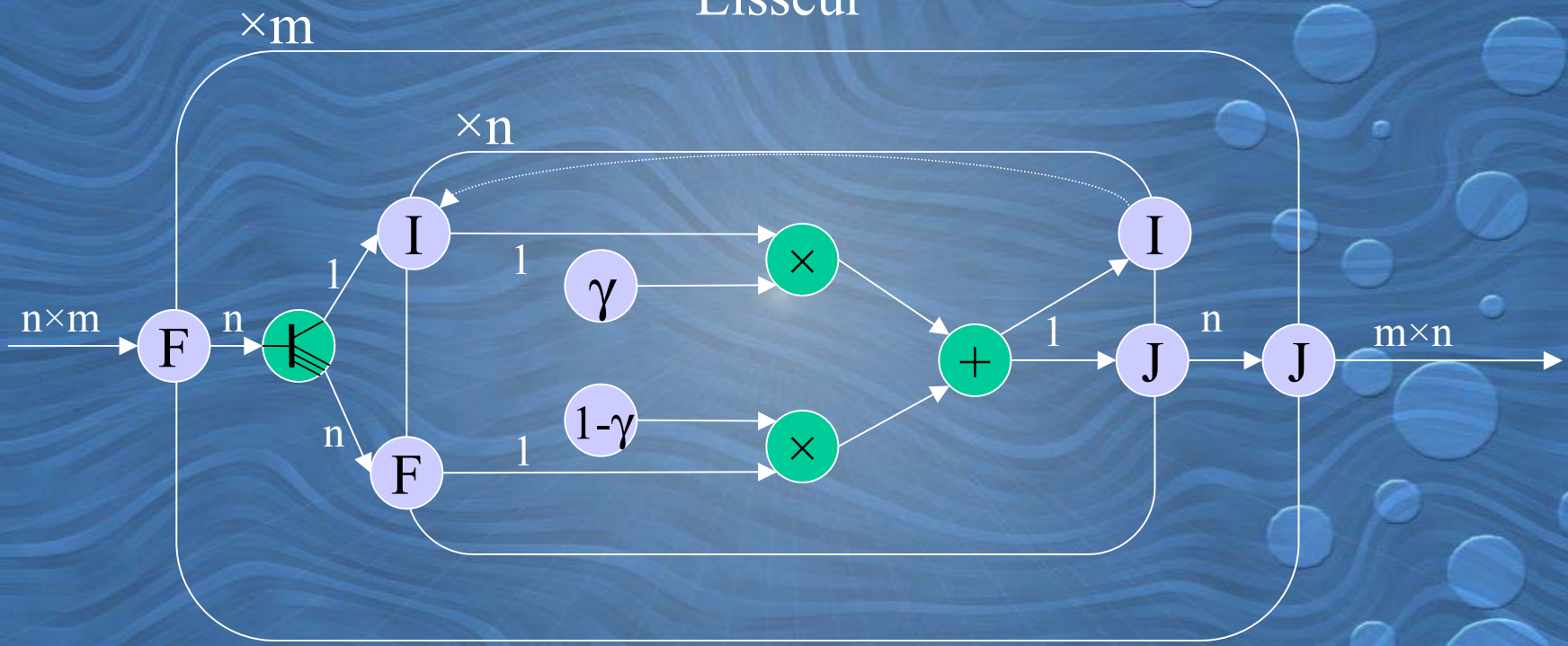
Application

Le filtre de Deriche



Application

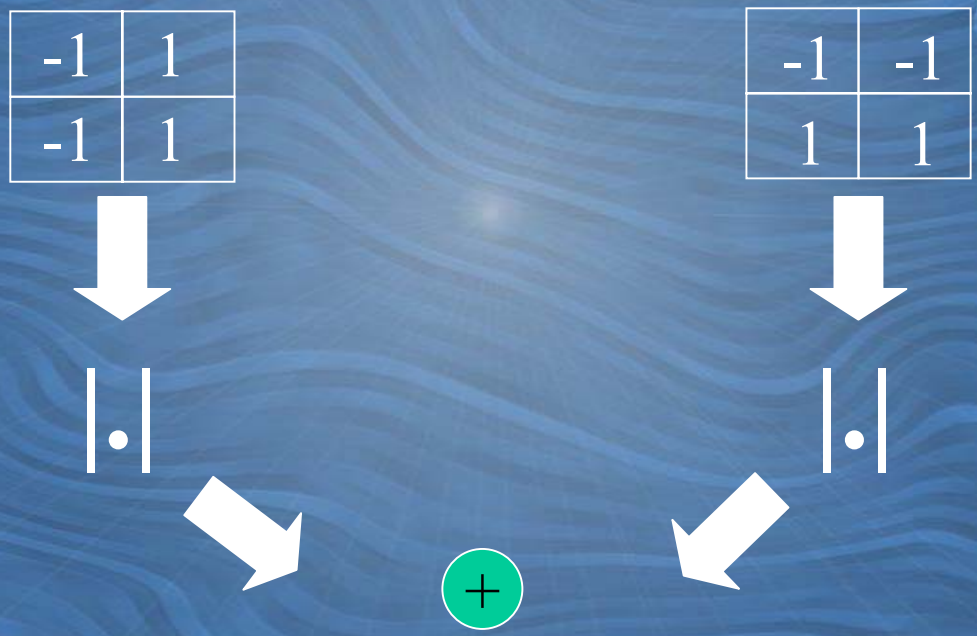
Lisseur



$$y(n)=\gamma x(n)+(1-\gamma)y(n-1)$$

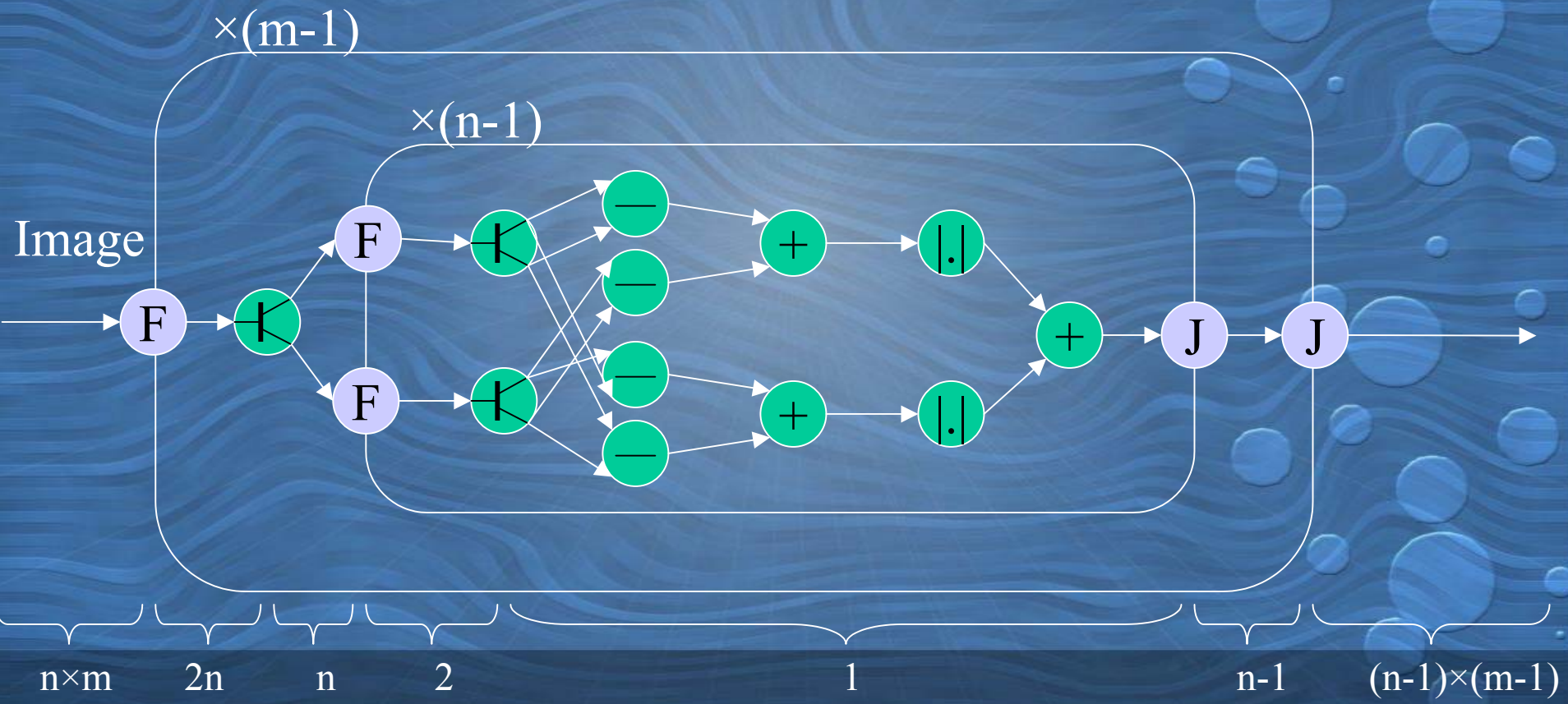
Application

Gradient



Application

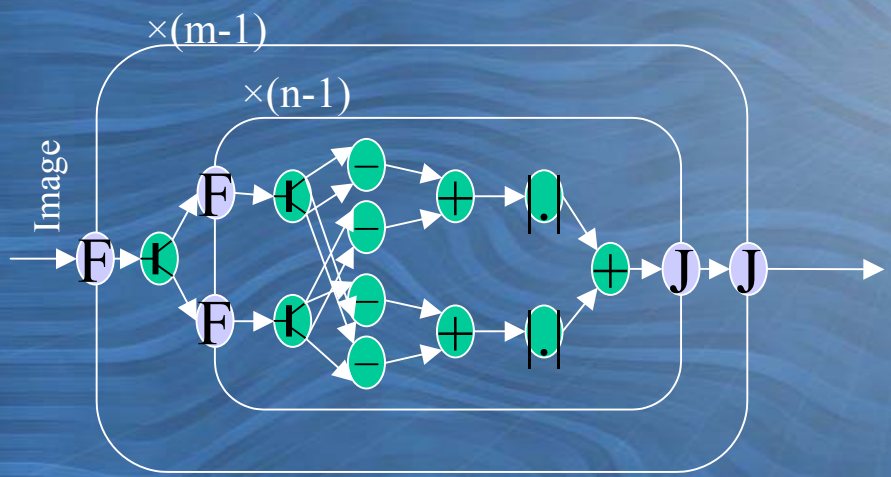
Gradient

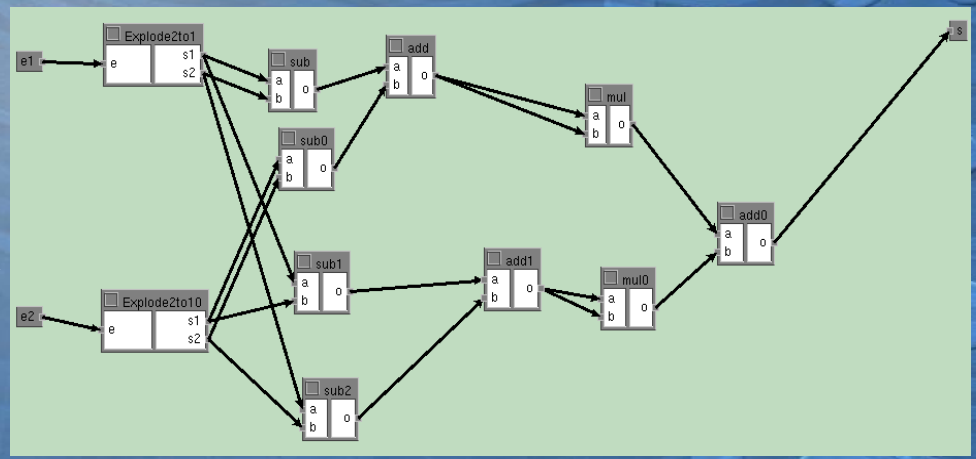


Application

Résultats

Saisie d 'un graphe sous forme de GFDD sous SynDEx





Application

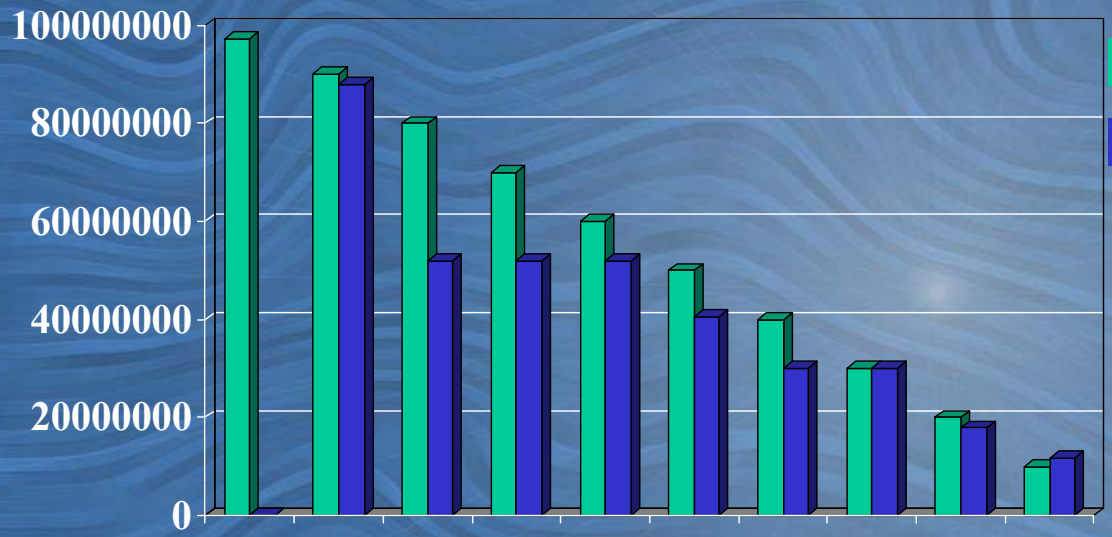
Résultats

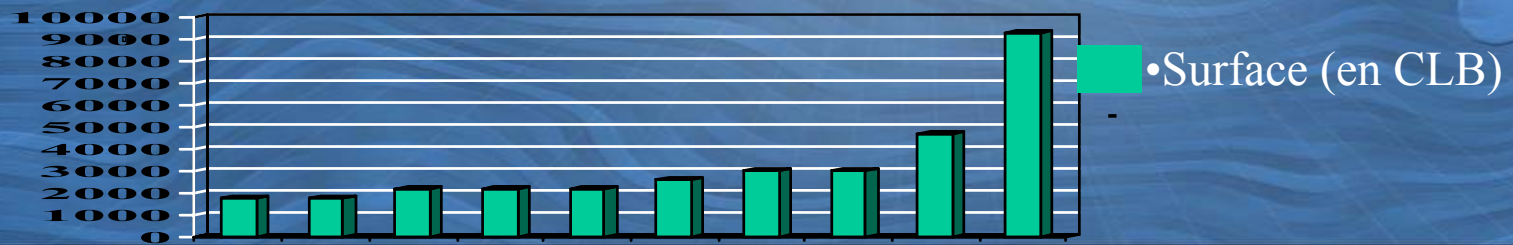
- Caractérisation:
  - mesures théoriques validées pour les opérateurs de base
  - cohérence des résultats trouvés:
    - data-path:
      - nombre de cycles
      - chemin critique
      - temps de cycle
    - control-path:
      - non testé

théorique	résultats
2400	2381
le chemin est le même	
cohérent	

Application

Résultats optimisation





Conclusion