Amélioration de l'applicabilité du pipeline de nid de boucles pour la synthèse de haut niveau

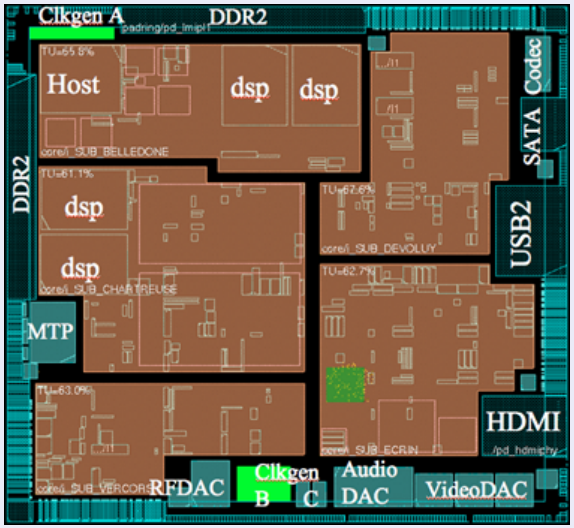
Antoine Morvan [1], Steven Derrien [2], Patrice Quinton [1]

Équipe CAIRN
[1] *INRIA-IRISA-ENS Cachan*
[2] *INRIA-IRISA-Université de Rennes 1*
Campus de Beaulieu, Rennes, France
{amorvan,sderrien,quinton}@irisa.fr

Cinquièmes rencontres de la communauté française de compilation

Introduction - Systèmes sur puce

Systèmes sur puce pour Systèmes embarqués



- Les systèmes embarqués sont partout
- Domaines d'application très variés
- Réduire les coûts en intégrant tout sur une seule puce
- Accélérateurs matériels **spécialisés** pour les tâches critiques

Introduction - Accélérateurs matériels spécialisés

Accélérateurs matériels spécialisés pour systèmes embarqués

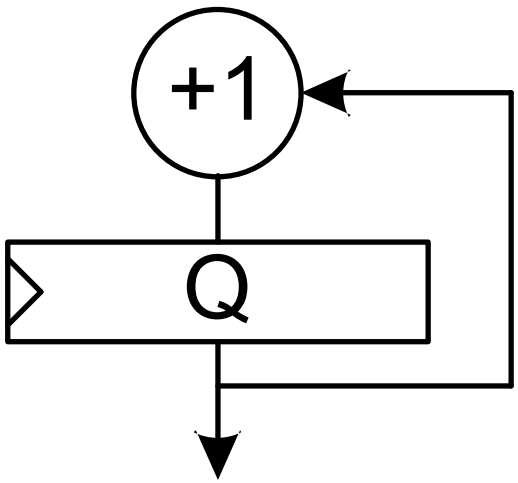
- Les systèmes embarqués ont besoin d'accélérateurs matériels
  - Performance, énergie, coûts surface ($\times 20 - \times 50$)
- Les accélérateurs spécialisés sont conçus en HDL (VHDL, Verilog)
  - Processus de conception très long

Introduction - Accélérateurs matériels spécialisés

Accélérateurs matériels spécialisés pour systèmes embarqués

- Les systèmes embarqués ont besoin d'accélérateurs matériels
  - Performance, énergie, coûts surface ($\times 20 - \times 50$)
- Les accélérateurs spécialisés sont conçus en HDL (VHDL, Verilog)
  - Processus de conception très long

Simple compteur représenté en 30 lignes ...




```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity COUNTER is
  generic (
    WIDTH : in natural := 32);
  port (
    RST   : in std_logic;
    CLK   : in std_logic;
    LOAD  : in std_logic;
    DATA  : in std_logic_vector(WIDTH-1 downto 0);
    Q     : out std_logic_vector(WIDTH-1 downto 0));
end entity COUNTER;

architecture RTL of COUNTER is
  signal CNT : unsigned(WIDTH-1 downto 0);
begin
  process(RST, CLK) is
  begin
    if RST = '1' then
      CNT <= (others => '0');
    elsif rising_edge(CLK) then
      if LOAD = '1' then
        CNT <= unsigned(DATA);
      else
        CNT <= CNT + 1;
      end if;
    end if;
  end process;

  Q <= std_logic_vector(CNT);
end architecture RTL;
```

Introduction - Synthèse de haut niveau

Vers des niveaux d'abstractions plus élevés

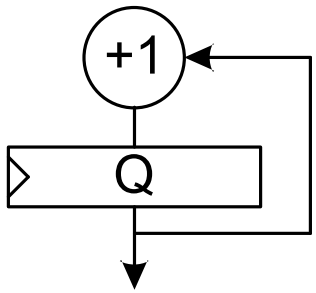
- Les outils de HLS travaillent sur des descriptions comportementales
  - C/SystemC en entrée, description matérielle en sortie
  - Gains en productivité significatifs ($\times 5 - \times 30$)
  - Utilisés en production dans l'industrie

Introduction - Synthèse de haut niveau

Vers des niveaux d'abstractions plus élevés

- Les outils de HLS travaillent sur des descriptions comportementales
  - C/SystemC en entrée, description matérielle en sortie
  - Gains en productivité significatifs ($\times 5 - \times 30$)
  - Utilisés en production dans l'industrie

Simple compteur représenté en 2 lignes



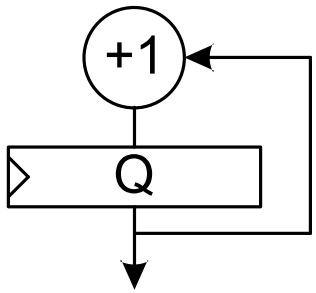

```
1  unsigned int Q = 0;
2  while (true) Q++;
```

Introduction - Synthèse de haut niveau

Vers des niveaux d'abstractions plus élevés

- Les outils de HLS travaillent sur des descriptions comportementales
  - C/SystemC en entrée, description matérielle en sortie
  - Gains en productivité significatifs ($\times 5 - \times 30$)
  - Utilisés en production dans l'industrie

Simple compteur représenté en 2 lignes




```
1  unsigned int Q = 0;
2  while (true) Q++;
```

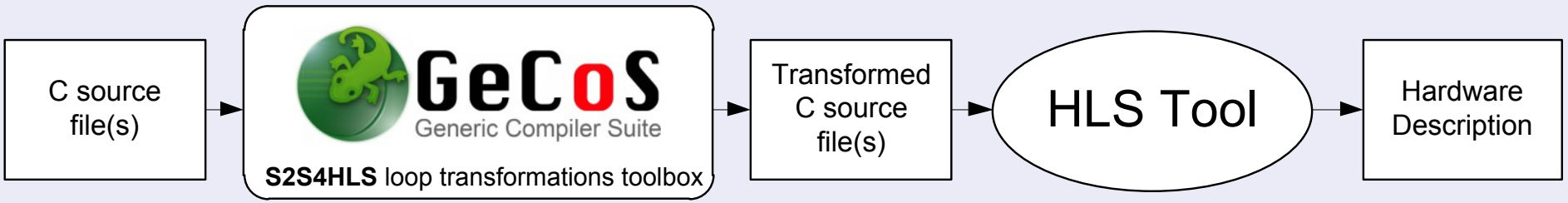
Qualité des résultats perfectible

- Les outils de HLS ne rivalisent pas avec un concepteur expert

Introduction - Source à Source pour la HLS

Projet INRIA/STMicro Nano2012 - S2S4HLS

- Dépasser les limitations des outils de HLS grâce au Source à Source
  - Outils de HLS sont fermés (boîtes noires)
  - HLS considéré comme un Back-End
- Transformation de boucles
  - Extraire le parallélisme, améliorer la localité, ...



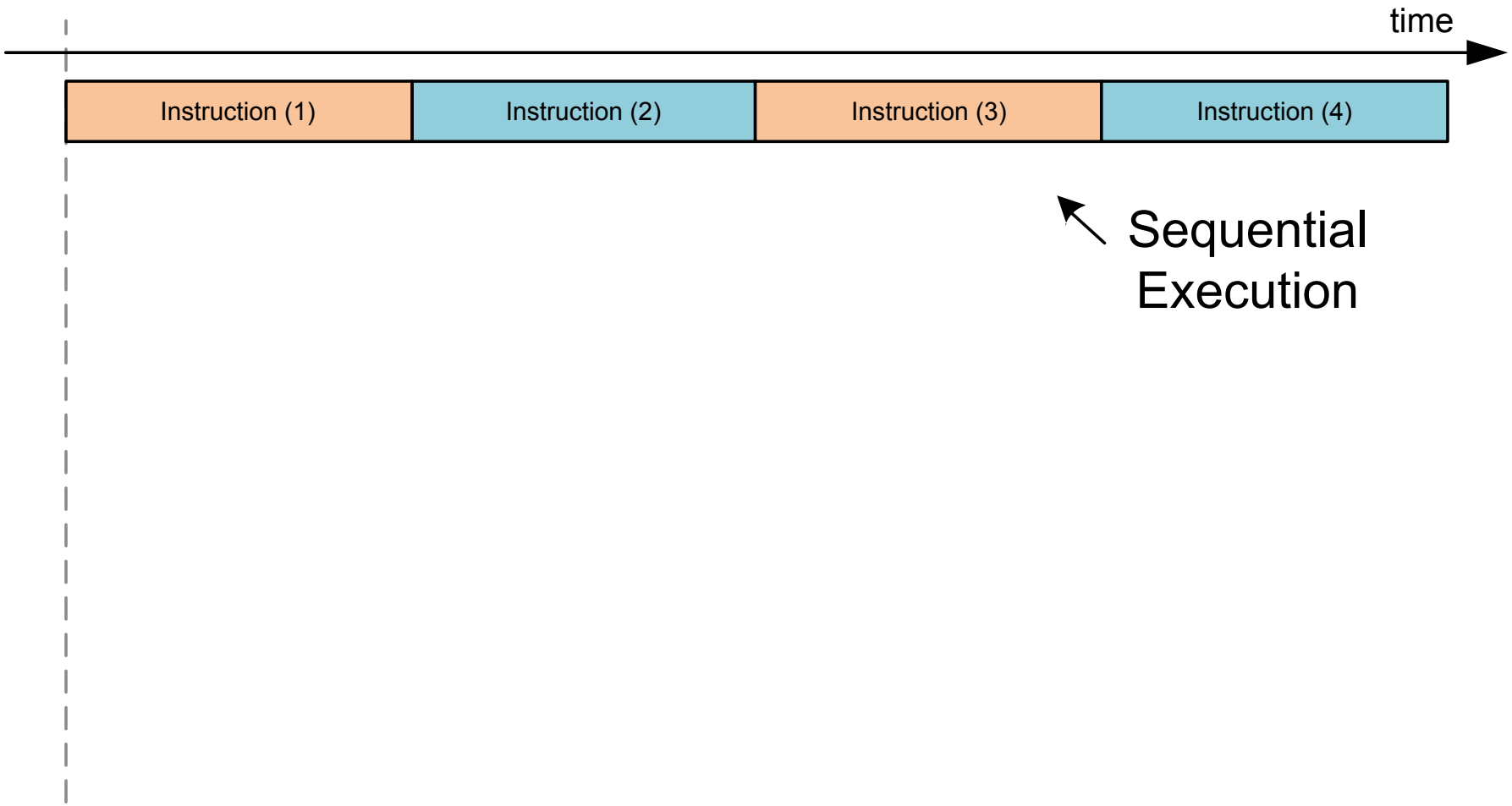
- Dans ce travail, nous améliorons l'applicabilité du pipeline de nid de boucles

Plan

1. Prérequis - Pipeline
   - Principe du pipeline
   - Pipeline de boucle en HLS
   - Pipeline de nid de boucles en HLS
2. Test de légalité
   - Condition de légalité
   - Le modèle polyédrique
   - Compter les points
   - Test de légalité
3. Correction - Insertion statique de bulles
4. Résultats & Conclusion

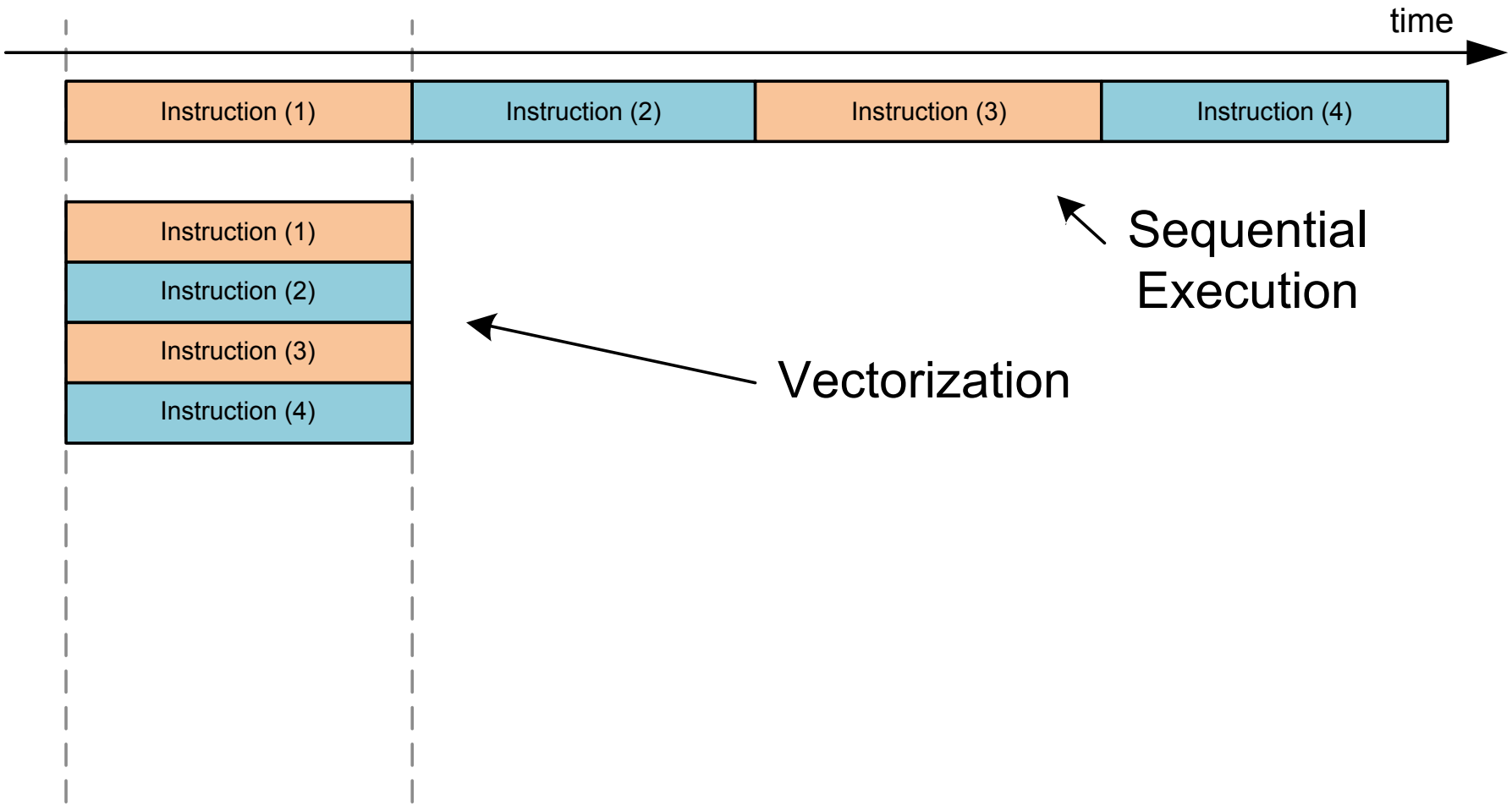
Prérequis - Principe du pipeline

Objectifs : exécution plus rapide . . .



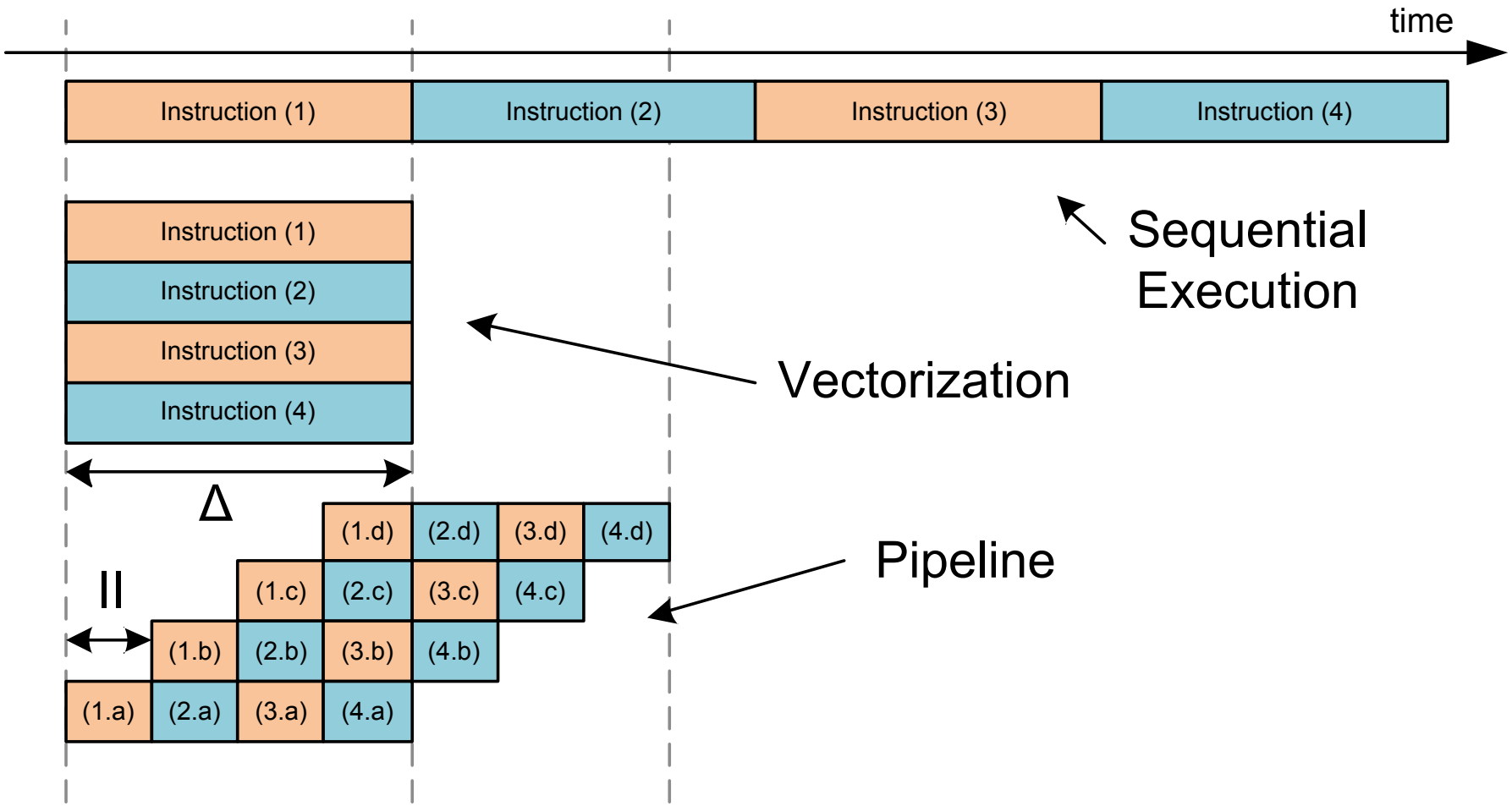
Prérequis - Principe du pipeline

Objectifs : exécution plus rapide . . .



Prérequis - Principe du pipeline

Objectifs : exécution plus rapide . . .



. . . mais à un coût modéré.

- 2 paramètres : Δ, la latence et *II*, l'intervalle d'initialisation

Prérequis - Pipeline de boucle en HLS

- Entrelacement de l'exécution des iterations de la boucle
- Pas de dépendance portée par la boucle

Exemple : $\Delta = 4$, $II = 1$ et $N = 16$

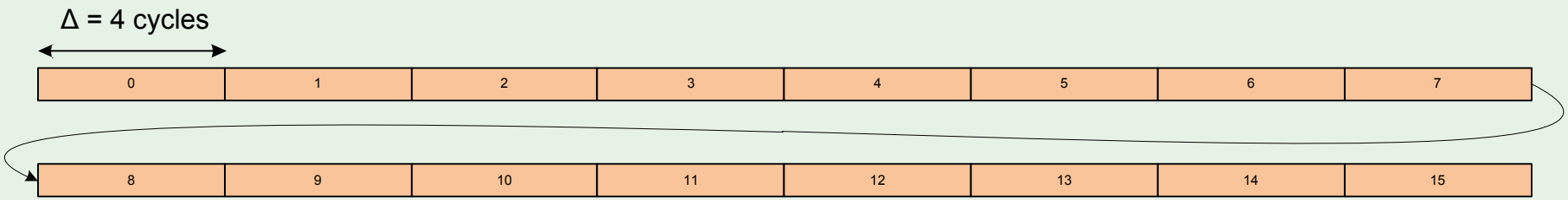
```
for(int i=0;i<N;i++)
  Y[i] = func(Y[i});
```

Prérequis - Pipeline de boucle en HLS

- Entrelacement de l'exécution des iterations de la boucle
- Pas de dépendance portée par la boucle

Exemple : $\Delta = 4$, $II = 1$ et $N = 16$

```
for(int i=0;i<N;i++)
  Y[i] = func(Y[i});
```



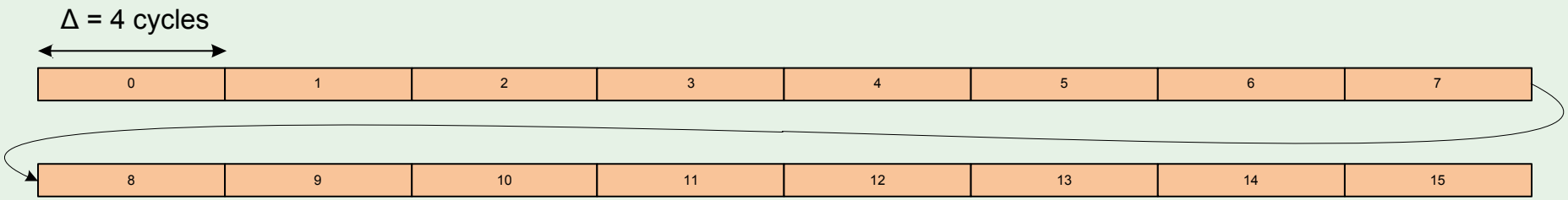
64 cycles

Prérequis - Pipeline de boucle en HLS

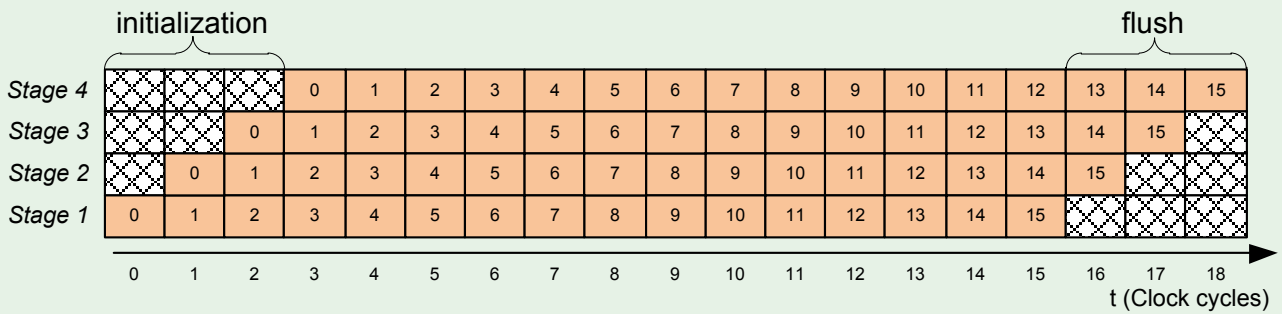
- Entrelacement de l'exécution des iterations de la boucle
- Pas de dépendance portée par la boucle

Exemple : $\Delta = 4$, $II = 1$ et $N = 16$

```
for(int i=0;i<N;i++)
  Y[i] = func(Y[i});
```



64 cycles

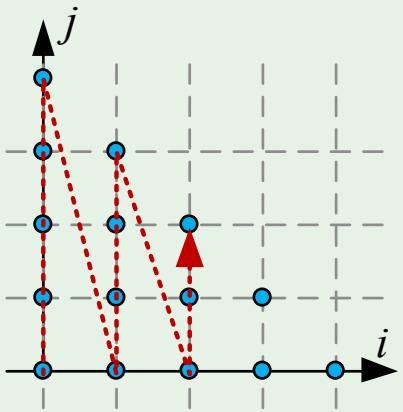


Prérequis - Pipeline de nid de boucles en HLS

- Pas de dépendance portée par la boucle interne
- Éventuellement des dépendances portées par les boucles externes

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```

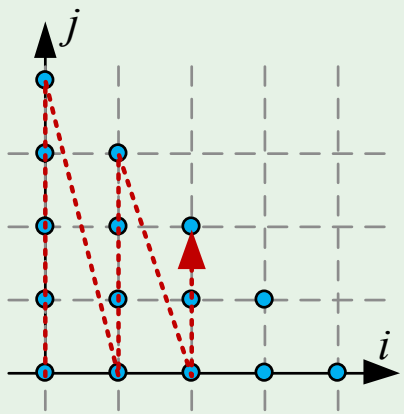


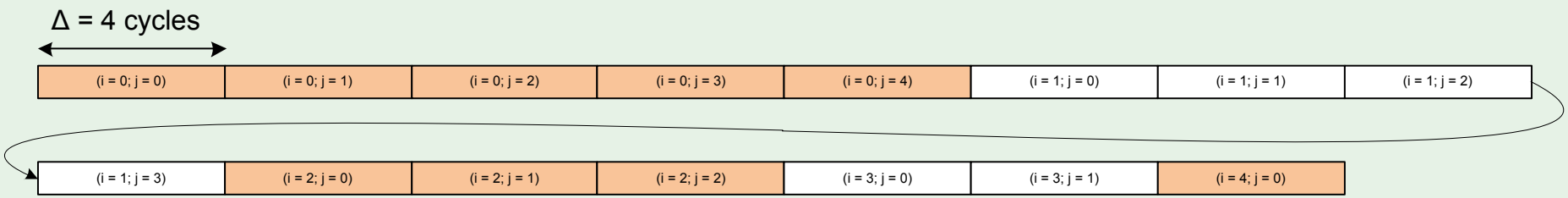
Prérequis - Pipeline de nid de boucles en HLS

- Pas de dépendance portée par la boucle interne
- Éventuellement des dépendances portées par les boucles externes

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```



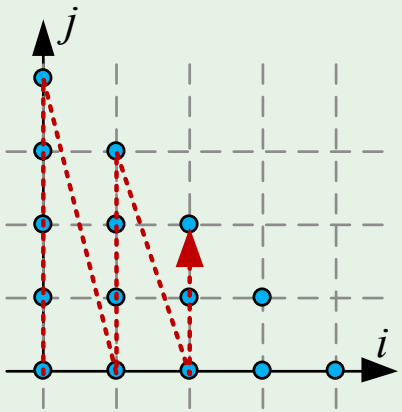


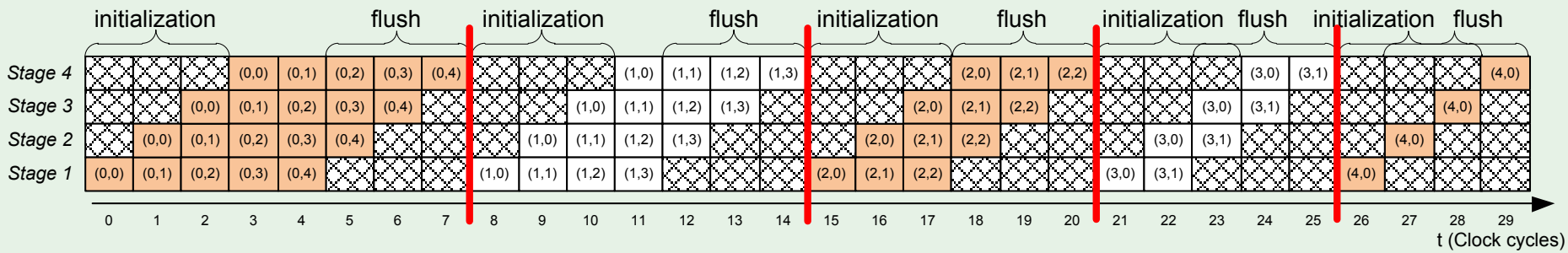
Prérequis - Pipeline de nid de boucles en HLS

- Pas de dépendance portée par la boucle interne
- Éventuellement des dépendances portées par les boucles externes

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```



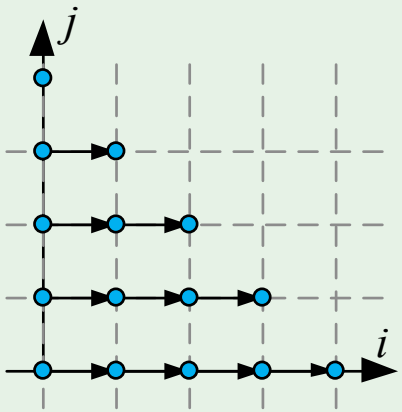


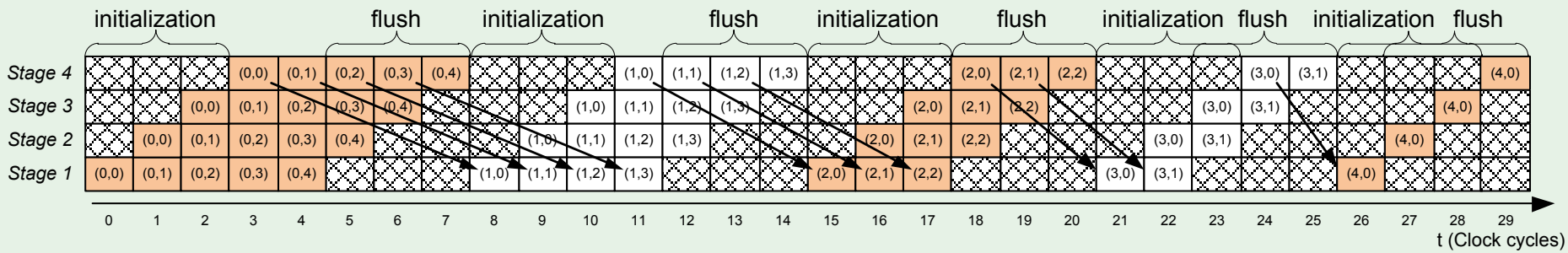
Prérequis - Pipeline de nid de boucles en HLS

- Pas de dépendance portée par la boucle interne
- Éventuellement des dépendances portées par les boucles externes

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```





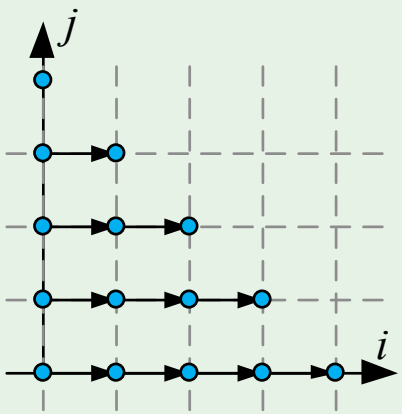
Prérequis - Pipeline de nid de boucles en HLS

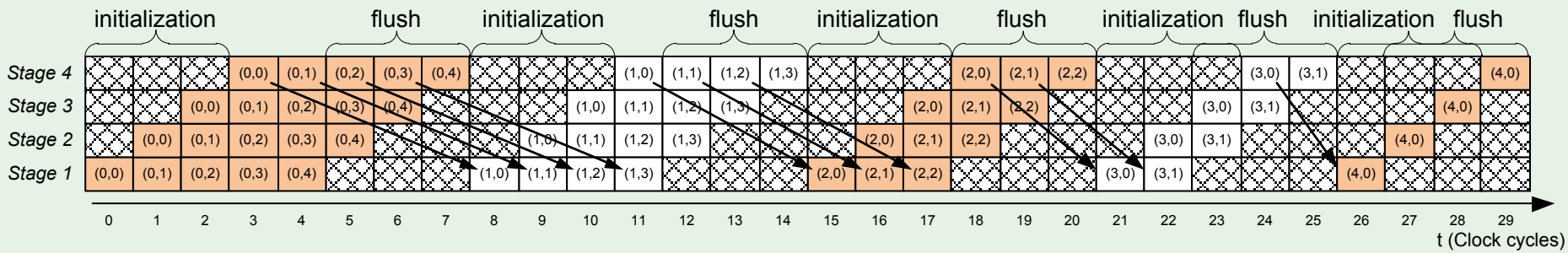
- Pas de dépendance portée par la boucle interne
- Éventuellement des dépendances portées par les boucles externes

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```

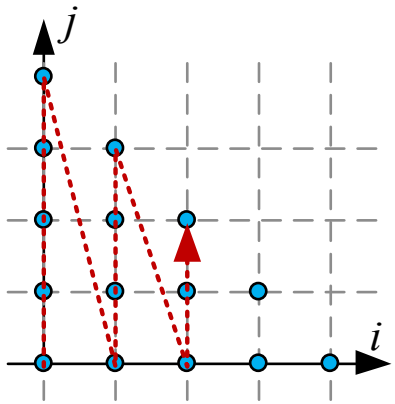
Flush overhead = 50% !



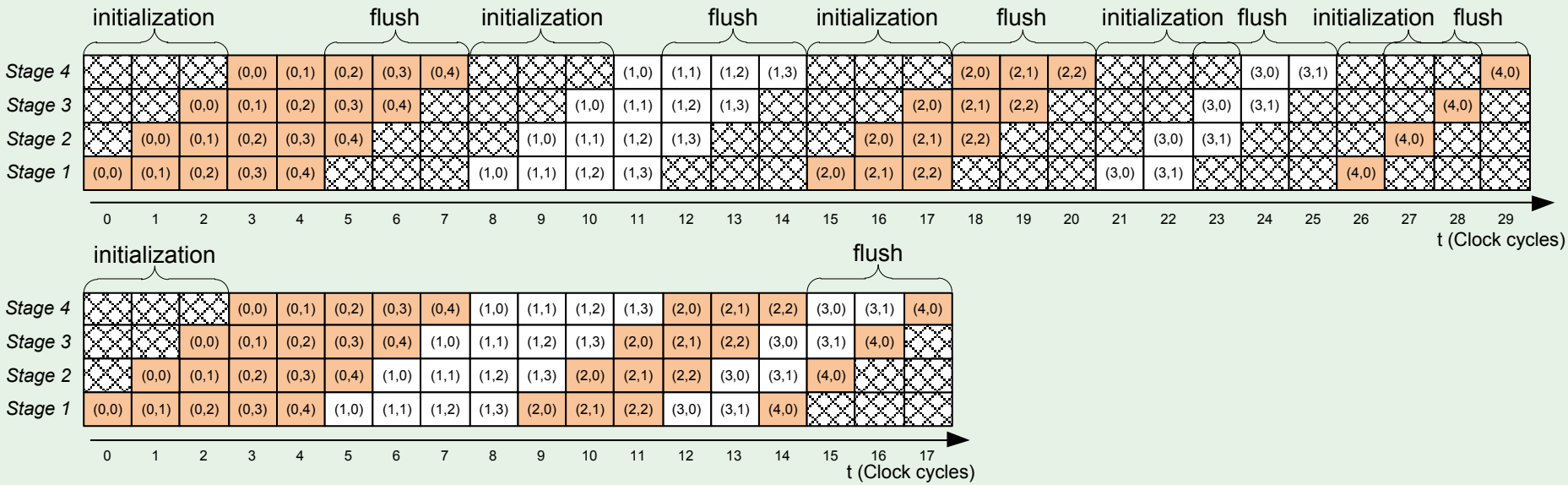


Prérequis - Pipeline de nid de boucles en HLS

- *Compresser* le pipeline
  - Entrelacer les boucles externes

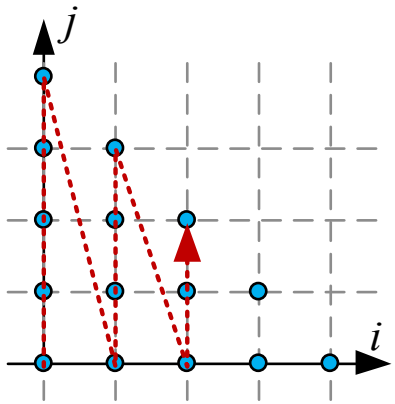


Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



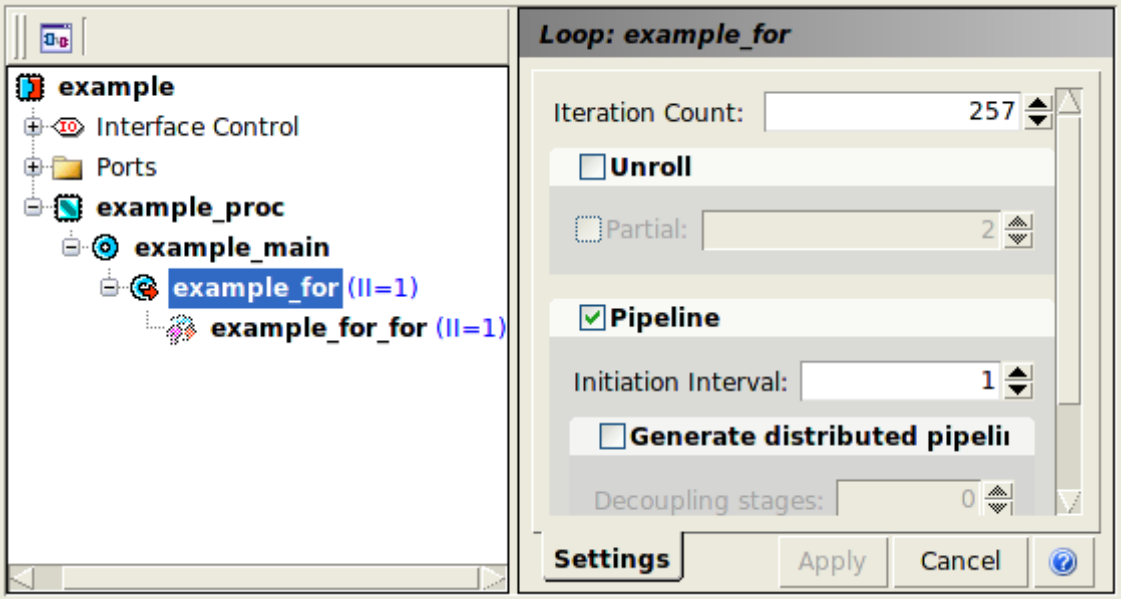
Prérequis - Pipeline de nid de boucles en HLS

- *Compresser* le pipeline
  - Entrelacer les boucles externes
  - Via une mise à plat des boucles



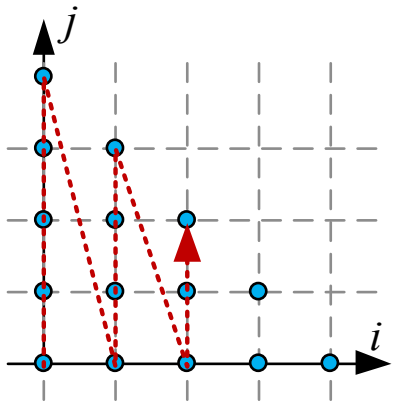
Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```

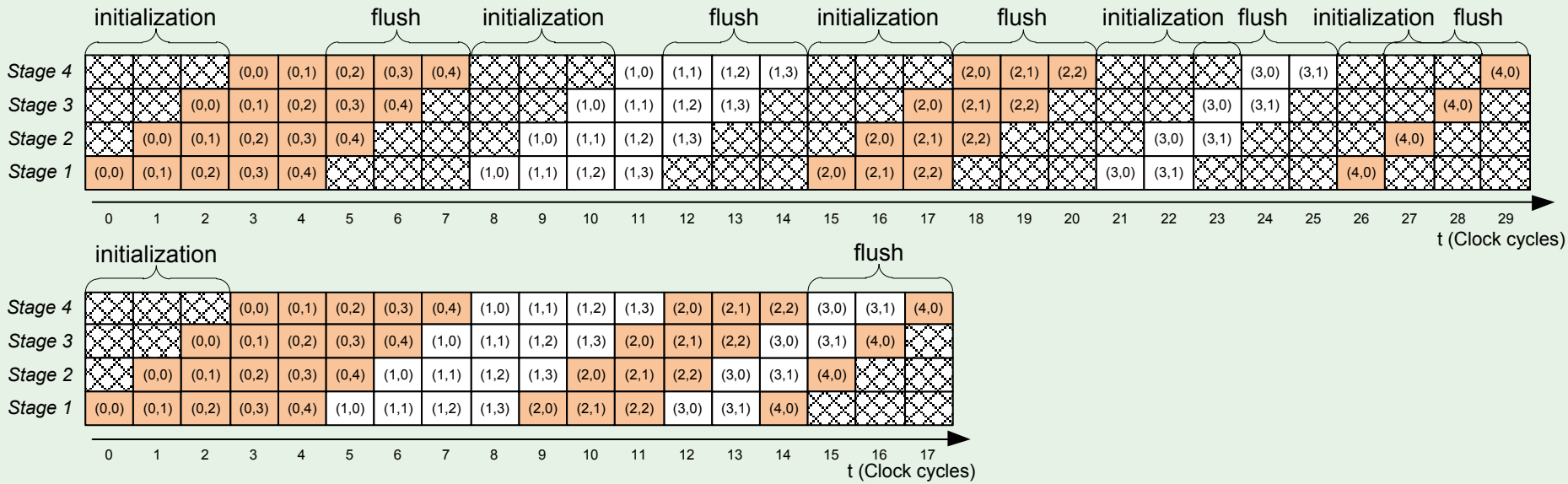


Prérequis - Pipeline de nid de boucles en HLS

- *Compresser* le pipeline
  - Entrelacer les boucles externes
  - Via une mise à plat des boucles

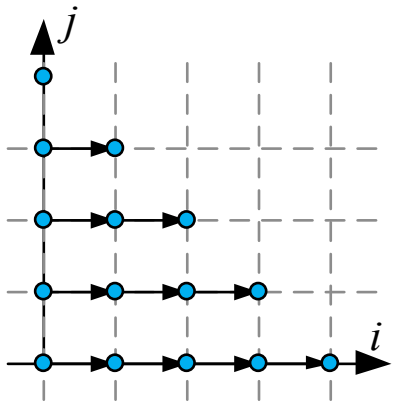


Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

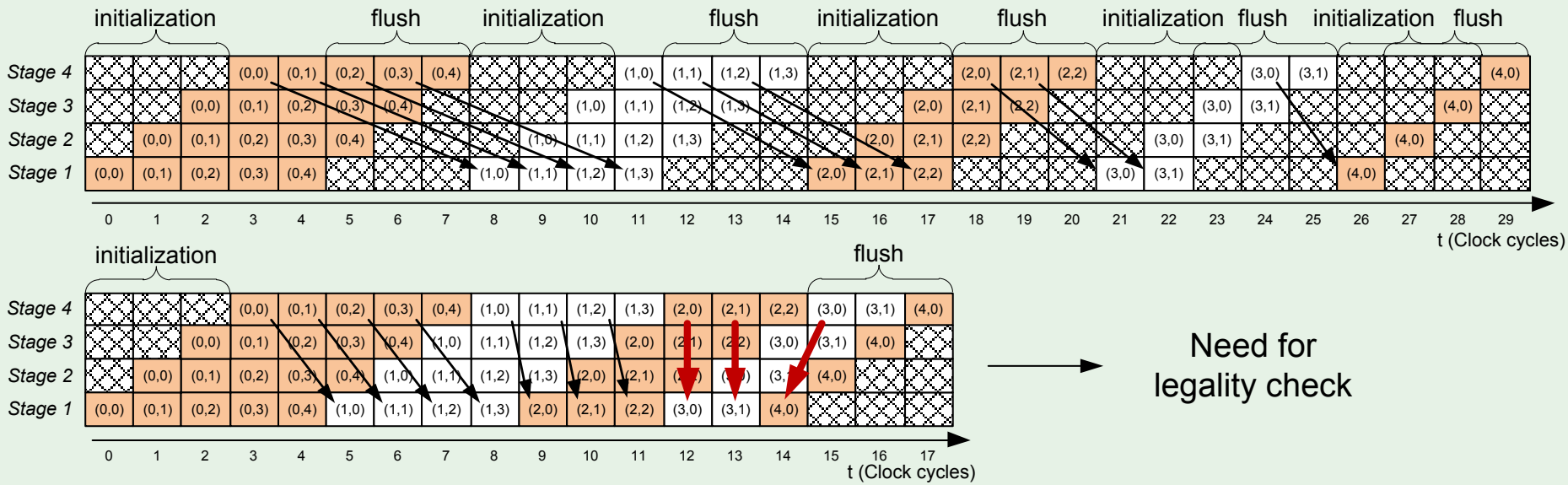


Prérequis - Pipeline de nid de boucles en HLS

- *Compresser* le pipeline
  - Entrelacer les boucles externes
  - Via une mise à plat des boucles
- Dépendances sur la boucle externe
  - **HLS produit un pipeline illégal**



Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



Nos contributions

Amélioration de l'applicabilité du pipeline de nid de boucles

- Test étendu de légalité du pipeline de nid de boucles
  - Applicable à une classe de programmes plus vaste
  - Static Control Parts (SCoPs)
    - Bornes et gardes affines
    - Nids de boucles parametrés
    - Dépendances non-uniformes
- Correction statique de pipeline de nid de boucles
  - Via l'insertion de bulles dans le pipeline **à la compilation**
- Implémentation et validation expérimentale
  - au sein de l'infrastructure de compilation Source à Source GeCoS

Plan

1. Prérequis - Pipeline
   - Principe du pipeline
   - Pipeline de boucle en HLS
   - Pipeline de nid de boucles en HLS
2. Test de légalité
   - Condition de légalité
   - Le modèle polyédrique
   - Compter les points
   - Test de légalité
3. Correction - Insertion statique de bulles
4. Résultats & Conclusion

Condition de légalité

Définition

Le pipeline de nid de boucles est légal si l'utilisation d'une valeur démarre au moins $\Delta - 1$ *cycles après sa définition.*

- $cycle(Use) > cycle(Def) + \Delta - 1$ ($\Delta =$ latence du pipeline)

Sachant qu'une itération démarre à chaque cycle ($II = 1$) :

- il faut s'assurer que $\Delta - 1$ itérations *Def* et *Use*

Condition de légalité

Définition

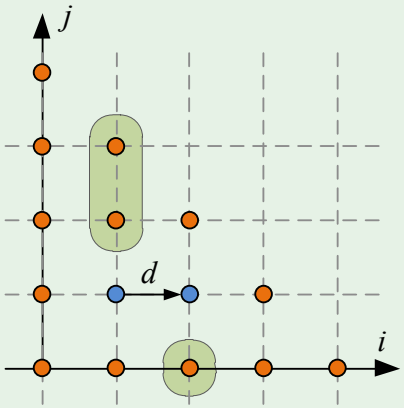
Le pipeline de nid de boucles est légal si l'utilisation d'une valeur démarre au moins $\Delta - 1$ *cycles après sa définition.*

- $cycle(Use) > cycle(Def) + \Delta - 1$ (Δ = latence du pipeline)

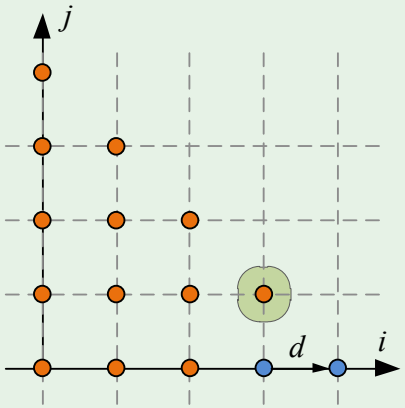
Sachant qu'une itération démarre à chaque cycle ($II = 1$) :

- il faut s'assurer que $\Delta - 1$ itérations *Def* et *Use*

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



$dist = 3$: OK

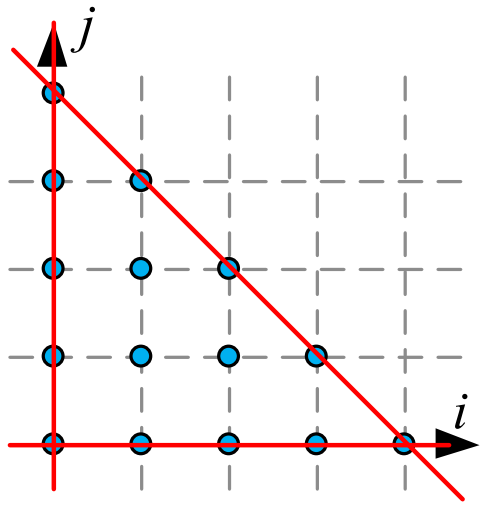


$dist = 1$: Pas OK

Le modèle polyédrique

Grâce au modèle polyédrique, nous avons :

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:     Y[j] = func(Y[j]);
  }
}
```

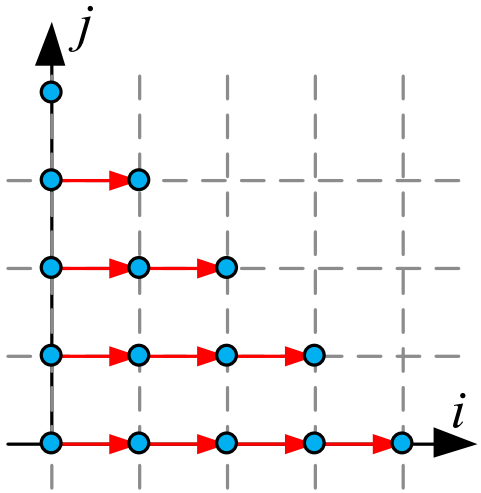


- $D_{S_0} = \{(i,j) | 0 \leq i < N, 0 \leq j < N - i\}$
  - Le domaine d'itération de S_0

Le modèle polyédrique

Grâce au modèle polyédrique, nous avons :

```
/* original source code */
for(int i=0;i<N;i++) {
  for(int j=0;j<N-i;j++){
S0:    Y[j] = func(Y[j]);
  }
}
```



- $D_{S_0} = \{(i,j) | 0 \leq i < N, 0 \leq j < N - i\}$
  - Le domaine d'itération de S_0

- $d : S_0(i,j) \rightarrow S_0(i+1,j)$ où $(i,j,i+1,j) \in D_{S_0} \times D_{S_0}$
  - *Polyhedral Reduced Dataflow Dependency*

Compter les points

Avec les informations précédentes :

- $D_{S_0} = \{(i,j) | 0 \leq i < N, 0 \leq j < N - i\}$
- $d : S_0(i,j) \rightarrow S_0(i+1,j)$ où $(i,j,i+1,j) \in D_{S_0} \times D_{S_0}$

Nous pouvons exprimer le nombre d'itérations qui séparent $S_0(i,j)$ et $S_0(i+1,j)$:

$$card \left(\begin{array}{c} (i,j) \rightarrow (i',j') \mid \\ (i,j,i',j') \in dom(d) \times D_{S_0} \\ (i,j) \prec (i',j') \prec (i+1,j) \end{array} \right)$$

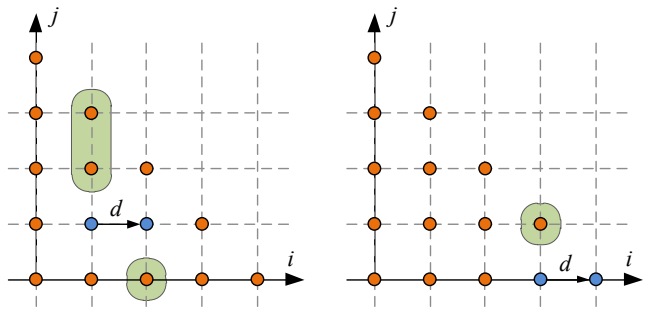
Compter les points

Avec les informations précédentes :

- $D_{S_0} = \{(i,j) | 0 \leq i < N, 0 \leq j < N - i\}$
- $d : S_0(i,j) \rightarrow S_0(i+1,j)$ où $(i,j,i+1,j) \in D_{S_0} \times D_{S_0}$

Nous pouvons exprimer le nombre d'itérations qui séparent $S_0(i,j)$ et $S_0(i+1,j)$:

$$card \left(\begin{array}{c} (i,j) \rightarrow (i',j') \mid \\ (i,j,i',j') \in dom(d) \times D_{S_0} \\ (i,j) \prec (i',j') \prec (i+1,j) \end{array} \right)$$



- Dans notre exemple : $S_0(i,j) \rightarrow (N - i - 1)$
  - Calculé grâce aux bibliothèques ISL & Barvinok

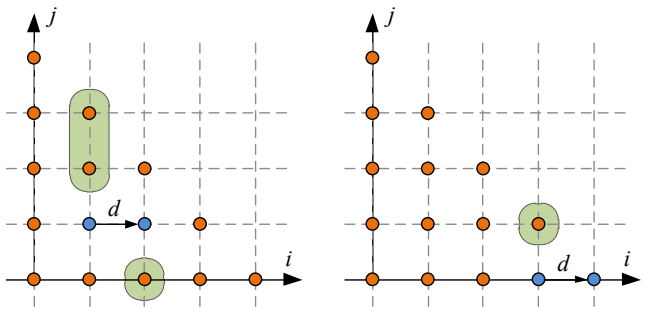
Compter les points

Avec les informations précédentes :

- $D_{S_0} = \{(i,j) | 0 \leq i < N, 0 \leq j < N - i\}$
- $d : S_0(i,j) \rightarrow S_0(i+1,j)$ où $(i,j,i+1,j) \in D_{S_0} \times D_{S_0}$

Nous pouvons exprimer le nombre d'itérations qui séparent $S_0(i,j)$ et $S_0(i+1,j)$:

$$card \left(\begin{array}{c} (i,j) \rightarrow (i',j') \mid \\ (i,j,i',j') \in dom(d) \times D_{S_0} \\ (i,j) \prec (i',j') \prec (i+1,j) \end{array} \right)$$



- Dans notre exemple : $S_0(i,j) \rightarrow (N - i - 1)$
  - Calculé grâce aux bibliothèques ISL & Barvinok
- Borne (min) conservative via l'expansion de Bernstein
  - Dans notre exemple : $min(N - i - 1)$ sur $dom(d) = 1$

Test de légalité

Condition de légalité : $\mathit{iteration}(\mathit{Use}) \succ \mathit{iteration}(\mathit{Def}) + \Delta - 1$

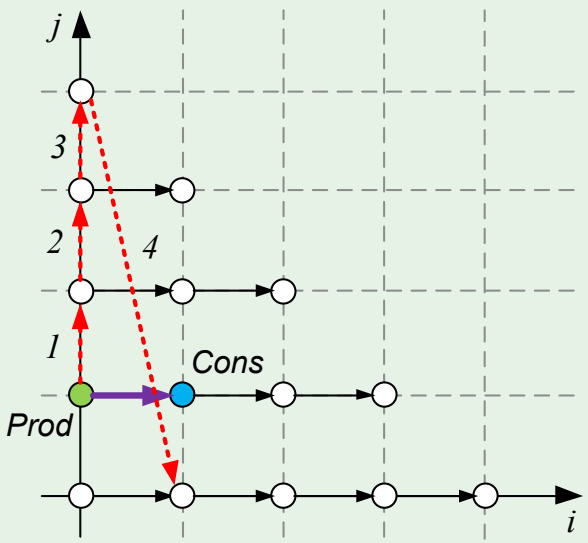
- Construire l'ensemble $\mathit{Prod}_{\mathit{errors}}$ où la condition est **fausse**
- Le pipeline de nid de boucles est légal ssi $\mathit{Prod}_{\mathit{errors}} = \emptyset$

Test de légalité

Condition de légalité : $\mathit{iteration}(\mathit{Use}) \succ \mathit{iteration}(\mathit{Def}) + \Delta - 1$

- Construire l'ensemble $\mathit{Prod}_{\mathit{errors}}$ où la condition est **fausse**
- Le pipeline de nid de boucles est légal ssi $\mathit{Prod}_{\mathit{errors}} = \emptyset$

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



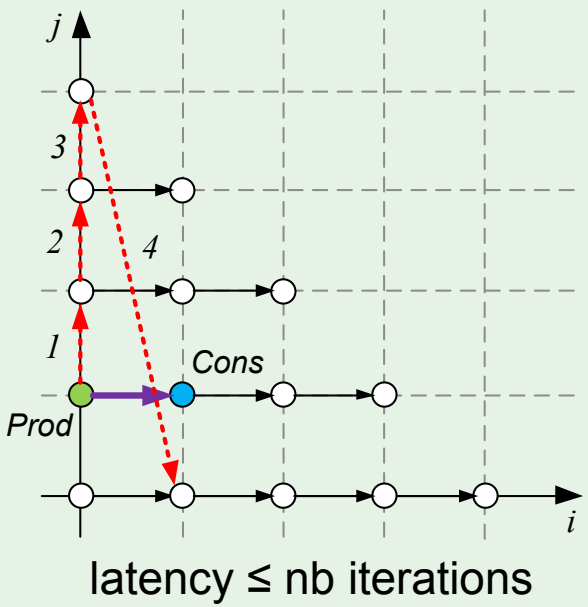
latency ≤ nb iterations

Test de légalité

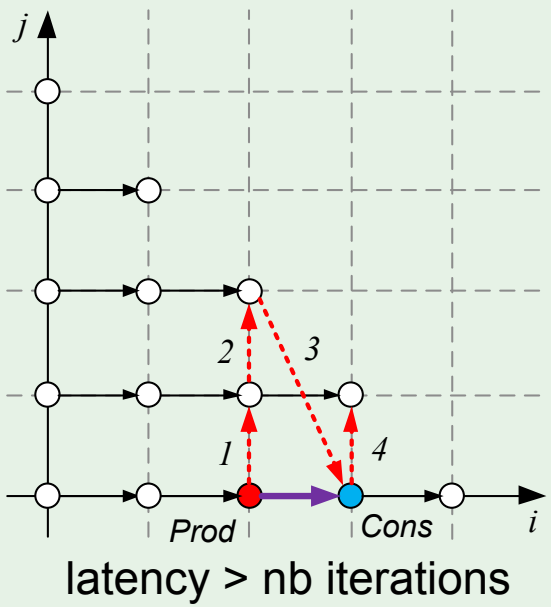
Condition de légalité : $\mathit{iteration}(\mathit{Use}) \succ \mathit{iteration}(\mathit{Def}) + \Delta - 1$

- Construire l'ensemble $\mathit{Prod}_{\mathit{errors}}$ où la condition est **fausse**
- Le pipeline de nid de boucles est légal ssi $\mathit{Prod}_{\mathit{errors}} = \emptyset$

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



latency ≤ nb iterations



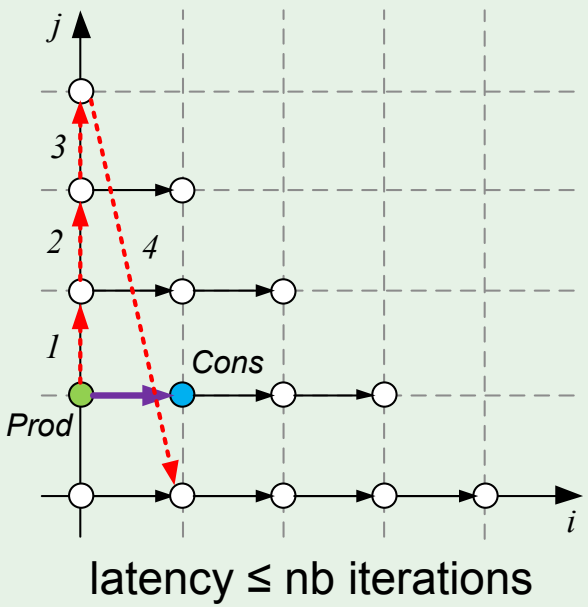
latency > nb iterations

Test de légalité

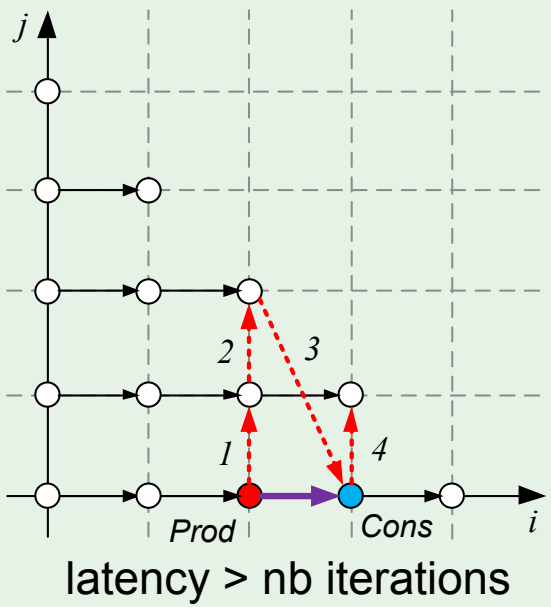
Condition de légalité : $\mathit{iteration}(\mathit{Use}) \succ \mathit{iteration}(\mathit{Def}) + \Delta - 1$

- Construire l'ensemble $\mathit{Prod}_{\mathit{errors}}$ où la condition est **fausse**
- Le pipeline de nid de boucles est légal ssi $\mathit{Prod}_{\mathit{errors}} = \emptyset$

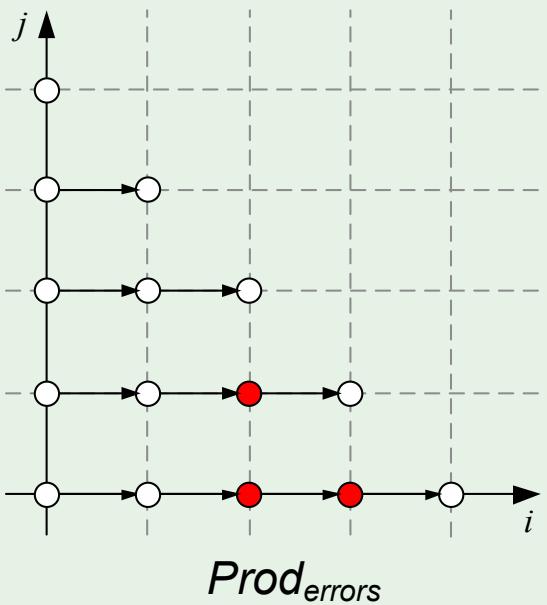
Exemple : $\Delta = 4$, $II = 1$ et $N = 5$



latency ≤ nb iterations



latency > nb iterations

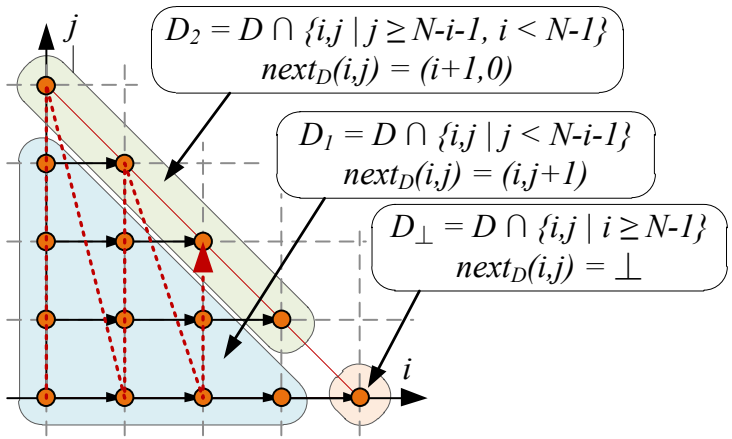


$\mathit{Prod}_{\mathit{errors}}$

Regarder Δ itérations plus loin

Approche basée sur la fonction *next* proposée dans la technique de génération de code de Boulet & Feautrier

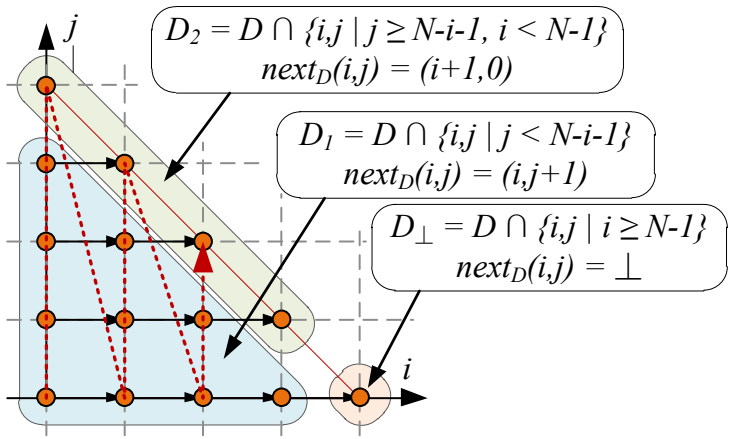
$$next(i,j) = \begin{cases} (i, j+1) & \text{if} & j < N - i - 1 \\ (i+1, 0) & \text{elseif} & i < N - 1 \\ \bot & \text{otherwise} & \end{cases}$$



Regarder Δ itérations plus loin

Approche basée sur la fonction *next* proposée dans la technique de génération de code de Boulet & Feautrier

$$next(i,j) = \begin{cases} (i, j+1) & \text{if} & j < N-i-1 \\ (i+1, 0) & \text{elseif} & i < N-1 \\ \bot & \text{otherwise} & \end{cases}$$



$$next^{\Delta}(i,j) = \overbrace{next \bullet next \bullet \ldots \bullet next}^{\Delta}(i,j)$$

Regarder Δ itérations plus loin

$$next^4(i,j) =$$

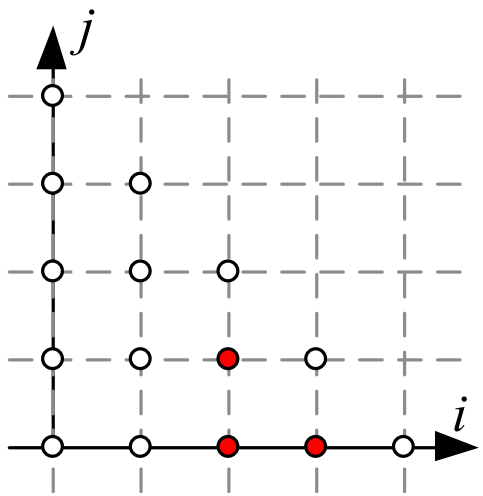
$$\begin{cases} (i, j+4) & \text{if} & j \leq N-i-5 \\ (i+1, 3) & \text{elseif} & i \leq N-5 \wedge j = N-i-1 \\ (i+1, 2) & \text{elseif} & i \leq N-4 \wedge j = N-i-2 \\ (i+1, 1) & \text{elseif} & i \leq N-3 \wedge j = N-i-3 \\ (i+1, 0) & \text{elseif} & i \leq N-4 \wedge j = N-i-4 \\ (N-1, 0) & \text{elseif} & i = N-3 \wedge j = 1 \wedge N \geq 3 \\ (N-2, 0) & \text{elseif} & i = N-4 \wedge j = 3 \wedge N \geq 4 \\ \bot & \text{else} & \end{cases}$$

Test de légalité

$Prod_{errors}$ est défini par :

$$\{(i,j) \in dom(d) \mid d(i,j) \prec next^{\Delta}(i,j)\}$$
$$\cup \quad \{(i,j) \in dom(d) \mid next^{\Delta}(i,j) \in \{\bot\}\}$$

(Construit grâce à ISL)

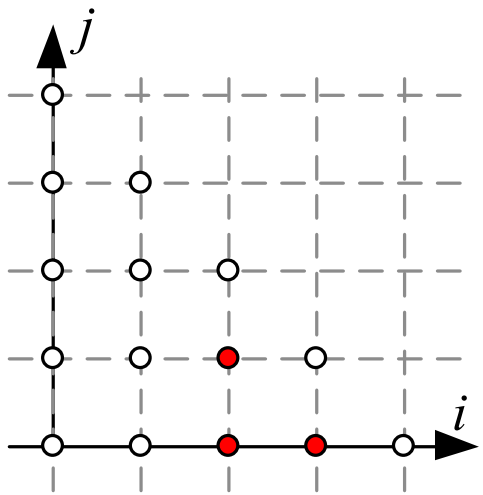


Test de légalité

$Prod_{errors}$ est défini par :

$$\{(i,j) \in dom(d) \mid d(i,j) \prec next^{\Delta}(i,j)\}$$
$$\cup \quad \{(i,j) \in dom(d) \mid next^{\Delta}(i,j) \in \{\bot\}\}$$

(Construit grâce à ISL)



$$Prod_{errors} = \{(i,j) \in D_{S_0} \mid N-4 < i < N-1 \wedge j < N-i-1\}$$

Plan

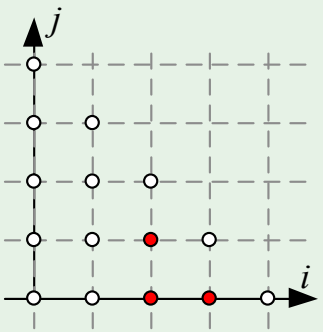
1. Prérequis - Pipeline
   - Principe du pipeline
   - Pipeline de boucle en HLS
   - Pipeline de nid de boucles en HLS
2. Test de légalité
   - Condition de légalité
   - Le modèle polyédrique
   - Compter les points
   - Test de légalité
3. Correction - Insertion statique de bulles
4. Résultats & Conclusion

Correction - Insertion statique de bulles

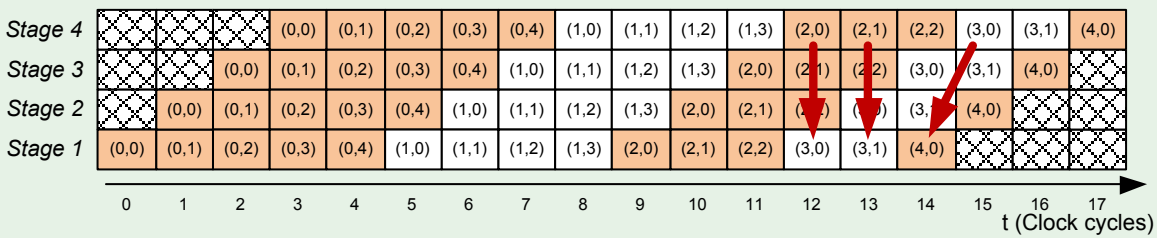
- Assurer la légalité en insérant des *"bulles"* à la compilation
  - ajouter des itérations *NOP* dans le domaine
- Mise à plat via la génération de code de Boulet & Feautrier
  - bulles matérialisées par des gardes dans le code source

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

$Prod_{errors}$




```
i=0;j=0;
while(i<N) {
#pragma ignore_mem_depcy Y
S0: Y[j] = func(Y[j]);
    if(j < N-i-1) j++;
    else j=0,i++;
}
```

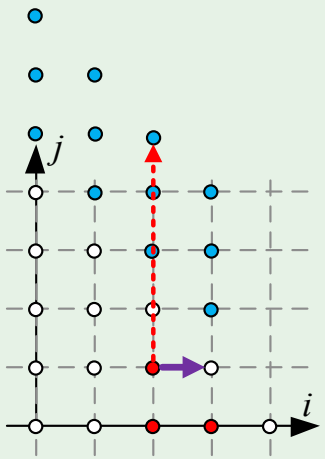


Correction - Insertion statique de bulles

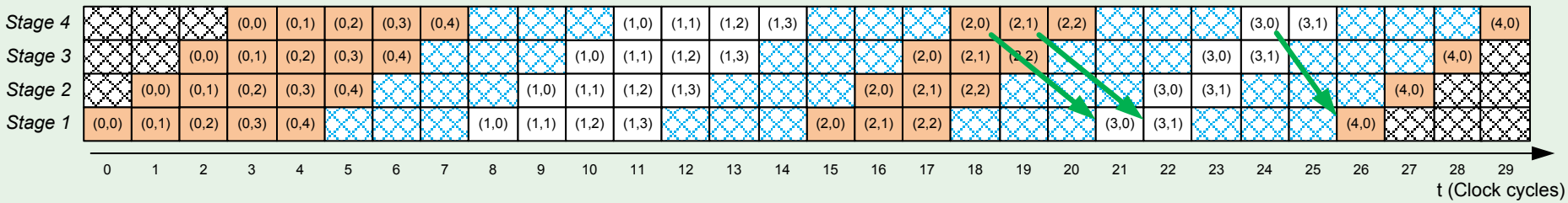
- Assurer la légalité en insérant des *"bulles"* à la compilation
  - ajouter des itérations *NOP* dans le domaine
- Mise à plat via la génération de code de Boulet & Feautrier
  - bulles matérialisées par des gardes dans le code source

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

Vider le pipeline partout $\rightarrow$ simple pipeline




```
i=0;j=0;
while(i<N) {
    if(j < N-i)
#pragma ignore_mem_depcy Y
S0:     Y[j] = func(Y[j]);
    if(j < N-i+2) j++;
    else j=0,i++;
}
```

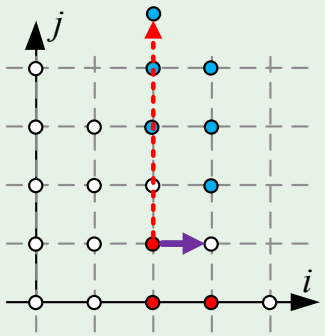


Correction - Insertion statique de bulles

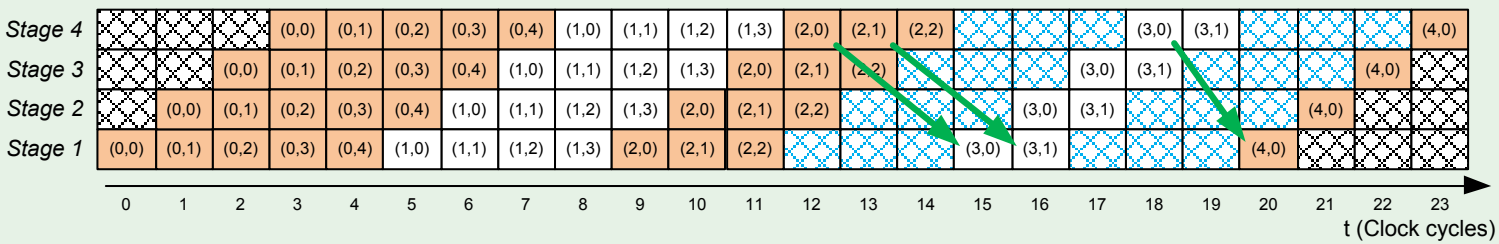
- Assurer la légalité en insérant des *"bulles"* à la compilation
  - ajouter des itérations *NOP* dans le domaine
- Mise à plat via la génération de code de Boulet & Feautrier
  - bulles matérialisées par des gardes dans le code source

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

Vider seulement quand nécessaire




```
i=0;j=0;
while(i<N) {
    if(j < N-i)
#pragma ignore_mem_depcy Y
S0:     Y[j] = func(Y[j]);
    if(j < N-i+2 && i>1 && i<4) j++;
    else if(j < N-i-1) j++;
    else j=0,i++;
}
```

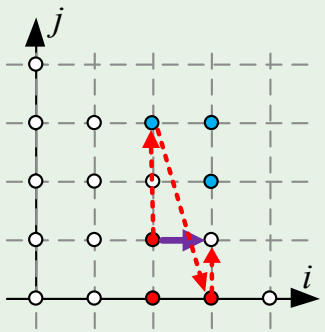


Correction - Insertion statique de bulles

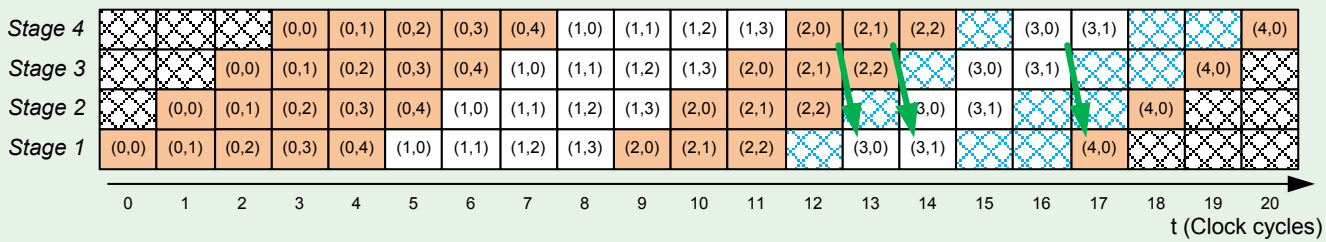
- Assurer la légalité en insérant des *"bulles"* à la compilation
  - ajouter des itérations *NOP* dans le domaine
- Mise à plat via la génération de code de Boulet & Feautrier
  - bulles matérialisées par des gardes dans le code source

Exemple : $\Delta = 4$, $II = 1$ et $N = 5$

Insertion de bulles optimisée




```
i=0;j=0;
while(i<N) {
    if(j < N-i)
#pragma ignore_mem_depcy Y
S0:     Y[j] = func(Y[j]);
    if(j<3 && i>1 && i<4) j++;
    else if(j < N-i-1) j++;
    else j=0,i++;
}
```



Plan

1. Prérequis - Pipeline
   - Principe du pipeline
   - Pipeline de boucle en HLS
   - Pipeline de nid de boucles en HLS
2. Test de légalité
   - Condition de légalité
   - Le modèle polyédrique
   - Compter les points
   - Test de légalité
3. Correction - Insertion statique de bulles
4. Résultats & Conclusion

Résultats qualitatifs

- Pipeline sur les 2 boucles les plus internes
  - la boucle la plus interne est parallèle
  - le 2ème niveau de boucle porte des dépendances
- Temps pour construire l'ensemble des bulles optimisé

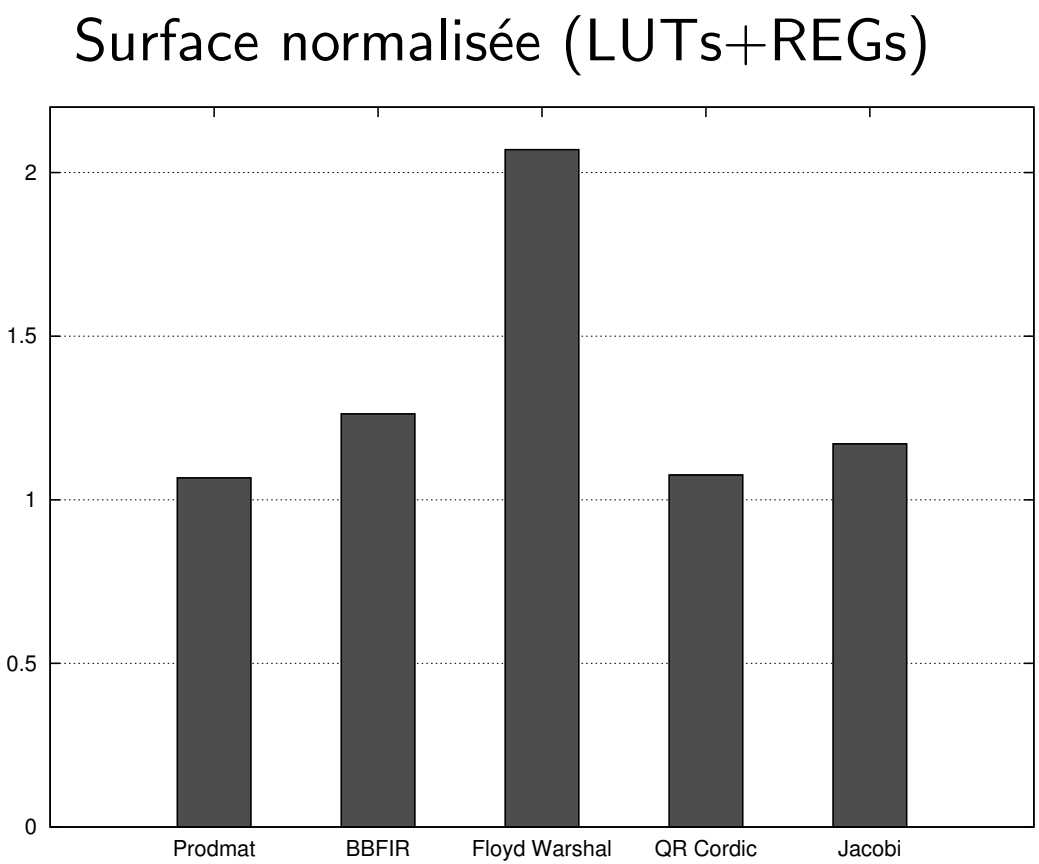
Kernel	Δ	**HLS Tool Nested Pipeline**	**Runtime (ms)**
Prodmat	4	**illegal**	30
BBFIR (2D)	4	no	93
Jacobi	8	no	236
Floyd Warshall	3	no	110
QR Cordic	13	no	327

Résultats quantitatifs - Coût en surface

Nid de boucles mis à plat avec insertion de bulles vs Pipeline 1D

Métrique

- Coût en surface : ressources matérielles (FPGA)

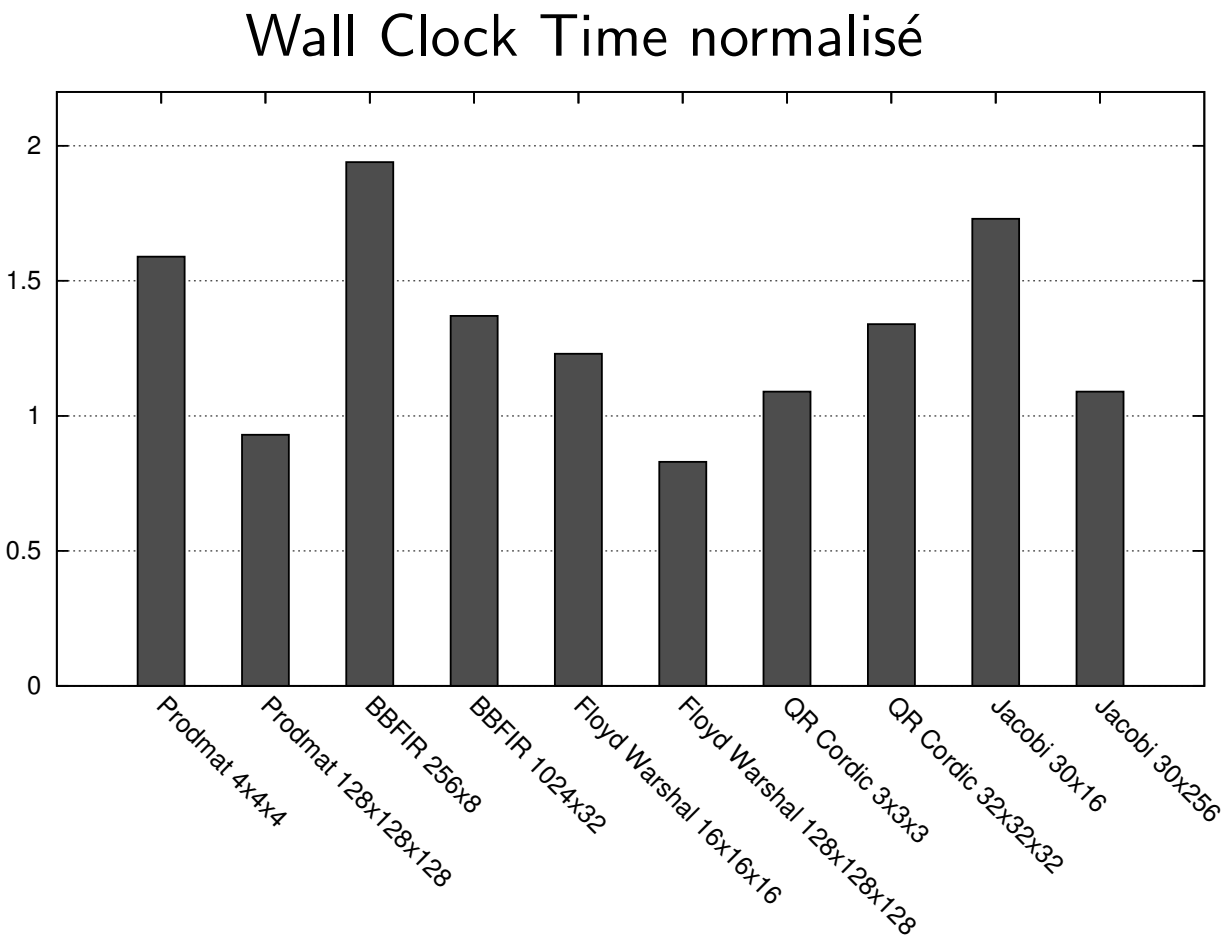


Résultats quantitatifs - Accélération

Nid de boucles mis à plat avec insertion de bulles vs Pipeline 1D

Métrique

- *Wall Clock Time* : performance (#cycles / fréquence$_{max}$)



Conclusion & Perspectives

Résumé

- Amélioration de l'applicabilité du pipeline de nid de boucles
- Technique de correction du pipeline
- Mise à plat des nids de boucles
- Implémentation et validation expérimentale

Conclusion

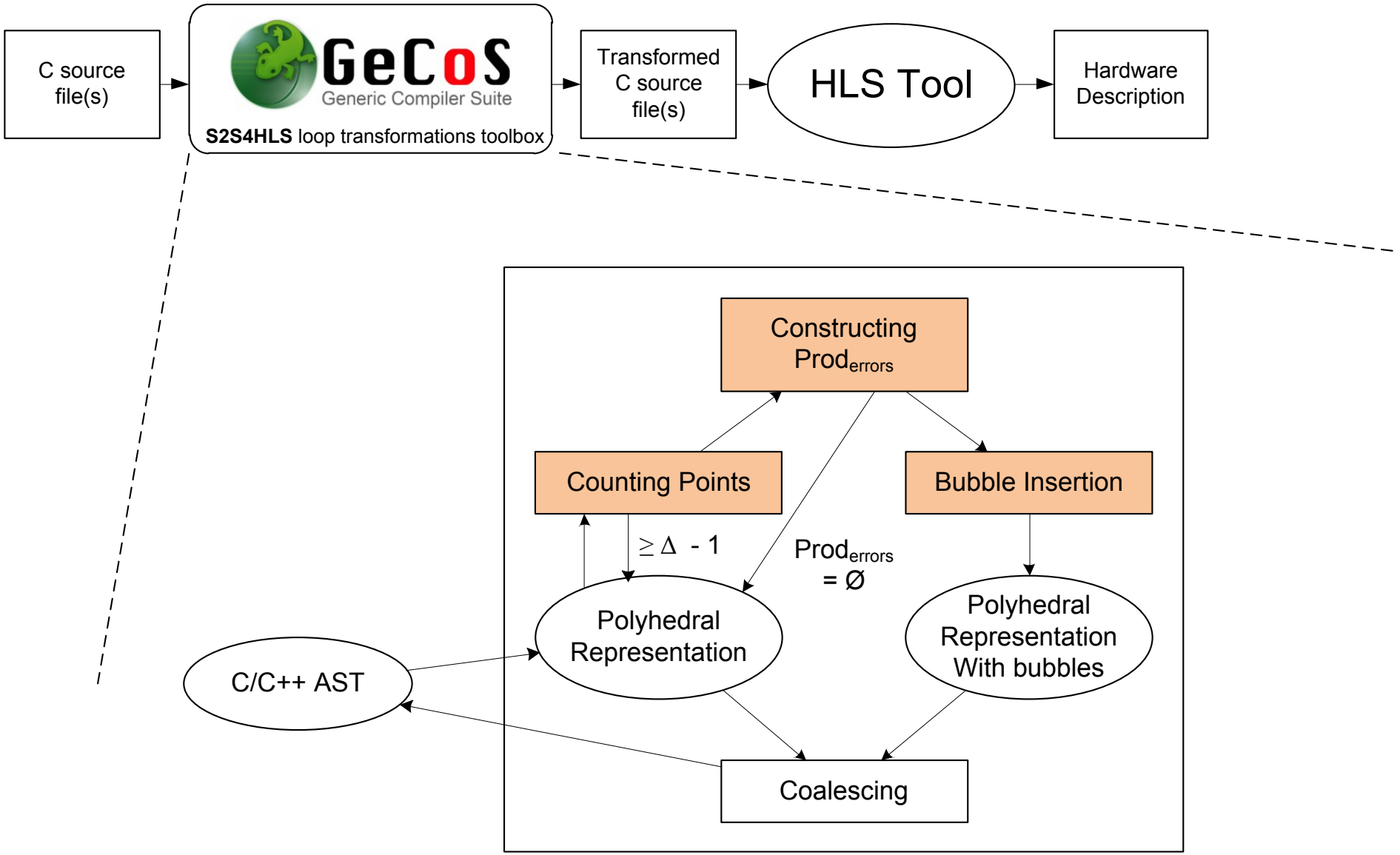
- HLS bénéficie des transformations de haut niveau
- Source à Source : une bonne solution pour contourner les limitations des outils de HLS

Perspectives

- Améliorer la génération de code pour limiter le surcoût en surface

Questions & Discussion

1. Prérequis - Pipeline
   - Principe du pipeline
   - Pipeline de boucle en HLS
   - Pipeline de nid de boucles en HLS
2. Test de légalité
   - Condition de légalité
   - Le modèle polyédrique
   - Compter les points
   - Test de légalité
3. Correction - Insertion statique de bulles
4. Résultats & Conclusion

Merci pour votre attention.
Questions?

Putting to work