ENSSAT EII3
Master 2 Recherche SISEA — parcours systèmes embarqués

TD Conception de Systèmes Intégrés

Synthèse de haut-niveau

Olivier Sentieys

1 Filtrage récursif

On s'intéresse à la mise en œuvre d'un algorithme de traitement du signal récursif, le filtre RII transverse :

$$y(n) = \sum_{i=0}^{N} a(i).x(n-i) + \sum_{i=1}^{P} b(i).y(n-i)$$

1. On désire mettre en œuvre les calculs de ce filtre sur une architecture pipeline spécifique. Celle-ci intègre des registres, multiplieurs et additionneurs.
   Représentez un graphe flot de signal de ce filtre lorsque $N = P = 4$ lorsque les indices de la $\sum$ sont croissants.
2. Sur le graphe précédent, indiquez le temps T_i qui sépare l'utilisation de chaque variable $y(n-i)$, de la fin du calcul de $y(n)$ (pour une itération du filtre). Chaque temps sera donné en fonction de $Tadd$ et $Tmult$, temps de fonctionnement des additionneurs et des multiplieurs.
3. Trouvez la fréquence maximale d'échantillonnage Fe_{max} du filtre.
4. Quel est le retard T_d en fonction de $Tmult$ et $Tadd$ qui sépare l'acquisition de $x(n)$ et le calcul de $y(n)$; on donnera ce retard en fonction de N et P. Indiquez en quoi ce retard peut être gênant.
5. On calcule les produits de convolution en faisant décroître les indices, i.e. en commençant les accumulations sur les échantillons d'indice les plus anciens, $x(n-N)$ et $y(n-P)$ respectivement. Dessinez le graphe flot de signal. Que deviennent dans ce cas Fe_{max} et T_d.
6. Pour $N = P = 4$, et $Tmult = 4ns$, $Tadd = 2ns$, indiquez l'ordonnancement du graphe aboutissant à une architecture pipeline à coût minimum pour une période d'échantillonnage de $12ns$.

2 Filtre non récursif RIF symétrique

1. Donner le graphe flot de données d'un filtre RIF symétrique sur $N = 8$ points dont l'équation est :
$$y_n = \sum_{i=0}^{3} a(i).[x(i) + x(7-i)]$$
2. Proposer une architecture pipeline pour implémenter les calculs du filtre dans les cas :
a) le temps de calcul est de $60ns$,
b) le temps de calcul est de $30ns$,
c) le temps de calcul est de $7ns$.
On demande d'expliquer clairement l'optimisation faite sur la minimisation du nombre de ressources. Pour chaque résultat, dessinez l'architecture pipeline et son fonctionnement.
On considérera que $Tadd = 5ns$, $Tmult = 7ns$.

3 Algorithme LMS sur circuit spécifique

Soit l'algorithme LMS suivant :

$$y_n = \sum_{i=0}^{N-1} \underline{h}_n(i).\underline{x}_n(i)$$
$$e_n = d_n - y_n$$
$$\underline{h}_{n+1}(i) = \underline{h}_n(i) + \mu.e_n.\underline{x}_n(i) \quad \forall i \in \{0 \dots N-1\}$$

On posera $N = 3$. Le temps d'une multiplication est de 2 cycles, celui d'une addition/soustraction ou d'un transfert mémoire est de 1 cycle. Dn, les vecteurs h et x viennent de la mémoire. Le vecteur h et la donnée Yn doivent être rechargés en mémoire. Le coût du matériel est défini dans le tableau ci-dessous. L'additionneur/soustracteur est un composant unique.

multiplieur	additionneur-soustracteur	registres	multiplexeur tristate
1	0.2	0.15	0.05

1. La contrainte de temps est fixée à 20 cycles. Combien d'opérateurs ($\times$, $\pm$) sont nécessaires pour respecter cette contrainte.
2. Dessiner le graphe flot de données de l'algorithme. Effectuer son ordonnancement par liste sur les ressources définies au 1.
3. En déduire la structure de l'architecture du circuit et son coût.
4. Peut on améliorer l'ordonnancement (ou le graphe lui même) afin de diminuer le coût.
5. Reprendre les questions 1 à 4 en fixant la contrainte de temps à 10 cycles.
6. Quelle est la fréquence maximale de réalisation du filtre LMS.

4 Circuit pour la résolution d'une équation différentielle.

La résolution numérique de l'équation différentielle 1 peut s'effectuer au moyen de l'algorithme ci dessous. Le résultat se trouve dans la variable y.

$$\frac{d^2y}{dx^2} + 5\frac{dy}{dx} + 3y = 0 \tag{1}$$

```
While x < a
  u1 = u - 5 * x * (u * dx) - 3 * y * dx;
  y1 = y + (u * dx);
  x1 = x + dx;
  x = x1; u = u1; y = y1;
End While;
```

L'architecture ne devra posséder qu'un seul multiplieur et un seul additionneur/soustracteur fonctionnant à une vitesse de Tcycle.

Les variables x, y, dx, et u sont placées dans des registres séparés et dédiés à ces variables.

Les variables 3 et 5 sont au sein d'une seule *register file* (regroupement de 2 registres accessibles 1 parmi 2) connecté au multiplieur.

1. Dessinez le graphe flot de données de l'intérieur de la boucle while sans aucun *déroulage* de boucle. Vous donnerez son ordonnancement ASAP ainsi qu'un ordonnancement sur les 2 opérateurs de l'architecture.
2. Dessinez l'architecture de l'unité de traitement permettant d'implémenter le cœur de la boucle. Vous donnerez également la table d'occupation des registres créés en plus de ceux cités dans l'introduction.
3. Expliquez quel serait le gain de cycle si on utilisait 2 multiplieurs pour ordonnancer ce graphe.
4. Expliquez comment gérer la boucle while en utilisant l'architecture précédemment définie. Indiquez alors la structure de l'unité de contrôle du circuit.
5. En vous basant sur cet exemple, expliquez ce qu'apporte la synthèse de haut niveau par rapport aux techniques actuelles de conception de circuits.

5 Transformations algorithmiques sur filtre FIR

Soit le filtre numérique à réponse impulsionnelle finie sur N points dont le graphe flot de signal est donné figure 1.

Les multiplieurs et les additionneurs ont un temps de 1 et respectivement un coût de 1 et 0.3. Les registres ont un temps négligeable et un coût de 0.2.

1. Quel est le chemin critique Tcc du graphe en fonction de N. Que se passe-t-il si la période d'échantillonnage est inférieur à Tcc. On considérera que l'ordonnancement du filtre sur 1 multiplieur et 1 additionneur engendre 4 registres (le coût de l'architecture vaut donc 2.1). On négligera les composants d'interconnexions.
2. Effectuez une transformation de type *tree-height reduction* (associativité). Dessinez le graphe transformé pour $N = 5$. En déduire son Tcc. Généraliser le calcul de Tcc en fonction de N, longueur du filtre. On considérera un ordonnancement du filtre sur 1

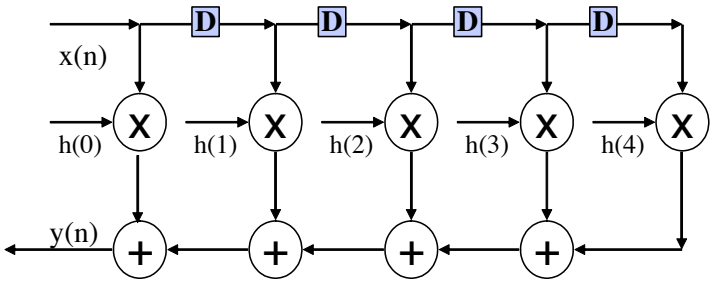


FIGURE 1 – Graphe du filtre RIF

multiplieur et 1 additionneur. Donner une estimation du coût de l'architecture. Conclure sur l'intérêt de cette transformation.

3. Expliquer comment effectuer un *retiming* afin d'arriver à la forme transposée du filtre (délais sur la chaîne d'additionneur). Quel est dans ce cas le temps du chemin critique du graphe. Donner, en particulier, une estimation du coût de l'architecture. Quels sont, à votre avis, les avantages et inconvénients de cette structure.

6 Retiming et ordonnancement d'un filtre IIR d'ordre 2

Soit le graphe flot de signal (IIR ordre 2, structure canonique) de la figure 2. Expliquez comment obtenir les deux versions *retimed* (graphes du bas de la figure 2) à partir du graphede départ (figure 2 haut).

Pour chacun des trois graphes donnez :

1. le temps du chemin critique Tcc du SFG ;
2. le temps de la boucle critique Tbc ;
3. une estimation du nombre d'opérateurs lorsqu'on ordonnance le graphe de manière à respecter un temps de retard inférieur à Tcc.

7 Synthèse d'architecture de la multiplication complexe

Une multiplication complexe prend en entrée 2 nombres complexes $(A+j.B)$ et $(C+j.D)$, génère 1 nombre complexe $(Re + j.Im)$, et a pour équation :

$$Re + j.Im = (A + j.B) \cdot (C + j.D)$$

Le temps d'un multiplieur est de 60 ns, celui d'un additionneur/soustracteur est de 30 ns. Le coût de ces opérateurs est respectivement de 4 et de 1. On dispose de la possibilité de piloter l'architecture à une période d'horloge interne de 30 ns. Les données A, B, C, et D sont lues sur les bus de données, et les résultats Re et Im doivent être écrits sur ces mêmes bus.

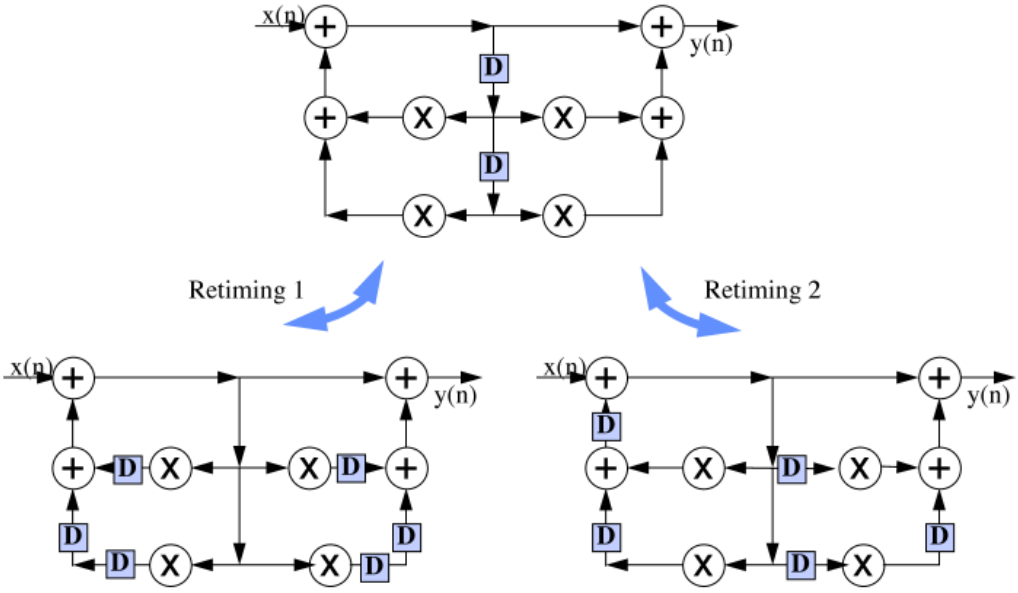


FIGURE 2 – Filtre IIR : graphe de départ (structure directe) et deux versions après *retiming*

1. Tracez le Graphe flot de cet algorithme. Donnez le temps du chemin critique Tcc.
2. Quel est le nombre minimum d'opérateurs pour exécuter ce graphe à une cadence de 240 ns ? Effectuez un ordonnancement de ce graphe de manière à respecter la contrainte.
3. Donnez le schéma de l'architecture obtenue.
4. Si on impose le fait que A, B, C, et D sont dans la même mémoire, quelles modifications sont apportées sur les résultats précédents ?
5. Donnez la description VHDL comportemental en vous basant sur l'exemple vu en cours. Pouvez vous estimer combien de lignes (approximativement) de code VHDL au niveau RTL sont nécessaires pour décrire cette architecture ? Que pouvez vous conclure après cette estimation.

8 Etude d'un circuit pour la mise en œuvre d'une FFT

On étudiera d'abord le calcul du papillon de la FFT, tâche élémentaire de l'algorithme, puis on s'intéressera à son implémentation globale pour une taille de vecteur égale à 8 points. Le papillon de la FFT radix 2 sur N points est donné figure 3. Il peut se calculer de plusieurs façons dont deux versions sont données ici et nommées respectivement **version 1** et **version 2**. Les W sont directement issus de la mémoire contenant une table des cosinus et sinus. On les considérera donc comme des constantes.

1. Représenter le graphe flot de données des calculs des versions 1 et 2 en indiquant les multiplications, additions et soustractions sur des nombres réelles à mettre en œuvre. On exprimera les sorties $X'r$, $X'i$, $Y'r$ et $Y'i$, les parties réelles et imaginaires des sorties X' et Y' en fonction des entrées X et Y.
2. Le temps d'un multiplieur est de 60ns, le temps d'un additionneur/soustracteur est de 30ns. On considérera qu'une multiplication par deux se fait directement par un décalage

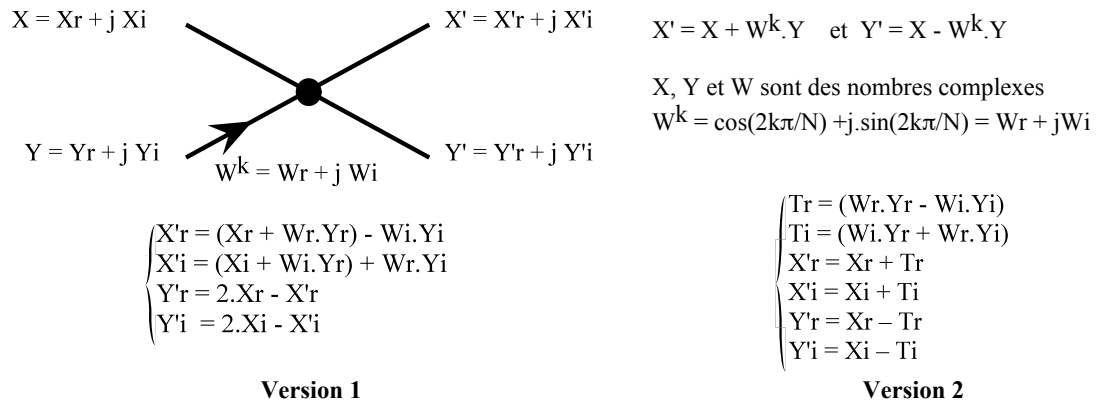


FIGURE 3 – Papillon de la FFT

du registre d'entrée d'un opérateur. Quelle est le temps du chemin critique Tcc dans les deux cas ?

3. On s'intéresse à l'implémentation VLSI du calcul de ce papillon. La cadence d'exécution d'un papillon est fixée à 360ns. On limitera le nombre de bus entre les mémoires et l'Unité de Traitement (UT) à deux, tout en autorisant le préchargement d'une donnée en registre. Combien d'opérateurs sont nécessaires ? Dans la suite du problème vous n'utiliserez qu'une seule des versions du calcul des papillons en motivant votre choix [1]. Effectuer un ordonnancement du graphe pour respecter les contraintes.
4. Dessiner l'architecture de l'unité de traitement du circuit ainsi conçu. On ne cherchera pas à optimiser le nombre de registres, en préférant dédier un registre à chaque donnée d'entrée (Yr, Yi, Wr, Wi, Xr, Xi).
5. On cherche maintenant à implanter la FFT sur 8 points selon le graphe classique donné figure 4.

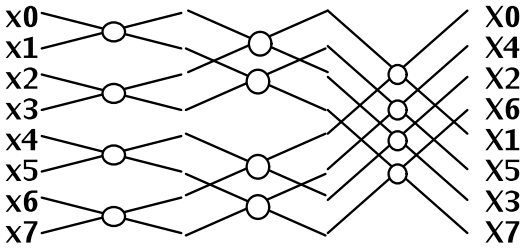


FIGURE 4 – Papillon de la FFT

Les vecteurs d'entrée et de sortie x et X sont des nombres complexes. Peut on exploiter le calcul d'une suite de papillons afin de réduire le temps d'exécution de celui ci ? Quel est le temps de calcul d'une FFT sur $N = 8$ points ?

9 Filtre en treillis

Les réponses aux questions suivantes sont à donner pour les deux versions du filtre en treillis données figure 5. Les multiplications Mi ont un temps d'exécution de 2 cycles tandis

1. Remarque : le choix de la version n'influence pas la notation.

que les additions Ai ont un temps d'exécution de 1 cycle. Exprimez les chemins demandés en utilisant le formalisme $\ldots \rightarrow Mi \rightarrow Ai \rightarrow \ldots$

1. Donner les chemins entre les points mémoire ou l'entrée et la sortie. En déduire le temps du chemin critique Tcc et de délai Td.
2. Donner les différentes boucles et en déduire le temps de la boucle critique Tbc.
3. Quelle est la signification de ces deux mesures ?

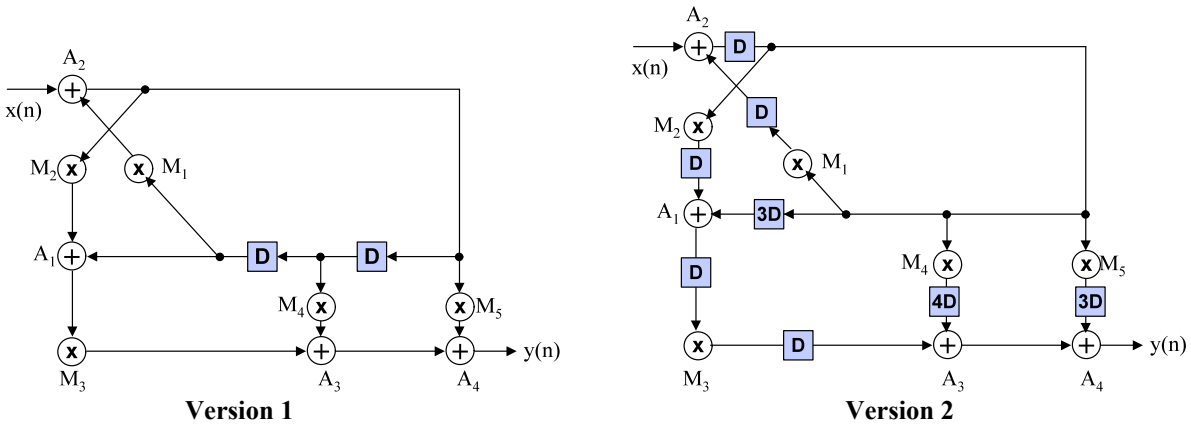


FIGURE 5 – Deux versions d'un filtre en treillis

10 Etude d'un circuit pour la mise en œuvre d'un filtre numérique

Soit le filtre à réponse impulsionnelle finie sur 8 points dont les coefficients $a(i)$ sont symétriques, i.e. $a(i) = a(7-i)$. Il est donc possible de factoriser la version 1 ci dessous

$$S = \sum_{i=0}^{7} a(i).e(i)$$

en une version 2 ne comportant plus que 4 multiplications telle que

$$S = \sum_{i=0}^{3} a(i)[e(i) + e(7-i)].$$

On considérera que $Tadd = 20ns$, $Tmult = 30ns$, que les données $e(i)$ et $a(i)$ viennent de mémoires, et que S retourne en mémoire en fin de calcul. Dans un premier temps le nombre de bus entre mémoire et unité de traitement ne sera pas limité.

1. Dessinez le graphe flot de signal ordonnancé au plus tôt de la version 2 de cet algorithme. Quel est le temps du chemin critique ?
2. On désire étudier une architecture pour implémenter les calculs de la version 2 du filtre, dans les cas suivants :
   (a) la cadence des calculs est de $240ns$
   (b) la cadence des calculs est de $120ns$

- Donnez le nombre d'opérateurs (additionneur, multiplieur) nécessaires dans les deux cas.
- Ordonnancez le graphe flot de données sur ce nombre d'opérateurs dans les deux cas. La cadence est-elle respectée ? Donnez la latence des calculs.
- Estimez le nombre minimum de bus et de registres dans les deux cas.

3. Dessinez l'architecture de l'unité de traitement (UT) obtenue après allocation des opérateurs et assignation des registres, dans le cas où la cadence vaut $120ns$. Expliquez clairement l'utilisation des ressources allouées par une table d'utilisation (opérateurs, registres, bus). En particulier, soyez vigilant à l'allocation des opérations aux opérateurs lorsque leur nombre est supérieur à 1.
4. Expliquez comment concevoir la structure de l'unité mémoire (UM). Vous donnerez le nombre et le type de bancs mémoire, le placement des données dans ceux ci, ainsi que la structure d'interconnexion par bus entre UT et UM.
5. On ajoute à la bibliothèque d'opérateurs un multiplieur pipeline de latence égale à $40ns$, de cadence égale à $20ns$, et comportant donc deux tranches de pipeline. Représentez l'ordonnancement du filtre en version 1 sur un additionneur et un multiplieur pipeline. Quel temps de calcul peut on atteindre avec cette version ? L'utilisation d'un additionneur pipeline pourrait elle apporter un gain en performance ?

11 Étude d'une architecture VLSI pour un filtre RII du 1^{er} ordre

Soit le filtre numérique à réponse impulsionnelle infinie du premier ordre défini par :

$$y(n) = a \cdot y(n-1) + x(n)$$

1. Après avoir tracé son Graphe Flot de Signal (*GFS*), donnez la fréquence maximale de fonctionnement du filtre (i.e. la fréquence d'échantillonnage du signal d'entrée F_e). On considérera qu'une multiplication et une addition sont effectuées en un cycle de 5 ns (technologie $0.18\mu m$, opérateurs sur 16 bits).
2. Exprimez $y(n-1)$ en fonction de $y(n-2)$, puis reportez cette expression dans $y(n)$ afin d'obtenir $y(n)$ fonction de $y(n-2)$ et des entrées. En déduire le nouveau *GFS* ainsi que sa fréquence maximale de fonctionnement. Concluez sur l'intérêt de cette transformation connue sous le nom de *LookAhead*.
3. Dessinez un ordonnancement de ce GFS permettant de suivre une cadence d'arrivée des échantillons de 5 ns (équivalente à une fréquence d'échantillonnage de 200 MHz). En déduire les caractéristiques d'une architecture (nombre d'opérateurs et nombre de registres, ...) permettant d'implanter cet ordonnancement. On considérera que les coefficients et les signaux $y(n-i)$ et $x(n-i), \forall i$ se trouvent dans des registres locaux à l'unité de traitement.

   Expliquez clairement vos résultats.
4. Généralisez cette transformation jusqu'à l'ordre N. Donnez l'expression de $y(n)$ en fonction $y(n-N)$. En déduire la cadence fonctionnement que l'on peut atteindre.

   Comment peut on trouver une architecture permettant de fonctionner à cette cadence ? Commenter sur un exemple où $N = 4$.

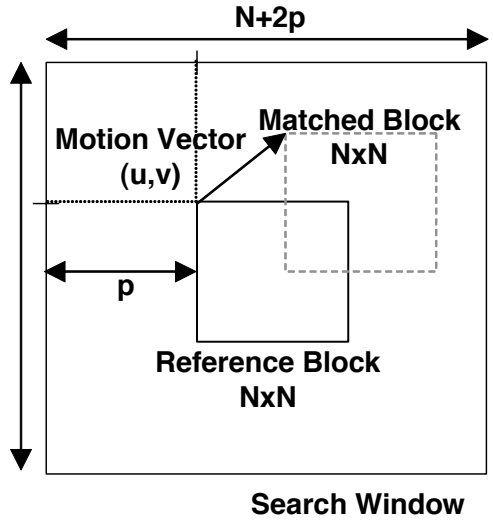
   Quelles sont donc les limitations de cette transformation ?

12 Étude d'un système sur silicium pour le codage vidéo

Dans les applications de codage vidéo, il est nécessaire, afin de diminuer le flux vidéo, de pouvoir estimer le mouvement entre des blocs de pixels de l'image courante et de l'image précédente. Cet algorithme, appelé Estimation de Mouvement (EM) représente la partie la plus complexe du codeur vidéo. On le retrouve dans tous les codeurs de type MPEGx ou H26x. Cela constitue donc un bloc intéressant à étudier dans le cadre de sa mise en œuvre sur un SOC.

Chaque image est partitionnée en **blocs de référence (BR)** de taille $N \times N$ pixels, chaque pixel étant codé sur 8 bits. Chaque BR est comparé avec des blocs de taille équivalente dans l'image précédente au sein d'une **fenêtre de recherche (FR)** de taille $(N + 2p) \times (N + 2p)$ pixels. Le bloc de la FR aboutissant à une distance minimale avec le BR est choisi comme vecteur de mouvement. La figure 6.a montre ce principe, tandis que l'algorithme 6.b est le calcul de l'estimation de mouvement du bloc BR sur l'ensemble de FR. Le calcul de la distance, représenté par les deux boucles centrales de l'algorithme 6.b est donné par :

$$distance(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |BR(i,j) - FR(i+u, j+v)|, \ for \ -p \leq u, v \leq p$$



a) Fenêtre de recherche dans l'EM

```
1.sadmin = MAXINT; mvx=0; mvy=0;
2.for (u=-p; u<=p; u++)
3.  for (v=-p; v<=p; v++) {
4.    sad = 0;
5.    for (i=0; i<N; i++) {
6.      for (j=0; j<N; j++) {
7.        sad = sad + ABS[BR(i,j)-FR(i+u,j+v)]
8.        /* if (sad>=sadmin)   break; */
9.      }
10.   }
11.   if (sad<sadmin) {
12.      sadmin = sad; mvx = u; mvy = v;
13.   }
14. }
```

b) Algorithme de l'EM

FIGURE 6 – Estimation de mouvement sur un bloc de l'image par rapport à un bloc de référence (EM)

1. Donner la complexité en nombre d'opérations [2] de l'estimation de mouvement d'un bloc BR sur FR (algorithme 6.b) en fonction de N et p.
2. Sur une image de taille $N_h \times N_w$ et sur un débit vidéo de D images par secondes, quelle est la complexité totale (en MOPS) de l'EM au sein du codage vidéo. On considérera

2. La valeur absolue $ABS()$ est une opération qui peut être implantée sur une ALU. Ne pas tenir des calculs d'indices.

que l'EM doit être calculée sur l'ensemble des blocs BR adjacents formant l'image. On prendra par la suite : $N = 8$, $p = 12$, $N_h = 288$, $N_w = 352$ (image de format CIF), $D = 20$ images/s.

Donner l'application numérique.

Quels peuvent être les avantages et les inconvénients de se servir de la ligne 8 (non considérée dans le début et dans la suite de ce problème).

3. **Implantation sur processeur VLIW**

   On s'intéresse tout d'abord à son implantation sur un DSP VLIW haute performance. Le processeur considéré est ici le ST200 de la famille Lx (voir transparent cours SoC) à 1 cluster et une fréquence d'horloge de 400 MHz. Ce processeur peut exécuter jusqu'à 4 opérations par cycle et possède 6 unités fonctionnelles (4 ALUs et 2 multiplieurs). Quelle est la puissance crête en MOPS que peut délivrer ce processeur ? Quelles sont les autres informations pertinentes sur cette architecture pour notre application d'estimation de mouvement ?

4. Donner une estimation du temps de calcul sur ce processeur.

5. **Implantation sur co-processeur dédié**

   On souhaite maintenant associer un processeur traitant les boucles des lignes 2 et 3 à un co-processeur accélérant les boucles centrales des lignes 5, 6 et 7. Quelle est la contrainte de cadence que doit respecter ce co-processeur ?

   En considérant que les fonctions +, - et ABS peuvent travailler à 400 MHz sur cet ASIC (i.e. le temps de traversée d'un opérateur permettant de réaliser les 3 fonctions de +, - et ABS est de 2.5 ns), quel est le nombre d'opérateurs arithmétiques que devra comporter ce co-processeur ? Expliquer clairement votre résultat.

6. On considérera le graphe flot de données de la boucle centrale de l'EM (lignes 6 et 7 de l'algorithme 6.b) une fois celle-ci totalement déroulée. Sur ce graphe flot de données, quel type de transformation doit être appliqué pour permettre un ordonnancement sur un nombre supérieur d'opérateurs ? Représenter cet ordonnancement sur le DFG transformé (sans forcément atteindre le nombre d'opérateurs trouvés dans la question précédente ; vous pouvez fixer par exemple un nombre d'opérateurs égal à 8).

   En optimisant la gestion de la boucle 5, donner le temps d'exécution de l'algorithme d'EM sur ce co-processeur.

7. Quel est le volume de données qui doit être communiqué entre la mémoire contenant les images courante et précédente et le co-processeur. On considérera que le co-processeur contient dans une mémoire locale le bloc BR et la fenêtre FR.

8. **Définition d'un SOC pour le codage vidéo**

   Dessiner et expliquer le synoptique d'un SOC –mixant processeur, co-processeur, mémoire, interconnexions et périphériques– qui vous parait bien adapté à l'implantation de l'application d'EM. Vous pourrez par exemple détailler rapidement pour les différents blocs les différentes possibilités de solutions architecturales.

   Qu'est ce qui vous parait le plus difficile dans la conception de ce système sur puce ?

9. Détaillez, en vous basant sur le cours, les différentes possibilités de spécifications (langage) et la méthodologie de validation (simulation) de l'architecture du SOC.

13 Étude des performances d'une structure en treillis

Soit la structure de filtre numérique en treillis représentée sur la figure 7. On considérera un additionneur de temps de propagation égal à un cycle et un multiplieur **pipeline à deux étages**, ayant une cadence de un cycle et ayant une latence de deux cycles.

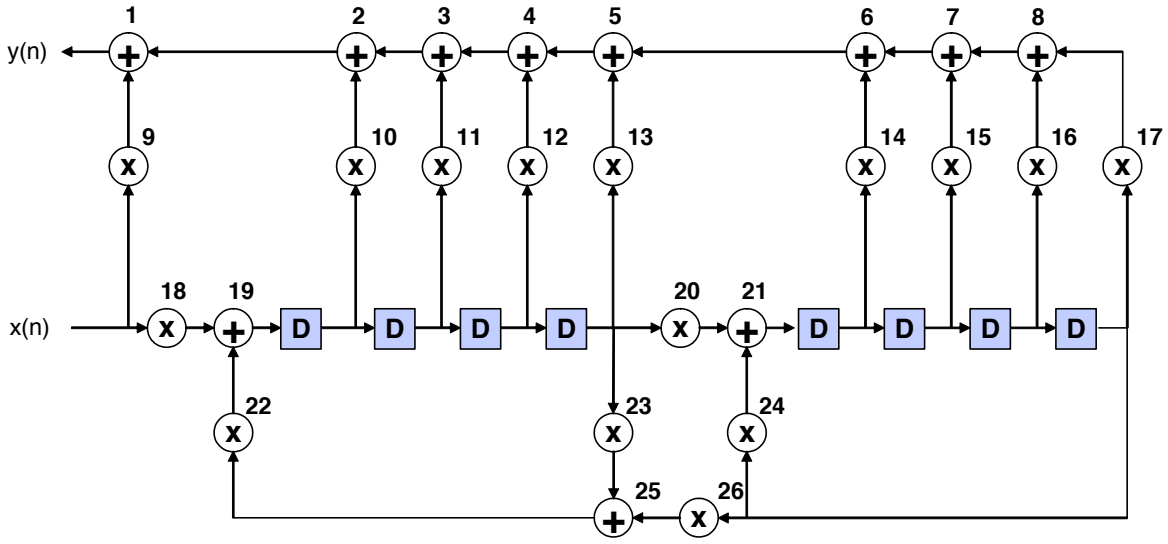


FIGURE 7 – SFG de la structure en treillis

1. Énumérer, en utilisant les numéros d'opérations, quelles sont les différentes boucles de ce graphe flot de signal (SFG). Quel est la temps de la boucle la plus critique (T_{bc}) ?
2. Quel est la temps du chemin critique (T_{cc}) ?
3. A quoi correspondent ces métriques ?

4. Déterminer le nombre minimum d'additionneurs et de multiplieurs permettant d'atteindre une cadence de traitement de 4 cycles (i.e. un nouvel échantillon $x(n)$ doit être filtré tous le 4 cycles).
5. Représenter un ordonnancement permettant d'atteindre cette cadence tout en minimisant la latence (temps de retard T_d) de traitement.
6. Proposer des transformations algorithmiques sur ce SFG permettant de réduire la latence du filtre à 4 cycles (ou moins). Expliquer clairement votre méthode et vos résultats, et dessiner éventuellement le SFG transformé.

14 Étude d'une architecture VLSI pour un filtre de Volterra

Soit le système discret défini par :

$$y(n) = \{[a_1.x(n-1) + a_2.x(n-2)] + c_1.z(n-1)\} + [b_1.w(n-1) + b_2.w(n-2)] \quad (2)$$

avec

$$z(n) = x(n).x(n-1) \tag{3}$$

et

$$w(n) = x(n).x(n) \tag{4}$$

On considérera qu'une multiplication et une addition sont effectuées en un cycle de 5 ns (technologie $0.18\mu m$, multiplication sur 16 bits et addition sur 32 bits).

1. Tracer son Graphe Flot de Signal (*SFG*) sans modifier la spécification du système d'équations ci dessus. Numéroter les différentes opérations.
2. Tracer ensuite son graphe flot de données (*DFG*) selon l'ordonnancement au plus tôt ASAP. Donner les mobilités des différentes opérations du graphe.
3. Donner l'**ordonnancement par listes** du *DFG* en considérant une allocation sur un multiplieur et un additionneur. Quel est le temps d'exécution de l'algorithme ?
4. Caractéristiques d'une architecture permettant d'implanter l'ordonnancement déterminé dans la question précédente.

   On considérera que les coefficients et les signaux $w(n-i)$, $z(n-i)$ et $x(n-i), \forall i$ se trouvent dans des registres locaux à l'unité de traitement.

   Expliquer comment déduire le nombre de registres à partir de l'ordonnancement et indiquer pour chaque registre les données qu'il contient.

   Dessiner la structure de l'architecture de l'unité de traitement. On considérera un bus d'entrée B véhiculant l'arrivée du nouvel échantillon $x(n)$.
5. On souhaite maintenant que l'architecture travaille à une cadence d'arrivée des échantillons de 5 ns (i.e ; une fréquence d'échantillonnage de 200 MHz). Quel ordonnancement peut on utiliser pour atteindre cette cadence ?

   Donner une estimation (expliquer vos résultats) :
   – du nombre d'opérateurs à utiliser ;
   – du nombre de registres nécessaires.

15 Etude de l'accélération offerte par le couplage d'un co-processeur à un processeur dans un SOC

Ce problème concerne l'étude de l'accélération offerte par le couplage d'un co-processeur à un processeur dans un SOC. On considérera une carte graphique pour accélérer le codage de séquences vidéo (e.g. format DivX) couplant un processeur VLIW et un co-processeur ASIC spécialisé sur un traitement particulier de cette séquence de codage (e.g. estimation de mouvement). Les caractéristiques du système sont données ci-dessous.

– Le processeur a une puissance crête de 800 MOPS pour une fréquence d'horloge de 200 MHz.
– Le bus de couplage fonctionne à 200 MHz sur 32 bits.
– Les images à coder par l'ASIC sont de taille maximum 512×340 ($N \times M$), chaque pixel étant codé sur 24 bits (3 couleurs sur 8 bits). Le résultat du calcul de l'ASIC représente un volume de $N \times M/2$ données de 32 bits.
– L'application à accélérer nécessite un nombre d'opérations de calcul de $N \times M \times 100$. Elle doit être exécutée dans le cas idéal en un temps inférieur à 5 ms.

– Le co-processeur ASIC doit : (1) recevoir l'image complète transférée pixel par pixel sur le bus 32 bits (temps T_{com1}), (2) effectuer son calcul (temps T_{exe}), (3) retourner le résultat du calcul sur le bus 32 bits (temps T_{com2}).

1. Calculer les temps de communication des données entre le processeur et l'ASIC T_{com1} et T_{com2}.
2. Exprimer l'accélération $Acc(T_{exe})$ offerte par l'ASIC, i.e. le rapport entre la puissance de calcul offerte par le processeur couplé à l'ASIC et celle du processeur seul. Tracer cette accélération en fonction de T_{exe}. Quelle est l'accélération maximale que l'on peut atteindre ?
3. A partir de quel T_{exe} la contrainte de temps réel est elle atteinte ?
4. Quelles solutions proposez vous pour augmenter les performances de ce système sur puce ?

16 Etude d'un filtre RII sous forme de cellules du 4ème ordre en cascade

L'étude traitée dans ce problème concerne la cellule élémentaire d'un filtre à réponse impulsionnelle infinie (RII) du quatrième ordre. On considérera deux structures de cette cellule, la structure directe

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) + \sum_{i=1}^{N} a_i y(n-i)$$

et la structure canonique

$$y(n) = \sum_{i=0}^{N} b_i w(n-i)\ ; \quad w(n) = x(n) \sum_{i=1}^{N} a_i w(n-i).$$

Les nœuds multiplications, notés « xbi », correspondent à une multiplication par le coefficient b_i (idem pour a_i) Les équations aux différences correspondant à ces deux structures sont données ci-dessous. On prendra $N = 4$.

On considérera un additionneur de temps de propagation égal à un cycle et un multiplieur pipeline à deux étages, ayant une cadence de un cycle et ayant une latence de deux cycles. Pour les applications numériques, on considérera un cycle égal à 5ns.

Les questions 1 à 4 sont à traiter pour les deux structures.

1. Énumérer, en utilisant les numéros d'opérations ou en les dessinant directement sur le graphe, les différentes boucles de ce graphe flot de signal (SFG) et leur temps respectif.
2. Quel est le temps de la boucle la plus critique Tbc ?
3. Quel est le temps du chemin critique Tcc ? Quel est le temps de retard Td ?
4. Quelle est la cadence d'échantillonnage maximale que l'on peut atteindre pour ce filtre du quatrième ordre ? A.N.

   Transformation du SFG de la structure canonique

5. Proposer des transformations algorithmiques sur le SFG de la structure canonique permettant de réduire le chemin critique (latence) du filtre à 4 cycles. Expliquer clairement votre méthode et vos résultats, et dessiner éventuellement le SFG transformé.
   Réalisation d'une architecture VLSI pour la structure canonique
   On considérera pour les questions 5 et 6 la structure canonique dans laquelle on posera $b_0 = 1$, ce qui supprime l'opération de multiplication correspondante.
6. Déterminer le nombre minimum d'additionneurs et de multiplieurs permettant d'atteindre une cadence de traitement de 4 cycles (i.e. un nouvel échantillon $x(n)$ doit être filtré tous le 4 cycles).
7. Indiquer, pour chaque nœud du graphe, la mobilité de l'opération exprimée en nombre de cycles.
8. Représenter un ordonnancement permettant d'atteindre cette cadence de 4 cycles tout en minimisant la latence et le temps de retard de traitement.
9. Quel est le nombre minimum de registres que contiendra l'architecture VLSI réalisant l'ordonnancement précédent ? Expliquez clairement votre réponse.
   Filtre d'ordre M sous forme cascade
   On considère un filtre d'ordre M, avec M multiple de 4, réalisé sous forme de mise en cascade d'un nombre C de cellules du quatrième ordre. Le filtre $H(z)$ d'ordre M est défini comme
   $$H(z) = \prod_{i=1}^{C} H_i(z)$$
   avec $C = M/4$.
10. La cadence d'échantillonnage maximale que l'on peut atteindre pour ce filtre d'ordre M est elle modifiée par rapport à celui d'une cellule du quatrième ordre ?
11. En considérant l'ordonnancement de la question 8, quel serait la cadence globale de traitement du filtre d'ordre M ?

17 Filtrage récursif en deux dimensions

Le filtrage en deux dimensions (2D) sur des données est une opération locale. Chaque échantillon de sortie est une combinaison linéaire des sorties et entrées précédentes et de la valeur d'entrée, dans un voisinage local 2D de l'entrée. On peut prendre comme exemple de donnée 2D une image selon les deux directions verticales (v) et horizontales (h).

La spécification de l'algorithme de filtrage est donnée pour information par l'équation suivante :

$$g(n_1, n_2) = \sum_{h=0}^{L} \sum_{v=0}^{M} b(h, v).f(n_1 - h, n_2 - v) + \underbrace{\sum_{h=0}^{L} \sum_{v=0}^{M}}_{h+v>0} a(h, v).g(n_1 - h, n_2 - v)$$

Le graphe flot de données correspondant à cette équation est donné figure 10 dans le cas où $M = L = 2$. L'entrée du filtre est $f(n_1, n_2)$, sa sortie est $g(n_1, n_2)$. Il est constitué de $M + 1$ cellules élémentaires d'ordre L disposées verticalement. La cellule élémentaire i, entourée en pointillés sur la figure 10, est un filtre récursif d'ordre L. La première partie de ce

problème est consacrée à l'étude de la cellule élémentaire, tandis que la second partie traite de l'algorithme complet. Pour la suite du problème, on considérera un additionneur et un multiplieur de temps de propagation égal à un cycle. Pour les applications numériques, on considérera le temps d'un cycle égal à 5ns.

1. Le graphe de la figure 8 représente la cellule élémentaire dans le cas où $L = 2$.
   - Énumérer, en les dessinant directement sur le graphe fourni, les différentes boucles du graphe flot de signal.
   - Donner le temps respectif de chaque boucle. Quel est le temps de la boucle la plus critique T_{bc} ?
   - Quel est le temps du chemin critique T_{cc} ?
2. Donner sur le graphe de la figure 8 un exemple de transformation par *retiming* par coupe du graphe et le reporter sur la figure 9.
3. Donner un ordonnancement ASAP du graphe flot de données (DFG) correspondant à la figure 8.
4. Indiquer, pour chaque nœud du graphe, la mobilité de l'opération exprimée en nombre de cycles.
5. Déterminer le nombre minimum d'additionneurs et de multiplieurs permettant d'atteindre une cadence de traitement de 4 cycles (i.e. un nouvel échantillon $f(n)$ doit être filtré tous le 4 cycles).
6. Effectuer un ordonnancement **par listes avec priorité sur la mobilité** sur le nombre d'opérateurs trouvés précédemment. Quelles sont les cadence et latence obtenues ?
7. Quel est le nombre minimum de registres que contiendra l'architecture VLSI réalisant l'ordonnancement précédent ? Expliquez clairement votre réponse.

8. Dans le graphe de la figure 10, les cellules élémentaires sont placées verticalement et $g(n_1, n_2)$ est rebouclé vers l'ensemble de ces cellules, tandis que $f(n_1, n_2)$ est diffusé.
   Indiquer les boucles inter-cellules ajoutées dans ce cas. Quel est le temps de la boucle la plus critique T_{bc} ? Quel est le temps du chemin critique T_{cc} ?

18 Etude d'un filtre RII

L'étude traitée dans ce problème concerne la cellule élémentaire d'un filtre à réponse impulsionnelle infinie (RII) donnée à la figure 11. On considérera un additionneur de temps de propagation égal à 8 ns et un multiplieur de temps de propagation égal à 20 ns.

1. Énumérez, en utilisant les numéros d'opérations ou en les dessinant directement sur le graphe, les différentes boucles de ce graphe flot de signal (SFG) et leur temps respectif. Quel est le temps de la boucle la plus critique Tbc ?
2. Quel est le temps du chemin critique Tcc ? Indiquez le chemin critique.
3. Proposez des transformations algorithmiques de type *retiming* permettant de réduire le chemin critique du filtre de façon à obtenir $Tbc = Tcc$. Expliquer clairement votre méthode et vos résultats, et dessinez le SFG transformé sur la figure 12 (vous pouvez utiliser la figure 13 également).

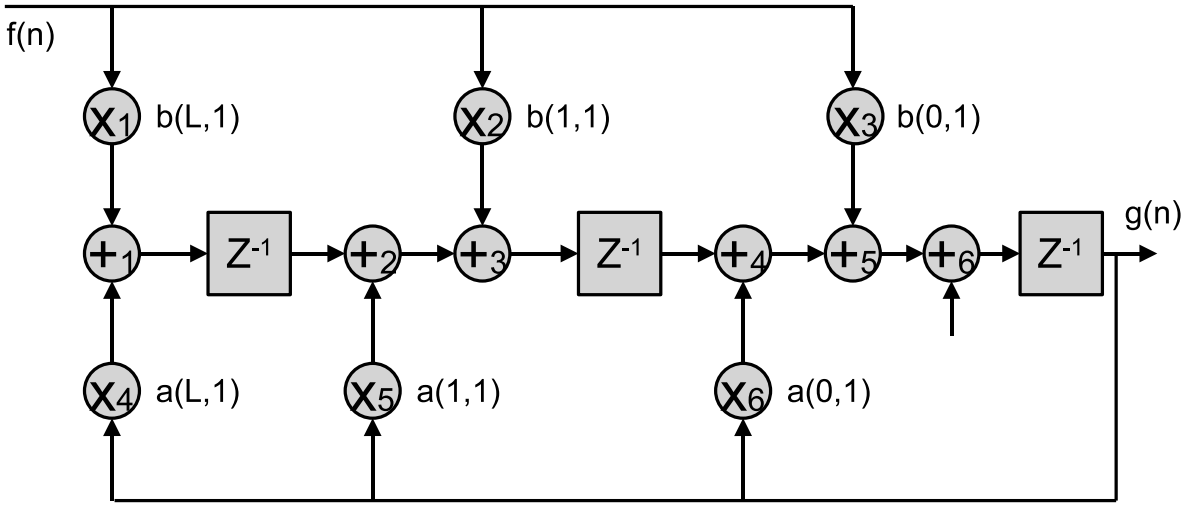


FIGURE 8 – Cellule élémentaire

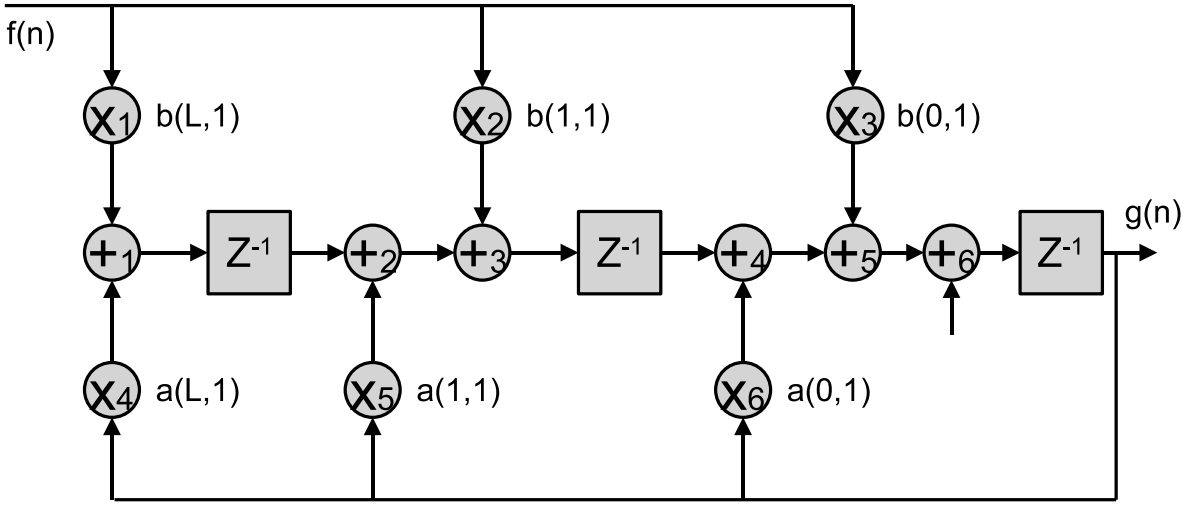


FIGURE 9 – Exemple de retiming

4. Déterminez le nombre minimum d'additionneurs et de multiplieurs permettant d'atteindre une cadence de traitement de 20 ns (i.e. un nouvel échantillon $x(n)$ doit être filtré toutes les 20 ns).
5. Représentez un ordonnancement permettant d'atteindre cette cadence de 20 ns. Vous représenterez trois itérations successives du filtre afin de vérifier les contraintes liées aux boucles critiques. Indiquez les clairement sur votre ordonnancement.

19 Transformations algorithmiques sur un SFG

Ce problème concerne l'optimisation du graphe de calcul représenté à la figure 14 par application de transformations algorithmiques. On considérera que les opérations A, B, C, D et E s'effectuent en deux cycles, tandis que les opérations F et G s'effectuent en un cycle. L'architecture dispose d'unités de calcul (UAL) permettant de réaliser les différentes opérations A à G. Exprimez les chemins demandés en utilisant le formalisme $\ldots \rightarrow B \rightarrow C \rightarrow \ldots$

1. Énumérez les différentes boucles de ce graphe flot de signal (SFG) et leur temps respectif. Quel est le temps de la boucle la plus critique Tbc ? Indiquez la boucle critique sur la figure 14.
2. Quel est le temps du chemin critique Tcc ? Indiquez le chemin critique sur les figures ?
3. Quel est le nombre d'UAL nécessaire pour effectuer ce calcul à une cadence de 4 cycles ?
4. Dessinez le graphe flot de données équivalant au graphe de la figure 14 ordonnancé au plus tôt (ASAP).
5. Proposez un ordonnancement de ce graphe sur le nombre d'UAL déterminé à la question 3 qui minimise la latence et respecte la cadence de 4 cycles. Dessinez plusieurs itérations du calcul pour clarifier votre démarche.
6. Proposez une transformation algorithmique de type *retiming* permettant de réduire le chemin critique du filtre à 4 cycles tout en réduisant le nombre de délai à 5 éléments. Expliquez clairement votre méthode et vos résultats. Dessinez le SFG transformé sur la figure 15.
7. Quel avantage possède le graphe transformé une fois ordonnancé dans les mêmes conditions qu'à la question 5 ?

20 Interpolation « luma » en codage vidéo HD

Lors de l'encodage d'une séquence vidéo par les codeurs les plus récents, les vecteurs de mouvement sont calculés très précisément sur des pixels interpolés au 1/2 pixel, voire au 1/4 de pixel. C'est l'objectif de l'interpolateur « luma ». On considère ici l'interpolation au demi-pixel, c'est-à-dire une partie de l'algorithme complet consistant à estimer pour chaque pixel de l'image trois nouveaux pixels placés entre chaque pixel. Par exemple, dans la figure 16, le pixel d'origine P entouré d'un cercle en pointillé va engendrer l'estimation du pixel c à sa droite, du pixel x en dessous (pour le pixel d'origine L sur la figure), et du pixel en diagonal bas-droite j. Chaque demi-pixel (a, x, j) est calculé par la formule indiquée sur la figure 16, formule issue directement du document de la norme H264. Le pseudo-code équivalent est donné ci-dessous. On ne tiendra compte ni des effets aux bords du bloc, ni de l'aspect adressage des pixels. L'interpolation s'effectue sur des macro-blocs 4x4, pour chaque macro-bloc de l'image.

1. Dans le contexte d'une séquence vidéo HD où chaque image à une taille de 1920x1080 pixels et la séquence devant être encodée à la cadence temps-réel de 60Hz (60 images/s), combien de macro-blocs 4x4 doivent être traités par seconde ? En déduire la puissance en MOPS de l'architecture nécessaire pour cette interpolation luma ?

   On considère par la suite un seul des trois calculs élémentaires :
   a=E-5*F+20*G+20*H-5*I+J (1)

2. Quelle est la contrainte de temps pour ce calcul du demi-pixel a ?
   Il est possible de réécrire l'équation (1) factorisée de la façon suivante :
   s0=E+J ; s1=F+I ; s2=G+H ; a=s0-5*s1+20*s2 (2)
   L'équation (2) peut à nouveau être réécrite de la façon suivante :
   s0=E+J ; s1=F+I ; s2=G+H ; a=[(s0-s1) + 4*(s2-s1)] + 16*s2 (3)
   L'avantage est ici la suppression des multiplications très coûteuses par des décalages par une puissance de deux qui ne coûte rien au niveau matériel.
3. Dessinez le graphe de l'équation 3. Les opérations de multiplication par 4 et par 16 seront représentées par << 2 et << 4. Celle-ci ne coûtant rien au niveau matériel (uniquement de l'aiguillage de fils), elles peuvent être omises par la suite. Quelle est la nouvelle puissance en MOPS de l'architecture ?
4. L'architecture peut contenir trois types d'opérateurs : additionneur (ADD), soustracteur (SUB) ou additionneur/soustracteur (A/S). En considérant que ces opérateurs ont un temps de traversée de 2ns (500MHz), combien de ces opérateurs sont nécessaires pour une cadence d'exécution de l'équation 3 en 6 ns (3 cycles) ? Quelle sélection, i.e. combien d'opérateurs de chaque type, vous paraît optimale ?
5. Proposez ensuite un ordonnancement de ce graphe respectant cette cadence de 3 cycles.

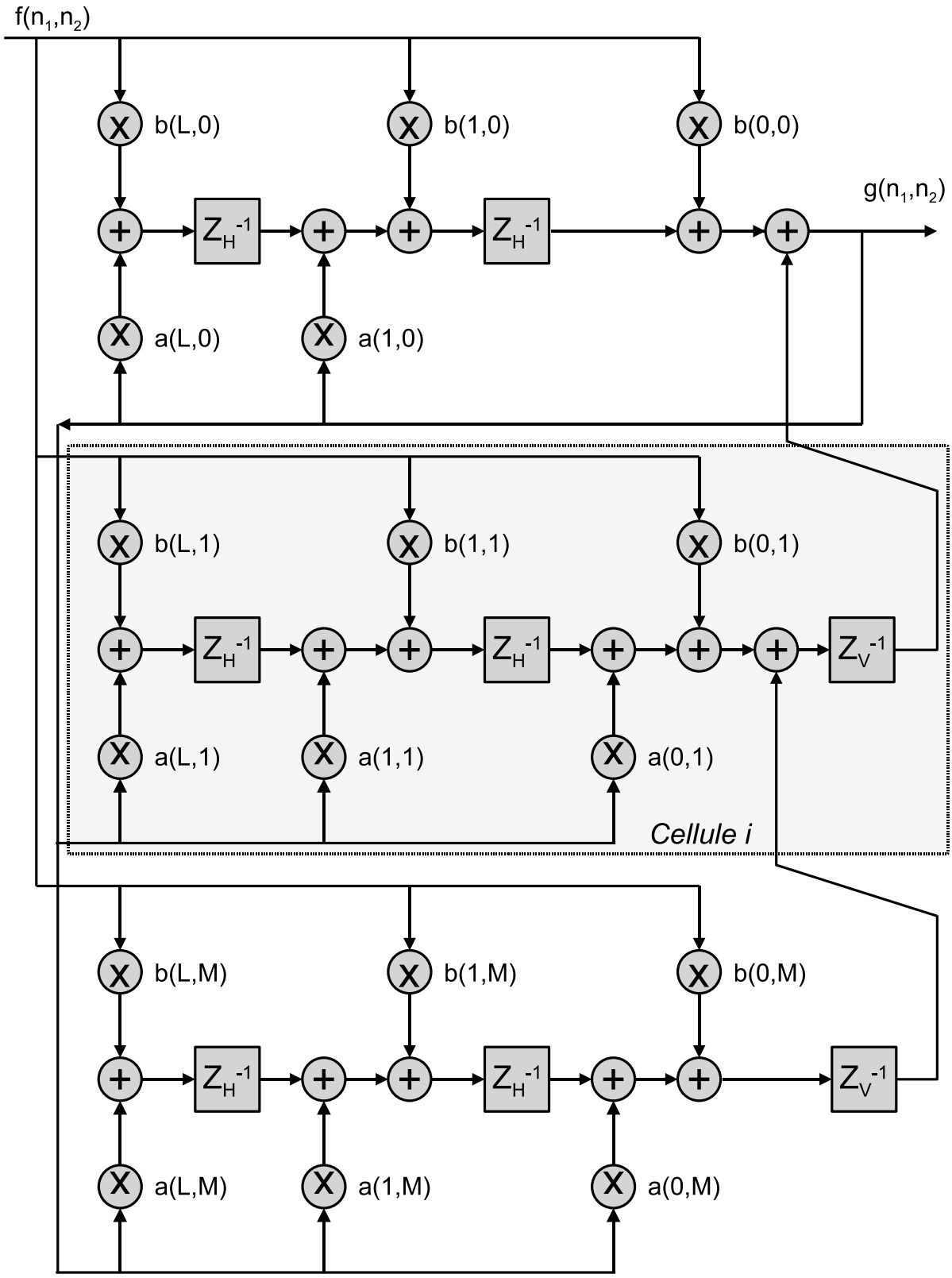


FIGURE 10 – Graphe complet du filtre 2D

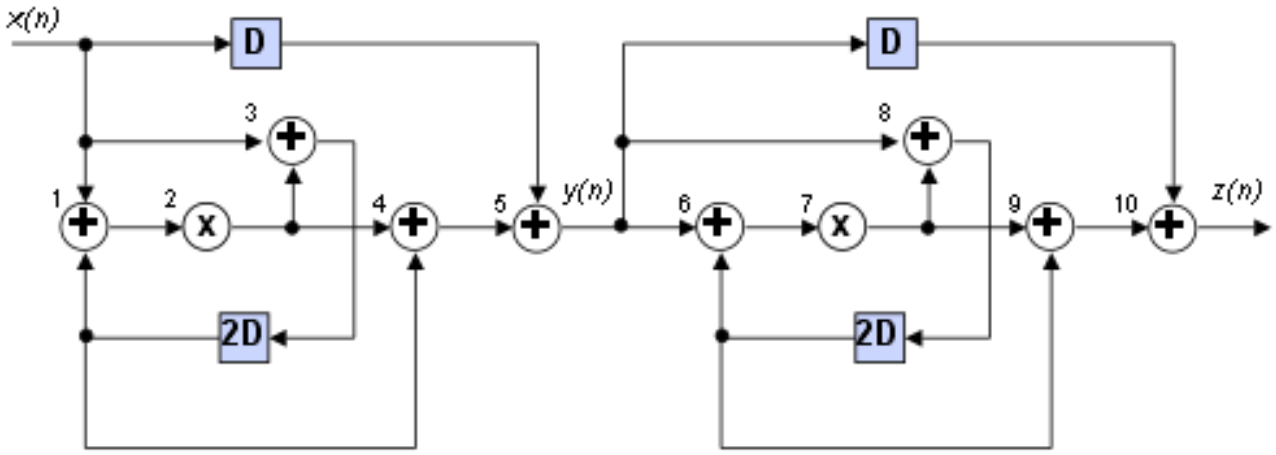


FIGURE 11 – filtre RII

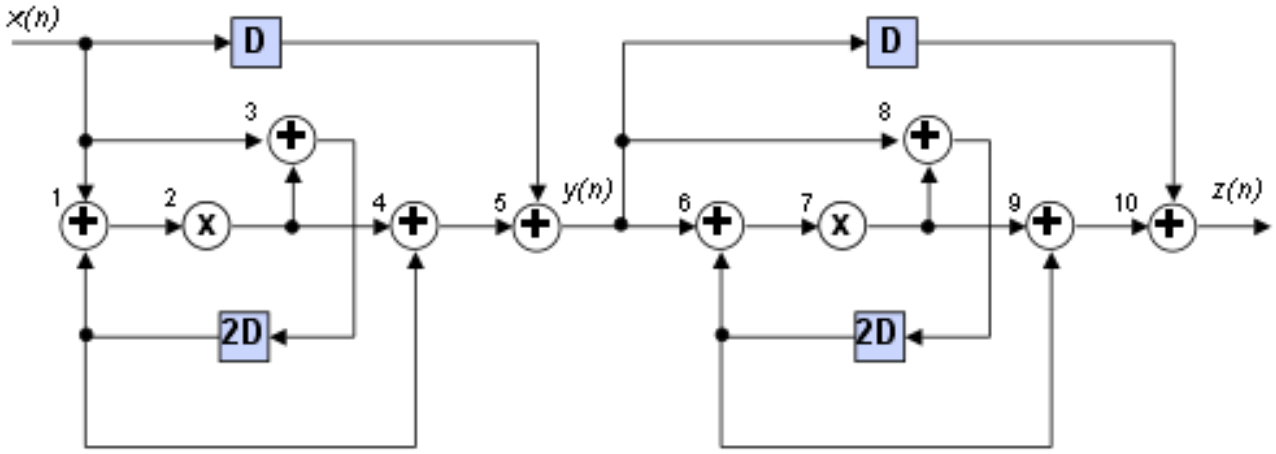


FIGURE 12 – filtre RII à transformer (1)

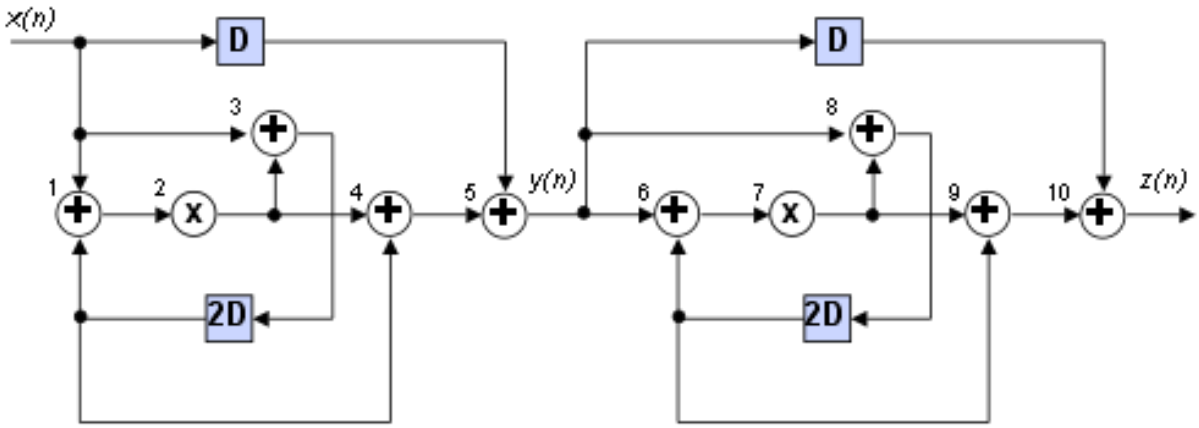


FIGURE 13 – filtre RII à transformer (2)

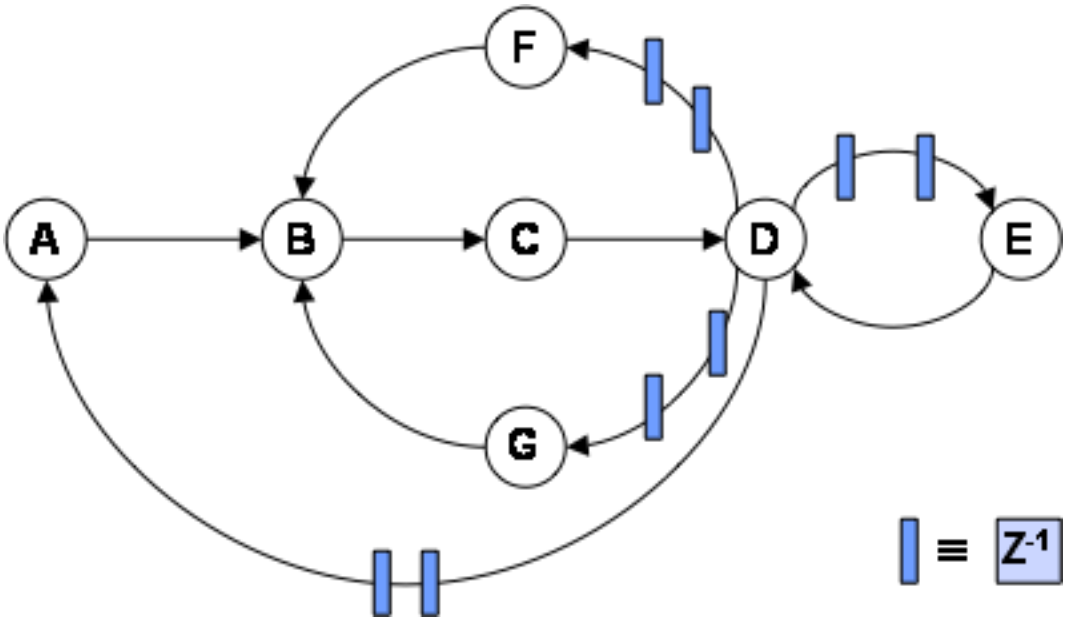


FIGURE 14 – version initiale du graphe de calcul

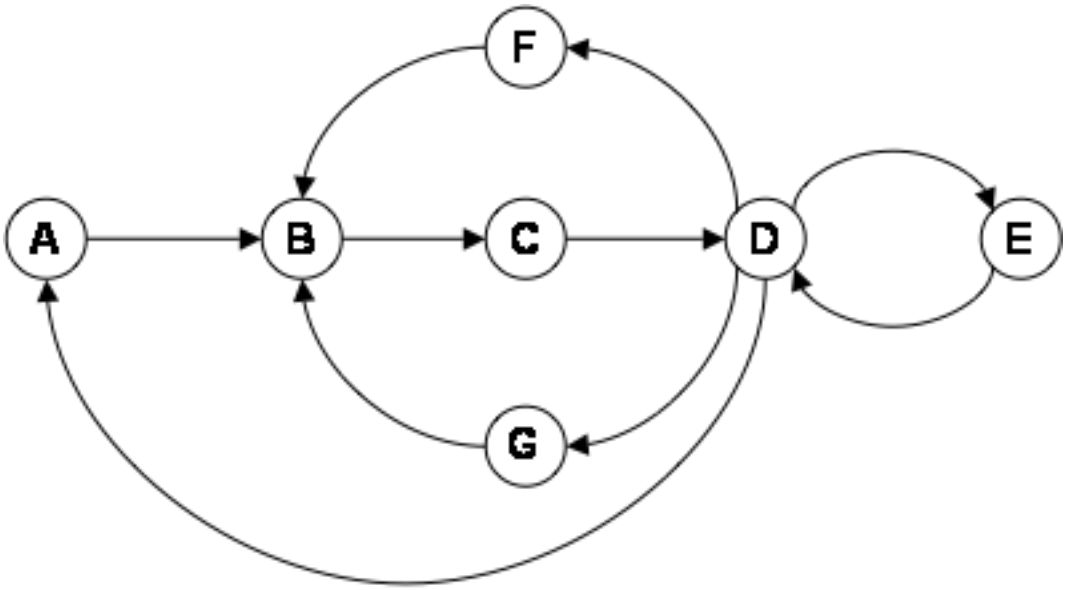


FIGURE 15 – version du graphe sans les délais pour illustration des transformations

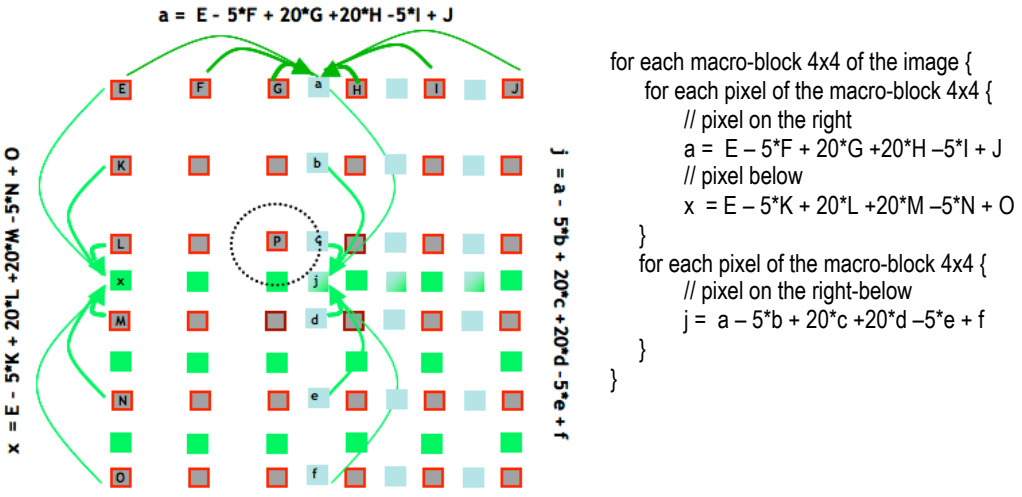


FIGURE 16 – Principe de l'interpolation luma