

ENSSAT EII3/DEA STIR

Corrections TD Architectures Spécialisées

Olivier Sentieys

23 novembre 2009

1. Filtrage récursif

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

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

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 données de ce filtre lorsque $N = P = 4$.

$$y(n) = [a(0).x(n) + a(1).x(n-1) + a(2).x(n-2) + a(3).x(n-3) + a(4).x(n-4)] \\ + [b(1).y(n-1) + b(2).y(n-2) + b(3).y(n-3) + b(4).y(n-4)]$$

Les évaluations dans les crochets se faisant de gauche à droite, le graphe flot de signal est celui de la figure 1.

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 T_{add} et T_{mult} , temps de fonctionnement des additionneurs et des multiplieurs.

Les boucles sont au nombre de 4. Chaque boucle part de $y(n)$, passe par 1, 2, 3 ou 4 éléments de retard z^{-1} , avant de retourner au point $y(n)$. Chaque boucle i possède un temps de calcul T_i valant :

$$T1 = T_{mult} + 4.T_{add}$$

$$T2 = T1$$

$$T3 = T_{mult} + 3.T_{add}$$

$$T4 = T_{mult} + 2.T_{add}$$

3. Trouvez la fréquence maximale d'échantillonnage du filtre.

Le nombre de période d'échantillonnage séparant $y(n)$ de $y(n-i)$ est i . En effet, $y(n)$ représente la valeur du signal $y(t)$ échantillonné à l'instant $t = nT$, où T est la période d'échantillonnage. Par conséquent le temps de calcul T_i doit être inférieur à $i.T$.

On a donc :

$$T_i \leq i.T, \forall i = 1 \dots P$$

$$T \geq \frac{T_i}{i}, \forall i = 1 \dots P \quad \Rightarrow \quad T \geq \max_i \left[\frac{T_i}{i} \right]$$

Remarque : $Fe \leq \min_i \frac{i}{T_i}$

On retrouve ici la notion de boucle critique vue en cours, où toute boucle dans le SFG peut limiter le fonctionnement en pipeline. $T_{bc} = T1/1$.

$$T_{e_{MIN}} = T_{mult} + 4T_{add} = 12ns$$

4. Quel est le retard T_d en fonction de T_{mult} et T_{add} 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.

Le retard (ou delay, ou latence dans le cas d'architectures pipelines) est le temps séparant l'acquisition de $x(n)$ de la production du résultat $y(n)$ correspondant. Dans notre cas il s'agit du chemin critique du SFG entre $x(n)$ et $y(n)$, c'est à dire :

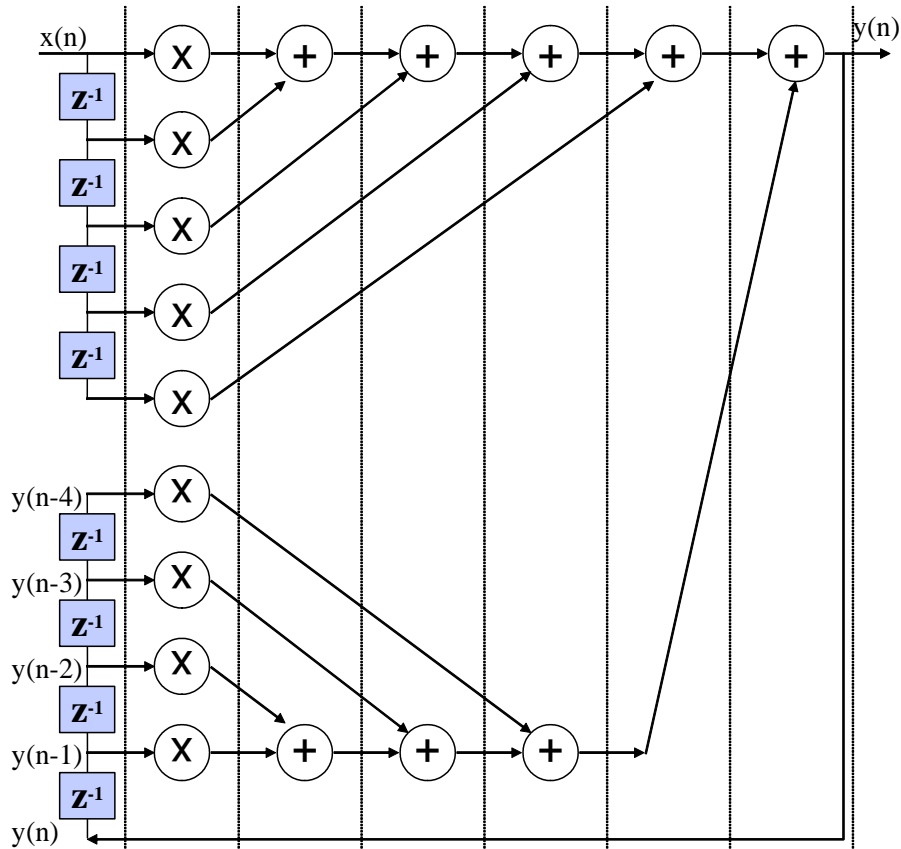


FIGURE 1 – Question 1 du problème 1

$$T_d = T_{mult} + (N + 1).T_{add}$$

Ce retard peut devenir gênant si il est très supérieur à T_e . Il peut par exemple devenir audible dans le cas d'un signal de parole, engendrer un dysfonctionnement dans le cas d'un système asservi, ...

5. On calcule les produits de convolution en faisant décroître le indices, c'est à dire en commençant les accumulations sur les échantillons d'indice les plus anciens. Que deviennent dans ce cas $F_{e_{max}}$ et T_d .

$$y(n) = [a(4).x(n - 4) + a(3).x(n - 3) + a(2).x(n - 2) + a(1).x(n - 1) + a(0).x(n)] \\ + [b(4).y(n - 4) + b(3).y(n - 3) + b(2).y(n - 2) + b(1).y(n - 1)]$$

$$T_{e_{MIN}} = T_{mult} + 2.T_{add} = 8ns$$

$$T_d = T_{mult} + 2.T_{add} = 8ns$$

6. Pour $N = P = 4$, et $T_{mult} = 4ns$, $T_{add} = 2ns$, indiquez l'ordonnancement du graphe aboutissant à une architecture pipeline à coût minimum pour une période d'échantillonnage de $12ns$.

$$\text{Le nombre de multipliers doit être } \geq \lceil \frac{N_{mult} \times T_{mult}}{T_e} \rceil = \lceil \frac{9 \times 4}{12} \rceil = 3$$

$$\text{Le nombre d'additionneurs doit être } \geq \lceil \frac{N_{add} \times T_{add}}{T_e} \rceil = \lceil \frac{8 \times 2}{12} \rceil = 2$$

Un premier ordonnancement est représenté figure 3.

On se rend compte que cet ordonnancement aboutit sur une impossibilité car la boucle critique est calculée en un temps supérieur à $T_e = 12ns$. Par conséquent, il est nécessaire d'ordonner la boucle en priorité comme sur la figure 4.

4. Synthèse d'architecture de la multiplication complexe

$$1. T_{cc} = T_{mult} + T_{add} = 90 \text{ ns}$$

2. Il faut au minimum 1 additionneur et 1 multiplieur. L'ordonnancement est donné à la figure 3

3. L'architecture obtenue contient les 2 opérateurs précités, ainsi que 5 registres (ou 4 registres et 1 multiplexeur).

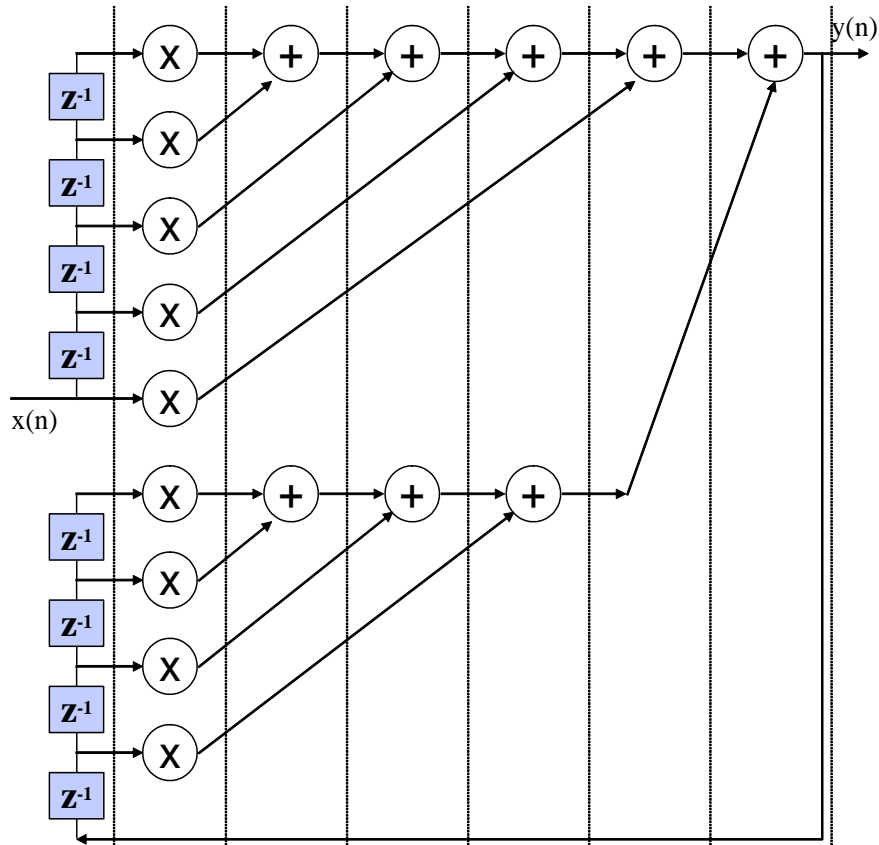


FIGURE 2 – Question 5 du problème 1

4. Si on impose un seul bus (ce qui est le cas général), les chargements des registres en tête du multiplieur ne peuvent se faire qu'à la cadence de 1 par cycle. Ceci impose maintenant que une multiplication prend 2 cycles. Cependant, en utilisant une fréquence d'horloge de 30 ns, et en modifiant légèrement l'ordonnancement, on peut maintenir la contrainte imposée de 240 ns.

5. Le VHDL comportemental pourrait être le suivant :

```
-- Multiplication complexe --
entity multiplication_complexe is
    port (A,B,C,D:in integer; Re,Im:out integer);
end multiplication_complexe;

architecture behavioral of multiplication_complexe is
    constant latency : time := 240 ns;
begin
    process
    begin
        Re <= a * c - b * d;
        Im <= a * d + b * c;
        Wait for latency;
    end process ;
end behavioral ;
```

6. Le code VHDL au niveau RTL est bien entendu beaucoup plus *verbeux*. Il faudrait décrire chaque opérateur (add, mult, registre, ...), les interconnecter, et enfin décrire le contrôleur sous forme de machine d'état. Le code comporterait ici 550 lignes si on utilisait la synthèse comportementale pour le générer ... On a donc évidemment un gain important sur la simplicité de la spécification

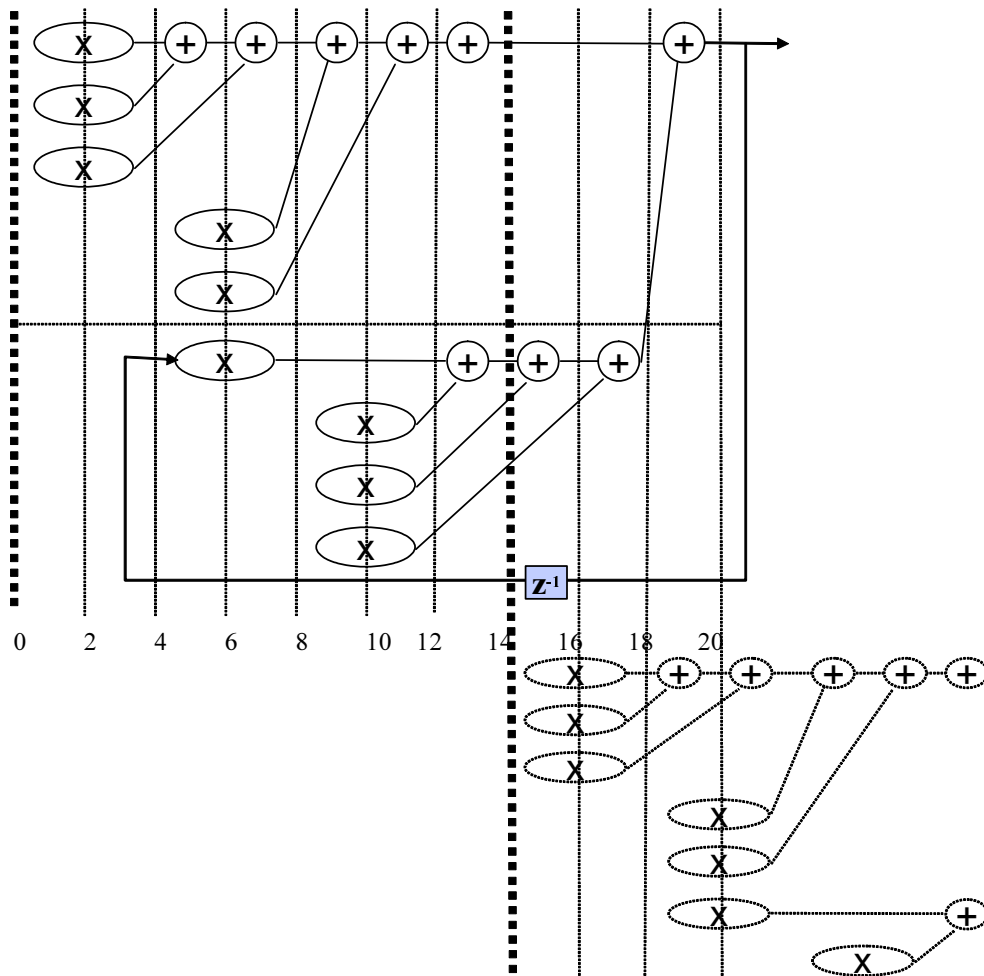


FIGURE 3 – Question 6 du problème 1

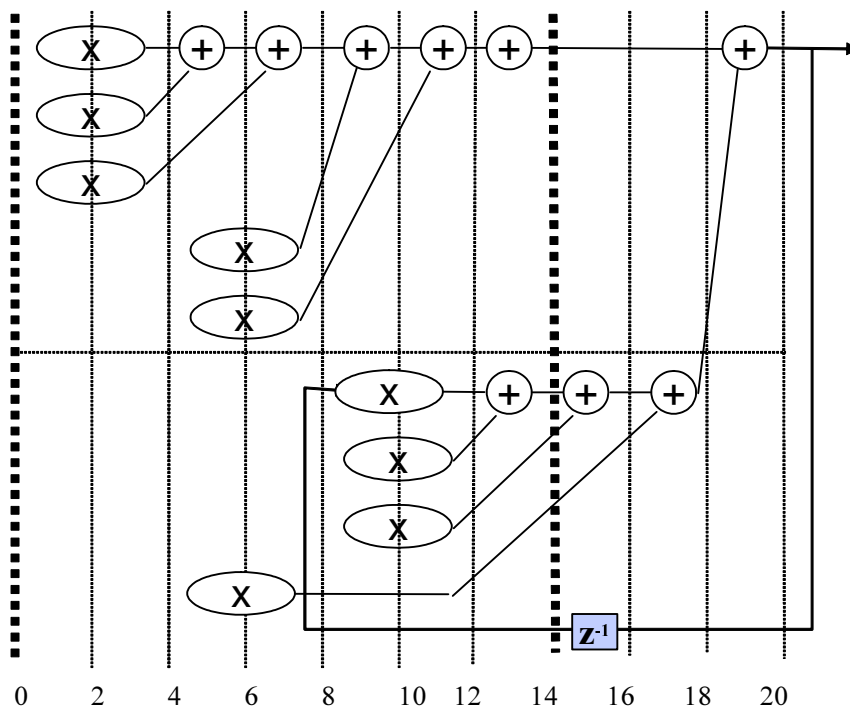


FIGURE 4 – Question 6 du problème 1 (suite)

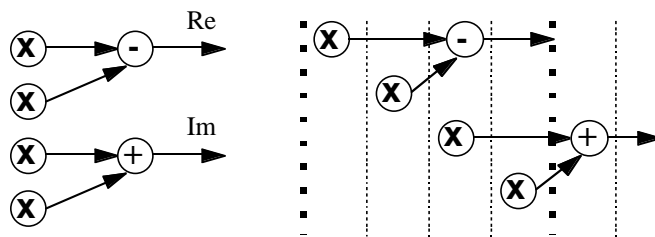


FIGURE 5 – Graphe flot de données de la multiplication complexe et son ordonnancement

impliquant un gain de conception et de simulation.