Accélérateurs matériels
Filtre FIR

Abel DIDOUH
François AUDOUIN

Table des matières

Architecture

L'objectif est de concevoir un filtre FIR d'ordre N_order où les données sont signées (en code complément à 2) sur N_data bits et les coefficients sont également signés (en code complément à 2) sur N_coeff bits. Dans les 2 cas, la virgule est située juste après le bit de signe.

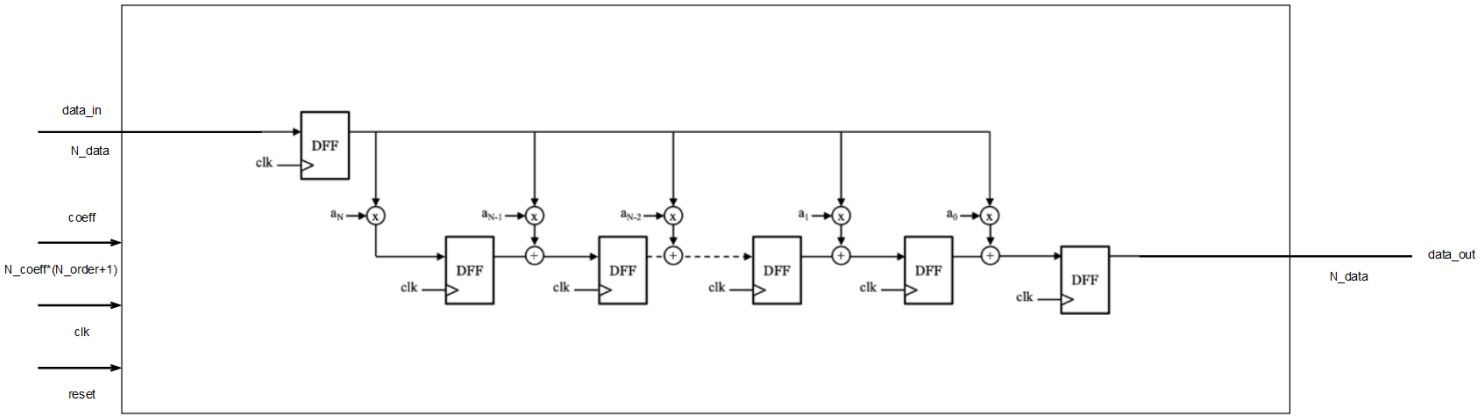
L'interface a pour paramètre générique :

Paramètre	Type	Description
`N_order`	`positive`	Ordre du filtre
`N_coeff`	`positive`	Nombre de bits des coefficients
`N_data`	`positive`	Nombre de bits des données

Les entrées et les sorties de l'architecture :

Port	Direction	Description
`clk`	Entrée	Horloge (active sur front montant)
`reset`	Entrée	**Initialisation synchrone active à 1**
`data_in`	Entrée (`N_data` bits)	Donnée à filtrer
`coeff`	Entrée (`N_coeff*(N_order+1)` bits)	Coefficients En MSB le coefficient correspondant à z^{-N_order}
`data_out`	Sortie (`N_data` bits)	Donnée filtrée

L'architecture est la suivante :



Le vecteur coeff est le vecteur coefficient de taille (N_coeff*(N_order+1)) bits. Au sein de ce vecteur, nous retrouvons chaque coefficient a_i avec i appartenant à l'intervalle [N_order, 0]. Le vecteur a_i possède une taille égale à N_coeff bits.

Fonction de transfert en Z

Nous avons :

$$dout = Q0 * z^{-1} + a0 * din * z^{-1}$$

$$dout = (Q1 * z^{-1} + a1 * din * z^{-1}) * z^{-1} + a0 * din * z^{-1}$$

$$dout = Q1 * z^{-2} + a1 * din * z^{-2} + a0 * din * z^{-1}$$

$$dout = (Q2 * z^{-1} + a2 * din * z^{-1}) * z^{-2} + a1 * din * z^{-2} + a0 * din * z^{-1}$$

$$dout = Q2 * z^{-3} + a2 * din * z^{-3} + a1 * din * z^{-2} + a0 * din * z^{-1}$$

...

$$dout = (Q_{N-1} * z^{-1} + a_{N-1} * din * z^{-1}) * z^{-N+1} + \sum_{0}^{N-2} a_i * din * z^{-i}$$

$$dout = Q_{N-1} * z^{-(N)} + a_{N-1} * din * z^{-(N)} + \sum_{0}^{N-2} a_i * din * z^{-i}$$

$$dout = Q_{N-1} * z^{-(N)} + a_{N-1} * din * z^{-(N)} + \sum_{0}^{N-2} a_i * din * z^{-i}$$

$$dout = Q_{N-1} * z^{-(N)} + \sum_{0}^{N-1} a_i * din * z^{-i}$$

$$dout = Q_N * z^{-1} * z^{-(N)} + \sum_{0}^{N-1} a_i * din * z^{-i}$$

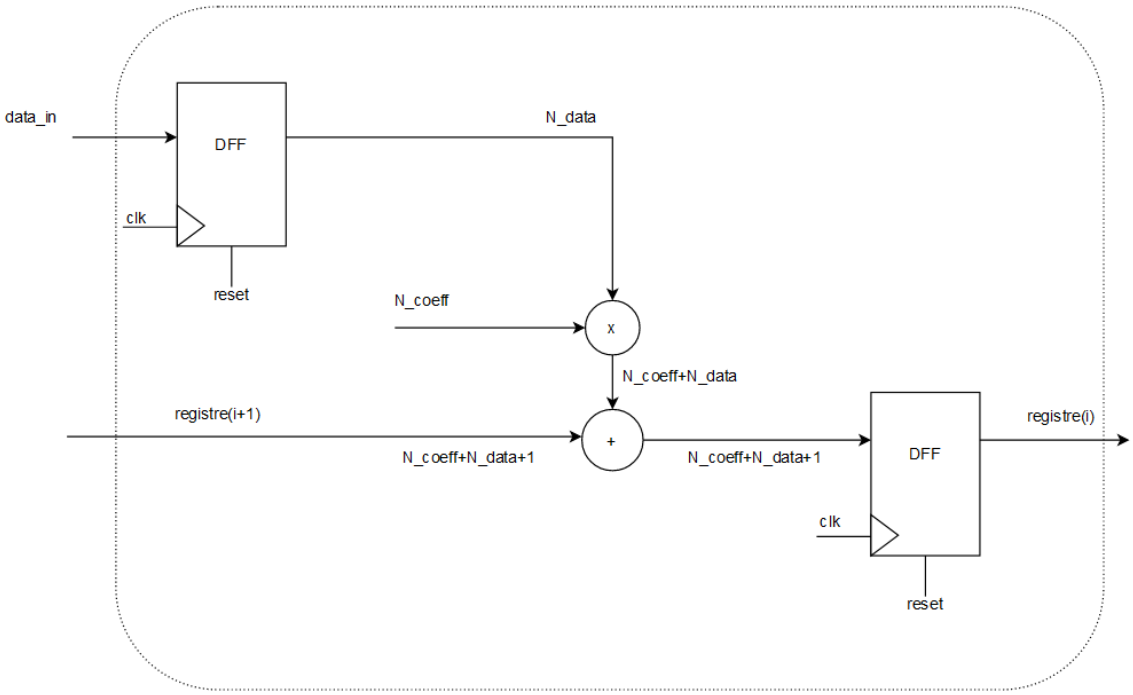
$$dout = din * z^{-1} * z^{-(N+1)} + \sum_{0}^{N-1} a_i * din * z^{-i}$$

$$dout = din * z^{-(N)} + \sum_{0}^{N-1} a_i * din * z^{-i}$$

$$dout = \sum_{0}^{N} a_i * din * z^{-i}$$

Taille des calculs, position virgule et troncature

Nous réfléchissons sur un bloc élémentaire :



Lorsque l'architecture multiplie le signal data_in sur N_data bits avec le signal a_N de taille N_coeff bits à la sortie du multiplieur le signal a une taille de N_data+N_coeff.

Lorsque l'architecture additionne le signal provenant du registre(i+1) de taille (N_coeff+N_data+1) bits et de la multiplication (N_coeff+N_data), nous obtenons un résultat sur (N_coeff+N_data+1) bits.

Prenons le cas où N_coeff est égale à 5, nous avons un vecteur a_i de taille 5.

Ai_4,	Ai_3	Ai_2	Ai_1	Ai_0

Et N_data est égale à 8 :

D7,	D6	D5	D4	D3	D2	D1	D0

Si l'on multiplie data_in par a_i nous obtenons un vecteur de taille N_data + N_coeff = 8 + 5 = 13.

D'après le cours « Accélérateur matériels » page 171 de Madame EXERTIER, nous avons

$$\begin{array}{r} N_{A1}\,,\ N_{A2} \\ \times \\ N_{B1}\,,\ N_{B2} \\ \hline N_{A1}+N_{B1}\,,\ N_{A2}+N_{B2} \end{array}$$

Nous avons une virgule juste après le bit de signe donc après multiplication la virgule se trouvera :

Ai_4,	Ai_3	Ai_2	Ai_1	Ai_0

x

D7,	D6	D5	D4	D3	D2	D1	D0

R12	R11,	R10	R9	R8	R7	R6	R5	…	R0

Si nous généralisons, nous obtenons :

Ai_Ncoeff,	Ai_Ncoeff-1	Ai_Ncoeff-2	…	Ai_0

x

D_Ndata,	D_Ndata-1	D_Ndata-2	…	D_0

R_Ncoeff+Ndata	R_Ncoeff+Ndata-1,	R_Ncoeff+Ndata-2	…	R_0

Nous pouvons passer à l'addition des deux vecteurs. C'est-à-dire du vecteur registre(i+1) et du vecteur provenant de la multiplication.

$$\begin{array}{r} N_{A1}\,,\ N_{A2} \\ + \\ N_{B1}\,,\ N_{B2} \\ \hline \max(N_{A1}\,,N_{B1})+1,\ \max(N_{A2}\,,N_{B2}) \end{array}$$

M_Ncoeff+Ndata	M_Ncoeff+Ndata-1,	M_Ncoeff+Ndata-2	…	M_0

+

Reg(i+1)_ Ncoeff+Ndata+1	Reg(i+1)_ Ncoeff+Ndata,	…	Reg(i+1)_0

Add_ Ncoeff+Ndata+1	Add_ Ncoeff+Ndata,	…	Add(i+1)_0

Nous enregistrons les résultats de chaque cellule élémentaire dans une matrice.

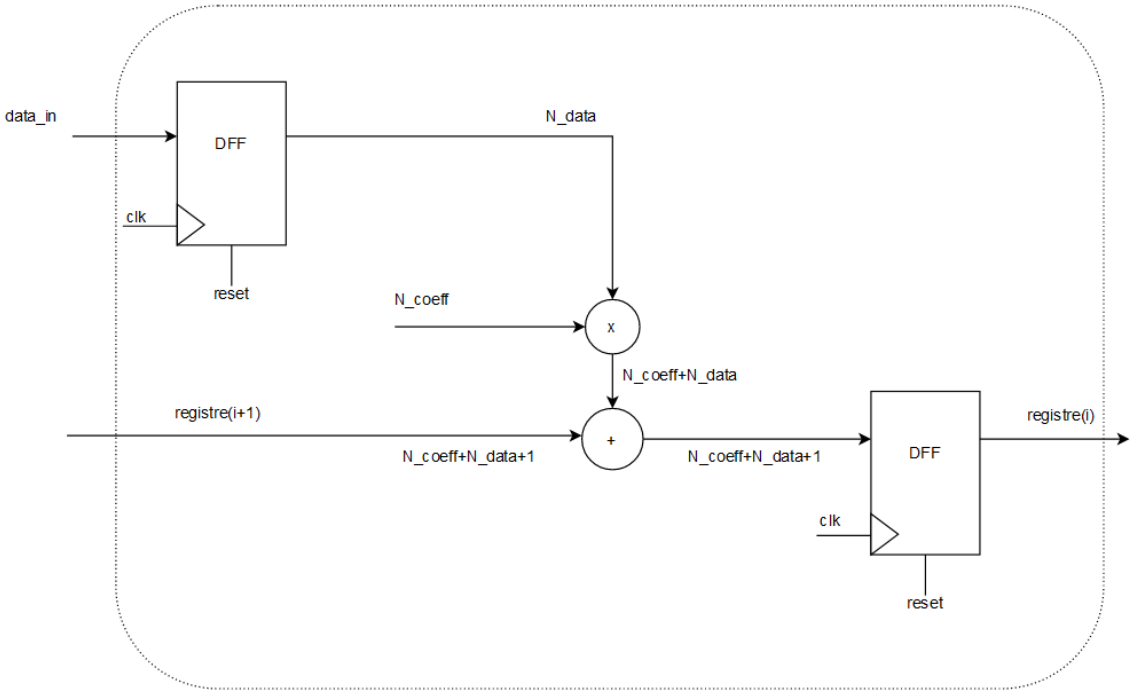
La taille de la matrice en y (profondeur) correspond à l'ordre du filtre. Tandis que la longueur de la matrice correspond à la taille du résultat de l'addition (Ncoeff+Ndata+1). Or à la sortie de notre filtre nous souhaitons avoir un vecteur de taille N_data bits. Il nous faut donc tronquer le résultat de l'addition se trouvant dans la cellule 0 de la matrice. La virgule se trouve à la position MSB-1 et nous devons prendre à partir de là les N_data bits qui précédent la virgule.

Add_ Ncoeff+Ndata+1	Add_ Ncoeff+Ndata,	Add_ Ncoeff+Ndata-1	…	Add_ Ncoeff-2-N_data	Add_0

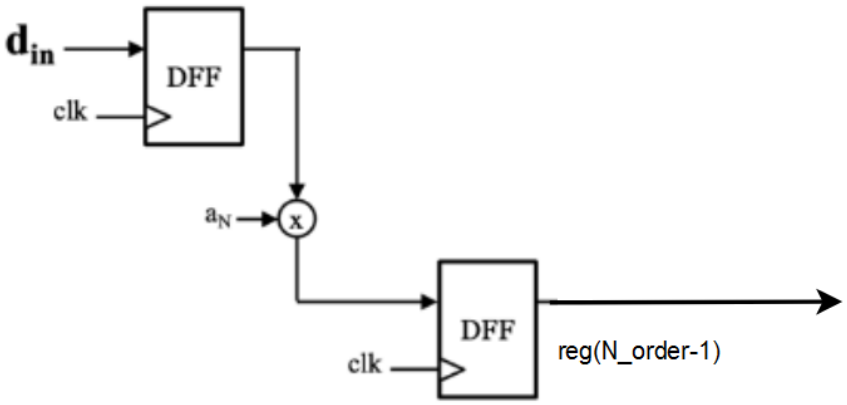
Vecteur data_out de taille N_data bits

VHDL

La cellule élémentaire que nous devrons décrire en VHDL est la suivante :



Cette cellule correspond au cas général néanmoins il y a un cas particulier qui est le suivant :



Pour ce faire, nous utiliserons une boucle for … loop qui décroit de N_order à 0.

Puis on utilise une condition if … then pour gérer le cas particulier. Sinon nous décrivons la cellule élémentaire générale. Nous utilisons des variables pour des calculs « brouillons » c'est-à-dire la multiplication et l'addition puis nous enregistrons la valeur de la variable signée dans une matrice.

Avant la fin du process, nous tronquons la valeur enregistrée dans la matrice à l'indice zéro pour obtenir une donnée utile.

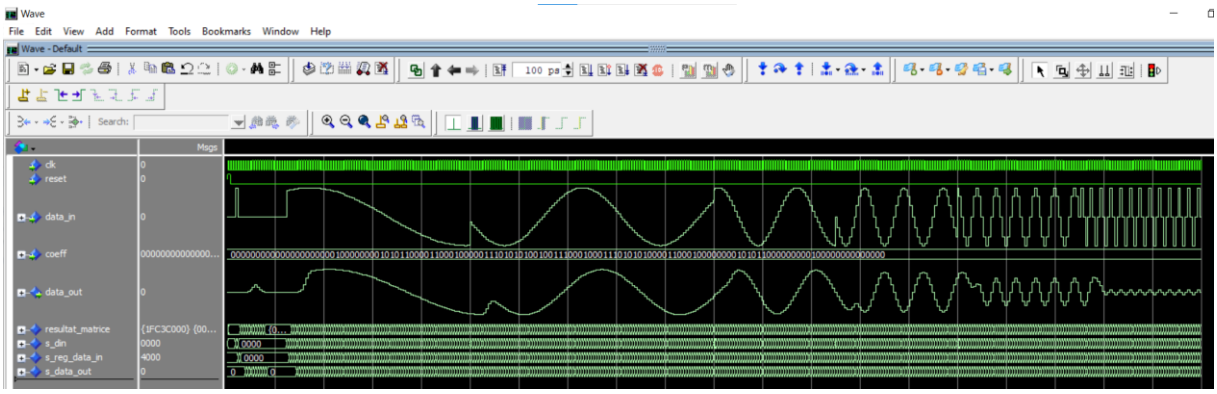
Nous pouvons passer à la simulation.

Simulation

Un testbench est fourni.

Nous créons un fichier simu.tcl et chrono.tcl

Nous obtenons la vue globale suivante :



Nous allons étudier en détail la première impulsion.

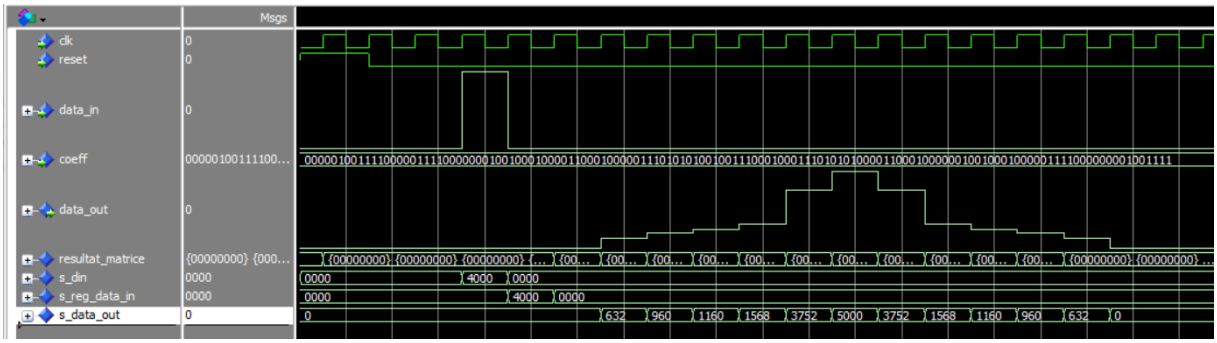
Nous allons également changer les coefficients. Cela nous permettra de corroborer avec l'ordre du filtre car nous allons mieux voir le changement de donnée sur le signal data_out.

Coefficient 0	3.8490644028E-02
Coefficient 1	5.8736879080E-02
Coefficient 2	7.0933303943E-02
Coefficient 3	9.5754693981E-02
Coefficient 4	2.2893489719E-01
Coefficient 5	3.0537582259E-01
Coefficient 6	2.2893489719E-01
Coefficient 7	9.5754693981E-02
Coefficient 8	7.0933303943E-02
Coefficient 9	5.8736879080E-02
Coefficient 10	3.8490644028E-02

Une impulsion d'amplitude I (I comme impulsion) correspond à une valeur de I en entrée pour un unique échantillon (soit une unique période d'échantillonnage). La réponse impulsionnelle d'un FIR de coefficients a_i (avec i allant de 0 à N) correspond à la suite (finie) de (N+1) valeurs : $I*a_o$, $I*a_1$ … $I*a_N$

La première impulsion est égale à 0.5.

Résultat attendu n°0	0.5*3.8490644028E-02	0,01924532
Résultat attendu n°1	0.5*3.8490644028E-02	0,02936844
Résultat attendu n°2	0.5*5.8736879080E-02	0,03546665
Résultat attendu n°3	0.5*7.0933303943E-02	0,04787735
Résultat attendu n°4	0.5*9.5754693981E-02	0,11446745
Résultat attendu n°5	0.5*2.2893489719E-01	0,15268791
Résultat attendu n°6	0.5*3.0537582259E-01	0,11446745
Résultat attendu n°7	0.5*2.2893489719E-01	0,04787735
Résultat attendu n°8	0.5*9.5754693981E-02	0,03546665
Résultat attendu n°9	0.5*7.0933303943E-02	0,02936844
Résultat attendu n°10	0.5*5.8736879080E-02	0,01924532
Résultat attendu n°11	0.5*3.8490644028E-02	0,01924532



Résultat n°1 obtenu en décimal	632
Résultat n°2 obtenu en décimal	960
Résultat n°3 obtenu en décimal	1160
Résultat n°4 obtenu en décimal	1568
Résultat n°5 obtenu en décimal	3752
Résultat n°6 obtenu en décimal	5000
Résultat n°7 obtenu en décimal	3752
Résultat n°8 obtenu en décimal	1568
Résultat n°9 obtenu en décimal	1160
Résultat n°10 obtenu en décimal	960
Résultat n°11 obtenu en décimal	632

Nous convertissons les valeurs décimales :

632	0.000 0010 0111 1000	0.01928710938
960	0.000 0011 1100 0000	0.029296875
1160	0.000 0100 1000 1000	0.03540039063

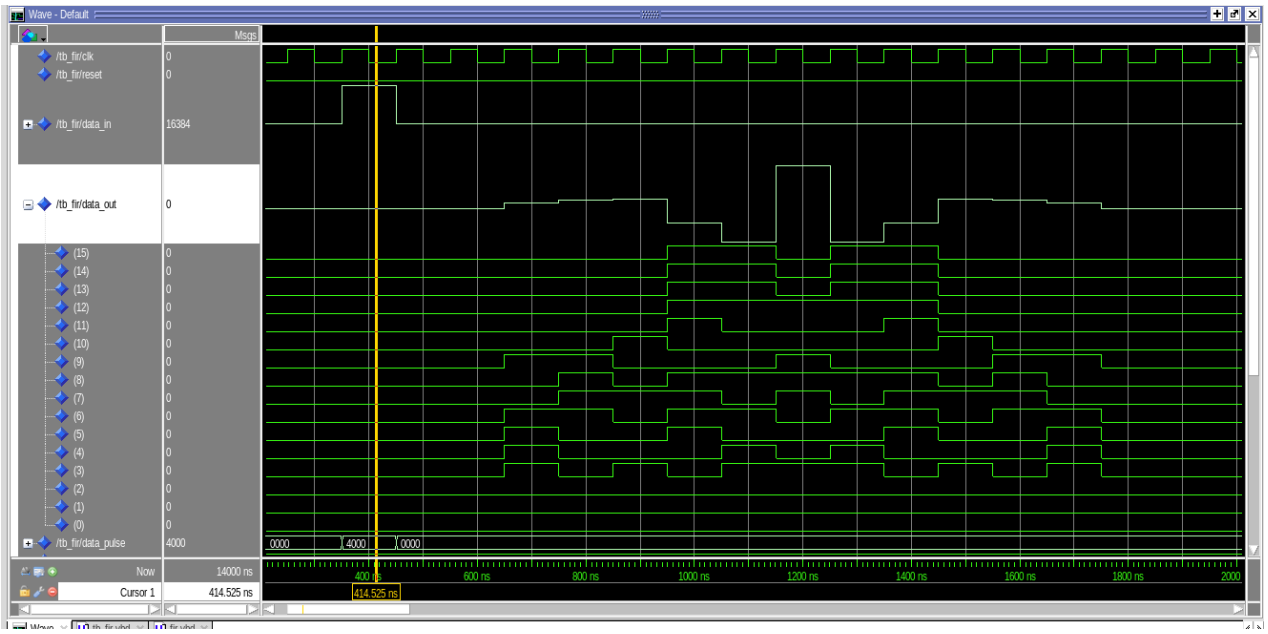
1568	0.000 0110 0010 0000	0.0478515625
3752	0.000 1110 1010 1000	0.1145019531
5000	0.001 0011 1000 1000	0.1525878906
3752	0.000 1110 1010 1000	0.1145019531
1568	0.000 0110 0010 0000	0.0478515625
1160	0.000 0100 1000 1000	0.03540039063
960	0.000 0011 1100 0000	0.029296875
632	0.000 0010 0111 1000	0.01928710938

Finalement les valeurs obtenues par simulations ne sont pas égales mais elles se rapprochent à 10^{-4} près.

Nous modifions le testbench en ajoutant des coefficients négatifs.

Coefficient 0	3.8490644028E-02
Coefficient 1	5.8736879080E-02
Coefficient 2	7.0933303943E-02
Coefficient 3	-9.5754693981E-02
Coefficient 4	-2.2893489719E-01
Coefficient 5	3.0537582259E-01
Coefficient 6	-2.2893489719E-01
Coefficient 7	-9.5754693981E-02
Coefficient 8	7.0933303943E-02
Coefficient 9	5.8736879080E-02
Coefficient 10	3.8490644028E-02

Nous obtenons la figure suivante :



Résultat attendu n°0	0.5*3.8490644028E-02	0,019245322
Résultat attendu n°1	0.5*3.8490644028E-02	0,02936844
Résultat attendu n°2	0.5*5.8736879080E-02	0,035466652
Résultat attendu n°3	0.5*7.0933303943E-02	-0,047877347
Résultat attendu n°4	0.5*9.5754693981E-02	-0,114467449
Résultat attendu n°5	0.5*2.2893489719E-01	0,152687911
Résultat attendu n°6	0.5*3.0537582259E-01	0,114467449
Résultat attendu n°7	0.5*2.2893489719E-01	-0,047877347
Résultat attendu n°8	0.5*9.5754693981E-02	-0,035466652
Résultat attendu n°9	0.5*7.0933303943E-02	0,02936844
Résultat attendu n°10	0.5*5.8736879080E-02	0,019245322
Résultat attendu n°11	0.5*3.8490644028E-02	0,019245322

Nous obtenons les résultats suivants provenant de la simulation

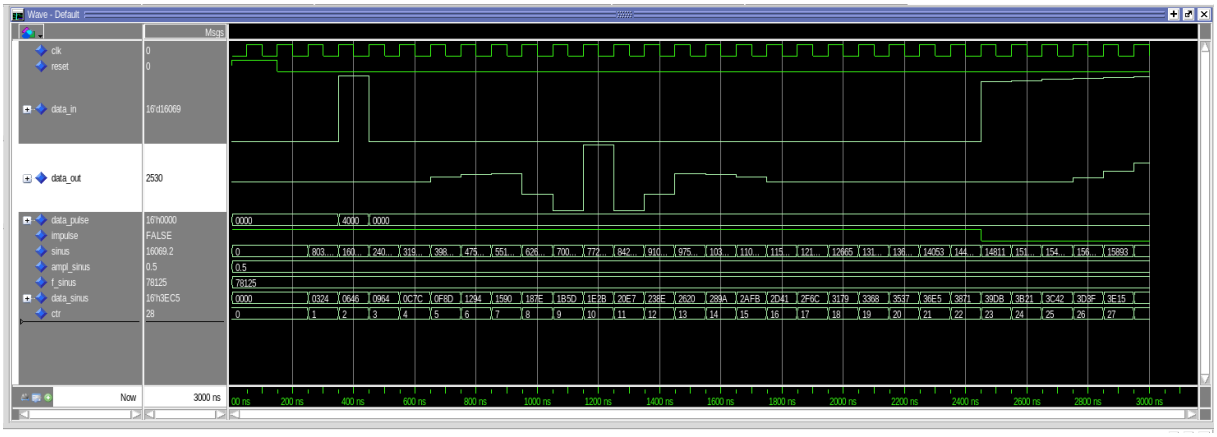
Résultat n°1 obtenu en décimal	632
Résultat n°2 obtenu en décimal	960
Résultat n°3 obtenu en décimal	1160
Résultat n°4 obtenu en décimal	-1568
Résultat n°5 obtenu en décimal	-3752
Résultat n°6 obtenu en décimal	5000
Résultat n°7 obtenu en décimal	-3752
Résultat n°8 obtenu en décimal	-1568
Résultat n°9 obtenu en décimal	1160
Résultat n°10 obtenu en décimal	960
Résultat n°11 obtenu en décimal	632

Nous convertissons les valeurs décimales :

632	0.000 0010 0111 1000	0.01928710938
960	0.000 0011 1100 0000	0.029296875
1160	0.000 0100 1000 1000	0.03540039063
-1568	1.111 1001 1110 0000	-0.0478515625
-3752	1.111 0001 0101 1000	-0.1145019531
5000	0.001 0011 1000 1000	0.1525878906
3752	0.000 1110 1010 1000	0.1145019531
-1568	1.111 0001 0101 1000	-0.0478515625
-1160	1.111 1001 1110 0000	-0.1145019531
960	0.000 0011 1100 0000	0.029296875
632	0.000 0010 0111 1000	0.01928710938

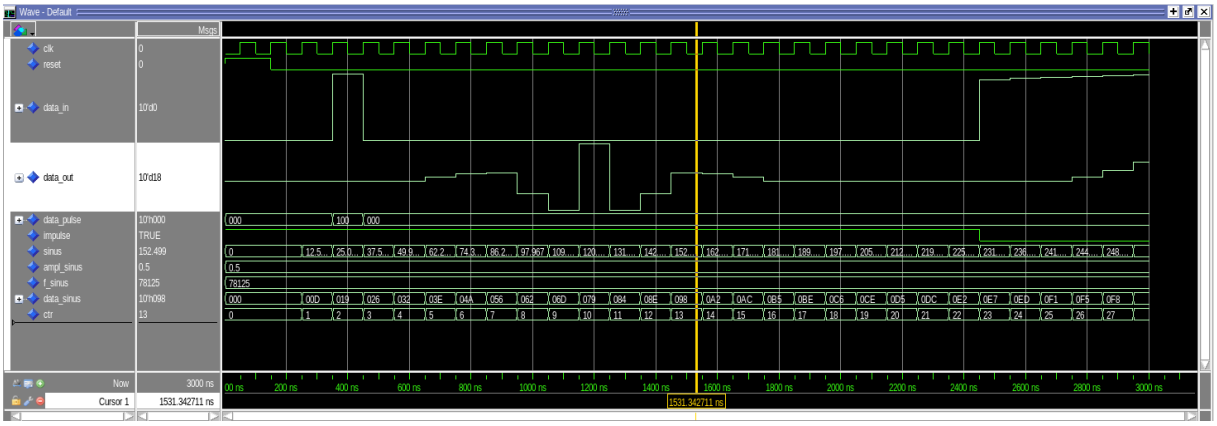
Nous obtenons bien les résultats souhaités.

Regardons la simulation « Behavioral Simulation »



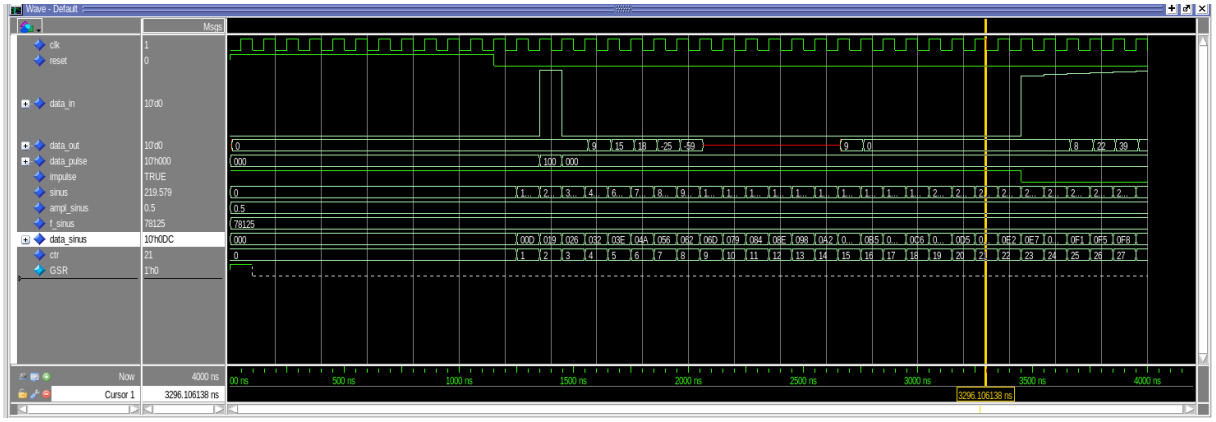
La simulation « Behavioral Simulation » est fonctionnel.

Regardons la simulation « Post-Implementation Functional Simulation »



La simulation est fonctionnelle.

Regardons la simulation « Post-Implementation Timing Simulation »



Cette simulation n’est pas fonctionnelle. On ne retrouve pas la donnée souhaitée en sortie de notre architecture.

Finalement, notre architecture est fonctionnelle jusqu'à l'étape de la simulation avec timing. En effet, lors de la simulation « Post Implementation Timing Simulation » nous obtenons des valeurs erronées. De ce fait une implémentation sur carte aboutira à un échec.

Implémentation

Analyse du rapport de synthèse

```
Module fir : Parameter N_order bound to: 8 - type: integer
Module fir : Parameter N_coeff bound to: 12 - type: integer
Module fir : Parameter N_data bound to: 10 - type: integer
```

Vivado reconnait les paramètres génériques suivant :

```
Report Check Netlist:
+------+------------------+-------+---------+-------+------------------+
|      |Item              |Errors |Warnings |Status |Description       |
+------+------------------+-------+---------+-------+------------------+
|1     |multi_driven_nets |      0|        0|Passed |Multi driven nets |
+------+------------------+-------+---------+-------+------------------+
```

Vivado informe qu'il n'y a aucun court-circuit.

```
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Registers :
                       23 Bit    Registers := 1
                       10 Bit    Registers := 10
---------------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------------
```

Vivado informe qu'il a reconnu un registre 23 bits et 10 registres de 10 bits.

Les 10 registres de 10 bits correspondent au registre de data_in sur 10 bits, au registre de data_out sur 10 bits et il y a un registre de 10 bits pour chaque étage du filtre.

```
Report Cell Usage:
+------+----------+------+
|      |Cell      |Count |
+------+----------+------+
|1     |BUFG      |     1|
|2     |DSP48E1   |     8|
|3     |DSP48E1_1 |     1|
|4     |LUT1      |     1|
|5     |FDRE      |    10|
|6     |IBUF      |   120|
|7     |OBUF      |    10|
+------+----------+------+
```

Il n'y a aucun latch.

```
DSP Report: Generating DSP resultat_matrice_reg[8]0, operation Mode
is: A*B2.
```

Un opérateur arithmétique à été réalisé dans un bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[8]0.
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: operator resultat_matrice_reg[8]0 is absorbed into DSP
resultat_matrice_reg[8]0.
```

On retrouve ici qu'un opérateur arithmétique qui a été utilisé par le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[7], operation Mode
is: (C+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisé dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[7].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[7] is absorbed into DSP
resultat_matrice_reg[7].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[7]0 is absorbed into DSP
resultat_matrice_reg[7].
DSP Report: operator resultat_matrice_reg[7]1 is absorbed into DSP
resultat_matrice_reg[7].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[6], operation Mode
is: (PCIN+A*B2)'.
```

Les 2 opérateurs arithmétiques ont été réalisé dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[6].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[6] is absorbed into DSP
resultat_matrice_reg[6].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[6]0 is absorbed into DSP
resultat_matrice_reg[6].
DSP Report: operator resultat_matrice_reg[6]1 is absorbed into DSP
resultat_matrice_reg[6].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[5], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[5].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[5] is absorbed into DSP
resultat_matrice_reg[5].
```

Le registre resultat_matrice a été absorbé dans le DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[5]0 is absorbed into DSP
resultat_matrice_reg[5].
DSP Report: operator resultat_matrice_reg[5]1 is absorbed into DSP
resultat_matrice_reg[5].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[4], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[4].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[4] is absorbed into DSP
resultat_matrice_reg[4].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[4]0 is absorbed into DSP
resultat_matrice_reg[4].
DSP Report: operator resultat_matrice_reg[4]1 is absorbed into DSP
resultat_matrice_reg[4].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[3], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[3].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[3] is absorbed into DSP
resultat_matrice_reg[3].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[3]0 is absorbed into DSP
resultat_matrice_reg[3].
DSP Report: operator resultat_matrice_reg[3]1 is absorbed into DSP
resultat_matrice_reg[3].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[2], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[2].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[2] is absorbed into DSP
resultat_matrice_reg[2].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[2]0 is absorbed into DSP
resultat_matrice_reg[2].
DSP Report: operator resultat_matrice_reg[2]1 is absorbed into DSP
resultat_matrice_reg[2].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[1], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[1].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[1] is absorbed into DSP
resultat_matrice_reg[1].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[1]0 is absorbed into DSP
resultat_matrice_reg[1].
DSP Report: operator resultat_matrice_reg[1]1 is absorbed into DSP
resultat_matrice_reg[1].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

```
DSP Report: Generating DSP resultat_matrice_reg[0], operation Mode
is: (PCIN+A*B2)'.
```

Les deux opérateurs arithmétiques ont été réalisés dans le bloc DSP.

```
DSP Report: register s_reg_data_in_reg is absorbed into DSP
resultat_matrice_reg[0].
```

Le registre s_reg_data_in a été absorbé dans le bloc DSP.

```
DSP Report: register resultat_matrice_reg[0] is absorbed into DSP
resultat_matrice_reg[0].
```

Le registre resultat_matrice a été absorbé dans le bloc DSP, c'est le registre de sortie qui a été utilisé.

```
DSP Report: operator resultat_matrice_reg[0]0 is absorbed into DSP
resultat_matrice_reg[0].
DSP Report: operator resultat_matrice_reg[0]1 is absorbed into DSP
resultat_matrice_reg[0].
```

On retrouve ici que 2 opérateurs arithmétiques ont été utilisés dans le bloc DSP.

Finalement, nous retrouvons bien pour la cellule particulière une opération de multiplication. Ainsi, le registre s_reg_data_in est absorbé dans le bloc DSP. Le bloc DSP effectue la multiplication. Le registre resultat_matrice[8] est absorbé. C'est le registre de sortie qui a été utilisé.

La cellule élémentaire utilise un bloc DSP. En effet, dans un premier temps, les deux opérateurs arithmétiques (+, X) sont réalisés au sein du bloc DSP. Le registre s_reg_data_in est absorbé dans le bloc DSP tout comme la sortie resultat_matrice[i] qui également absorbé dans le bloc DSP.

```
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
INFO: [Synth 8-4471] merging register 's_reg_data_in_reg[9:0]' into 's_reg_data_in_reg[9:0]'
[/user/audouinf/SEI_Semestre2/ACC_MAT/tp_fir/rtl/fir.vhd:40]
```

Vivado nous informe qu'il va fusionner le registre s_reg_data_in dans le registre s_reg_data_in. Nous ne comprenons pas ce choix fait par le synthétiseur. (Mais en écrivant la suite du rapport, nous comprenons que s_reg_data_in est dupliqué dans chaque bloc DSP dans le BREG.)

```
INFO: [Synth 8-3333] propagating constant 0 across sequential
element (\resultat_matrice_reg[8][22] )
```

Vivado a bien reconnu le zéro que l'on a concaténé à la multiplication pour que la taille du vecteur soit exacte.

DSP: Preliminary Mapping Report (see note below. The ' indicates corresponding REG is set)

Module Name	DSP Mapping	A Size	B Size	C Size	D Size	P Size	AREG	BREG	CREG	DREG	ADREG	MREG	PREG
fir	A*B2	12	10	-	-	22	0	1	-	-	-	0	0
fir	(C+A*B2)'	12	10	23	-	23	0	1	0	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1

Le registre M (en sortie du multiplieur) n'est pas utilisé.

La cellule particulière utilise un bloc DSP. Le registre PREG n'est pas utilisé. Le registre BREG sur 10 bits est utilisé pour s_reg_data_in (finalement s_reg_data_in est dupliqué).

Pour les cellules élémentaires qui suivent et utilisent des blocs DSP, le registre PREG est utilisé pour le registre resultat_matrice[i] sur 23 bits. Le registre BREG sur 10 bits est utilisé pour s_reg_data_in.

Module Name	DSP Mapping	A Size	B Size	C Size	D Size	P Size	AREG	BREG	CREG	DREG	ADREG	MREG	PREG
fir	A*B2	12	10	-	-	22	0	1	-	-	-	0	0

Cela correspond à la cellule particulière. D'après la documentation, le port A est un port d'entrée pour les opérations de préadministration, de multiplication, d'addition/soustraction/accumulation, d'ALU ou de concaténation. Le port A prend le vecteur correspondant au coefficient et le port B prend le vecteur correspondant à s_reg_data_in.

Module Name	DSP Mapping	A Size	B Size	C Size	D Size	P Size	AREG	BREG	CREG	DREG	ADREG	MREG	PREG
fir	(C+A*B2)'	12	10	23	-	23	0	1	0	-	-	0	1

Le vecteur résultat de la cellule particulière est envoyé sur le port d'entrée C. Le vecteur passe de 22 bits à 23 bits car on concatène un zéro au vecteur résultat de la cellule particulière qui est sur 22 bits. Il y a une multiplication entre le vecteur coefficient de l'ordre – 1 avec la donnée s_reg_data_in qui a été dupliqué. Le registre PREG est utilisé par résultat_matrice[N_ordre-1].

Module Name	DSP Mapping	A Size	B Size	C Size	D Size	P Size	AREG	BREG	CREG	DREG	ADREG	MREG	PREG
fir	(PCIN+A*B2)'	12	10	-	-	23	0	1	-	-	-	0	1

Comme précédemment, le bloc DSP multiplie le coefficient correspondant à l'ordre i avec s_reg_data_in qui a été dupliqué dans BREG. D'autre part, nous remarquons la présence de PCIN. En effet, ce port correspond selon la fiche

technique de Xilinx à un port d'entrée de données en cascade depuis PCOUT de la tranche DSP48E1 précédente vers l'additionneur. Le registre de sortie du bloc précédent qui se trouve dans PREG est connecté au bloc DSP actuel. Finalement, le registre PREG de l'actuel bloc DSP est utilisé pour resultat_matrice[i].

Cela se reproduit jusqu'à l'ordre 0.

Précédemment, nous avions relevé la présence d'un registre de 23 bits. Nous sommes certains que ce registre est le registre C qui est utilisé pour faire la jonction entre la cellule particulière et la première cellule élémentaire.

Des changements (Youpi merci Vivado ! 😊)

```
DSP Final Report (the ' indicates corresponding REG is set)
+------------+--------------+--------+--------+--------+--------+--------+------+------+------+------+-------+------+------+
|Module Name | DSP Mapping  | A Size | B Size | C Size | D Size | P Size | AREG | BREG | CREG | DREG | ADREG | MREG | PREG |
+------------+--------------+--------+--------+--------+--------+--------+------+------+------+------+-------+------+------+
|fir         | (A*B')'      | 30     | 18     | -      | -      | 22     | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (C+A*B')'    | 30     | 18     | 22     | -      | 0      | 0    | 1    | 1    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 0      | 0    | 1    | -    | -    | -     | 0    | 1    |
|fir         | (PCIN+A*B')' | 30     | 18     | -      | -      | 23     | 0    | 1    | -    | -    | -     | 0    | 1    |
+------------+--------------+--------+--------+--------+--------+--------+------+------+------+------+-------+------+------+
```

Pour la cellule particulière, le port d'entrée A est sur 30 bits. Le port A correspond au coefficient à l'ordre N. Le port B est sur 18 bits. Ce port correspond à s_reg_data_in qui est dupliqué dans BREG. Finalement, le bloc DSP ne vas prendre que les 22 premiers bits (au-delà ils sont nuls). Il va créer un registre C pour mémoriser le résultat de la cellule particulière pour pouvoir faire le pont avec la cellule élémentaire.

Le bloc DSP de la cellule élémentaire se trouvant après celle de la cellule particulière effectue son calcul. Comme précédemment et avec les blocs DSP qui viendront après le port A est sur 30 bits et le port B est sur 10 bits. Nous remarquons que le port P ne prend plus de vecteur car la taille est nulle. Cela est la conséquence que les blocs DSP utilisent PCIN pour propager le résultat.

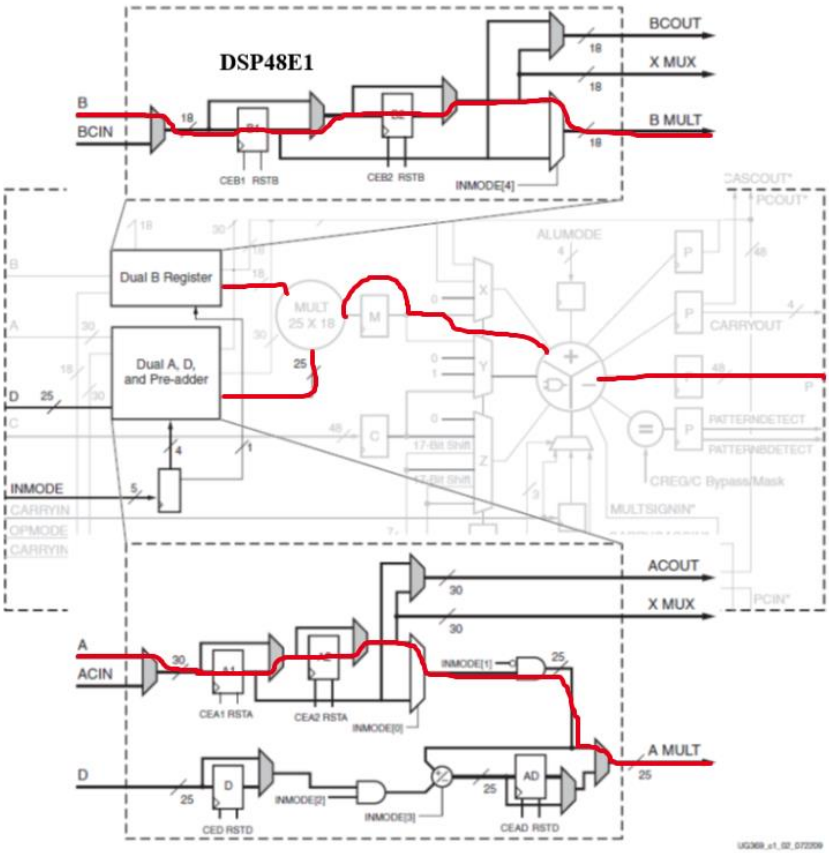
Finalement, à la fin nous avons bien une sortie sur le port P avec une taille de 23 bits qui correspond à la taille de N_coeff + N_data + 1.

```
Report Cell Usage:
+------+----------+------+
|      |Cell      |Count |
+------+----------+------+
|1     |BUFG      |     1|
|2     |DSP48E1   |     8|
|3     |DSP48E1_1 |     1|
|4     |LUT1      |     1|
|5     |FDRE      |    10|
|6     |IBUF      |   120|
|7     |OBUF      |    10|
+------+----------+------+
```

Vivado a reconnu 8 DSP, 10 bascules, 1BUFG pour l'arbre d'horloge, 120 ports d'entrées (118 de données + horloge et reset) et 10 ports de sorties (données).

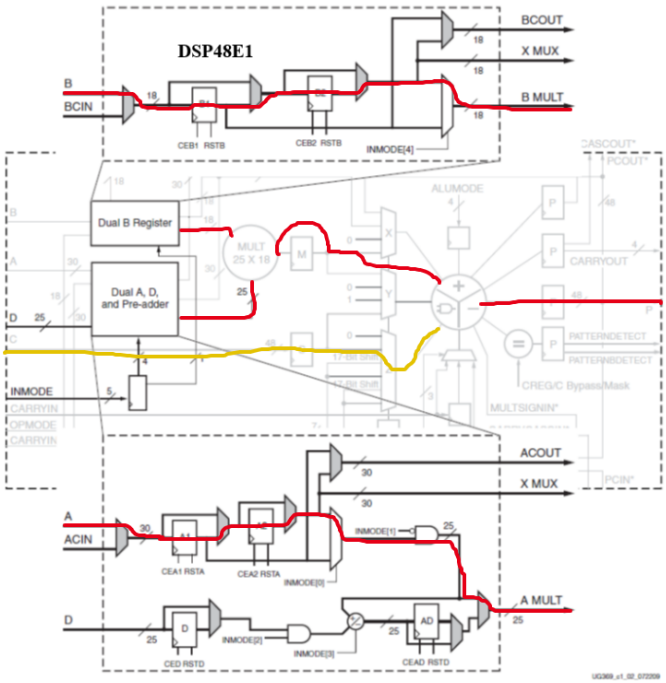
Explication de l'outil de synthèse sur l'utilisation du bloc DSP :

<u>Pour A*B :</u>



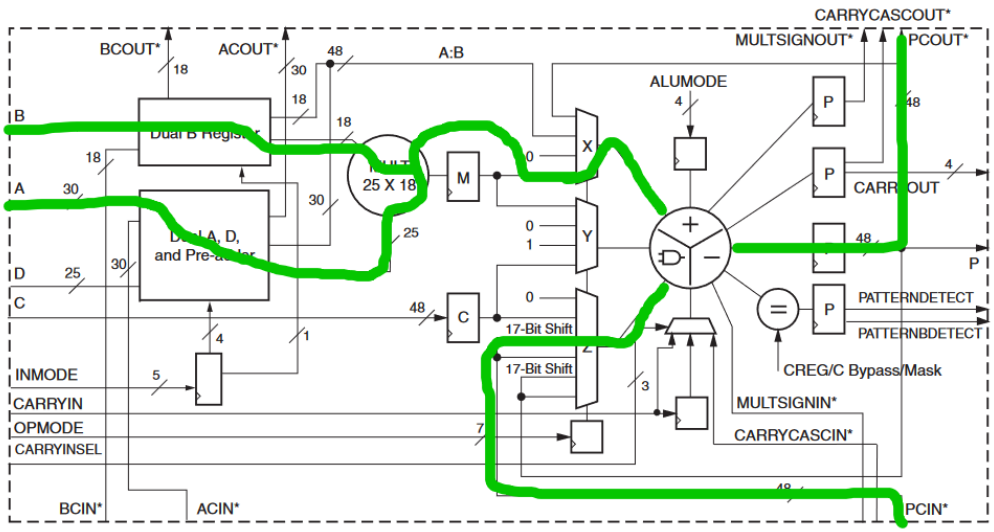
Le registre M n'est pas utilisé, on en déduit qu'il existe un chemin qui relie le multiplieur directement avec l'additionneur/soustracteur.

Pour C+A*B :



Le registre M n'est pas utilisé, on en déduit qu'il existe un chemin qui relie le multiplieur directement avec l'additionneur/soustracteur.

Pour PCIN+A*B :



Extrait de la doc AMD-Xilinx

Le registre M n'est pas utilisé, on en déduit qu'il existe un chemin qui relie le multiplieur directement avec l'additionneur/soustracteur.

Analyse du rapport de ressources

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	1	0	0	63400	<0.01
LUT as Logic	1	0	0	63400	<0.01
LUT as Memory	0	0	0	19000	0.00
Slice Registers	10	0	0	126800	<0.01
Register as Flip Flop	10	0	0	126800	<0.01
Register as Latch	0	0	0	126800	0.00
F7 Muxes	0	0	0	31700	0.00
F8 Muxes	0	0	0	15850	0.00

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
0	Yes	-	Reset
0	Yes	Set	-
10	Yes	Reset	-

Relevé des ressources

- 3 slices
- 1 LUT
- 10 DFF

2. Slice Logic Distribution

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	3	0	0	15850	0.02
SLICEL	1	0			
SLICEM	2	0			
LUT as Logic	1	0	0	63400	<0.01
using O5 output only	0				
using O6 output only	1				
using O5 and O6	0				
LUT as Memory	0	0	0	19000	0.00
LUT as Distributed RAM	0	0			
LUT as Shift Register	0	0			
Slice Registers	10	0	0	126800	<0.01
Register driven from within the Slice	0				
Register driven from outside the Slice	10				
LUT in front of the register is unused	9				
LUT in front of the register is used	1				
Unique Control Sets	1		0	15850	<0.01

3. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	135	0.00
RAMB36/FIFO*	0	0	0	135	0.00
RAMB18	0	0	0	270	0.00

4. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	9	0	0	240	3.75
DSP48E1 only	9				

5. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	130	1	0	210	61.90
IOB Master Pads	62				
IOB Slave Pads	63				
Bonded IPADs	0	0	0	2	0.00
PHY_CONTROL	0	0	0	6	0.00
PHASER_REF	0	0	0	6	0.00
OUT_FIFO	0	0	0	24	0.00
IN_FIFO	0	0	0	24	0.00
IDELAYCTRL	0	0	0	6	0.00
IBUFDS	0	0	0	202	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	24	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	24	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	300	0.00
ILOGIC	0	0	0	210	0.00
OLOGIC	0	0	0	210	0.00

6. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	1	0	0	32	3.13
BUFIO	0	0	0	24	0.00
MMCME2_ADV	0	0	0	6	0.00
PLLE2_ADV	0	0	0	6	0.00
BUFMRCE	0	0	0	12	0.00
BUFHCE	0	0	0	96	0.00
BUFR	0	0	0	24	0.00

Relevé des ressources :

- 0 block RAM
- 9 DSP
- 1 arbre d’horloge (BUFGCTRL)
- 0 PLL

Analyse du rapport de timings

```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints
    -------      -------  ---------------------  -------------------
      7.366        0.000                      0                  368

    WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints
    -------      -------  ---------------------  -------------------
      0.319        0.000                      0                  368

    WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    --------     --------  ----------------------  --------------------
       4.500        0.000                       0                    20


------------------------------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------------------------------

Clock        Waveform(ns)       Period(ns)      Frequency(MHz)
-----        ------------       ----------      --------------
sys_clk_pin  {0.000 5.000}      10.000          100.000


------------------------------------------------------------------------------------------------
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------------------------
    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints
    -------      -------  ---------------------  -------------------
      7.366        0.000                      0                  368

    WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints
    -------      -------  ---------------------  -------------------
      0.319        0.000                      0                  368

    WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    --------     --------  ----------------------  --------------------
       4.500        0.000                       0                    20
```

Relevé :

- Slack = WNS = 7.366 ns
- WNS ≥ 0 et WHS ≥ 0

Calcul fréquence maximale :

$$fmax = \frac{1}{(10 - 7.366)10^{-9}} = 379{,}65\ MHz$$

STA : chemin critique

Pour le setup :

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             7.366ns  (required time - arrival time)
  Source:                 resultat_matrice_reg[8]0/CLK
                            (rising edge-triggered cell DSP48E1 clocked by sys_clk_pin  {rise@0.000ns
fall@5.000ns period=10.000ns})
  Destination:            resultat_matrice_reg[7]/C[0]
                            (rising edge-triggered cell DSP48E1 clocked by sys_clk_pin  {rise@0.000ns
fall@5.000ns period=10.000ns})
  Path Group:             sys_clk_pin
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (sys_clk_pin rise@10.000ns - sys_clk_pin rise@0.000ns)
  Data Path Delay:        0.872ns  (logic 0.434ns (49.777%)  route 0.438ns (50.223%))
  Logic Levels:           0
  Clock Path Skew:        -0.026ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    5.022ns = ( 15.022 - 10.000 )
    Source Clock Delay      (SCD):    5.324ns
    Clock Pessimism Removal (CPR):    0.275ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         (clock sys_clk_pin rise edge)
                                                      0.000     0.000 r
    E3                                                0.000     0.000 r  clk (IN)
                         net (fo=0)                   0.000     0.000    clk
    E3                   IBUF (Prop_ibuf_I_O)         1.482     1.482 r  clk_IBUF_inst/O
                         net (fo=1, routed)           2.025     3.506    clk_IBUF
    BUFGCTRL_X0Y16       BUFG (Prop_bufg_I_O)         0.096     3.602 r  clk_IBUF_BUFG_inst/O
                         net (fo=19, routed)          1.722     5.324    clk_IBUF_BUFG
    DSP48_X0Y40          DSP48E1                                     r  resultat_matrice_reg[8]0/CLK
  -------------------------------------------------------------------    -------------------
    DSP48_X0Y40          DSP48E1 (Prop_dsp48e1_CLK_P[0])
                                                      0.434     5.758 r  resultat_matrice_reg[8]0/P[0]
                         net (fo=1, routed)           0.438     6.196    resultat_matrice_reg[8][0]
    DSP48_X0Y41          DSP48E1                                     r  resultat_matrice_reg[7]/C[0]
  -------------------------------------------------------------------    -------------------

                         (clock sys_clk_pin rise edge)
                                                     10.000    10.000 r
    E3                                                0.000    10.000 r  clk (IN)
                         net (fo=0)                   0.000    10.000    clk
    E3                   IBUF (Prop_ibuf_I_O)         1.411    11.411 r  clk_IBUF_inst/O
                         net (fo=1, routed)           1.920    13.331    clk_IBUF
    BUFGCTRL_X0Y16       BUFG (Prop_bufg_I_O)         0.091    13.422 r  clk_IBUF_BUFG_inst/O
                         net (fo=19, routed)          1.600    15.022    clk_IBUF_BUFG
    DSP48_X0Y41          DSP48E1                                     r  resultat_matrice_reg[7]/CLK
                         clock pessimism              0.275    15.298
                         clock uncertainty           -0.035    15.262
    DSP48_X0Y41          DSP48E1 (Setup_dsp48e1_CLK_C[0])
                                                     -1.701    13.561    resultat_matrice_reg[7]
  -------------------------------------------------------------------
                         required time                         13.561
                         arrival time                          -6.196
  -------------------------------------------------------------------
                         slack                                  7.366
```

Le chemin critique se trouve entre le registre resultat_matrice cellule 8 et le registre resultat_matrice cellule 7. Finalement, le chemin critique se trouve entre la cellule particulière et la première cellule élémentaire.

Chemin critique :