Station hybride (DSP/FPGA) pour un système rapide de reconnaissance automatique de la parole

Hybrid station (DSP/FPGA) for fast automatic speech recognition system

Hamza Atoui*, Mouhamed Boughazi & Mouhamed Fezari

Département d'électronique, Université Badji Mokhtar, BP 12, Annaba, 23000, Algérie.

Info. Article

Historique de l'article

Reçu le 28/03/2018
Révisé le 12/09/2019
Accepté le 23/09/2019

Mots-clés

DSP-FPGA-DTW-LMS-Reconnaissance de la parole.

RESUME

La reconnaissance automatique de la parole (RAP) a fait l'objet de plusieurs investigations depuis le début des années 50. La nature complexe de la parole oblige, l'auditeur humain à utiliser plusieurs niveaux de traitement pour ce domaine. Le 1er niveau est la détermination des caractéristiques du signal de la parole lui-même, c'est-à-dire l'analyse acoustique. Les niveaux phonétique, lexical et sémantique, etc. c'est dire combien chez l'être humain la reconnaissance et la compréhension de la parole sont fortement liées. Le but du présent travail est la réalisation d'un système embarqué (DSP/FPGA) pour le domaine de RAP fonctionnant en temps réel. Pour atteindre ce but, nous avons matérialisé quelques parties du système RAP implémenté sur DSP comme le filtre LMS et le cœur de la distance DTW (accélérateurs matériels) sur FPGA. Cette implantation nous permet d'accélérer le processus de la reconnaissance donc une augmentation considérable du dictionnaire de références.

Keywords

DSP-FPGA-DTW-LMS- Speech Recognition

ABSTRACT

The automatic speech recognition (ASR) has been the subject of several research studies since the early 50s. And because of the complex nature of speech, human listener uses several levels of treatment for this domain. The first level is the determination of the speech signal characteristics itself, i.e the acoustic analysis. Then comes the phonetics, lexical and semantic levels, etc. This shows how human's recognition and speech understanding are strongly related. The purpose of this work is to design an embedded ASR system based on (DSP/FPGA) functioning/operating in real time. To achieve this goal, we have materialized and implemented some parts of the ASR system on DSP and other parts such as the LMS filter and the core of the DTW distance (hardware accelerators) on FPGA, this hybrid implementation allows us to accelerate the process of automatic speech recognition, so it can be used on real time applications to help us reach a considerable increase amount of words in the reference dictionary.

* ***Auteur correspondant***
Hamza Atoui
Département d'électronique, Université Badji Mokhtar, BP 12, Annaba, 23000, Algérie.
Email : hamza.atoui@gmail.com

©UBMA 2020

1. INTRODUCTION

Un système RAP est un système capable de reconnaître une forme inconnue parmi un dictionnaire de références. N'importe quel système de reconnaissance de formes comprend deux phases, la phase d'apprentissage pour créer le dictionnaire de références et la phase de reconnaissance à base d'un classifieur pour reconnaitre la forme en question. Ces deux phases en RAP ont des parties en commun comme le prétraitement de signal de la parole et l'extraction des paramètres. La première partie de ce travail est dédiée à une présentation générale de notre système implémenté sur DSP. Le positionnement du problème qui nous pousse à utiliser le circuit FPGA est présenté dans la deuxième partie. La troisième partie est consacrée aux accélérateurs matériels LMS [1], [2] et DTW CORES et finalement la conclusion et les perspectives sont données dans la dernière partie.

2. VUE GENERALE DU SYSTEME

Les systèmes de reconnaissance de formes se basent sur une ou plusieurs approches. Les deux approches suivantes peuvent être utilisées pour la reconnaissance automatique de la parole. L'approche analytique (parole continue) : on essaye de reconnaitre les composantes élémentaires de la parole qui sont les syllabes/phonèmes. Cette approche permet de traiter du gros vocabulaire. L'approche globale (mot isolé) : l'unité de base est le mot lui-même. Le mot est considéré comme une entité indivisible, une petite phrase de très courte durée peut aussi être considérée comme une seule unité. Dans notre implémentation, on a implémenté l'approche globale qui peut être utilisée pour commander par exemple un robot AVG, un bras manipulateur, une chaise roulante d'un handicapé … etc. les deux figures suivantes présentent les deux phases de notre système sur SDK6711 (Fig. 1 et Fig. 2).

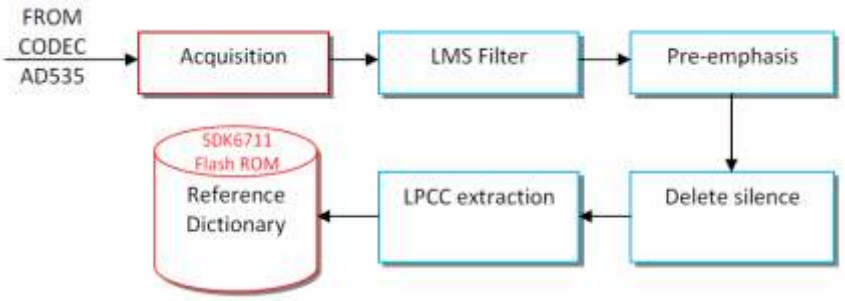

Figure 1. Phase d'apprentissage

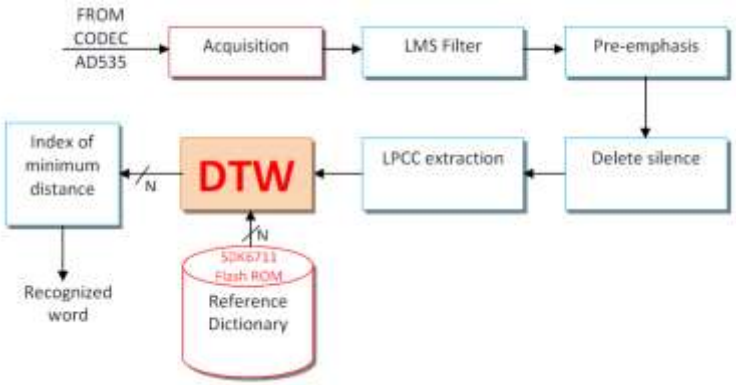

Figure 2. Phase de reconnaissance

3. FILTRE LMS

La terminologie de (LEAST MEAN SQUARE), algorithme inventé par Widrow-Hoff en 1960 pour l'apprentissage de réseaux de neurones [3] (mise à jour des poids synaptiques). Ce filtre est appareillé dans la classe des filtres adaptatifs et à cause de la puissance de LMS contre les bruits de fonds, on l'a utilisé pour filtrer le bruit introduit par l'environnement [4]. Ce filtre est composé de deux parties, une pour calculer la sortie du filtre et l'autre pour faire la mise à jour des coefficients du filtre. La première partie est un simple filtre FIR donnée par l'équation suivante (1) :

©UBMA 2020

$$y(n) = \sum_{k=0}^{N-1} c(k) \times x(n-k) \quad (1)$$

y : signal de sortie

x : signal d'entrée

c : coefficients du filtre FIR

N : nombre de coefficients du filtre FIR

La deuxième partie présente la mise à jour des coefficients par les équations suivantes (2) :

$$c(n) = c(n-1) + \mu \times err \times x(n-1) \quad (2)$$

$$err = y(n-1) - d(n-1)$$

c : coefficients du filtre FIR

err : erreur entre la sortie du filtre et la valeur désirée

x : signal d'entrée

y : signal de sortie

d : signal désiré

μ : facteur de convergence

Les figures suivantes présentent respectivement le schéma synoptique général du filtre LMS et le filtrage d'un signal sinusoïdal noyé dans un bruit AWGN sous Model-Sim (Fig.3 et Fig.4).

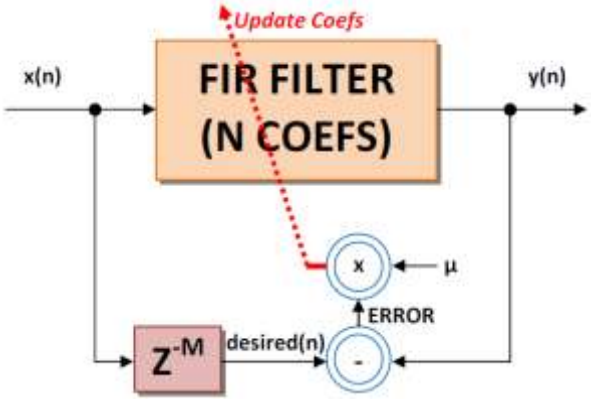

Figure 3. Synoptique du filtre LMS

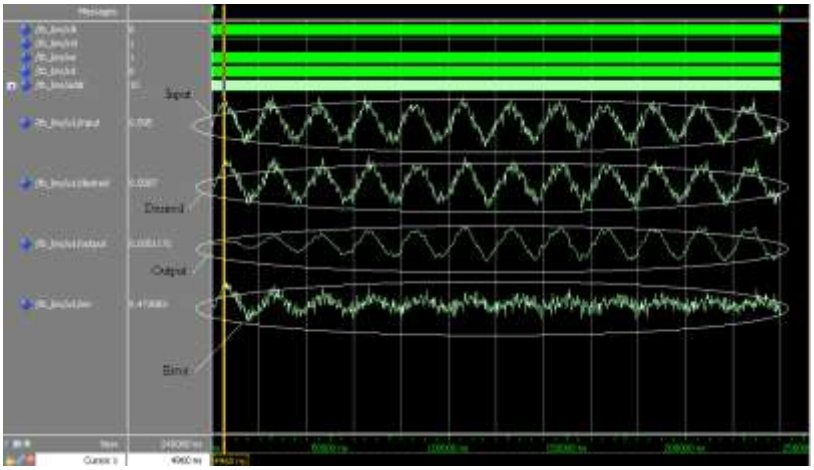

Figure 4. Graphes de simulation de LMS sous Model-Sim

©UBMA 2020

4. PREACCENTUATION

On peut considérer le système d'acquisition (AD535 de TI) comme un filtre passe bas qui fait l'atténuation des hautes fréquences du signal de la parole [5], et dans pas mal de cas l'information utile discriminante existe dans les hautes fréquences. Donc pour lutter contre cet effet on passe notre signal par un filtre FIR égaliseur de spectre qui s'appelle filtre de préaccentuation de la forme suivante (3) dans le plan Z [6] (Fig5).

$$\begin{cases} H(z) = 1 - \alpha \times Z^{-1} \\ \alpha \leq 0.95 \end{cases} \quad (3)$$

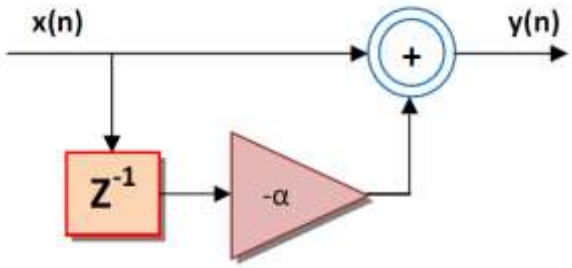

Figure 5. Filtre de préaccentuation

5. ELIMINATION DE SILENCE

Il existe un grand nombre de variantes d'algorithmes de la détection de début/fin d'un mot (VAD : Voice Activity Detection) [7]. Parmi ces algorithmes, on a utilisé l'algorithme Energy Based VAD (4). Ce dernier se base sur un seuillage par le calcul de logarithme compressé de l'écart type d'une trame du signal de la parole ne dépassant pas les 10 ms [8].

$$\begin{cases} S_i = 10 \times \log\left(\sqrt{\dfrac{\sum_j \left(x_j - \overline{x_i}\right)^2}{N-1}} \right) \end{cases} \quad (4)$$

S_i : logarithme compressé de la trame i

x_j : échantillon j de la trame i

x_i : valeur moyenne de la trame i

N : nombre des échantillons dans la trame i

Cet algorithme nécessite après décision un filtre à moyenne mobile pour éliminer les glitches créés par des événements fugitifs dans le signal de la parole (Fig6).

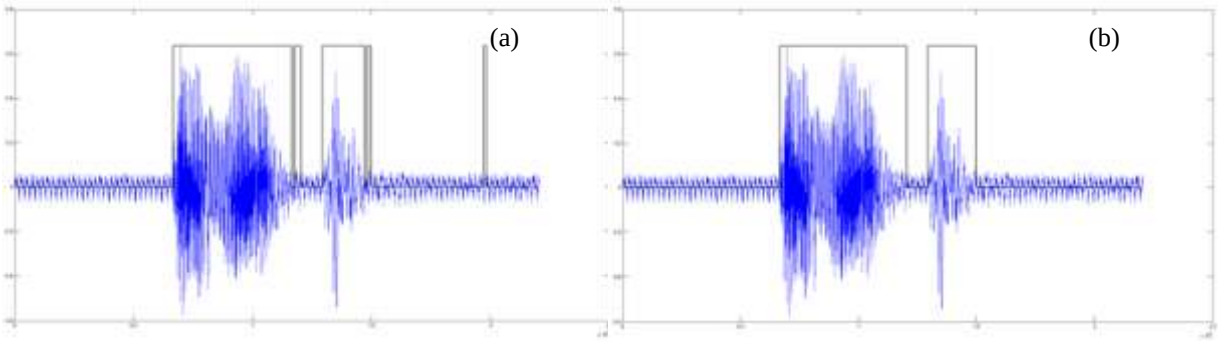

Figure 6. (a) avant le filtre à moyenne mobile et (b) après filtrage des glitches

6. EXTRACTION DES PARAMETRES

Un module de paramétrisation du signal est un module de prétraitement acoustique basé sur le calcul des coefficients. Son rôle est de fournir et extraire des informations caractéristiques et pertinentes du signal pour produire une présentation moins redondante du signal brut [9]. Ses paramètres sont calculés sur un bloc d'échantillons de taille fixe. Il correspond à un temps de parole autour de 30 ms. La suite de vecteur d'analyse est obtenue en déplaçant ce bloc d'un temps autour de 20 ms (présence d'un recouvrement), ce qui apparente à

©UBMA 2020

une analyse de type fenêtre glissante (Fig. 7). Le choix des paramètres est particulièrement important, car il s'agit de la base du système RAP. Les meilleurs paramètres devraient respecter les conditions suivantes **:**

- Grande variation entre les différents mots de dictionnaire de références.
- Faible variation pour le même mot.
- Robustesse aux effets du medium (distorsion, bruit,… etc.).
- Facile à calculer (simplicité de l'implémentation).

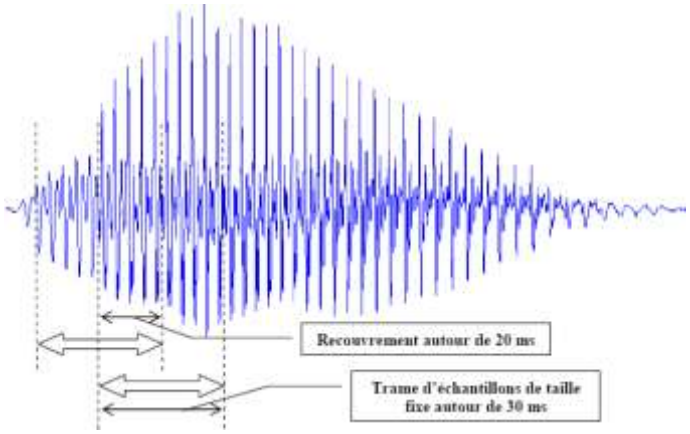

Figure 7. Segmentation de signal de la parole par des trames de taille fixe avec recouvrement

Ces conditions sont idéales, il existe toujours un compromis entre ces conditions, donc le choix d'un tel paramètre se fait également selon le cahier de charges du système (taille physique, endroit d'exploitation, plateforme de développement…etc.). Les paramètres acoustiques couramment utilisés sont les coefficients cepstraux [10]. L'analyse cepstrale résulte du modèle de production, son but est d'effectuer la déconvolution « source/conduit ». Les deux familles de coefficients cepstraux les plus utilisés en RAP sont issues de deux analyses différentes pour obtenir le spectre. Lorsque le spectre d'amplitude résulte d'une FFT sur le signal de parole prétraité, lissé par une suite de filtres triangulaires répartis selon l'échelle de MEL, ses coefficients sont appelés MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) [10] (Fig. 8). Lorsque le spectre correspond à une analyse LPC, les coefficients se déduisent des coefficients LPC par un développement de Taylor, d'où leur nom est LINEAR PREDICTION CEPSTRAL COEFFICIENTS (LPCC) (Fig. 9).

Figure 8. Extraction des coefficients MFCC

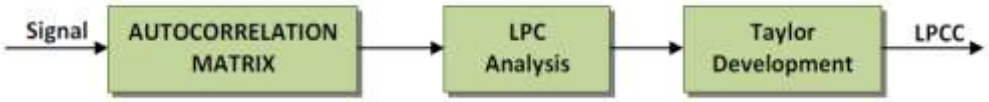

Figure 9. Extraction des coefficients LPCC

A cause de la complexité du calcul des coefficients MFCC, on a choisi la voie LPCC pour implémenter sur DSP (le développement de Taylor est remplacé par la récursivité de Levinson-Durbin).

7. CLASSIFIEUR DTW

Après les opérations de prétraitement et l'extraction des paramètres, il reste à utiliser un classifieur capable d'identifier un mot inconnu [11]. Les premiers classifieurs de calcul de distance considèrent que les matrices paramètres du mot inconnu et la référence sont alignées, donc on fait toujours une troncature ou une insertion dans les matrices paramètres pour les rendre alignées. L'alignement temporel linéaire est insuffisant dans le cas de fortes distorsions entre les événements vocaux de deux prononciations différentes d'un mot donné. C'est pourquoi, on utilise un alignement temporel non linéaire (programmation dynamique) afin d'adapter au mieux l'algorithme de RAP. L'algorithme DTW (Dynamic Time Warping) tient compte de ce phénomène [12]. Le principe de base est d'essayer de trouver le chemin optimal à parcourir parmi l'ensemble des distances entres les

©UBMA 2020

vecteurs acoustiques. Le point de départ de cet algorithme est le théorème d'optimalité, ce théorème peut être énoncé comme suit : « tout chemin optimal est constitué de sous chemin optimaux ». Tout ce qui nous intéresse dans ce travail est d'implanter un accélérateur matériel de DTW. Le calcul de cette distance entre deux séries (A & B) prend quelques étapes sont (5) (Fig.10)

Condition initiale : $g(1,1) = 2 \times d(1,1)$

Calcul de la distance cumulée par la contrainte symétrique de Sakoe & Shiba

$$g(i,j) = \min \begin{pmatrix} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2 \times d(i, j) \\ g(i-1, j) + d(i, j) \end{pmatrix} \quad (5)$$

Largeur de bande : $j - r \le i \le j + r$

Normalisation de la distance finale : $DTW(A,B) = g(n,m) / C$

Avec $C = n + m$

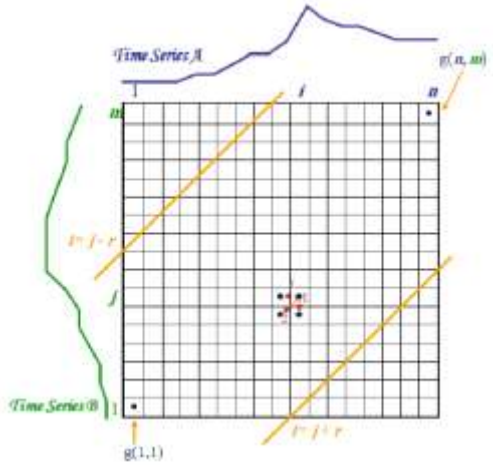

Figure 10. DTW entre les séries A et B

8. IMPLEMENTATION SUR LE KIT SDK6711

Les deux phases dans les figures 1 et 2 sont implémentées sur le KIT SDK6711 de Texas Instrument (TI) [13].

8.1 Présentation de SDK6711

Le KIT SDK6711 est équipée par :

- CPU DSP TMS320C6711 cadencé par une fréquence de 150 MHz.
- Mémoire externe SDRAM de 16 MB.
- Mémoire externe Flash ROM de 128 KB.
- 3 LEDS & DIPSWITCH à 4 positions.
- 2 connecteurs d'extension.
- CODEC audio (ADC/DAC) le AD535 16 bits.
- Interface de debug/Load sur le port parallèle du PC-IBM.

L'environnement logiciel est le CODE COMPOSER STUDIO (CCS ver : 3.00) pour un développement en C/C++ & assembleur. Le DSP TMS320C6711 est un CPU d'architecture VLIW de 6 unités ALU, 2 multiplieurs sur deux chemins de données A et B (32 registres de 32 bits) et la possibilité de faire des calculs en virgule flottante simple et double précision en format IEEE-754. Une puissance de calcul théorique arrive à 900 MIPS (cette puissance de calcul reste théorique vu que la limite d'exploiter toutes les unités en même temps avec des ressources différentes).

8.2 Positionnement du problème

Le calcul en virgule flottante prolonge le pipeline de TMS320C6711 en dix étages au lieu de six, on ne peut pas faire plus de deux accès mémoire et deux multiplications par paquet de 8 instructions, le contrôle des aléas de pipeline se fait par le programmeur, le développement en assembleur par paquet de 8 instructions nécessite un effort immense [14] qui nous pousse vers la programmation en série classique ou de développer en C/C++. Cette dernière rend la puissance de calcul de notre DSP ne dépassant pas les 90 MIPS. Toutes ses contraintes rendent le TMS320C6711 incapable de faire l'acquisition et le filtrage LMS en temps réel si le nombre de coefficients du

©UBMA 2020

filtre dépasse les huit coefficients. En plus, le calcul de la distance DTW entre des matrices de grande taille rend la décision en termes du temps visible à l'utilisateur (temps de calcul important). Donc le développement des algorithmes qui fonctionnent en temps réel, nécessite un µP puissant en calcul (consommation électrique importante, échauffement inutile, coups très élevé…etc.). Heureusement, à cause du développement technologique des circuits FPGA en termes espace programmable, vitesse de déroulement, l'existence des DSP CORES physiques sur la puce de circuit FPGA, donnent aux développeurs une puissance d'implantation des algorithmes et des architectures complexes par un minimum de ressources matérielles [15]. L'union entre un circuit DSP et un autre FPGA donne une puissance de calcul en temps réel importante qui nous poussent à réaliser un système de RAP rapide sur un dictionnaire de références de taille importante par rapport au même système implémenté sur DSP seulement.

9. IMPLANTATION MATERIELLES DE LMS ET DTW

Avant de passer vers l'implantation de LMS et DTW CORES sur le circuit FPGA chez Xilinx Virtex5, notre LMS et DTW sont connectés aux DSP à travers l'interface mémoire EMIF [16], [17], [18]. Cette interface mémoire est capable de travailler en deux modes, le mode synchrone (S-Mode) pour interfacer des mémoires dynamiques (DRAM), et le mode asynchrone (A-Mode) pour interfacer des mémoires statiques (SRAM). Les figures suivantes présentent les chronogrammes de l'interface EMIF en mode asynchrone pour la lecture et l'écriture (Fig.11).

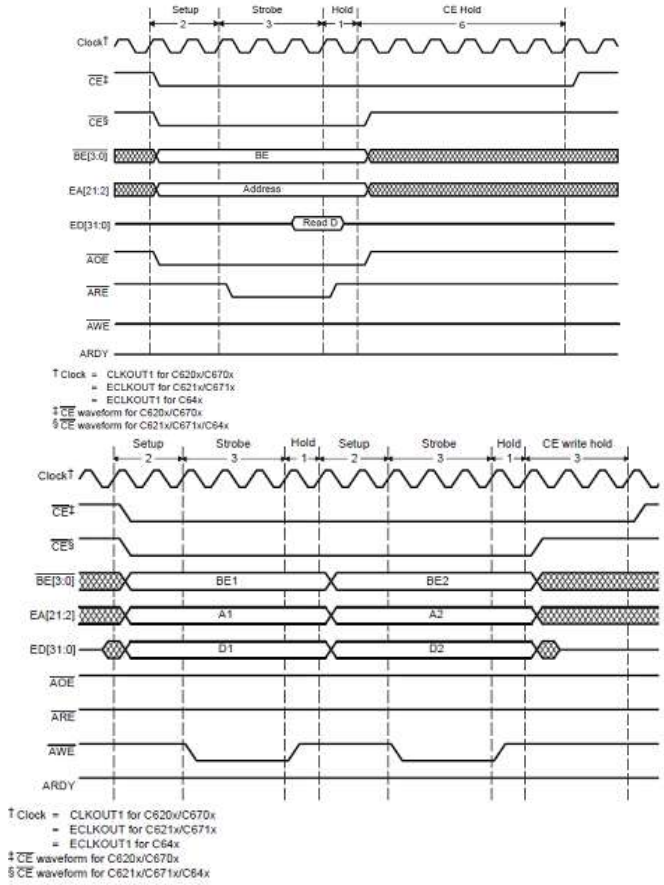

Figure 11. (a) EMIF en lecture (A-Mode) et (b) EMIF en écriture (A-Mode)

L'opération de lecture prend 12 tops d'horloge sur 4 phases [SETUP -2 tops-, STROBE -3 tops-, HOLD -1 top- et CE READ HOLD -6 tops-]. L'opération d'écriture prend 9 tops d'horloge sur 4 phases [SETUP -2 tops-, STROBE -3 tops-, HOLD -1 top- et CE WRITE HOLD -3 tops-]. La fréquence max donnée par le fabricant TI est 100 MHz programmable par soft en manipulant les registres de l'interface EMIF. La figure suivante (Fig. 12) présente la liaison entre le SDK6711 et la carte FPGA Vertex5 de Xilinx à travers l'interface mémoire EMIF.

©UBMA 2020

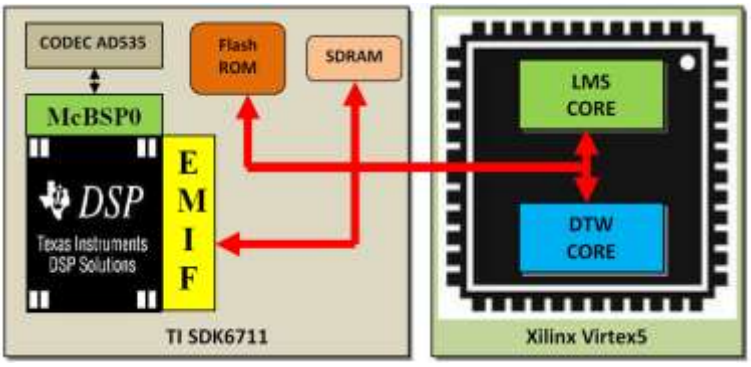

Figure 12. Station hybride DSP/FPGA

9.1 Implantation de LMS

L'implantation des deux CORES se base sur une FSMD (Finite State Machine with Data-path) machine à états finis avec un chemin de données. Le chemin de données du filtre LMS est composé d'une partie qui fait le calcul de la sortie du filtre par un FIR et une partie concernant la mise à jour des coefficients plus un buffer de M registres à décalage pour faire une copie de bruit du fond au début de l'opération du filtrage LMS. La figure suivante présente la synoptique générale du CORE (Fig. 13).

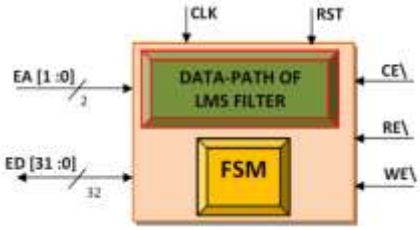

Figure 13. Synoptique générale de LMS CORE

Le tableau 1 résume la description des lignes du CORE

Tableau 1 : description des lignes du CORE

Ligne	**Description**
CE\	**C**HIP **E**NABLE active niveau bas
RE\	**R**EAD **E**NABLE active niveau bas
WE\	**W**RITE **E**NABLE active niveau bas
RST	Demande de **RESET** (initialisation) externe
CLK	Signal d'horloge **CLOCK** (fréquence max de 100 MHz)
EA [1 :0]	2 bits d'adresse pour sélectionner un registre parmi quatre
ED [31 :0]	Bus de données 32 bits bidirectionnel

Légende : EA = **E**xternal **A**ddress bus ED = **E**xternal **D**ata bus MHz = **M**ega **H**ert**z**.

A l'intérieur du CORE, on a quatre registres pour manipuler l'opération de filtrage LMS. Le tableau 2 résume la description de ses registres.

Tableau 2 : description des registres.

Adresse	**Registre**	**Description**	**Accès**
00 (0x0)	INPUT_LMS	Registre 32 bits compatible IEEE-754 présente l'entrée du filtre LMS	R/W
01 (0x1)	MU_REG	Registre 32 bits compatible IEEE-754 présente le facteur de convergence du filtre LMS	R/W
10 (0x2)	CSR_REG	Registre à 2 bits pour contrôler le filtre LMS	R/W
11 (0x3)	OUTPUT_LMS	Registre 32 bits compatible IEEE-754 présente la sortie du filtre LMS	R

©UBMA 2020

Légende : R = **R**eadable bits (bits en lecture) W = **W**ritable bits (bits en écriture) R/W = **R**e**W**ritable bits (bits en lecture/écriture).

L'écriture dans le registre INPUT_LMS provoque un décalage automatique d'une position dans les buffers FIR et valeur désirée. Les deux bits de registre CSR_REG sont le bit START et le bit CLR, l'écriture d'un "1" dans le bit START déclenche le calcul de la sortie de FIR puis la mise à jour des coefficients (ce bit revient à zéro automatiquement à la fin du calcul). L'écriture d'un "1" dans le bit CLR fait l'initialisation du chemin de données du filtre (ce bit revient à zéro automatiquement à la fin de l'initialisation).

9.1.1 Architecture interne (DATAPATH)

A ce stade, on a exploité le parallélisme pour implanter la partie FIR du LMS. Cette architecture fait diminuer les niveaux de latence (délais de propagation), si le nombre de coefficients est égal à huit donc, on a quatre niveaux et pour seize coefficients on a cinq niveaux seulement. Le chemin de données de la mise à jour des coefficients se base sur des unités MAC (Multiplication/Accumulation) par coefficient. Le nombre de niveaux de latence est toujours égal à trois quelque soit le nombre de coefficients comme indique la figure suivante qui présente l'architecture interne du CORE (un LMS de huit coefficients) (Fig. 14).

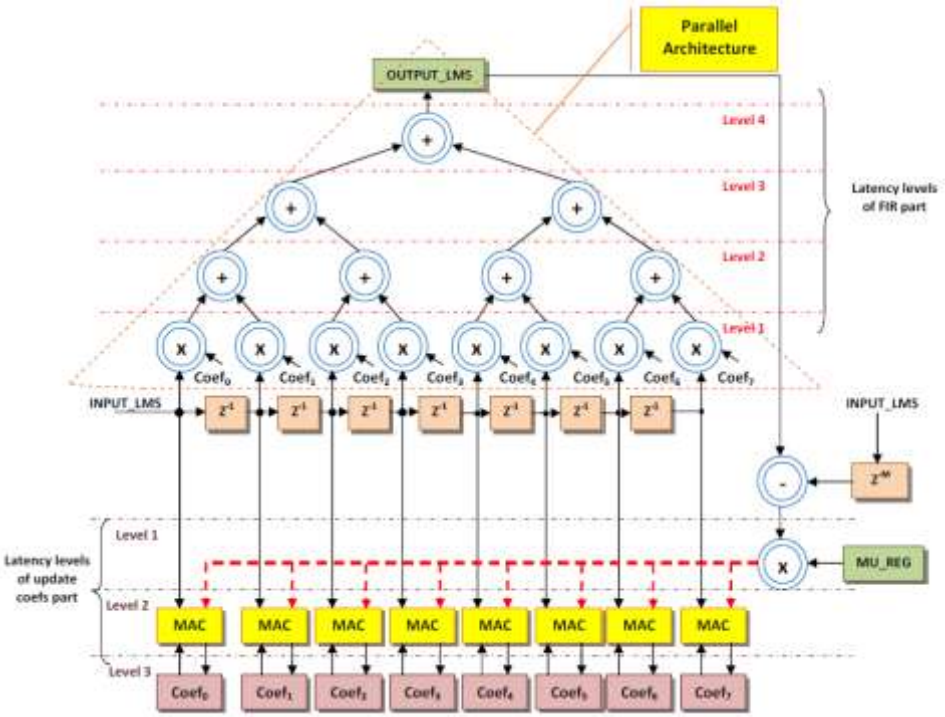

Figure 14. Architecture interne de LMS CORE à huit coefficients

9.1.2 Le séquenceur (FSM)

Le schéma dans la figure 15 nécessite une machine à états finis pour cadencer les deux opérations calcul FIR et mise à jour des coefficients. Cette FSM utilise l'horloge de l'interface EMIF de DSP (100 Mhz) et d'autre part respecte les niveaux de latence de chaque partie. Après synthèse de l'architecture interne du CORE, le logiciel ISE donne une fréquence maximale égale à 12.649 MHz sur seize coefficients. Cette fréquence nous donne le nombre de tops d'horloge nécessaire pour voir un état stable devant les registres coefficients du filtre ($Coef_i$) et le registre de sortie du filtre LMS (OUTPUT_LMS), donc le nombre d'états dans la FSM. Les cinq niveaux de latence de FIR sur seize coefficients nécessitent huit tops d'horloge (huit états), et trois niveaux de latence de mise à jour des coefficients nécessitent six tops d'horloge (six états). Le graphe d'état suivant présente la FSM du CORE (Fig. 15).

©UBMA 2020

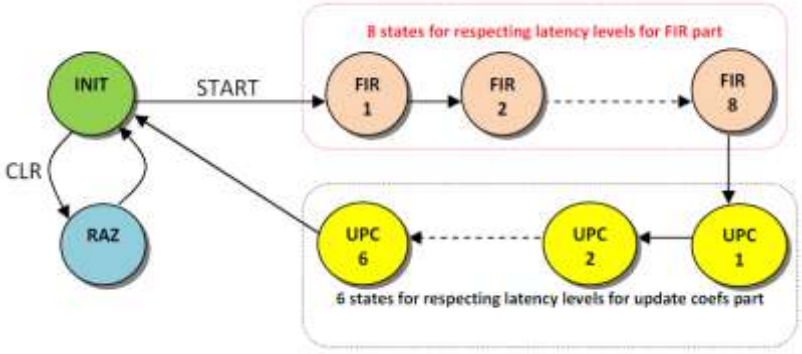

Figure 15. Graphe d'état de séquenceur de LMS

Le tableau 3 résume la description de chaque état de la FSM.

Tableau 3 : description de chaque état de la FSM

Etat	Description
INIT	Etat de repos après RESET matériel
RAZ	Etat d'initialisation après une demande par le bit CLR dans le registre CSR_REG
FIR1 :FIR7	Etats d'attente pour voir une valeur stable devant le registre OUTPUT_REG
FIR8	Etat de mémorisation de la sortie de FIR dans le registre OUTPUT_REG
UPC1 :UPC5	Etats d'attente pour voir une valeur stable devant les registres $Coef_i$
UPC6	Etat de mémorisation de la sortie mise à jour des Coefs dans les registres $Coef_i$

Légende : $Coef_i$ = le registre de l'ième coefficient du filtre LMS CSR= Control Status Register.

9.1.3 Pseudo-code d'exploitation DSP-LMS

Le pseudo-code suivant utilise le codec AD535 et le LMS CORE pour faire l'acquisition et le filtrage en temps réel durant une seconde par le DSP TMS320C6711. La fréquence d'échantillonnage du codec est fixé par le constructeur TI, elle est égale à 8 KHz donc 8000 échantillons par seconde (Fig16).

```
#define START 0x01
#define CLR 0x02
#define MU_VAL 0.00001

// Init LMS CORE
CSR_REG ← CLR;
// LOAD CONVERGENCE FACTOR TO MU_REG
MU_REG ← MU_VAL;
// ACQUISITION WITH LMS FILTRING FOR 1 SECOND
FOR k=0:1:8000-1
      X ← READ_AD535 (); // X is 16 bits integer value
      Y ← NORMALIZE(X); // Y is float value between +-1
      INPUT_LMS ← Y; // load simple in LMS CORE
      CSR_REG ← START; // start LMS CORE
      CSR_REG ← START; // 15 clock ticks for done operation
      BUFF[k] ← OUTPUT_LMS; // read result
END FOR
```

Figure 16. Pseudo-code d'exploitation de LMS CORE par le DSP

9.1.4 Rapport d'implantation sur FPGA

Le tableau 4 présente le rapport d'implantation de LMS CORE sur la Virtex5 XC5VLX330T-2FF1738 (Tab. 4)

©UBMA 2020

Tableau 4 : rapport d'implantation de LMS CORE sur la Virtex5 XC5VLX330T-2FF1738

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1,205	207,360	1%
Number of Slice LUTs	28,005	207,360	13%
Number of bonded IOBs	39	960	4%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of DSP48Es	66	192	34%

Légende : LUT = Look Up Table IOB = Input Output Bonded
BUFG/BUFGCTRL = Buffer Gate Control DSP48E = Digital Signal Processeor 48 bits.

9.2 Implantation de DTW

Comme cité précédemment les deux CORES utilisent une FSMD (Finate State Machine with Data-path). D'après les équations (5), le calcul de la distance DTW se base sur le calcul des distances cumulées par le calcul des distances locales et la contrainte symétrique de H.Sakoe et S.Chiba [12]. Dans la littérature, il existe pas mal de formules pour calculer la distance locale entre les vecteurs des deux matrices paramètres, et la plus utiliser est la distance euclidienne. Mais dans notre travail, on a choisi la distance City Block par ce qu'elle est la plus rapide et la plus simple à implanter sur FPGA. La contrainte de H.Sakoe et S.Chiba est la contrainte la plus adapté au modèle de la parole par ce qu'elle respecte le déplacement d'un segment à sa position d'origine soit en avant ou en arrière [12]. La figure suivante présente la synoptique générale du CORE (Fig17)

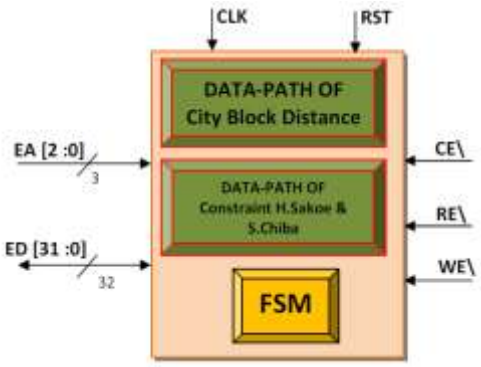

Figure 17. Synoptique générale de DTW CORE

Les mêmes signaux de communication que le LMS CORE à l'exception de bus d'adresse, vu que le DTW CORE manipule 8 registres internes comme l'indique le tableau 5.

Tableau 5 : signaux de communication

Adresse	**Registre**	**Description**	**Accès**
000 (0x0)	**R1_REG**	Registre 32 bits compatible IEEE-754 présente le premier membre de la distance City Block	R/W
001 (0x1)	**R2_REG**	Registre 32 bits compatible IEEE-754 présente le deuxième membre de la distance City Block	R/W
010 (0x2)	**ACC_CB (d(i,j))**	Registre 32 bits compatible IEEE-754 présente l'accumulateur de la distance City Block	R/W
011 (0x3)	**CSR_REG**	Registre à 1 bit pour contrôler le DTW CORE	R/W
100 (0x4)	**X_REG (g(i-1,j-1))**	Quatre registres 32 bits compatible IEEE-754 présentent les membres de calcul de la contrainte de H.Sakoe & S.Chiba	R/W
101 (0x5)	**Y_REG (g(i-1,j))**		R/W
110 (0x6)	**Z_REG (g(i,j-1))**		R/W
111 (0x7)	**DC_REG (g(i,j))**		R/W

*Légende : R/W = **ReW**ritable bits (bits en lecture/écriture).*

©UBMA 2020

Le registre CSR_REG contient seulement un bit qui s'appelle START. L'écriture d'un "1" dans ce bit provoque le calcul de la contrainte (ce bit revient à zéro automatiquement à la fin du calcul).

9.2.1 Architecture interne (DATAPATH)

La distance City Block entre deux vecteurs se fait par le cumul de la valeur absolue de la différence élémentaire des deux vecteurs comme l'indique la formule suivante (6) [12] :

$$D(V_1,V_2)=\sum_k |V_1(k)-V_2(k)| \quad (6)$$

Le chemin de données de la figure 18 accélère un petit peu le calcul par rapport au DSP.

Après la synthèse de ce chemin, ISE donne une fréquence maximale égale à 64.593 MHz, donc 2 tops d'horloge sont largement suffisants pour voir un état stable devant le registre ACC_CB.

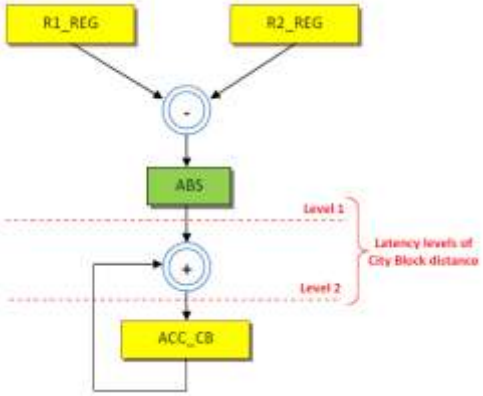

Figure 18. Chemin de données de la distance City Block

La contrainte de H.Sakoe et S.Chiba est une opération entre quatre éléments d'une matrice 2x2 glissante d'une position sur les colonnes puis sur les lignes de la matrice des distances cumulées (g). Le glissement de matrice 2x2 donne la possibilité de charger le registre X_REG par le contenu du registre Y_REG et le registre Z_REG par le contenu du registre DC_REG pour la prochaine distance cumulée après le calcul de la distance cumulée en cours. Plus l'implantation parallèle de la contrainte elle-même qui accélère considérablement le remplissage de la matrice des distances cumulées. La figure 19 présente le chemin de données de la contrainte.

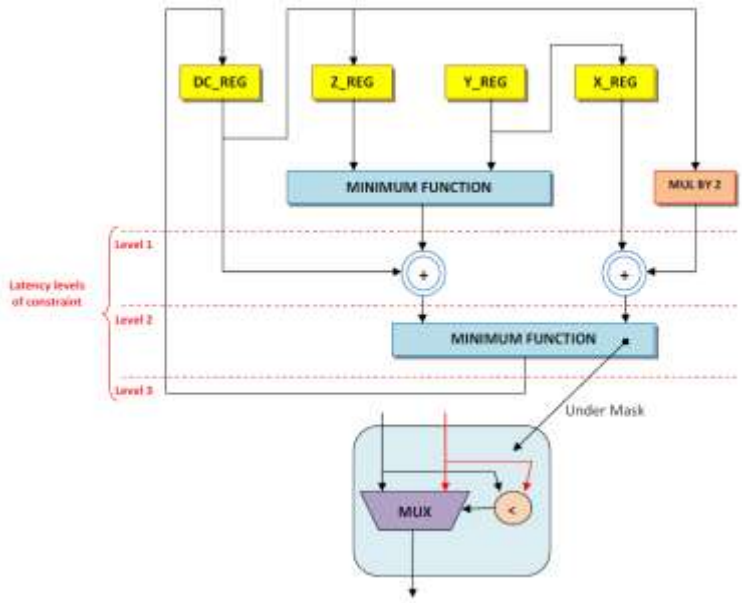

Figure 19. Chemin de données de la contrainte

©UBMA 2020

La synthèse du chemin de données de la contrainte donne une fréquence maximale égale à 45.858 MHz donc trois tops d'horloge pour la contrainte et un top pour le glissement.

9.2.2 Le séquenceur (FSM)

Le séquenceur est une simple FSM à huit états, quatre états pour la contrainte et trois états pour le calcul de la distance City Block (on a ajouté un état supplémentaire pour rendre le nombre d'états multiple de deux) et un autre état présente l'état de repos du séquenceur. Le déclanchement du calcul de la distance City Block se fait automatiquement après l'écriture dans le registre R2_REG. Mais le démarrage du calcul de la contrainte se fait par écriture d'un « 1 » dans le bit START du registre CSR_REG comme l'indique le graph d'état de la figure20.

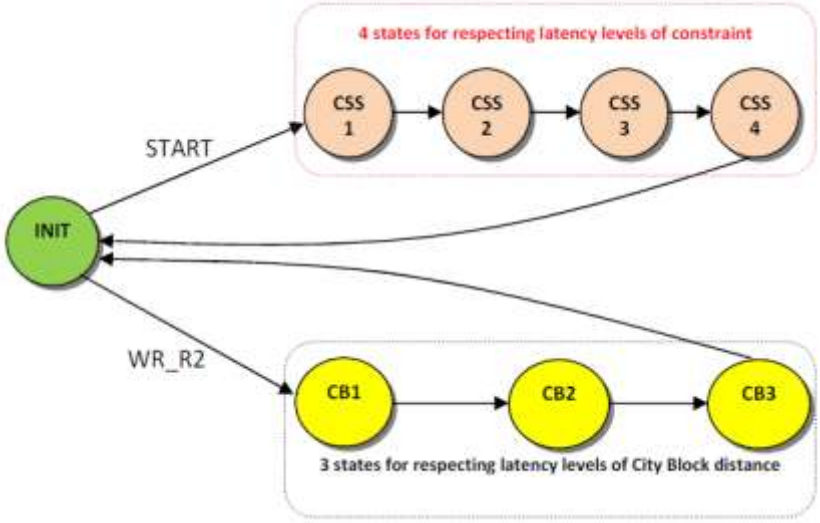

Figure 20. Graphe d'état de séquenceur de DTW CORE

Le tableau 6 résume la description de chaque état de la FSM.

Tableau 6 : description de chaque état de la FSM.

Etat	Description
INIT	Etat de repos après RESET matériel
CSS1 :CSS2	Etats d'attente pour voir une valeur stable devant le registre DC_REG
CSS3	Etat de mémorisation de la contrainte dans le registre DC_REG
CSS4	Etat de glissement (X_REG ← Y_REG & Z_REG← DC_REG)
CB1 :CB2	Etats d'attente pour voir une valeur stable devant le registre ACC_CB
CB3	Etat de mémorisation de la somme dans le registre ACC_CB

Légende : ACC= Accumulateur ; CB= City BlockREG= Registre ; DC= Distance cumulée.

9.2.3 Pseudo-code d'exploitation DSP-DTW

Le pseudo-code de la figure 21 fait le calcul de la distance DTW entre deux matrices A_{10xN} et B_{10xM} (10 coefficients LPCC par vecteur acoustique).

9.2.4 Rapport d'implantation sur FPGA

Le tableau 7 présente le rapport d'implantation de DTW CORE sur la Virtex5 XC5VLX330T-2FF1738.

©UBMA 2020

```
#define START 0x01
#define N 50 // number of vectors in A matrix
#define M 60 // number of vectors in B matrix
#define Q 10 // number of coefs per vector

// Init X_REG & Z_REG
X_REG ← 0;
Z_REG ← REAL_MAX;
For i=0:1:N-1
      For j=0:1:M-1
            // compute City Block distance
            ACC_CB ← 0;
            For k=0:Q-1
                  R1_REG ← A[k,i];
                  R2_REG ← B[k,j]; // start city block distance
            End For
            // compute constraint of H.Sakoe & S.Chiba
            DC_REG ← ACC_CB; // load local distance
            Y_REG ← REAL_MAX;
            If (i-1)> 0 then Y_REG ← g[i-1,j]; End If
            CSR REG ← START; // start constraint
            g[i,j] ← DC_REG; // read result
      End For
      // prepare for next line
      X_REG ← g[i,0];
      Z_REG ← g[i+1,0];
End For
// Normalization of DTW distance
DTW = g[N-1,M-1]/(N+M);
```

Figure 21. Pseudo-code d'exploitation de DTW CORE par le DSP

Tableau 7 : rapport d'implantation de DTW CORE sur la Virtex5 XC5VLX330T-2FF1738.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	231	207,360	1%
Number of Slice LUTs	3,502	207,360	1%
Number of bonded IOBs	40	960	4%
Number of BUFG/BUFGCTRLs	3	32	9%
Number of DSP48Es	0	192	0%

Légende : LUT = Look Up Table IOB = Input Output Bonded
BUFG/BUFGCTRL = Buffer Gate Control DSP48E = Digital Signal Processeor 48 bits.

10. COMPARAISON

Le tableau 8 présente une comparaison en termes de temps écoulé pour reconnaître un mot dans une base de référence de 20 mots entre le DSP seul et la plateforme hybride DSP/FPGA sans compter le temps d'acquisitions.

Tableau 8 : comparaison en termes de temps écoulé

Plateforme	**Temps écoulé**
DSP	Environ de 3 secondes
DSP/FPGA	Environ de 0.1 secondes

Légende : DSP= Digital Signal Processor
FPGA : Field Programmable Gate Array

CONCLUSION ET PERSPECTIVES

Dans cet article, il a été présenté l'implémentation de l'approche globale de la reconnaissance de la parole par DTW sur deux plateformes (DSP seul et DSP/FPGA) afin de faire une comparaison entre les deux sur l'axe vitesse d'exécution. L'hybridation DSP/FPGA donne à la fois la souplesse d'implémenter en soft d'un

©UBMA 2020

algorithme sur DSP et d'autre part d'invoquer le parallélisme matériel (les accélérateurs matériels) par l'implantation sur FPGA. Le but principal de ce travail est d'implanter des accélérateurs matériels qui améliorent le temps de réponse du système et augmentent la taille du dictionnaire de références. Les résultats obtenus sont très encourageant puisque l'approche hybride DSP/FPGA est 33 fois plus rapide que le classique approche à base de DSP. L'apparition de la nouvelle gamme des circuits FPGA de l'entreprise Xilinx la Virtex-UltraScale+ a une gravure inferieur à 20nm avec des DSPCOREs fonctionnant en virgule fixe et en virgule flottante selon le besoin, pousse l'implantation des accélérateurs matériels vers un horizon de meilleurs performances en terme de vitesse d'exécution et optimisation architecturale. Ce travail peut être amélioré par l'implantation d'un accélérateur pour l'extraction des paramètres LPCC dans le circuit FPGA. Une deuxième piste d'amélioration de la discrimination son/parole pourrait être l'utilisation d'autres types de paramètres que les paramètres classiques MFCC. Les résultats obtenus nous montrent qu'une application peut être facilement programmée en C, et ensuite automatiquement convertie en VHDL et implémentée sur un FPGA d'une manière optimale.

REFERENCES

[1] A. Aldahoud, H. Atoui and M. Fezari, 2013. Robust Automatic Speech recognition System Implemented in a Hybrid Design DSP-FPGA, *Intern. J. of Signal Processing, Image Processing and Pattern Recognition*, Vol. 6 (5), 333-342.

[2] M. Ferazi, H. Atoui and I. M. M. El Emary, 2012. Hybrid Design bades on DSP/FPGA for Real-time Adaptive Filter in Robust Voice Command System. *ICESTI12 Annaba Algeria*.

[3] B. Widrow and S.D.Stearns, 1985. Adaptive Signal Processing. Pearson, 496 p.

[4] J. Petrone, 2004. Adaptive Filter Architectures for FPGA Implementation. Florida University, USA, 97 p.

[5] Texas Instrument Corp, 2003. A DSP/BIOS AD535 Codec Device Driver for the TMS320C6x11 DSK, SPRA850A.

[6] Les Thede, 2004. Practical Analog And Digital Filter Design. Artech House Microwave Library, 270p.

[7] J. Ramirez, J. M. Gorriz and J. C. Segura, 2007. Voice Activity Detection. Fundamentals and Speech Recognition System Robustness, *Robust Speech Recognition and Understanding*, ISBN 987-3-90213-08-0, 1–22.

[8] L.R Rabiner and M.R. Sambur, 1975. An Algorithm for determining Endpoints of Isolated Utterances, *The Bell System Technical Journal*, Vol.54 (2), 297-315.

[9] X. Wang, J. Lin, Y.Sun, H. Gan and L.Yao, 2009. Applying Feature Extraction of Speech Recognition on VoIP Auditing, *International Journal of Innovative Computing, Information and Control*, vol. 5 (7), 1851-1856.

[10] Sahidullah, Md. Saha: Design, 2012. Analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication* Vol.54 (4), 543565.

[11] F. A.A. Laleye and C. Motamed E. C. Ezin, 2016. Speech phoneme classification by intelligent decision-level fusion. *Lecture Notes in Electrical Engineering*, Vol.383, 63–78.

[12] H.Sakoe and S.Chiba, 1978. Dynamic Programming Optimization For Spoken Word Recognition, *IEEE Trans, on Acoustic, Speech, ans Signal Processing*, Vol.26 (1), 43-49.

[13] Texas Instrument Corp, 2011. TMS320C6000 DSP CPU and Instruction Set Reference. SPRU189.

[14] P. Yin, 2011. Introduction to TMS320C6000 DSP Optimization. SPRABF2, Texas Instrument Corp.

[15] U. Meyer-Baese, 2014. Digital Signal Processing with Field Programmable Gate Arrays (4th Edition). Springer, 930p.

[16] Texas Instrument Corp, 2009. TMS320C6000 DSP Peripherals Overview Reference Guide. SPRU190.

[17] Tietche, B.H., Romain, O., Denby, B. & Dieuleveult, F.D., 2012. FPGA-based simultaneous multichannel fm broadcast receiver for audio indexing applications in consumer electronics scenarios, *IEEE Transactions on Consumer Electronics*, DOI: 10.1109/TCE.2012.6414980, Vol.58 (4), 1153-1161.

[18] B. Happi Tietche, O. Romain & B. Denby, 2013. Pratical FPGA-Based Architecture for Arbitrary Ratio Sample Rate Conversion, *Journal of Signal Processing Systems*, Springer, DOI 10.1007/s11265-013-0840-5.

NOMENCLATURES

RAP : Reconnaissance Automatique de la Parole.

FPGA : Field programmable Gate Array.

FSMD : Finite State Machine with Data-path.

DTW : Dynamic Time Warping.

LMS : Least Mean Square.

DSP : Digital Signal Processor.

FFT : Fast Fourier Transform.

VAD : Voice Activity Detection.

SDK : System Development Kit.

CODEC : Coder Decoder.

©UBMA 2020