MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de fin d'études

En vue d'obtention du diplôme d'ingénieur d'Etat en ELECTRONIQUE

Option : ***COMMUNICATION***

Thème

CONCEPTION ET REALISATION D'UNE PLATEFORME D'ACQUISITION RECONFIGURABLE A BASE D'UN CIRCUIT FPGA

Dirigé par :

M$^{r.}$ M.LAGHROUCHE

M$^{r.}$ N.DERGUINI

Présenté par :

M^{r}. AIT BENALI M'hand

Mr. AIT BENALI Abdelghali

Promotion : 2010-2011.

Remerciements

Arrivé au terme de ce travail nous tenant à exprimer nos plus sincères remerciements à nos promoteurs **M^r^ M. Mountassar**, **M^r^ M. Laghrouche**, **M^r^ N. Derguini** pour leur conseilles et orientations.

On remercie l'équipe ***MEMS*** du centre du développement des technologies avancées CDTA, **M^r^ M. Barchich** pour sa disponibilité au niveau du laboratoire maquette et **M^r^ Ben fdila** pour avoir mis à notre disposition le matériels du laboratoire microélectronique.

Sommaire

Chapitre II : Les circuits logiques programmables ASICs et PLD.

Chapitre III : Réalisation pratique de la plateforme d'acquisition

Liste des figures

Introduction générale :

Une chaine d'acquisition permet d'extraire des informations qui caractérisent les grandeurs physiques qui sont les paramètres agissant sur processus (température, pression, humidité…etc.).

Dans ce projet, nous nous proposant l'étude et la réalisation d'une plateforme d'acquisition et de restitution de données multi capteurs reconfigurable à base un FPGA. La fonction de ce dispositif va permettre en premier lieu de convertir huit grandeurs physiques en grandeurs électriques au moyen des cartes capteurs/conditionneurs, ensuite les signaux analogiques seront reçus par les huit entrées de l'ADC0808 dont la fonction est l'aiguillage et la conversion des signaux analogiques en un signal numérique, la donnée numérique va être ensuite envoyée au PC via le port série à travers un module UART implémenté dans un FPGA pour traitement et affichage des résultats de mesure. Dans notre cas nous réaliserons comme exemple un système qui permettra la mesure de la température, pression et le niveau sonore.

Ce travail est réalisé au sien du département d'électronique de l'université MOULOUD MAMMERI en collaboration avec le centre de développement de technologies avancées CDTA.

Le mémoire est Il est organisé en trois chapitres.

Dans le premier chapitre, des rappelles sur les capteurs sont données, à savoir les déférents types de capteurs, principe de détection, conditionneurs…etc.

Dans le deuxième chapitre on donne un aperçu général sur les différentes familles des circuits logiques ASICs et PLDs, ensuite une description générale des circuits FPGAs.

Le dernier chapitre est consacré à l'étude et la réalisation pratique des différentes parties de la chaine d'acquisition. Ce chapitre est organisé en trois parties :

Dans la première partie on donne un aperçu sur les caractéristiques et le principe de fonctionnement des différentes cartes analogiques (carte d'alimentation, conditionnement des capteurs, carte de multiplexage et conversion analogique numérique).

La deuxième partie est sur l'implémentation sur FPGA de module de et le développement de l'application de visualisation sur PC des grandeurs mesurées.

Chapitre I

I.1.Introduction :

Dans sa structure, une chaine d'acquisition doit pouvoir assurer au moyen des différents dispositifs des fonctions tels que l'extraction de l'information concernant chaque une des grandeurs physiques à connaitre et sa traduction en signal électrique au moyen de capteurs et conditionneurs, Traitement analogique du signal qui consiste en l'amplification et filtrage ,Sélection parmi des signaux disponibles du seul signal requis à l'aide de multiplexeur et la conversion du signal sous forme numérique au moyen de l'échantillonneur bloqueur et convertisseur analogique numérique .Le schéma synoptique suivant montre les déférentes parties constitutives de la chaine d'acquisition.

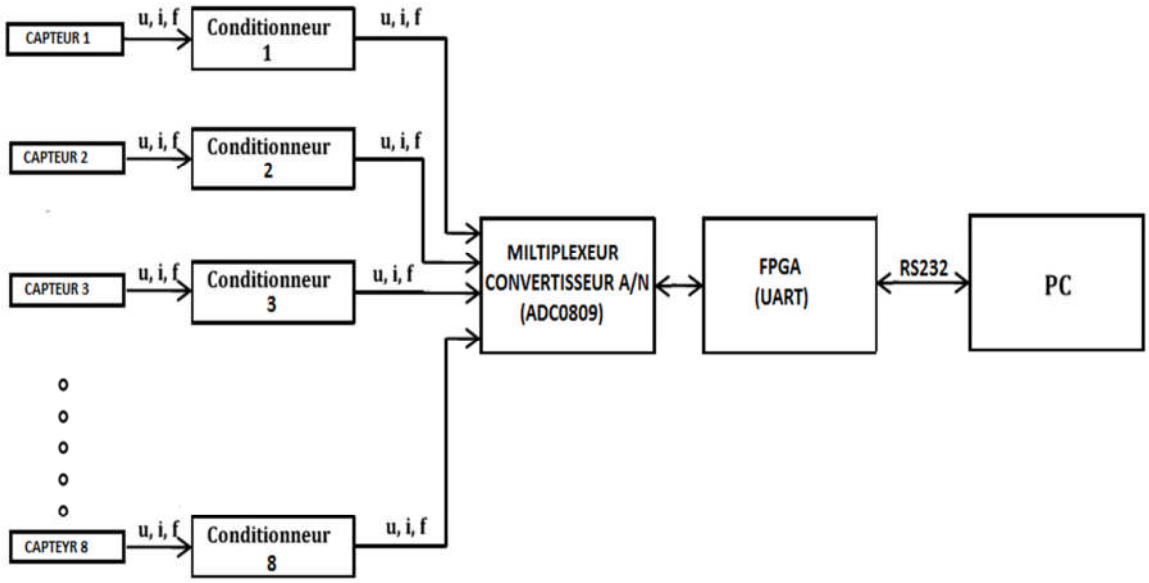


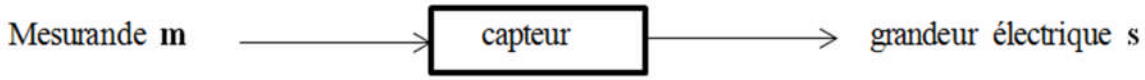
Figure I.1 : Schémas synoptique d'une chaine d'acquisition.

I.2.Les Capteurs :

I.2.1. Définition :

Le capteur est le premier élément d'une chaine de mesure, qui permet de convertir une grandeur physique **(m)** dite « mesurande » en une grandeur électrique **s(m)**, fonction connue du mesurande, telle que chaque valeur **s** de cette grandeur électrique puisse être liée de façon à la valeur **m** du mesurande.

La grandeur électrique **s (m)** est à l'origine de ce signal électrique que traite le système de mesure.



I.2.2. Classification des capteurs :

Etant un élément de circuit électrique, le capteur se présente, selon sa sortie, comme capteur actif ou capteur passif.

I.2.2.1. Capteur actif :

Comme son nom l'indique, ce capteur fonctionne en générateur. Son principe repose sur un effet physique qui assure la conversion en énergie électrique de l'énergie propre au mesurande : énergie thermique, mécanique ou de rayonnement. Les plus importants parmi ces effets sont récapitulés dans le *tableau I.1*.

Mesurande	Effet utilisé	Grandeur de sortie
Température	Thermoélectricité	Tension
Force, pression, accélération	Piézoélectricité	Charge
Vitesse	Induction électromagnétique	Tension
Position (aimant)	Effet Hall	Tension

Figure I.2 : effets physiques de base des capteurs actifs.

I.2.2.2. Capteur passif:

Le principe des capteurs de ce type est de mesurer leurs impédances dont l'un des paramètres déterminants est sensible au mesurande (géométrie et dimensions, propriétés électriques des matériaux : résistivité **ρ**, perméabilité magnétique **μ**, constante diélectrique **ε)**.

Les paramètres géométriques ou dimensionnels de l'impédance peuvent varier si le capteur comporte soit un élément mobile, soit un élément déformable qui va provoquer une variation du signal électrique délivré par le capteur.

Les propriétés électriques des matériaux, selon la nature de ces derniers, peuvent être sensibles à des grandeurs physiques variées : température, éclairement, pression, humidité,...etc. Si une seule de ces grandeurs est susceptible d'évolution, toutes les autres étant maintenues constantes il s'établit une correspondance univoque entre la valeur de cette grandeur et celle de l'impédance du capteur. La courbe d'étalonnage traduit cette correspondance et permet, à partir de la mesure de l'impédance de déduire la valeur de la grandeur physique agissante (mesurande).

Le *tableau I.2* donne un aperçu des divers mesurande susceptibles de modifier les propriétés électriques des matériaux utilisés dans la réalisation des capteurs.

Mesurande	**Caractéristique électrique sensible**
Température	Résistivité
Déformation	Résistivité Perméabilité magnétique
Humidité	Résistivité
Position (aimant)	Résistivité

Figure I.3 : Principes physiques et matériaux.

L'impédance d'un capteur passif et ses variations ne sont mesurables qu'en intégrant le capteur dans un circuit électrique, par ailleurs alimenté et qui est son conditionneur. Les conditionneurs les plus utilisés sont : le montage potentiométrique, le pont d'impédances, les circuits oscillant et l'amplificateur opérationnel.

I.2.2.3. Capteurs composites. Corps d'épreuve:

Pour des raisons de cout ou de facilité d'exploitation, on peut être amené à utiliser un capteur, non pas sensible au mesurande mais à l'un de ses effets. Le corps d'épreuve est le dispositif qui, soumis au mesurande étudié en assure une première traduction en une autre grandeur physique non électrique, dit mesurande secondaire, qu'un capteur adéquat traduit alors en grandeur électrique (*figure I.3*).L'ensemble formé par le corps d'épreuve et un capteur actif ou passif constitue un ***Capteur composite***.

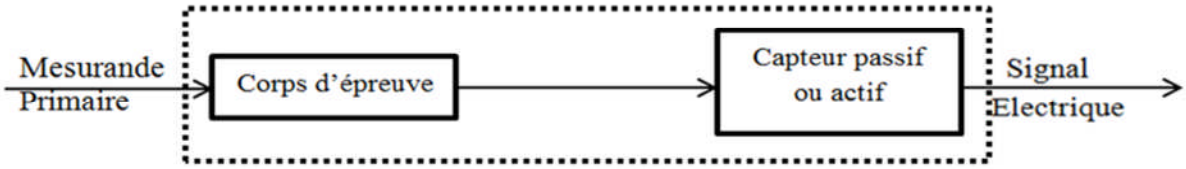


Figure I.4 : Structure d'un capteur composite.

Les corps d'épreuve sont très utilisés pour la mesure de grandeurs mécaniques (traction, pression,...etc.): celles-ci imposent au corps d'épreuve des déformations ou des déplacements auxquels un capteur approprié est sensible.

I.2.3. Capteur intégré :

Un capteur intégré est un composant réalisé par les techniques de la Microélectronique et qui regroupe sur un substrat de silicium commun le capteur proprement dit, le corps d'épreuve éventuel et des circuits électroniques de conditionnement du signal (*figure. I.4*).

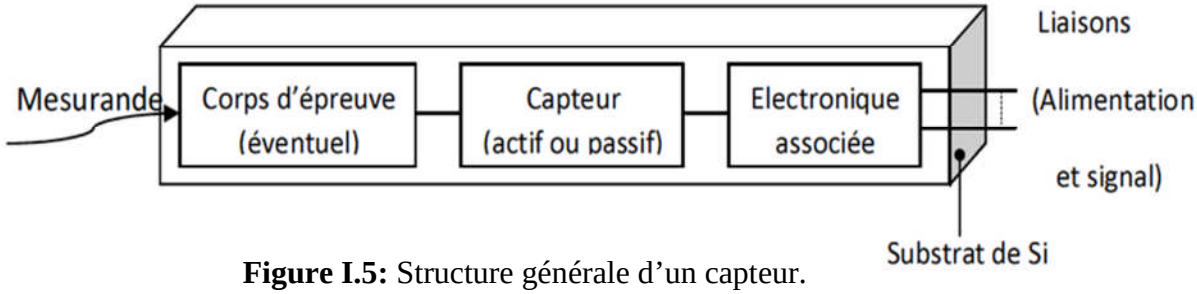


Figure I.5: Structure générale d'un capteur.

I.2.4. Caractéristiques statistiques d'un capteur :

Supposons que l'on fasse n mesures m_1….. m_i…... m_n d'un même mesurande m.

> ***Valeur moyenne :***

On appelle valeur moyenne de **m** la quantité $\bar{m}$ telle que :

$$\bar{m} = \frac{\sum_{i=1}^{1} m_i}{n} \qquad \text{(I.1)}$$

> ***Variance :***

La variance est une mesure arbitraire servant à caractériser la dispersion d'une distribution ou d'un échantillon.

$$\mathbf{V} = \frac{\sum_{i=1}^{n}(m_i - \bar{m})^2}{n} \qquad \text{(I.2)}$$

> ***Ecart-type :***

L'écart-type mesure la dispersion de cette série de valeurs (m1....m_n) autour de la moyenne $\bar{\mathbf{m}}$.

$$\boldsymbol{\sigma = \sqrt{V} = \sqrt{\frac{\sum_{n=1}^{n}(m_i - \bar{m})^2}{n}}} \qquad \text{(I.3)}$$

I.2.4.1.Fidélité - Justesse – Précision :

> ***Fidélité :***

Elle définit la qualité d'un capteur à délivrer une mesure répétitive sans erreurs. L'erreur de fidélité correspond à l'écart type obtenu sur une série de mesures correspondant à un mesurande constant.

> ***Justesse :***

C'est l'aptitude d'un capteur à délivrer une réponse proche de la valeur vraie et ceci indépendamment de la notion de fidélité. Elle est liée à la valeur moyenne obtenue sur un grand nombre de mesures par rapport à la valeur réelle.

> ***Précision :***

Elle définit l'écart en pourcentage que l'on peut obtenir entre la valeur réelle et la valeur obtenue en sortie du capteur. Ainsi un capteur précis aura à la fois une bonne fidélité et une bonne justesse.

Ces trois propriétés caractérisent le capteur et son étalonnage. On dit qu'un capteur est fidèle si l'écart-type qu'il fournit est faible, qu'il est juste s'il est dépourvu d'erreur systématique et qu'il est précis s'il est à la fois juste et fidèle.

I.2.4.2.Résolution :

Elle correspond à la plus petite variation du mesurande que le capteur est susceptible de déceler.

I.2.4.3. Sensibilité :

Elle détermine l'évolution de la grandeur de sortie **S** en fonction de la grandeur d'entrée **m** en un point donné $\mathbf{m_i}$. C'est la pente de la tangente à la courbe issue de la caractéristique du capteur.

$$S = \left(\frac{dS}{dm}\right)_{m=m_i} \quad (I.4)$$

Dans le cas d'un capteur linéaire, la sensibilité ***S*** du capteur est constante.

Il faut noter que la sensibilité d'un capteur peut être fonction du conditionneur auquel il est associé.

I.2.4.4. Finesse :

C'est la qualité d'un capteur à ne pas venir modifier par sa présence la grandeur à mesurer. Cela permet d'évaluer l'influence du capteur sur la mesure. On la définit non seulement vis à vis du capteur mais aussi vis à vis de l'environnement d'utilisation du capteur.

I.2.4.5. Linéarité :

Zone dans laquelle la sensibilité du capteur est indépendante de la valeur du mesurande, qui peut être définie par une droite approchant au mieux la caractéristique réelle du capteur. On définit à partir de cette droite l'écart de linéarité qui exprime en pourcentage l'écart maximal entre la courbe réelle et la droite approchant la courbe.

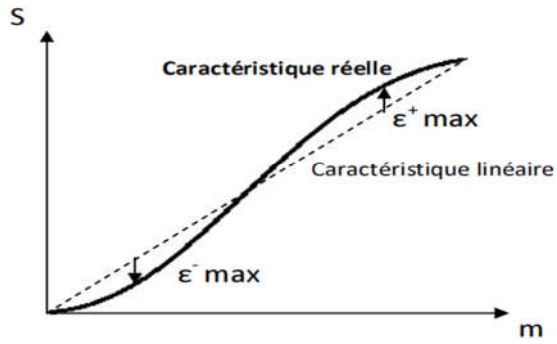


Figure I.6 : Exemple de linéarisation de caractéristiques.

I.3. Conditionneur associé :

Le conditionnement de la mesure consiste à rendre exploitable la mesure issue du capteur. L'association capteur-conditionneur détermine le signal électrique et ses caractéristiques. On effectue une adaptation de la source du signal à la chaîne de mesure complète.

I.3.1. Conditionneurs de capteurs actifs :

On distingue trois types :

I.3.1.1. Capteur source de tension :

On peut adopter le modèle suivant pour la sortie du capteur auquel on vient connecter une impédance (Z_i) correspondant à l'impédance d'entrée du conditionneur (Z_C).

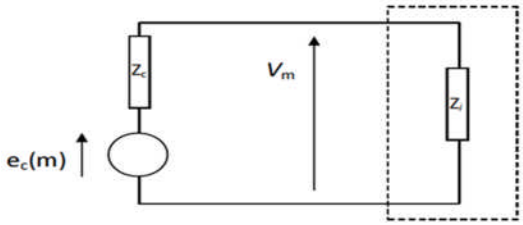
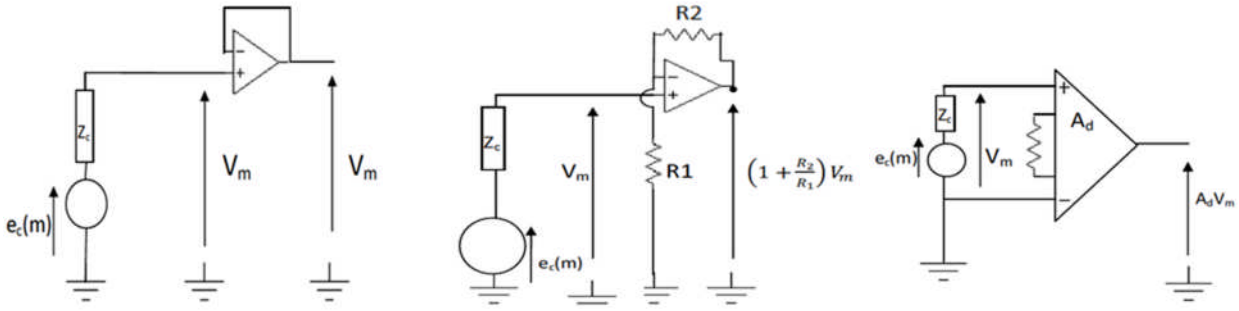


Figure I.7: Modèle du capteur source de tension.

$$V_{m=e_c\frac{z_i}{z_i-z_c}\approx e_c} \quad \text{Pour} \quad \boldsymbol{z_i \gg z_c} \qquad \text{(I.5)}$$

On utilisera des dispositifs à forte impédance d'entrée de manière à obtenir une tension en sortie du conditionneur aussi proche que la tension en sortie du capteur ($\mathbf{V_m \approx e_c}$).

On pourra utiliser un montage suiveur ou un amplificateur différentiel plus classiquement appelé amplificateur d'instrumentation.

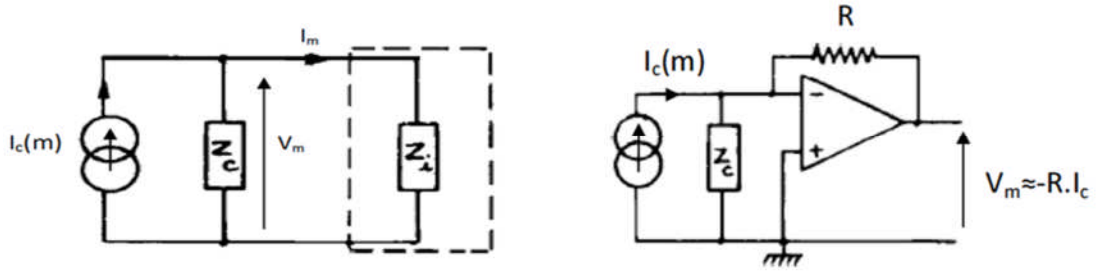


a) b) c)

Figure I.8: a) Montage suiveur, b) amplificateur opérationnel, c) amplificateur d'instrumentation

I.3.1.2. Capteur source de courant :

Dans ce cas, le capteur peut se modéliser par une source de courant avec une impédance en parallèle.



a) **b)**

Figure I.9:a) Modèle du capteur type source de courant, **b)** Convertisseur courant-tension

On fait appel dans ce cas à un convertisseur courant-tension de manière à obtenir une tension proportionnelle au courant de sortie du capteur.

I.3.1.3. Capteur source de charge :

Ce capteur se comporte en générateur et présente une impédance interne capacitive. C'est le cas d'un cristal piézo-électrique. Il faut faire attention dans le cas où l'on vient brancher une impédance équivalente résistive à ses bornes. Cette résistance peut engendrer une décharge trop rapide de la capacité empêchant toute mesure.

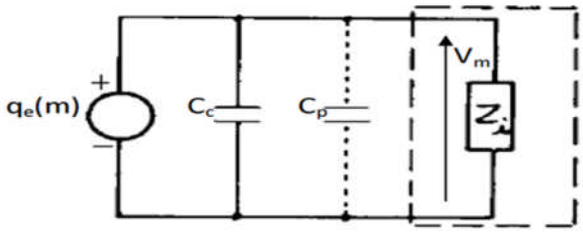


Figure I.10 : Modèle du capteur type source de charge.

Dans ce cas, il est préférable d'utiliser un amplificateur de charge dont le principe est présenté ci-dessous.

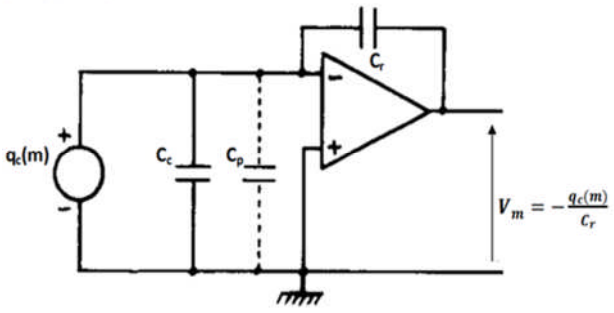


Figure I.11: Amplificateur de charge.

I.3.2. Conditionneurs de capteurs passifs :

Comme ce capteur donne une image du mesurande par l'intermédiaire d'une impédance, on associe toujours une source externe de tension ou de courant au capteur.

Deux grands principes de conditionneurs peuvent être employés :

- ✓ Montage en pont : on récupère alors une tension proportionnelle au mesurande.
- ✓ Montage oscillant : la fréquence du signal de sortie est modulée par le mesurande.

I.3.2.1. Montage potentiométrique :

Deux cas peuvent se présenter.

> **Cas de résistances :**

On utilise un simple pont diviseur alimenté par une source de tension continue V_e. L'impédance interne de la source (R_s) et l'impédance de l'appareil de mesure (R_d) doivent être prises en compte. Le capteur est modélisé par la résistance R_c.

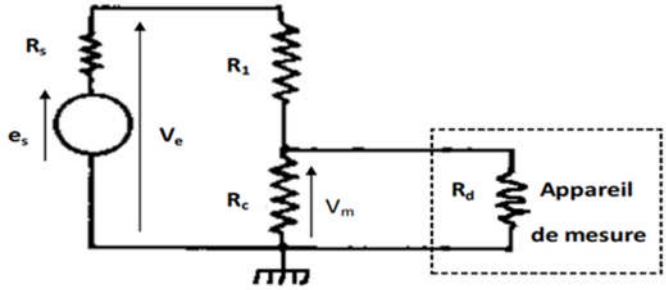


Figure I.12: Modèle du montage potentiométrique.

En négligeant R_s et R_d, on obtient :

$$V_m = \frac{R_c}{R_C+R_1} V_e \qquad (I.6)$$

Remarque : Cas d'une alimentation en courant :

Figure I.13: Modèle du montage potentiométrique.

L'utilisation d'une source de courant I_s rend le montage directement linéaire si l'on néglige l'impédance interne de la source, c'est à dire :

$$V_m = R_c I_s \tag{I.7}$$

> **Cas d'impédances complexes (zj):**

Le capteur est capacitif (détecteur de niveau par exemple) ou inductif (détecteur de position). On utilise alors une source d'alimentation sinusoïdale associée à un pont diviseur.

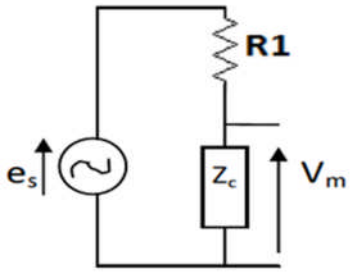


Figure I.14 : Montages en pont dans le cas d'impédances complexes.

En supposant $\boldsymbol{R_1 \ll |Z_C|}$ et que l'impédance du capteur varie de Z_{c_0} à $Z_{c_0} + \Delta Z_C$ on obtient une variation de tension à ses bornes qui vaut :

$$\Delta V_m = \frac{e_s}{R_1} \Delta Z_c \tag{I.8}$$

De même en utilisant une source de courant I_s :

$$\Delta V_m = I_s \Delta Z_c \tag{I.9}$$

I.3.2.2. Montage en pont :

L'utilisation d'un montage potentiométrique présente le défaut d'avoir en sortie la présence d'une tension, et ceci en l'absence de variations du mesurande. L'emploi d'un montage en pont présente l'avantage de s'affranchir de cette tension continue

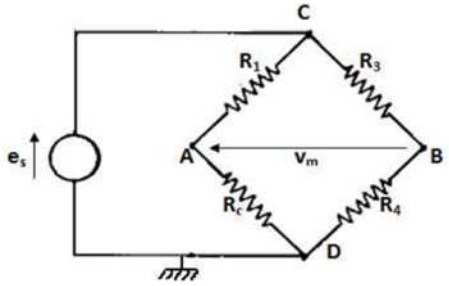


Figure I.15 : montage en pont.

Après avoir calculé les potentiels aux points **A** et **B**, on obtient une tension de mesure encore appelée tension de déséquilibre du pont :

$$V_m = V_A - V_B = \frac{R_C R_3 - R_1 R_4}{(R_1 + R_c)(R_4 + R_3)} e_s \quad (I.10)$$

Si on veut une tension nulle en l'absence d'évolution du mesurande (cas stable $R_c = R_{C_0}$), on trouve la condition d'équilibre d'un pont de Wheatstone :

$$\overline{R_C R_3 = R_1 R_4} \quad (I.11)$$

Cas particulier R1=R3=R4=R:

On suppose que le mesurande évolue autour d'une valeur $\boldsymbol{R_{C_0}}$ **:** $\boldsymbol{R_C}$**=** $\boldsymbol{R_{C_0}}$+Δ**R**, avec $\boldsymbol{R_{C_0}}$**=R**. Comme ce pont correspond à un double montage potentiométrique, on aura alors dans ce cas une sensibilité maximale ; l'expression de $\boldsymbol{V_m}$ devient :

$$\boldsymbol{V_m} = \frac{\boldsymbol{e_s}}{\boldsymbol{4}} \frac{\boldsymbol{\Delta R / R}}{\boldsymbol{(1 + \Delta R / 2R)}} \quad (I.12)$$

En petites variation de $R_c\left(\Delta R / R \ll 1\right)$, on peut linéariser la relation entre $\boldsymbol{V_m}$ ***et*** $\boldsymbol{\Delta R}$

$$V_m = \frac{e_s}{4} \frac{\Delta R}{R} \quad (I.13)$$

I.3.2.3. Montage oscillant :

Un circuit oscillant (LC) présente une fréquence de résonance $\mathbf{F_o}$ telle que:

$$L_0 = \frac{1}{2\pi\sqrt{LC}} \quad (I.14)$$

Si on insère un capteur capacitif ou inductif dans un tel circuit, ses variations entraîneront une variation Δ**F** de la fréquence d'oscillation du circuit. En supposant des petites variations, on obtient une évolution :

$$\frac{\Delta F}{F_0} = -\frac{\Delta L}{2L_0} \quad \text{ou} \quad \frac{\Delta F}{F_0} = \frac{\Delta C}{2C_0} \quad (I.15)$$

Dans le cas d'un capteur capacitif, on peut utiliser un multivibrateur astable à amplificateur opérationnel.

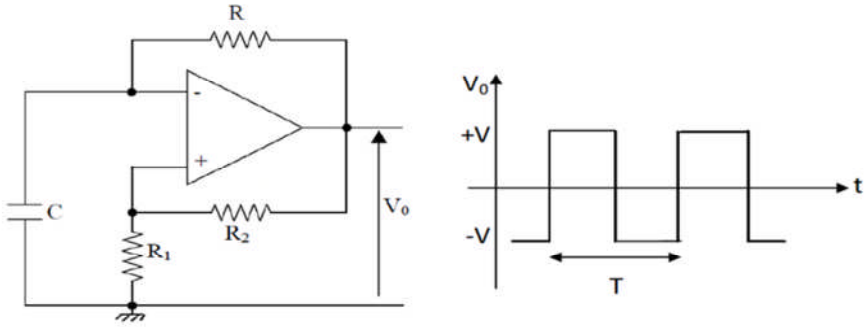


Figure I.16 : Montage astable à circuit R-C.

La période des oscillations est directement reliée à la valeur de la capacité par la relation :

$$T = 2.RC.\ln\left(1+\frac{2R_1}{R_2}\right) \qquad (I.16)$$

I.4. Le multiplexage :

Le multiplexage analogique est une opération qui permet dans le cas d'acquisition de plusieurs signaux analogique la sélection d'un canal bien déterminé afin d'aiguiller son signal vers les dispositifs de traitement (filtre anti repliement, convertisseur analogique –numérique) situés en aval . Dans notre système cette opération est assurée par un multiplexeur à huit voies analogiques.

I.4.1.Description du multiplexeur analogique :

Un multiplexeur analogique est constitué:

- D'un réseau d'interrupteurs (N) à base des transistors à effet de champ relient au même point qui est la sortie, chaque interrupteur contrôle une voie.
- D'un décodeur, sur N bits avec $N=2^{n,}$ qui permet la sélection de la voie analogique à échantillonner.

I.5. La conversion analogique numérique :

I.5.1.Définition :

L'opération de la conversion analogique –numérique consiste à transformer une grandeur électrique continue dans le temps et en amplitude en une grandeur numérique exprimée sur N bits .cette grandeur représente dans le système de codage qui lui est affectée, un nombre proportionnel à la grandeur analogique.

Cette conversion est assurée par un convertisseur analogique numérique recevant à son entrée une tension analogique Ve, délivre en sortie un mot de n bits correspondant, selon un code binaire déterminé à valeur numérique N associée à cette tension analogique.

Soit : N= (b_{n-1}……..$b_{0)}$ la valeur numérique.

Quel que soit le convertisseur analogique–numérique utilisé, le processus de conversion passe par deux étapes : l'échantillonnage temporel et la quantification des amplitudes.

I.5.2.Echantillonnage :

La fonction d'échantillonnage consiste à prélever des échantillons du signal analogique s(t) à des instants régulièrement espacés par une période Te (période d'échantillonnage) a une fréquence Fe dite fréquence d'échantillonnage.

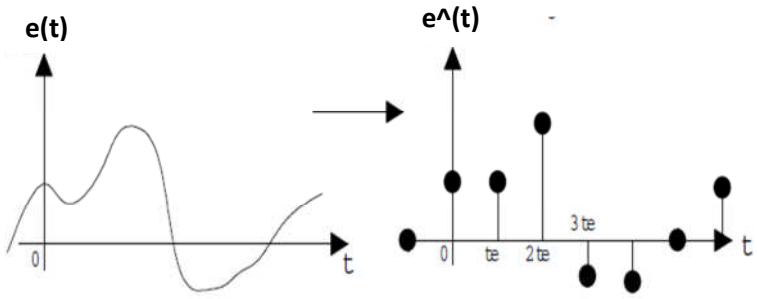


Figure I.17 : Principe d'échantillonnage d'un signal analogique.

I.5.2.1.Théoréme de Shannon :

Pour que le signal échantillonné soit sans perte d'information il faut prendre la fréquence d'échantillonnage Fe supérieur ou égale à deux fois la fréquence maximale du signal a échantillonner.

$$Fe \geq 2Fmax \tag{I.17}$$

I.5.2.2.Echantillonnage et maintient :

Cette fonction est réalisée par un circuit échantillonneur-bloqueur. Cadencé par un signal d'horloge de fréquence Fe, ce circuit prélève, à chaque impulsion d'horloge une valeur de la tension du signal analogique et la mémorise (en chargeant un condensateur) pour la transmettre en suite au convertisseur analogique-numérique pour l'évaluer et la quantifier.

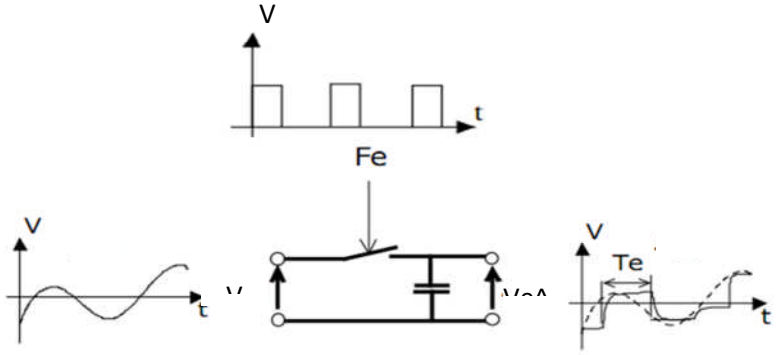


Figure I.18: Principe de fonctionnement d'un échantillonneur bloqueur.

I.5.2.3. Repliement de spectre :

Le phénomène de repliement de spectre apparait quand la fréquence Fmax du signal devient supérieure à un demi de la fréquence d'échantillonnage :

$$Fmax > Fe/2 \tag{I.18}$$

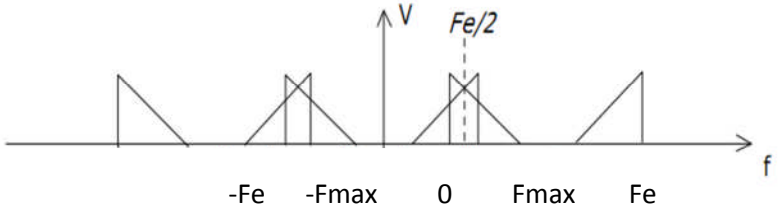


Figure I.19: Echantillonnage provocant le repliement du spectre.

S'il y'a repliement de spectre, il n'est plus possible de retrouver le spectre du signal d'origine .Dans ce cas, l'opération d'échantillonnage modifie les caractéristiques du signal d'entrée. Donc pour éviter ce phénomène il faut toujours respecter le théorème de Shannon.

I.5.2.4.Filtre anti-repliement :

Dans la réalité le spectre d'un signal analogique est généralement de largeur infinie (à cause du bruit, ou de signaux interférents non désirés), il y'a donc toujours un phénomène de repliement spectral susceptible de ramener dans la bande utile, du bruit ou un signal d'interférence. D'où la nécessité d'inclure un filtre passe bas anti-repliement ayant pour fréquence de coupure :

$$Fc = Fe/2 \tag{I.19}$$

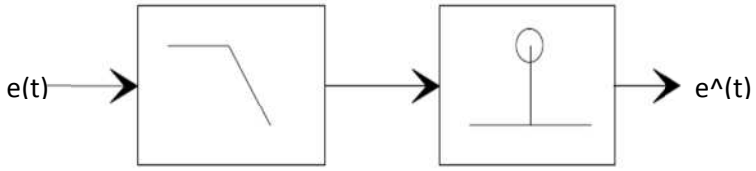


Figure I.20: Utilisation d'un filtre en amont de l'échantillonneur.

Dans le cadre d'un filtre anti-repliement, en générale on fait appel à un filtre passe bas du type Butterworth dont le gain est donné par la relation :

$$H(f) = \frac{1}{\sqrt{\left(1 + \frac{f^{2n}}{f_c}\right)}} \tag{I.20}$$

Ce filtre présente l'avantage de répondre au critère de platitude dans la bande passante et un retard de groupe constant jusqu'à $fc/2$.

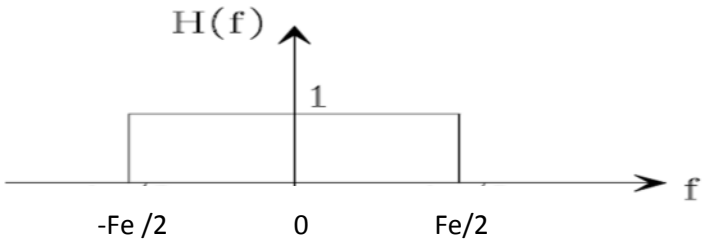


Figure I.21: Gabarit idéal du filtre anti-repliement.

I.5.3.La quantification :

La quantification est l'étape de conversion analogique-numérique, elle consiste à transformer une tension réelle V_{analog} en un nombre binaire V_{num}= (bi)$_{0 \le i \le N-1, \text{ choisi}}$ parmi un ensemble fini et prédéterminé de valeurs .Un convertisseur analogique-numérique (CAN) est caractérisé par deux principaux paramètres : son nombre de bits N(ou résolution) et sa dynamique d'entrée $\Delta V_{in} = [V_{min}, V_{max}]$.A partir de cela, le **quantum 'q'** ou LSB (less Significant Bit) nommé aussi pas de quantification ,est déterminé par la relation :

$$q = \frac{\Delta V_{in}}{2^N - 1} = LSB \tag{I.21}$$

C'est la quantité élémentaire dont sont multiple toutes les tensions de sortie .Pour une tension analogique d'entrée V_{analog}, le CAN fait correspondre une tension V_{num} telle que :

$$\left|V_{analog} - V_{num}\right| \leq q/2 \tag{I.22}$$

Dans le cas de l'utilisation d'un code numérique en binaire naturel :

$$V_{num}=[b_{N-1}2^{N-1}+\cdots+b_1 2^1+b_0 2^0] \tag{I.23}$$

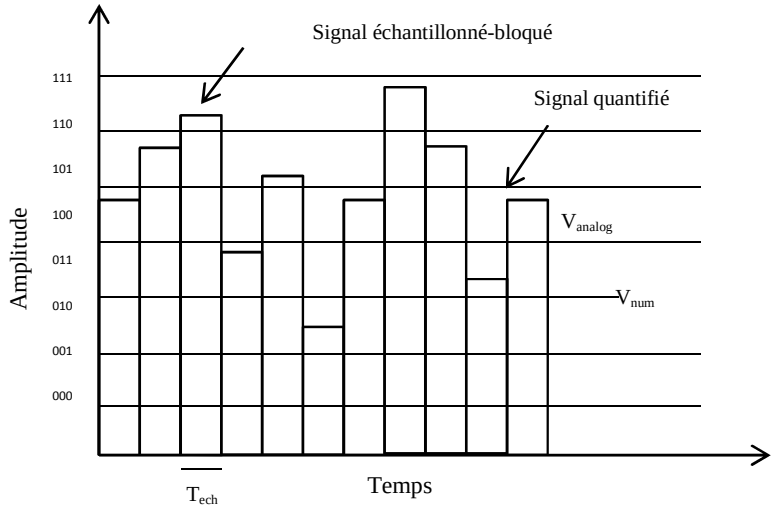


Figure I.22: Quantification d'un signal analogique.

I.5.4.Les convertisseurs analogiques numériques

I.5.4.1. Caractéristiques des convertisseurs analogiques numériques :

a)-La résolution :

Elle traduit le nombre de bits sur lesquels est donné le résultat de la conversion.

b)-La précision :

C'est l'écart entre la caractéristique de transfert de CAN idéal et celle d'un CAN réel. Cet écart est exprimé en nombre de LSB (bit de poids faible).

c)-La vitesse de conversion :

Il existe différents principes de conversion analogique numérique. Les plus rapides sont les CAN flash, d'autres moins rapides tels que les CAN à approximations successives et les CAN à rampe numérique.

I.5.4.2.Le convertisseur à approximations (pesées) successives :

On détermine les valeurs des différents bits l'un après l'autre en commençant par le MSB (bit de poids fort).

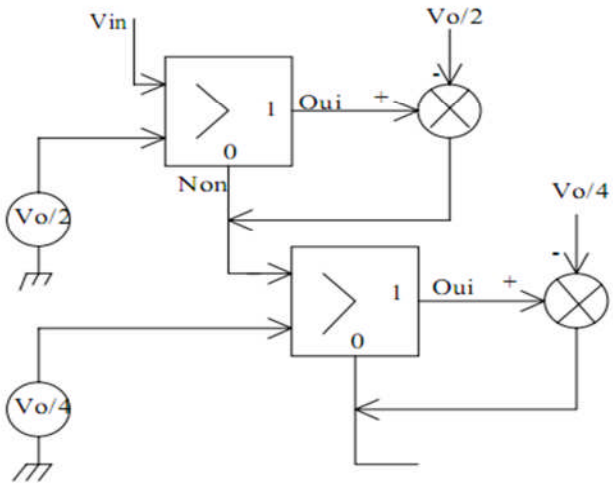


Figure I.23: Principe de l'approximation successive.

Le signal d'entrée d'amplitude Vin est comparé à une tension de référence: Vo/2. S'il est supérieur, on lui retranche cette valeur et on met le bit de comparaison à '1', sinon on met le bit de comparaison à '0' et on le compare à la tension suivante.

Il compare :

$$\text{Vin a } \frac{\text{Vref}}{2} \text{ puis } \text{Vin} - \frac{\text{Vref}}{2} b_{n-1} \text{ puis à } \frac{Vref}{4} \text{ …etc...} \qquad \text{(I.24)}$$

$$\text{Vin} = \frac{\text{Vref}}{2} b_{n-1} + \frac{\text{Vref}}{4} b_{n-2} + \cdots$$

Ce convertisseur est plus lent que le convertisseur Flash. Ainsi pour 16 bits, il lui faut en moyenne un temps de conversion (Tc) de 10µs.Il est très adapté à des signaux audio.

I.5.4.3. Choix d'un convertisseur analogique numérique :

Vu l'utilisation de plusieurs voies analogiques dans notre système, et la lenteur des signaux délivrés par les capteurs nous jugerons que le convertisseur analogique numérique ADC 0809 est le plus approprié pour notre circuit. Ce convertisseur se présente dans un boitier plastique de 28 broches doté :

- D'un multiplexeur analogique a huit voies dont la sélection se fait via un bus à trois bits.
- D'un comparateur pour fixer la dynamique d'entrée du convertisseur.
- D'un bloc de contrôle de timing qui contient trois entrées, l'une pour le signal d'horloge (clk) et l'autre pour enclencher le début de la conversion, et la troisième pour la fin de conversion.
- D'un convertisseur analogique –numérique a sortie parallèles, dont la résolution est sur huit bits, et le temps de conversion Tc = 100 µs d'où Fc=10khz.
- La fréquence d'échantillonnage est Fe = Fc /8

I.6.Conclusion :

Dans ce chapitre en a présenté les déférant blocs constitutif de la partie analogique de la chaine d'acquisition de données notamment les capteurs et leur conditionneurs associe suivie de l'amplification des signaux analogiques résultant on suite on a évoqué le principe d'aiguillage du multiplexeur et en dernier on a évoqué la fonction d'échantillonnage et de conversion analogique numérique. Dans le chapitre suivant on présentera les déférentes familles des circuits logiques programmables PLD et ASICs.

Chapitre II

II.1.Introduction :

Dans ce chapitre, nous allons évoquer les différents types des circuits intégrés programmables et reprogrammables, allant d'une simple mémoire programmable au circuit FPGA (Field Programmable Gate Array) de plus de 10 millions de portes logiques reprogrammables, en passant par les circuits spécifiques à une application. Nous allons tout d'abord présenter les différents types de circuits ASICs. Ensuite nous présentons les circuits programmables PLD (Programmable Logic Device), en suivant l'ordre chronologiques, jusqu'aux FPGAs, et on termine ce chapitre par la présentation des différentes caractéristiques d'un circuit FPGA de famille VIRTEX-II.

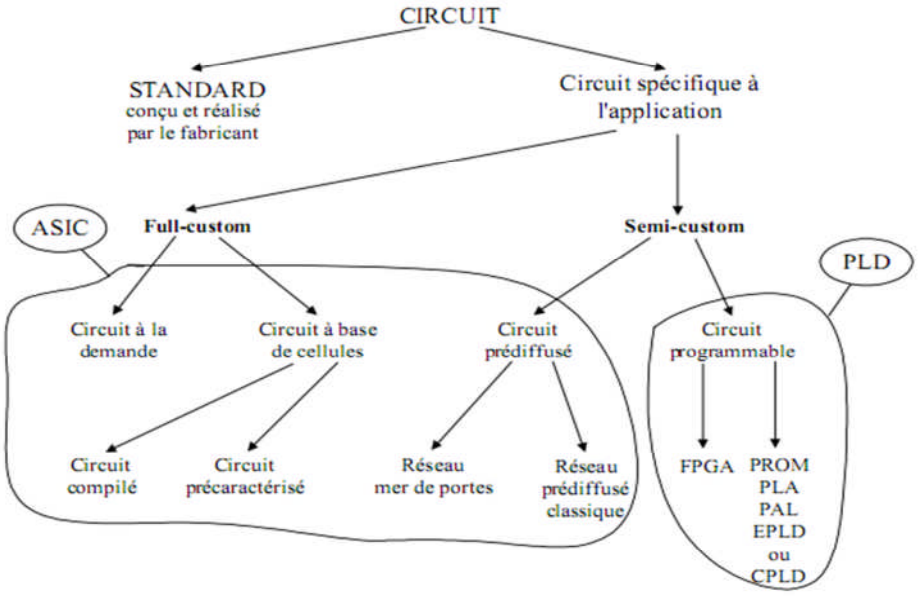


Figure II.1 : classification des circuits logiques programmables.

II.2.Les ASICs :

Un ASIC (application Specific Integrated Circuit) est un circuit spécialisé .En générale il regroupe un grand nombre de fonctionnalités uniques et sur mesure.

Les ASICs fournissent précisément la fonctionnalité nécessaire pour une tache spécifique .Ils sont conçus pour une tache bien définie et cela leurs permets d'être plus petits, moins. Chers, plus rapide et de consommer moins de puissance qu'un processeur programmable.

II.2.1.Classification des ASICs :

II.2.1.1. Les circuits personnalisés :

Les circuits intégrés appelés full-custom ont comme particularité de posséder une architecture dédiée à chaque application et sont donc complètement définis par les concepteurs. La fabrication nécessite la création de l'ensemble des masques pour la réalisation (6 pour les transistors plus 2 par couche métal). Les temps de fabrication de ces masques et de production des circuits sont de ce fait assez longs. Ces circuits sont ainsi appropriés pour de grandes séries. L'avantage du circuit full-custom réside dans la possibilité d'avoir un circuit ayant les fonctionnalités strictement nécessaires à la réalisation des objectifs de l'application, et donc un nombre minimal de transistors (donc la surface de puce la plus petite et le coût le plus faible). Parmi les circuits full-custom, on distingue :

- les circuits à la demande,
- les circuits à base de cellules.

II.2.1.1.1. Les circuits à la demande :

Ces circuits sont directement conçus et fabriqués par les fondeurs. Ils sont spécifiques car ils répondent à l'expression d'un besoin pour une application particulière. Le demandeur utilise le fondeur comme un sous-traitant pour la conception et la réalisation et n'intervient que pour exprimer le besoin. Ces circuits spécifiques utilisent au mieux la puce de silicium. Chaque circuit conçu et fabriqué de cette manière doit être produit en très grande quantité pour amortir les coûts de conception.

II.2.1.1.2. Les circuits à base de cellules :

Les circuits à base de cellules (CBIC : Cell Based Integrated Circuit) permettent des complexités d'intégration allant jusqu'à plusieurs dizaines de millions de portes. Dans cette catégorie de circuits, on distingue les circuits à base de cellules précaractérisées et les circuits à base de cellules compilées.

II.2.1.1.2.1. Les cellules précaractérisées :

Les cellules précaractérisées sont des entités logiques plus ou moins complexes. Il peut s'agir de cellules de base (portes, bascules, etc.) mais aussi de cellules mémoires (ROM, RAM) ou encore de sous-systèmes numériques complexes (UART, cœur de microprocesseur,

PLA, ...). Toutes ces cellules ont été implantées et caractérisées au niveau physique (d'où la notion de cellules précaractérisées) par le fondeur. La fonctionnalité globale de l'application à réaliser s'obtient en choisissant les cellules appropriées dans une bibliothèque fournie par le fondeur.

II.2.1.1.2.2. Les circuits à base de cellules compilées :

Les circuits à base de cellules compilées sont en fait basés sur l'utilisation de cellules précaractérisées. A la différence des circuits précaractérisés, les cellules ne sont pas utilisables directement mais au travers de modules paramétrables ou modules génériques. Chaque module est créé par la juxtaposition de n cellules de même type. La différence entre circuits précaractérisés et circuits compilés provient essentiellement de l'outil utilisé pour générer les dessins des masques de fabrication. Ces outils sont appelés des compilateurs de silicium.

II.2.2. Les circuits semi-personnalisés :

Dans la famille des circuits semi-custom, on distingue deux groupes :

- les circuits prédiffusés,
- les circuits programmables.

II.2.2.1. Les circuits prédiffusés :

Parmi les circuits prédiffusés, on distingue les prédiffusés classiques ("Gate Array"), et les ASICs structurés.

II.2.2.1.1. Les circuits prédiffusés classiques :

Les circuits prédiffusés classiques possèdent une architecture interne fixe qui consiste, dans la plupart des cas, en des rangées de portes séparées par des canaux d'interconnexion. L'implantation de l'application se fait en définissant les masques d'interconnexion pour la phase finale de fabrication. Ces masques d'interconnexion permettent d'établir des liaisons entre les portes et les plots d'entrées/sorties.

II.2.2.1.2 Les ASICs structurés :

L’idée de base de l’ASIC structuré, c’est d’offrir une offre logicielle simplifiée au client (faible coût par rapport aux précaractérisés) mais avec la bibliothèque d’IPs (blocs de propriété intellectuelle tels que les microprocesseurs, contrôleurs Ethernet, …) des

précaractérisés : la simplicité et le coût des FPGAs avec les potentialités du précaractérisés. La réalité physique est bien entendue assez éloignée de la réalité marketing.

II.3. Les circuits programmables PLD :

Les PLDs (Programmable Logic Device) sont des circuits logiques programmables par l'utilisateur, ils se décomposent en deux familles :

- les PROMs, les PLAs, les PALs et les EPLDs,
- les FPGAs.

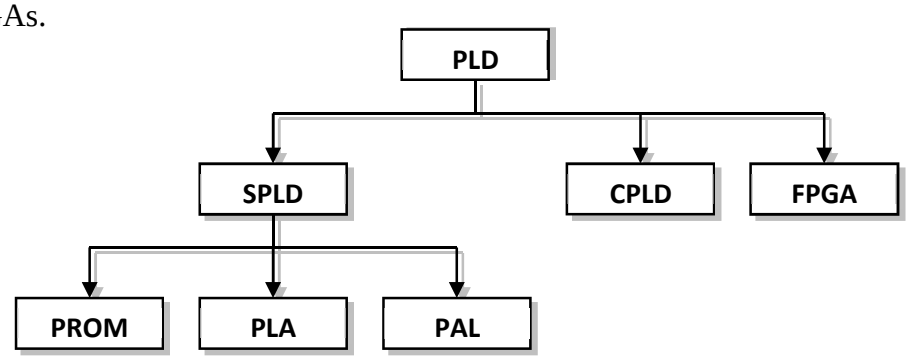


Figure II.2 : classification des PLD

II.3.1. La PROM :

Cette mémoire est l'association d'un réseau de ET fixes, réalisant le décodage d'adresse, et d'un réseau de OU programmables, réalisant le plan mémoire proprement dit. On peut facilement comprendre que, outre le stockage de données qui est sa fonction première, cette mémoire puisse être utilisée en tant que circuit logique. La figure ci-dessous représente la structure logique d'une PROM bipolaire à fusibles.

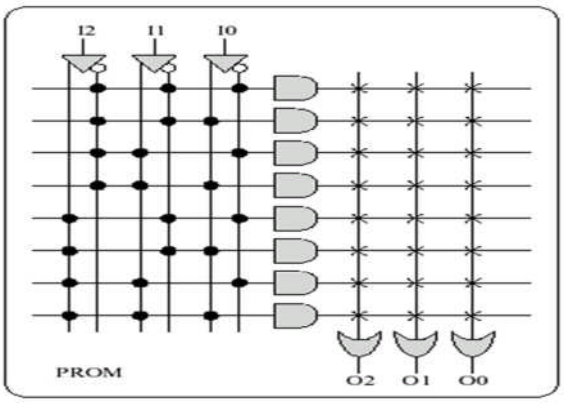


Figure II.3 : architecture d'une PROM.

II.3.2. Les PLAs :

Le circuit PLA (Programmable Logic Array) a été développé, il y a plus de 20 ans. Il reprend la technique des fusibles des PROMs bipolaires. La programmation consiste à faire sauter les fusibles pour réaliser la fonction logique de son choix. La structure des PLAs est une évolution des PROMs bipolaires. Elle est constituée d'un réseau de ET, OU programmables. Sa structure logique est la suivante :

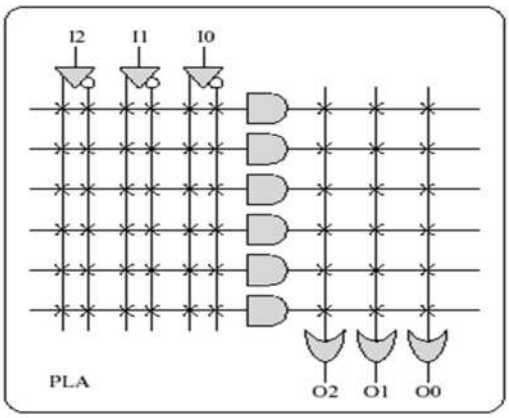


Figure II.4 : architecture d'un PLA.

II.3.3. Les PALs :

Les PALs (Programmable Array Logic) imposent un réseau de OU fixes et un réseau de ET programmables. La technologie employée est la même que pour les PLAs. L'avantage de cette architecture est l'augmentation de la vitesse par rapport aux PLAs. En effet, comme le nombre de connexions programmables est diminué, la longueur des lignes d'interconnexion est réduite. Le temps de propagation entre une entrée et une sortie est par conséquent plus faible.

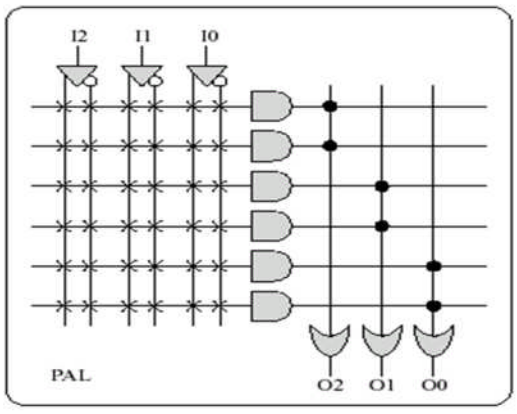


Figure II.5 : architecture d'un PAL.

II.3.4 .Les EPLDs :

Les EPLDs (Erasable Programmable logic Device) sont des circuits programmables électriquement et effaçables, soit par exposition aux UV pour les plus anciens, soit électriquement. Ces circuits, développés en premier par la firme ALTERA, sont arrivés sur le marché en 1985. Les EPLDs sont une évolution importante des PALs CMOS. Ils sont basés sur le même principe pour la réalisation des fonctions logiques de base.

Les EPLDs font appel à la notion de macro-cellule qui permet, par programmation, de réaliser de nombreuses fonctions logiques combinatoires ou séquentielles.

II.3.5 Les FPGA (LCA) :

Le circuit FPGA (Field Programmable Logic Device) ou LCA (Logic Cell Array) est un circuit prédiffusé programmable. Le concept du FPGA est basé sur l'utilisation d'une LUT (Look Up Table) comme élément combinatoire de la cellule de base. En première approximation, cette LUT peut être vue comme une mémoire (16 bits en général) qui permet de créer n'importe quelle fonction logique combinatoire de 4 variables d'entrées. Chez Xilinx, on appelle cela un générateur de fonction ou Function Generator. La figure suivante représente la cellule type de base d'un FPGA.

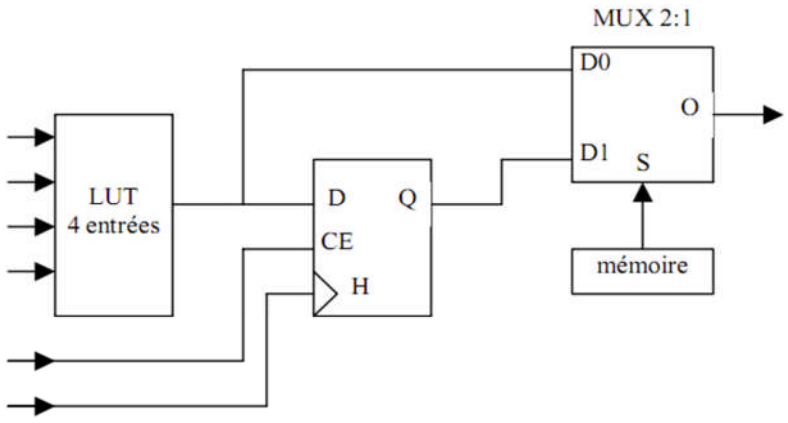


Figure II.6 : cellule de base de circuit

Elle comprend une LUT 4 entrées et une bascule D (D Flip-Flop). La bascule D permet la réalisation de fonctions logiques séquentielles. La configuration du multiplexeur 2 vers 1 de sortie autorise la sélection des deux types de fonction, combinatoire ou séquentielle.

Les cellules de base d'un FPGA sont disposées en lignes et en colonnes. Des lignes d'interconnexions programmables traversent le circuit, horizontalement et verticalement, entre les diverses cellules. Ces lignes d'interconnexions permettent de relier les cellules entre elles, et avec les plots d'entrées/sorties. Les connexions programmables sur ces lignes sont réalisées par des transistors MOS dont l'état est contrôlé par des cellules mémoires SRAM. Ainsi, toute la configuration d'un FPGA est contenue dans des cellules SRAM.

II.3.5.1.Architecture générale des FPGAs :

Les circuits FPGA sont constitués de plusieurs blocs :

- Les éléments combinatoires et séquentiels CLB (Configurable Logic Bloc).
- Les éléments d'entrée / sortie IOB (Input Output Bloc), ils sont associés aux broches de circuit.
- Les éléments de mémorisation.
- Les éléments de contrôle et d'acheminement de l'horloge DCM (Digital Clock Manager)
- Les éléments de routage PSM (Programmable Switch Manger) qu'est une matrice d'interconnexions.

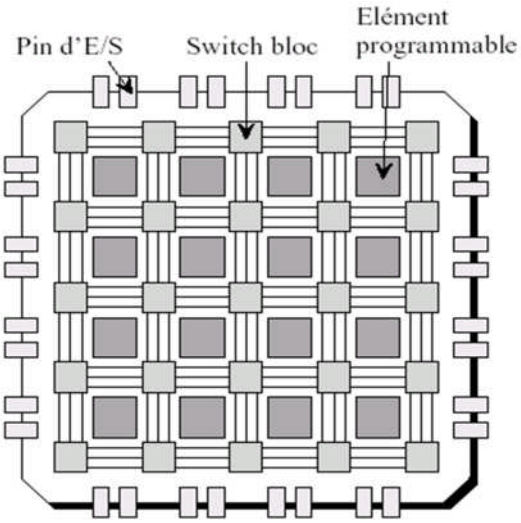


Figure II.7 : architecture générale d'un FPGA.

II.3.5.1.1.Les éléments Logiques CLB :

Ce sont les éléments de base de circuit FPGA, leur configuration permis de réaliser n'importe quelle fonction combinatoire et séquentielle. Ils sont généralement constitués d'une ou plusieurs LUTs (Look Up Table) qui contienne après configuration la table de vérité de la fonction logique à réaliser ou alors un ensemble de valeurs qui sont mémorisées.

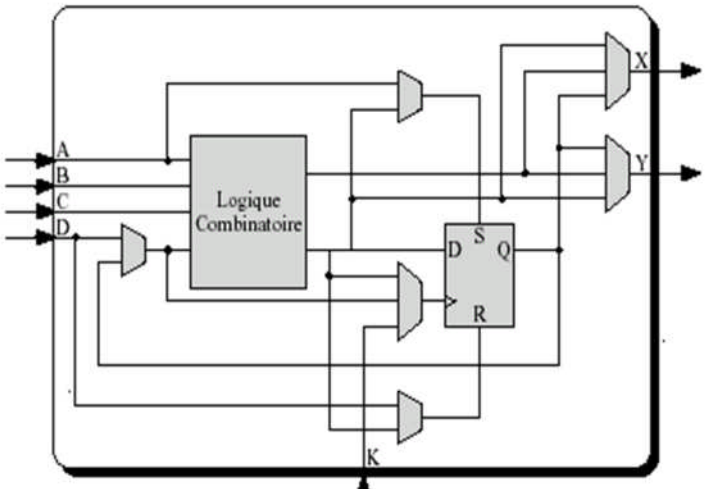


Figure II.8 : architecture d'un CLB de circuit xc2000.

II.3.5.1.2.Les éléments d'entrées /sorties (IOB) :

Les Blocs d'entrées-sorties (IOBs) permettent de contrôler l'échange des données entre les pins d'entrée-sortie et les blocs logiques internes. Chaque IOB supporte un échange de données bidirectionnelles et la logique 3 états.

Le périmètre du FPGA, représente l'interface entre l' FPGA et le milieu extérieur, il permet de configurer les broches du circuit selon les options suivantes :

- Input / Output
- TBUF (tampon à trois états logiques : ON, OFF, haute impédance)
- Inverseur
- Bascule flip-flop.

II.3.5.1.3.Les éléments de mémorisation :

L'intégration des blocs mémoires dans l'architecture des FPGA est devenue irréfutable, avec l'apparition des applications qui demandent de grandes capacités de stockage (comme le traitement de signal). De cette façon les temps d'accès à la mémoire sont diminués puisqu'il n'est pas nécessaire de communiquer avec des éléments extérieurs au circuit. Du fait

de l'intégration de ces blocs, dans des architectures déjà exécutantes, un point crucial apparait qui est le routage entre les parties mémoires et les ressources logiques. La caractéristique la plus importante de ce routage est sa flexibilité .Si le routage il n'est pas assez flexible alors le circuit est difficilement routable, s'il est trop flexible il y a surconsommation de surface de silicium.

II.3.5.1.4.Les éléments de routage :

Les ressources de routage représentent la plus grosse partie de silicium consommé sur la puce réalisant le circuit. Ces ressources sont composées de sagement qui permettent de relier les différents éléments entre eux via des matrices PSM (Programmable Switch Manager) d'interconnections.

Les interconnexions à usage générale sont composées des segments verticaux et horizontaux qui encadrent chaque CLB et qui peuvent être reliés entre eux par une matrice de commutation. Chaque segment peut être connecté à des segments qui lui sont adjacents ou perpendiculaire en utilisant des points de connexion. Les lignes directes fournissent des chemins entre les CLB adjacents et entre les CLBs et les cellules d'entrée / sortie.

Les lignes longues sont des lignes verticales et horizontales qui n'utilisent pas les matrices de communication .Elles parcourent toutes les zones d'interconnexion. Elles sont utilisées pour véhiculer les signaux qui doivent parcourir un long trajet. Elles égalisent les délais entre les signaux de façon à permettre un décalage minimum entre deux points distants de la ligne .Ces lignes conviennent pour véhiculer les signaux d'horloge. L'ensemble des signaux de connexion est appelé PIP (Programmable Interconnecte points).

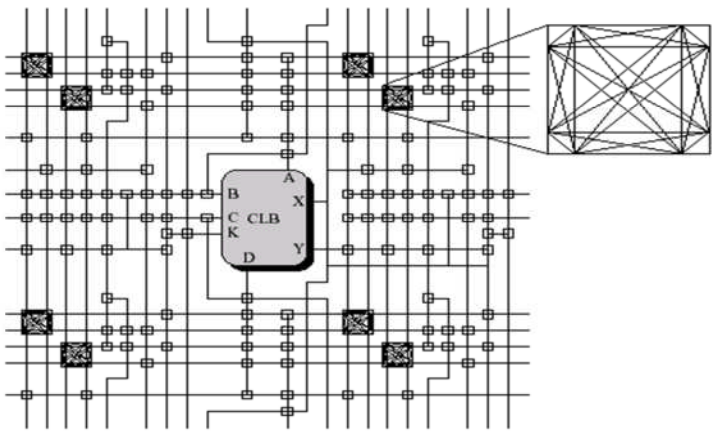


Figure II.9 : schéma des interconnexions.

II.3.5.1.5.Les éléments de contrôle et d'acheminement des horloges :

Les circuits FPGA sont prévus pour recevoir une ou plusieurs horloges. Des entrées peuvent être spécialement réservées à ce type de signaux, ainsi que des ressources de routage spécialement adaptées au transport d'horloges dans tout le circuit sur de longues distances (bufférisassions des lignes). Ainsi, pour être sûre d'avoir la même horloge dans tout le circuit (Synchronisation des signaux) les signaux doivent recevoir des éléments d'asservissement des horloges (PLL : Phase Locked Loop ou DLL : dalay Locked Loop ou DCM : Digital Clock Manager) qui permettent souvent de créer à partir d'une horloge d'autres horloges à des fréquences multiples de la fréquence de l'horloge incidente.

II.3.6. Les familles des circuits FPGAs de XILINX :

La première génération de LCA est le XC2000 en 1985, elle comprend des produits d'une complexité allant de 600 à 1500 portes logiques utilisables, suivi par XC3000,XC4000, XC5000,XC6000, SPARTAN ,VIRTEX : la septième génération des FPGAs (1999) .Cette famille vise de très grandes capacités (4 millions de portes logique) et les hautes vitesses (supérieure à 100MHZ) . Pour notre application nous allons travailler sur un FPGA de la famille VIRTEX-II.

II.3.6.1. La famille VIRTEX-II :

La famille viertex-II ouvre une gamme de complexité allant jusqu'à 4 millions de portes logique utilisables .celle-ci fournit une architecture régulière flexible, et programmable de blocs logiques configurable (CLBs), entourés par un périmètre de blocs Input/output programmables (IOBs) et possède des multiplieurs de 18bits x 18bits, ainsi que des mémoires SRAM et des lignes d'horloge DCM et GCM (Global Clock Mux).

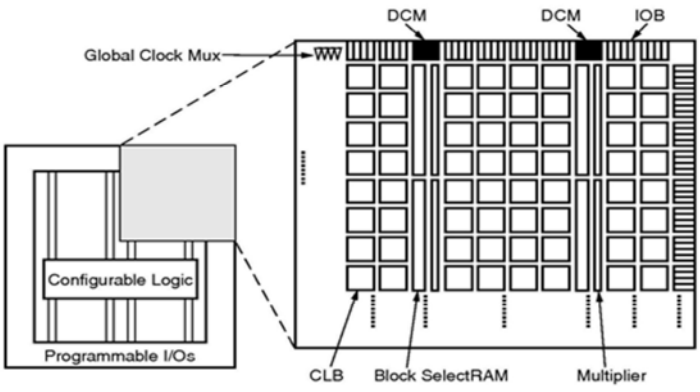


Figure II.10 : Architecture d'un circuit VIRTEX-II.

II.3.6.1.1.Architecture interne d'un circuit VIRTEX-II :

a) Structure d'un bloc CLB :

Chaque CLB est constitué de quatre cellules logiques réparties en deux tranches identiques de deux Slices chacune avec deux retenues indépendantes et une chaine de décalage commun .Chaque slice contient essentiellement :

- **Un générateur de fonctions à quatre entrées :**

réalisé à l'aide d'une table associative LUT (Look Up Table) a quatre entrées .Les multiplexeurs groupent les générateurs de fonctions d'une tranche d'un CLB afin de réaliser n'importe quelle fonction de cinq , six ,sept ou huit variables booléennes .Chaque table permet la conception d'une mémoire synchrone 16x1bit .Une fois appariées ,les deux LUTs d'un slice offrent une mémoire synchrone 16x2bits,32 x 1bit ou 16x 1bit a double accès (Dual Port Synchronous RAM).

- **L'élément de stockage :**

Le circuit possède des signaux d'initialisation (set et reset) synchrones ou asynchrones.

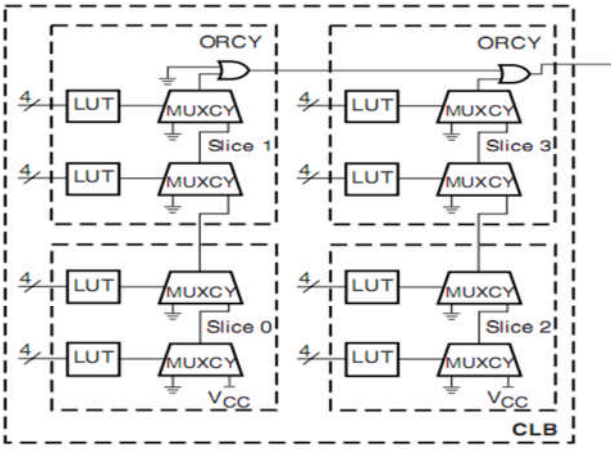


Figure II.11 : Structure d'un CLB.

b) II.3.6.1.1.2.Les blocs d'entrée/sortie (IOB ou input/output block) :

Il contient l'interface entre les broches du circuit et les CLBs. Nous pouvons ainsi modifier le système implémenté sur le circuit FPGA sans interférer avec l'attribution des broches. Cette caractéristique s'avère importante si nous souhaitons développer des nouvelles versions d'un produit tout en conservant la compatibilité au niveau du boitier.

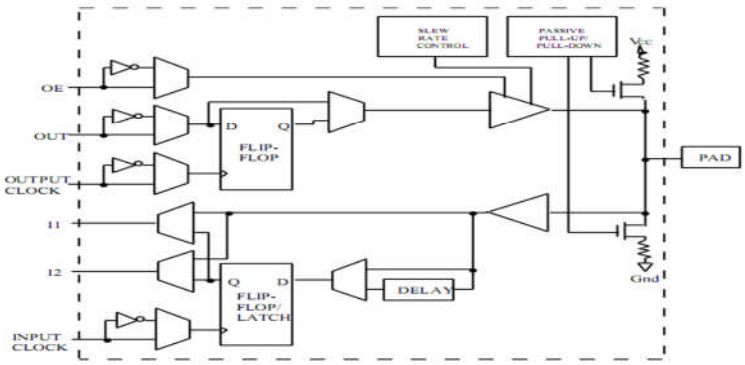


Figure II.12 : Structure d'un IOB.

Outre la possibilité de stoker de l'information dans les LUTs, le circuit VRTEX-II offre des mémoires SRAM ou Block Select RAM+), organisées en blocs .Situées de part et d'autre de la matrice du CLB. Le circuit possède aussi des lignes d'horloge, ces dernières permettent de contrôler les décalages d'horloge à l'intérieur du FPGA ou entre plusieurs circuits, ainsi que de déphaser, doubler ou diviser l'horloge.

c) Multiplieur 18bits x18bits :

Le multiplieur 18bits x 18bits est un multiplieur à chiffre signé. Chaque entrée est sur 18bits et la sortie est sur 36bits .Ce multiplieur est généré par le core générateur de circuit VIRTEX-I

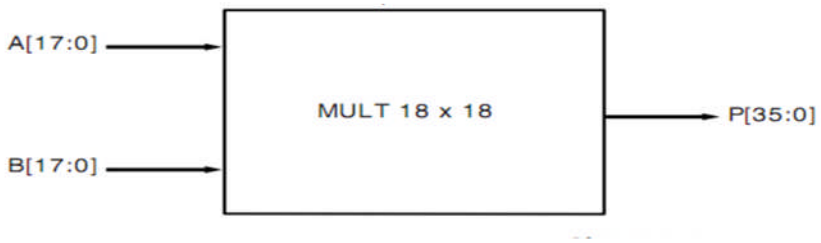


Figure II.13 : Multiplieur 18bits x18bits.

II.4.Conclusion :

Dans ce chapitre nous avons donné un aperçu générale sur les différents familles des circuits intégrés numérique ASICs et PLDs, ensuite nous avons fait la description générale des circuits FPGAs qui suscitent le plus d'intérêts car, d'une part ils présentent une meilleure flexibilité et fournissent le plus de fonctionnalités au concepteur et d'autre part les outils nécessaires à leurs développement sont de plus en plus disponibles et performants. Comme nous allons effectuer l'implémentation de l'architecture de notre circuit sur un FPGA de famille VIRTEX-II de Xilinx, nous avons fait une description détaillée de ce dernier.

Le chapitre suivant est consacré à la réalisation pratique des déférents blocs de la plateforme d'acquisition.

Chapitre III

III.1.Introduction :

Dans ce chapitre on se propose l'étude et la réalisation pratique de la plateforme d'acquisition. Cette plateforme est composée d'une partie analogique constituée d'un réseau de huit capteurs conditionnés .Chaque capteur assure la transformation de la variation du mesurande en un signal électrique dont la variation est exprimé en fonction du même mesurande. Le signal électrique issu du capteur filtré ensuite numérisé grâce à un convertisseur analogique numérique dont la résolution est sur huit bits parallèles.

Le signal numérique est acheminé vers l'ordinateur via une carte numérique à base d'un FPGA de famille Virtex –II assurant deux fonctions, la première est la commande d'un multiplexeur analogique dont la fonction est la sélection de la voie analogique a échantillonner, la deuxième est la liaison série (RS232) avec l'ordinateur. Le signal reçu par l'ordinateur est traité et visualisé par une application de de visualisation sous environnement Labview.

III.2.La partie analogique :

Cette carte est composée :

- D'une carte de multiplexage et conversion analogique –numérique.
- D'une carte de conditionnement du capteur de température LM335.
- D'une carte de conditionnement du capteur de pression MPX5100.
- D'une carte électronique d'un sonomètre.
- D'une carte d'alimentation +15v,-15v, +5v.

III.2.1.Carte de multiplexage et conversion analogique numérique :

Cette carte comporte huit supporte SIL08 pour l'insertion des cartes de conditionnement de chaque capteurs et un circuit intégré ADC0809 pour le multiplexage et conversion analogique numérique.

La première broche de chaque supporte est reliée à l'une des voies analogique de circuit intégré ADC0809, les quatre suivantes servent d'alimentation pour les déférentes cartes. La figure II.1 représente le schéma électrique de la carte.

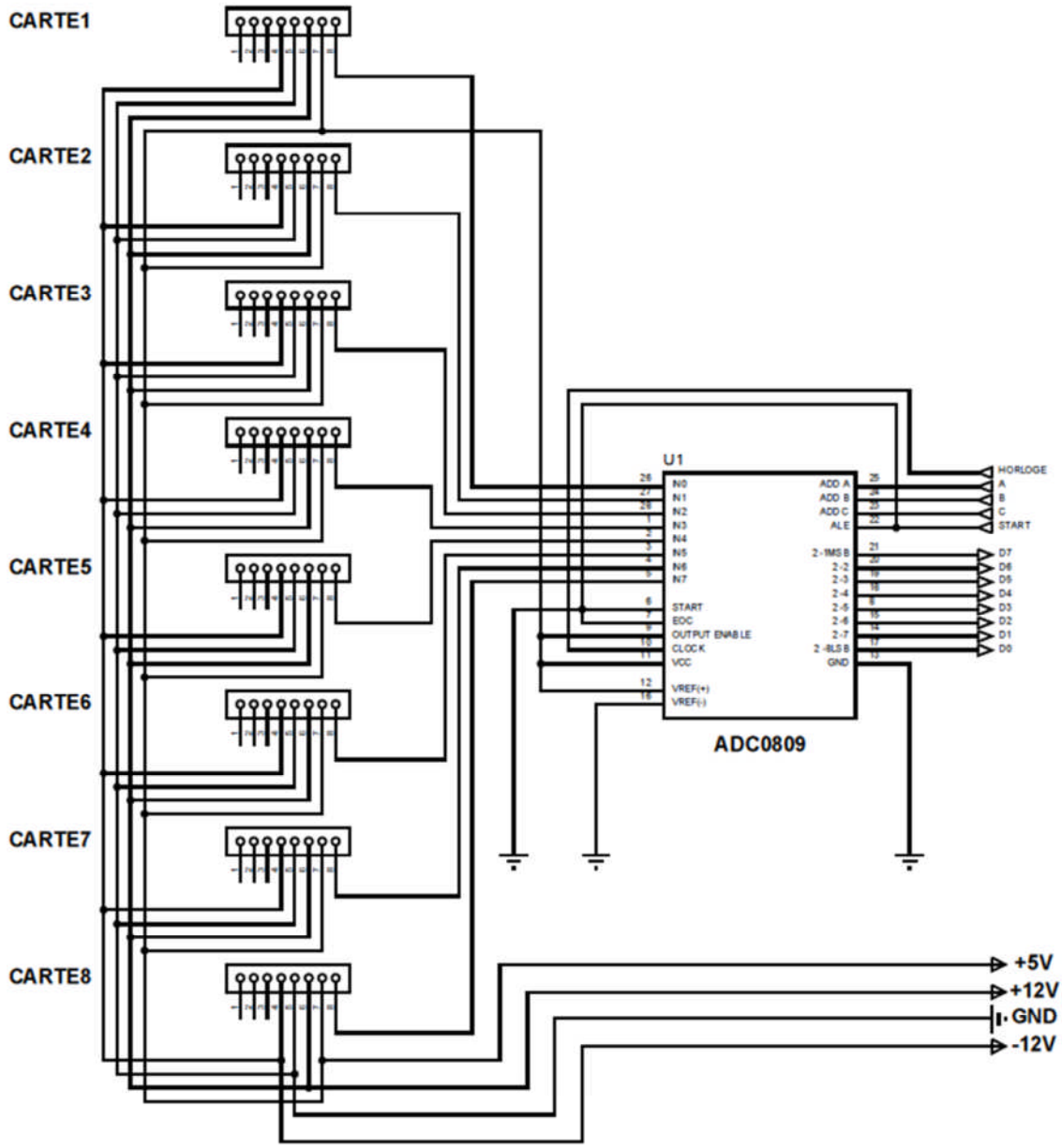


Figure III.1 : schémas électrique de la carte de conversion analogique numérique.

III.2.1.1.Présentation de l'ADC0809 :

Le composant ADC0809 est un circuit d'acquisition de données, comprenant un Convertisseur Analogique Numérique de 8 bits, un Multiplexeur de 8 voies et un circuit logique de contrôle compatible multiprocesseurs. La figure III.2 montre les déférents blocs de l'ADC0809.

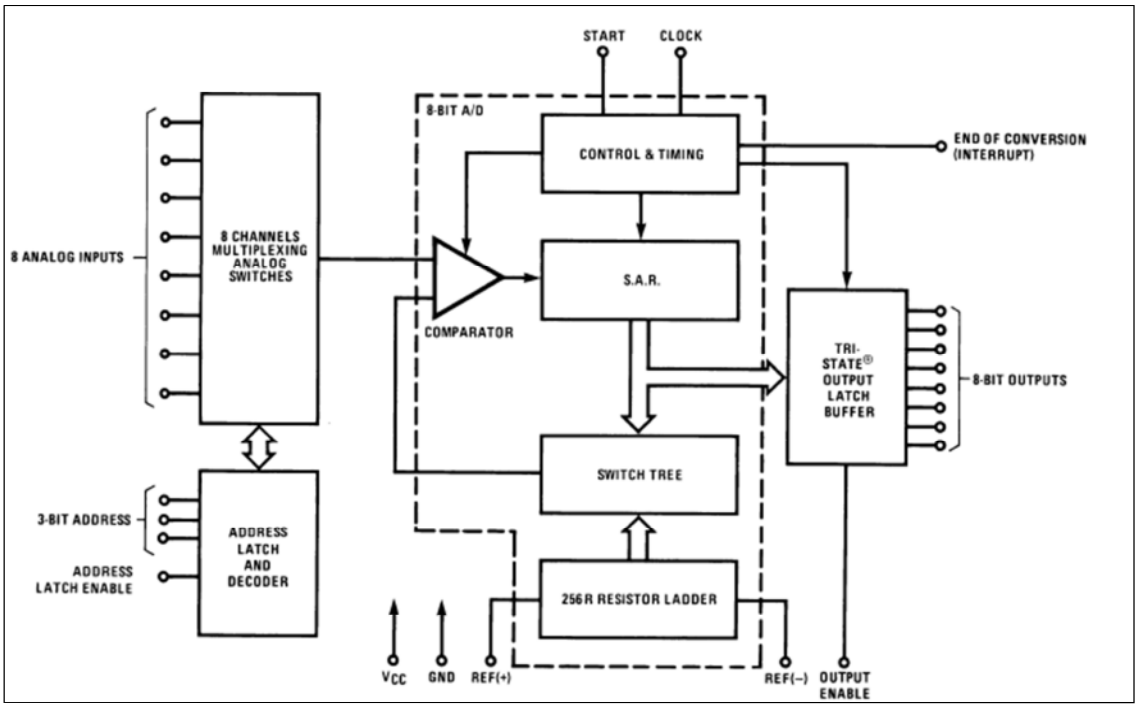


Figure III.2 : schéma bloc de l'ADC0809.

La technique de CAN utilisée est celle des approximations successives mettant en œuvre, outre la logique de contrôle, un réseau R/2R de résistances, un réseau de commutateurs analogique (Switch) ainsi qu'un comparateur.

Le multiplexeur a huit voies permet d'accorder à une quelconque entrée analogique parmi huit, selon le code binaire des adresses de poids faible 'A2 A1 A0'='C B A' présenté à l'entrée du décodeur d'adresses interne au composant. Le code des adresses est verrouillé sur une impulsion au niveau haut appliquée sur l'entrée ALE (Broche 22).

- **Valeurs limites :**

Tension d'alimentation VCC_{max} = 6,5V.

Tension sur chaque broche, exceptées celles de contrôle: de - 0,3V à VCC + 0,3V.

Tension des entrées de contrôle (START; 0E; CLOCK; ALE; ADDA; ADDB; ADDC) : de - 3V à +15V.

Puissance dissipée: 875 mW.

Fréquence d'horloge: 10 Khz < F_{CLK} <1280 Khz.

- **Caractéristiques:**

1) Tension d'alimentation VCC_{max} = 5V.
2) Résolution: 8bits.

3) Faible consommation : 15mW.
4) Durée d'un cycle de conversion : 100µs.
5) Plage de température de fonctionnement : de - 40°C à +85°C ou de -55°C à +125°C.

III.2.2.Carte de conditionnement du capteur de température LM335 :

- **Principe de fonctionnement :**

Le LM335 est un capteur de température intégré qui fournit une tension proportionnelle à la température (en kelvin).Sa sensibilité est de 10 mV / K. A 0°C (273 K), cela donne donc une tension continue de 2,73 V.

L'amplificateur de différence, construit autour d'un amplificateur opérationnel TL081, qui augmente la sensibilité d'un facteur 10 et assure un changement d'échelle. En sortie de l'amplificateur on obtient une tension de 100 mV / °C ce qui donne :

- 0 V à 0 °C
- 2 V à 20 °C
- 5 V à 50 °C

Le capteur LM335 est alimenté avec une tension +5v et l'amplificateur avec des tensions symétriques +15v, 0,-15v.La figure III.3 montre les schémas électrique de la carte de conditionnement du capteur.

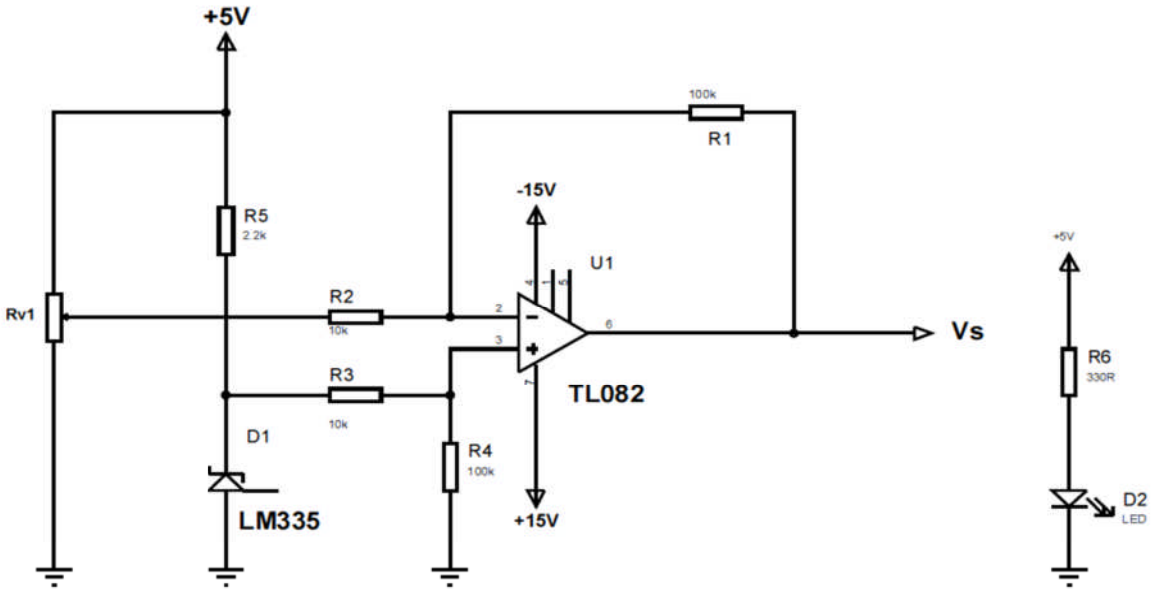


Figure III.3 : schémas électrique de la carte de conditionnement du capteur.

III.2.3.Carte de conditionnement du capteur de pression MPX5100 :

Le MPX5100AP est capteur résistif qui fournit, entre ses broches 2 et 3, une tension flottante proportionnelle à la pression ambiante. La figure III.4 montre la courbe caractéristique du capteur.

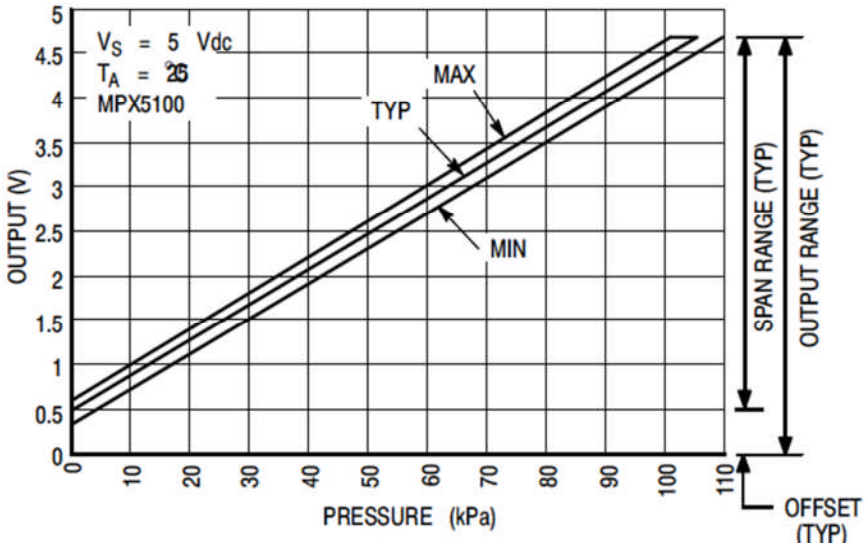


Figure III.4 : courbe caractéristique du capteur MPX5100.

La sensibilité du capteur dépend de sa tension d'alimentation. C'est pourquoi on utilise une alimentation (+12 V) stable pour alimenter le capteur. La sensibilité est alors d'environ 0,024 mV / hPa, soit 24 mV pour 1000 hPa.

Le signal de sortie du capteur est amplifié par un amplificateur de différence AD620 de manière à avoir une sensibilité de 4 mV / hPa, soit 4 V pour 1000 hPa.

L'amplification en tension correspondante est égale à : 4 V / 24 mV = 167 = 44 dB

L'amplification de l'amplificateur de différence AD620 est donnée par la relation :

A = 1 + (49400 / Rv1)

Ce qui donne théoriquement : Rv1 = 298 ohms.

En pratique, Rv1 est ajustée avec un trimmer de 500 ohms.

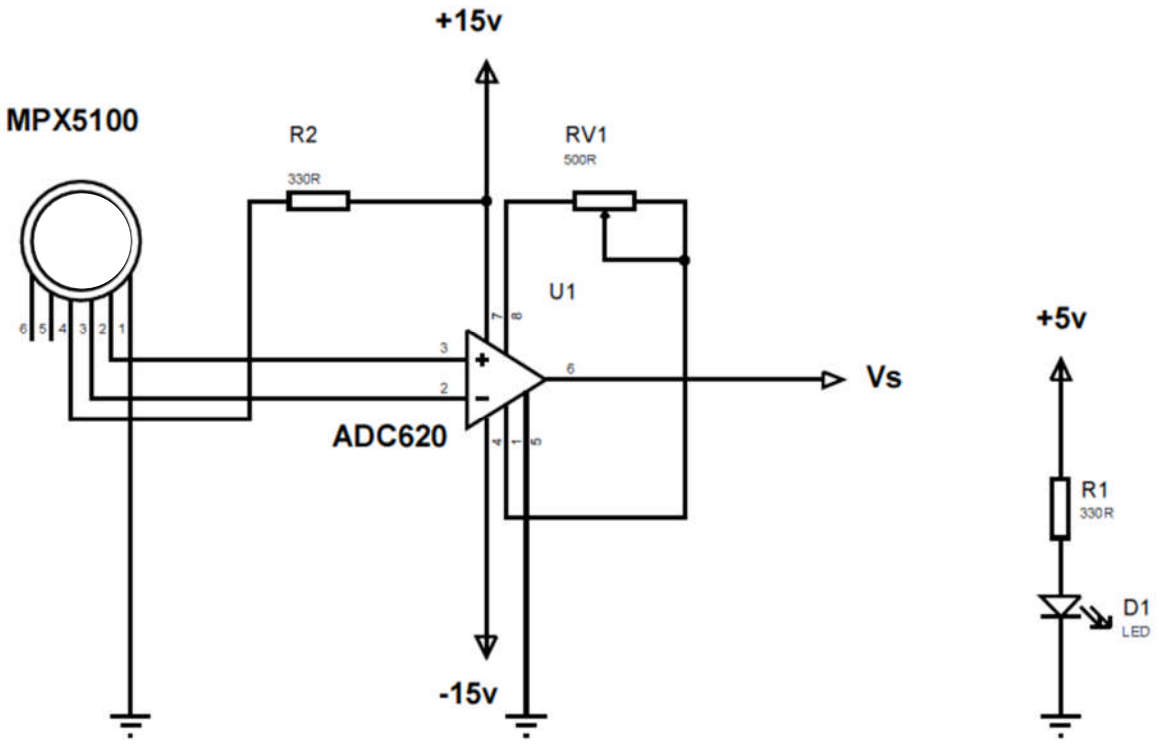


Figure III.5: schémas électrique de la carte de conditionnement du capteur.

III.2.4.Le sonomètre :

Il s'agit d'un dispositif électronique qui mesure en temps réel le niveau sonore en décibels .La plage de mesure varie de 20 dB à 100dB. Le capteur utilisé pour ce dispositif est un microphone Electric. La figure III.6 montre le schéma électrique de circuit de conditionnement du capteur sonore (microphone).

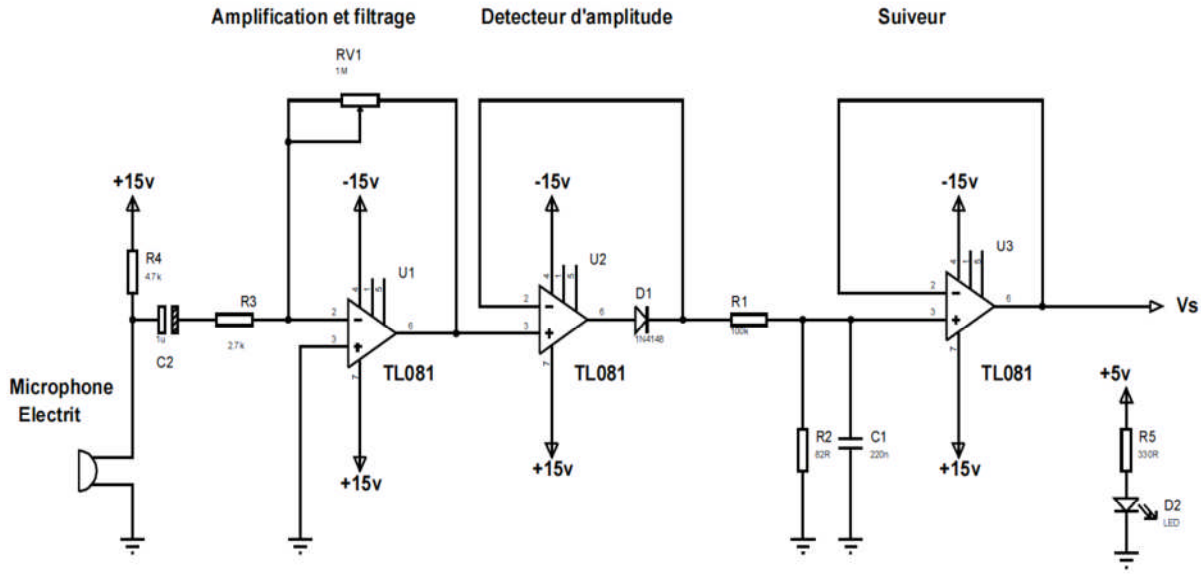


Figure III.6 : schéma électrique de la carte de conditionnement du capteur sonore.

III.2.4.1.Principe de fonctionnement :

III.2.4.1.2.Le capteur sonore (microphone) :

Le capteur (microphone) fournit une tension dont la composante alternative est proportionnelle à la pression acoustique du son.

La sensibilité du capteur s'exprime en dB avec la référence : 0dB⇔1V par µbar. Pour le microphone de type Electret, cette sensibilité est typiquement comprise entre -35 dB et -50dB.Pour une sensibilité de -40dB, cela correspond à 0.01v/µbar.

La bande passante du capteur (microphone) va de 50Hz à 10Khz.

III.2.4.1.3.Correspondance entre le niveau de tension de microphone et le niveau sonore :

Pour un microphone de sensibilité de -40dB la réponse est de 0.01v

niveau sonore	**pression acoustique efficace**	**tension efficace de la composante alternative** (broche 1 du microphone) **(sensibilité du microphone : - 40 dB)**
0 dB (seuil d'audibilité)	0,000 2 µbar	2 µV
20 dB	0,002 µbar	20 µV
40 dB	0,02 µbar	200 µV
60 dB	0,2 µbar	2 mV
80 dB	2 µbar	20 mV
100 dB	20 µbar	200 mV
120 dB (seuil de douleur)	200 µbar	2 V

III.2.4.1.4.Amplification et filtrage :

Le circuit U1 (TL081) et C2, R3, Rv forment un amplificateur sélectif de type passe-bande dont L'amplification est réglable avec le potentiomètre Rv (de type logarithmique) de façon à avoir 11 V crête en sortie pour un son de niveau 100 dB. L'amplification (gain) est donnée par la relation :

$$|A| = Rv/R3$$

III.2.4.1.5.Détecteur d'amplitude :

Le détecteur d'amplitude permet d'extraire l'enveloppe supérieure du signal alternatif délivré par l'amplificateur U1et cela à l'aide de la diode D.

III.2.4.1.6.Le suiveur :

Le suiveur fait simplement tampon entre le détecteur d'amplitude et l'entrée du convertisseur analogique-numérique. En sortie du suiveur, la correspondance entre tension et niveau sonore est :

- 5 V <=> 100 dB
- 500 mV <=> 80 dB
- 50 mV <=> 60 dB
- 5 mV <=> 40 dB
- 0,5 mV <=> 20 dB

III.2.5.Carte d'alimentation :

Cette carte délivre deux une tension (+5V) pour alimenter le convertisseur analogique numérique, et une tension symétrique (-15v, 0, +15v) pour alimenter les capteurs et les amplificateurs opérationnels.

Ces différentes tensions sont obtenues à partir de la tension de secteur (220V/50Hz) après les opérations de transformation, redressement, filtrage et régulation.

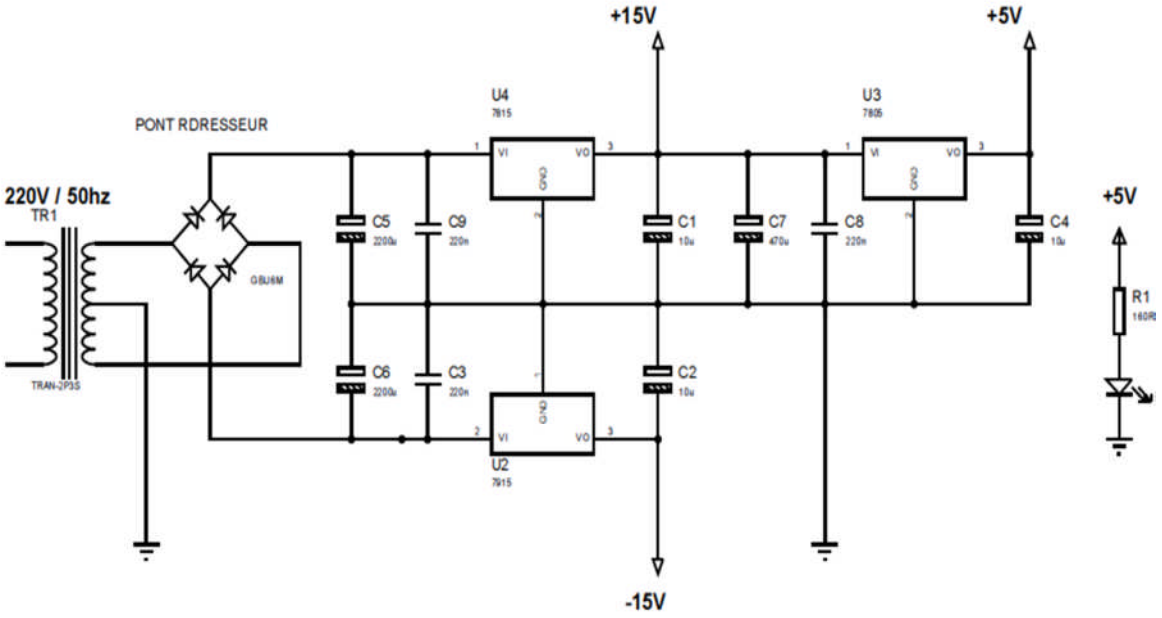


Figure III.7: Schéma électrique de la carte d'alimentation.

III.3.La partie numérique :

Cette partie est construite au tour d'une carte de développement FPGA de famille VIRTEX –II xc2v1000 utilisée pour l'implémentation d'un module UART dont la fonction est d'assurer la liaison série (rs232) avec l'ordinateur. La carte est dotée :

- D'un port RS232.
- De deux ports d'entrée / sortie.
- De huit Switch.
- De deux afficheurs sept segments.
- D'un circuit FPGA xc2v1000 d'une intégration d'un million de porte logiques.
- D'un port JTAG pour la programmation.

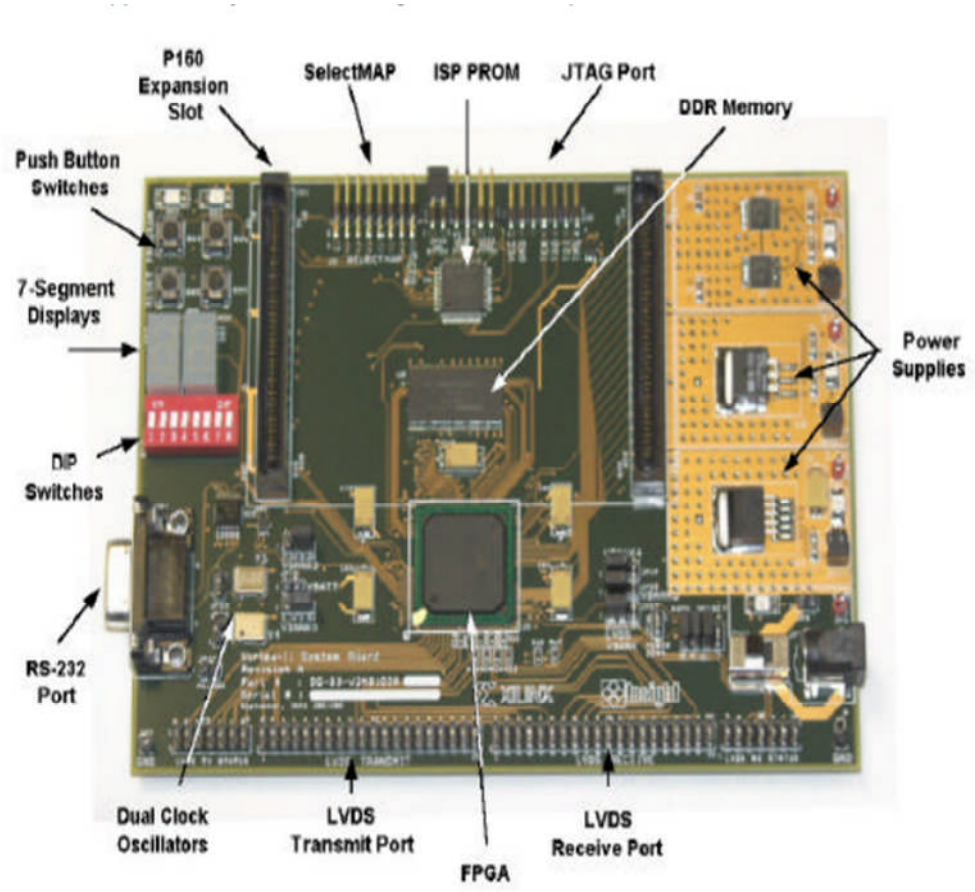


Figure III.8: Carte de développement FPGA VIRTEX –II

III.3.1.Principe de fonctionnement d'une liaison série RS232 :

La liaison série RS232 est une liaison sérié asynchrone qui assure la transmission de données entre deux dispositifs. Son principal avantage réside dans le fait que le nombre de fils nécessaire à la transmission est réduit. Elle peut être effectuée en utilisant uniquement trois fils (TxD, RxD et la masse).l'octet à transmettre est envoyé bit par bit par l'émetteur sur la ligne TxD, vers le récepteur RxD qui le reconstitue. L'émission d'une donnée de 8 bits est effectuée à travers la transmission d'une trame de 11 bits, Le premier bit est le bit *start* en suite Les huit bits de données après le bit de *parité* et le dernier est le bit *stop*.

Figure III.9: format d'une trame envoyée par liaison sérié

La transmission peut se faire dans les deux sens, émission et réception simultanées. Le niveau logique bas du bit *Start* indique au récepteur qu'une nouvelle séquence de données arrive. ceci a pour effet que le récepteur va considérer les huit bits suivant comme étant des bits de données et le bit suivant les huit bits de données comme bit de parité .le bit *stop* de niveau logique haut est utilisé pour indiquer la fin de la trame. Le bit de parité peut être utilisé comme étant un bit paire ou impaire pour indiquer qu'il n'y a pas d'erreur dans l'octet de données.

III.3.2.Module de liaison série UART :

Le protocole de communication sérié sur FPGA est assuré par une interface UART (*Universal Asynchronous Receiver Transmitter*).l'interface UART implanté sur la carte FPGA est constitué de deux modules : un module de réception et un module de transmission.

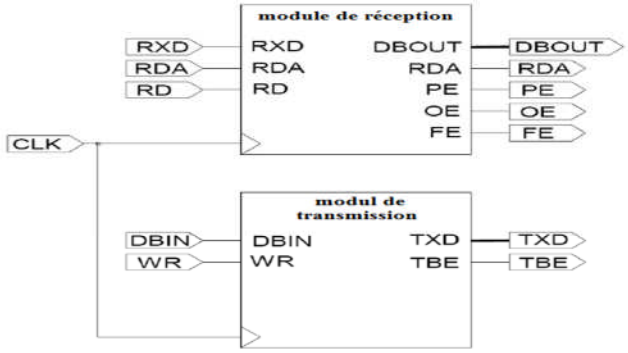


Figure III.10: module UART

III.3.2.1.Le module de réception :

Ce module de l'UART reçoit une donnée sérié disponible sur le port RxD et la convertie en une donnée parallèle placée sur le port DBOUT. Le module de réception comprend un contrôleur de donnée sérié, deux compteurs utilisés pour la synchronisation, un registre à décalage et un contrôleur de bits d'erreur.

- Le registre à décalage est utilisé pour la sauvegarde des données qui arrive à travers RxD.
- Le contrôleur de données sérié est utilisé pour la synchronisation de la phase d'acquisition de données au moyen d'un compteur pour la temporisation entre deux bits ressues et un compteur de bits pour garder la trace de nombre de bits ressues.
- Le contrôler de bit d'erreur analyse la donnée ressue pour trois type d'erreur : l'erreur de parité, erreur de format et l'erreur de débordement.

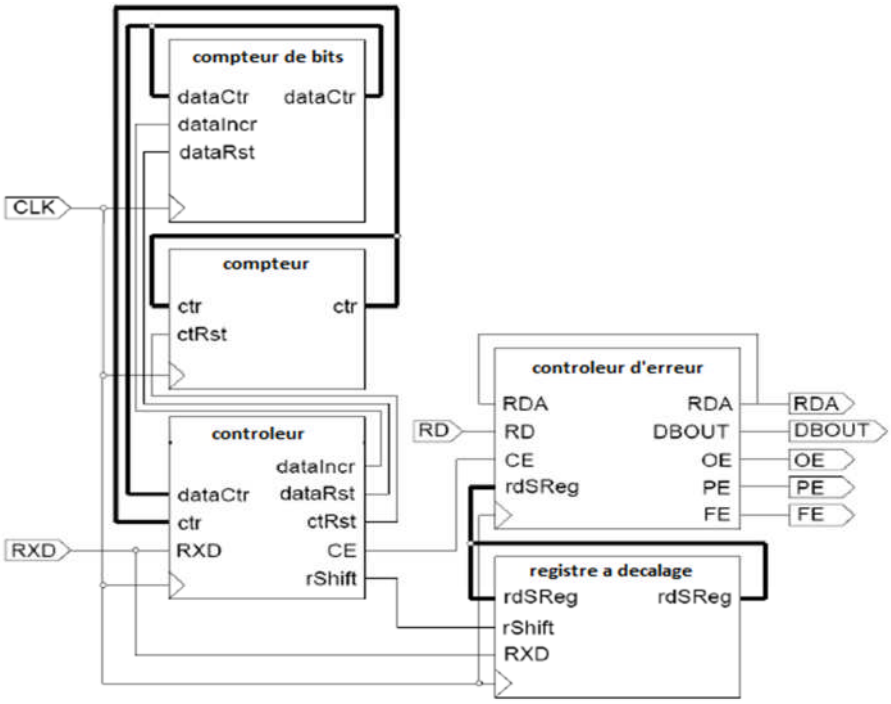


Figure III.11: module de réception

III.3.2.2.Le module de transmission :

Ce module a pour fonction de transmettre la donnée parallèle disponible sur le port DBIN en série via TxD vers le PC. Le baude rate de la transmission est le même que celui de la réception.

Pour transmettre l'octet disponible sur DBIN, le module de transfert dispos d'un contrôleur de transfert, deux compteurs de synchronisation et un registre a décalage.

Le contrôleur de transfert utilise les deux compteurs de synchronisation pour contrôler la vitesse à laquelle l'octet de donnée doit être transmis à travers TxD et le nombre de bit à transmettre.

Le port TxD correspond au bit le moins significatif du registre à décalage. La transmission de la donnée ce fait par un simple décalage à droite de registre a décalage.

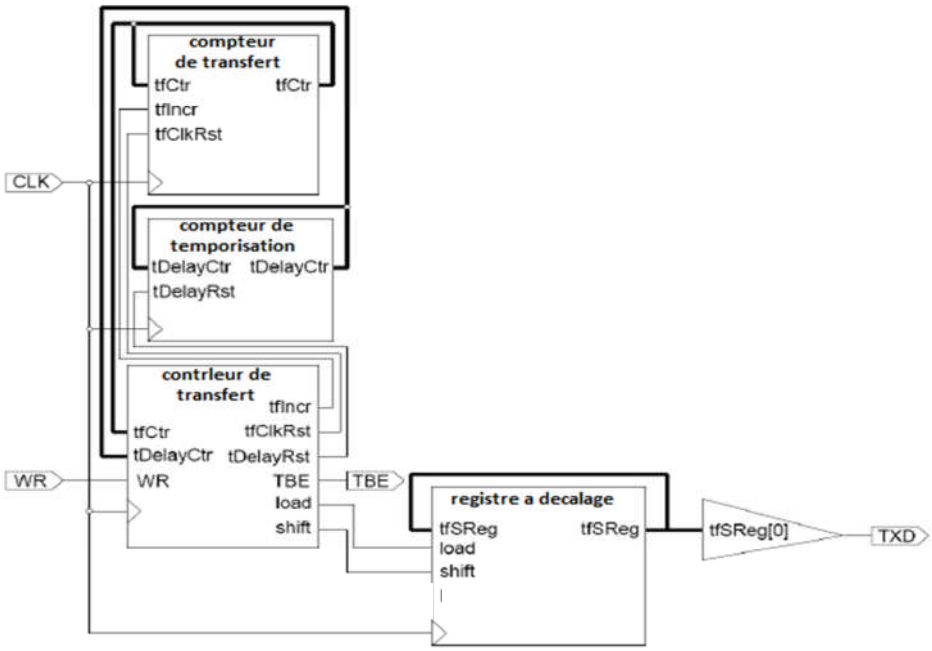


Figure III.12: module de transmission

III.3.2.3.La description VHDL de l'UART :

Les deux modules composant l'UART sont décrits par deux machines d'états comme le montre la figure III.3, une machine d'état de transmission et une machine d'état de réception.

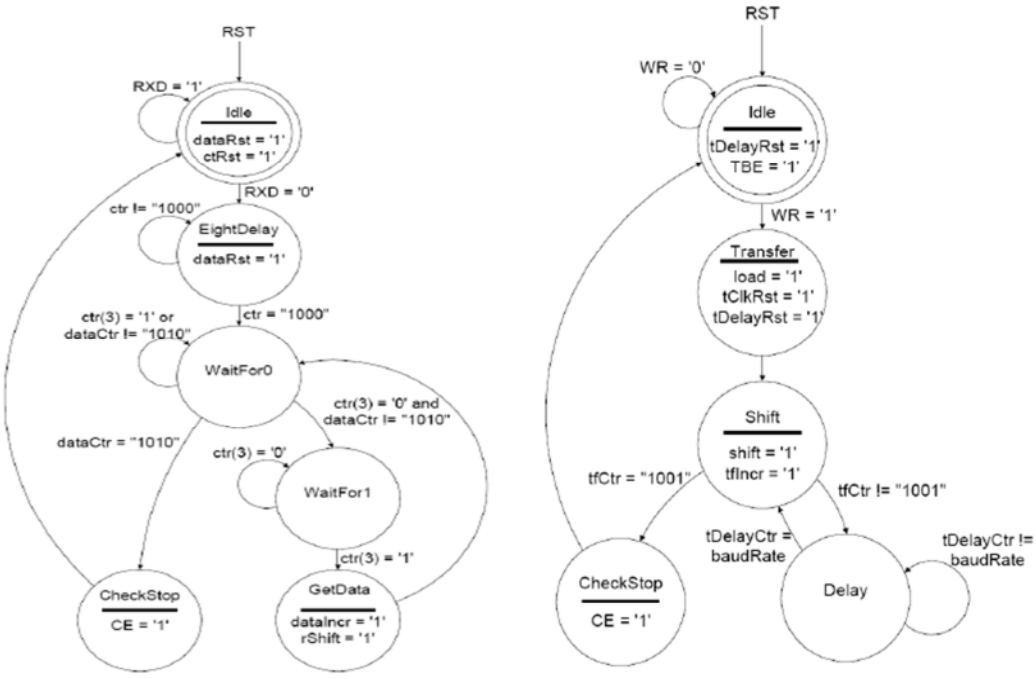


Figure III.13 : a) machine d'état de réception b) machine d'état de transmission

Le code VHDL suivant traduit les machines d'états de réception et transmission

```
--------------------------------------------------------------------------------
-- Auteur : Ait Benali Abdelghali
--          Ait Benali M'hand
--Projet de fin d'étude: conception d'une carte d'acquisition universel
--------------------------------------------------------------------------------

--Déclaration des librairies
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity UART is
   Port ( TXD   :out std_logic   := '1';
          RXD   :in  std_logic;
          CLK   :in  std_logic;                        --horloge principal
          DBIN  :in  std_logic_vector (7 downto 0);  --Data Bus in
          DBOUT :out std_logic_vector (7 downto 0);  --Data Bus out
          WR    :in  std_logic;                       --Write Strobe
          RST   :in  std_logic   := '0');            --Reset
end UART;

architecture Behavioral of original is
```

```
-------------------------------------------------------------------------
-- Déclaration des machines d'états
-------------------------------------------------------------------------

--Machine d'état de réception

     type rstate is (
          strIdle,              -- état idle
          strEightDelay,        --temporisation de 8cycles d'horloge
          strGetData,           --sauvegarde des 8 bits de données
          strCheckStop          --retour à l'état de repos
     );

--Machine d'état de transfert

     type tstate is (
          sttIdle,              --état idle
          sttTransfer           --chargement de la donnée dans
                                --le registre a décalage
          sttShift              --décalage a droit de registre
          );

-------------------------------------------------------------------------
-- Déclaration des signaux
-------------------------------------------------------------------------

--diviseur de baude rate.
   constant baudDivide : std_logic_vector(6 downto 0)   := "1101101";


--Registre de réception
     signal rdReg     :  std_logic_vector(7 downto 0)   := "00000000";

--Registre à décalage de réception
     signal rdSReg    :  std_logic_vector(8 downto 0)   := "111111111";

--Register de transmission

     signal tfReg     :  std_logic_vector(7 downto 0);

--Register à décalade transmission
     signal tfSReg    :  std_logic_vector(9 downto 0) := "1111111111";

--compteur utilisé par rClk
     signal clkDiv    :  std_logic_vector(7 downto 0)   := "00000000";

--compteur utilisé par tClk
     signal rClkDiv   :  std_logic_vector(3 downto 0)   := "0000";

--Compteur utilisé  pour la temporisation
     signal ctr :  std_logic_vector(3 downto 0)         := "0000";

--Compteur utilisé pour la temporisation dans la transmission
     signal tfCtr     :  std_logic_vector(3 downto 0)   := "0000";

--Horloge de réception
     signal rClk:  std_logic       := '0';

--Horloge de transmission
```

```
      signal tClk:  std_logic;

--Conte le nombre de bits lu
      signal dataCtr   :  std_logic_vector(3 downto 0)   := "0000";

--Enclenche le bloc de contrôle d'erreur
      signal CE        :  std_logic;

      signal ctRst     :  std_logic := '0';
      signal load      :  std_logic := '0';
      signal shift     :  std_logic := '0';
      signal tClkRST   :  std_logic := '0';
      signal rShift    :  std_logic := '0';
      signal dataRST   :  std_logic   := '0';
      signal dataIncr  :  std_logic   := '0';

--Etat présent de la machine d'état de réception
      signal strCur    :  rstate  := strIdle;


--Etat suivant de la machine d'état de réception
      signal strNext   :  rstate;

--Etat présent de la machine d'état de transmission
      signal sttCur    :     tstate := sttIdle;

--Etat suivent de la machine d'état de transmission
      signal sttNext   :  tstate;


-------------------------------------------------------------------------
-- Fonction du module
-------------------------------------------------------------------------

begin
      DBOUT <= rdReg;
      tfReg <= DBIN;

--Fonction de diviseur d'horloge

      process (CLK, clkDiv)

            begin
                  if (Clk = '1' and Clk'event) then
                        if (clkDiv = baudDivide) then
                              clkDiv <= "00000000";
                        else
                              clkDiv <= clkDiv +1;
                        end if;
                  end if;
            end process;

      process (clkDiv, rClk, CLK)

--Definition de rclk

      begin
            if CLK = '1' and CLK'Event then
```

```
                if clkDiv = "1101101" then
                      rClk <= not rClk;
                else
                      rClk <= rClk;
                end if;
          end if;
     end process;

     process (rClk)
          begin
                if (rClk = '1' and rClk'event) then
                      rClkDiv <= rClkDiv +1;
                end if;
          end process;

     tClk <= rClkDiv(3);

--Definition de tclk

     process (rClk, ctRst)
          begin
                if rClk = '1' and rClk'Event then
                      if ctRst = '1' then
                            ctr <= "0000";
                      else
                            ctr <= ctr +1;
                      end if;
                end if;
          end process;

     process (tClk, tClkRST)
          begin
                if (tClk = '1' and tClk'event) then
                      if tClkRST = '1' then
                            tfCtr <= "0000";
                      else
                            tfCtr <= tfCtr +1;
                      end if;
                end if;
          end process;

 --Processes de contrôle d'erreurs

     process (rClk,CE)
          begin

                if rClk = '1' and rClk'event then
                      if CE = '1' then

                            rdReg(7 downto 0) <= rdSReg (7 downto 0);
                      end if;
                end if;
          end process;

--Processus de contrôle de registre a décalage de réception

     process (rClk, rShift)
          begin
```

```
                if rClk = '1' and rClk'Event then
                      if rShift = '1' then
                            rdSReg <= (RXD & rdSReg(8 downto 1));
                      end if;
                end if;
          end process;

--Processus de contage de nombre de bits ressues

     process (rClk, dataRST)
          begin
                if (rClk = '1' and rClk'event) then
                      if dataRST = '1' then
                            dataCtr <= "0000";
                      elsif dataIncr = '1' then
                            dataCtr <= dataCtr +1;
                      end if;
                end if;
          end process;

--Machine d'état de réception

     process (rClk, RST)
          begin
                if rClk = '1' and rClk'Event then
                      if RST = '1' then
                            strCur <= strIdle;
                      else
                            strCur <= strNext;
                      end if;
                end if;
          end process;

--Processus de gestion de réception de données

     process (strCur, ctr, RXD, dataCtr, rdSReg, rdReg)
          begin
                case strCur is

                      when strIdle =>
                            dataIncr <= '0';
                            rShift <= '0';
                            dataRst <= '0';

                            CE <= '0';
                            if RXD = '0' then
                                  ctRst <= '1';
                                  strNext <= strEightDelay;
                            else
                                  ctRst <= '0';
                                  strNext <= strIdle;
                            end if;

                      when strEightDelay =>
                            dataIncr <= '0';
                            rShift <= '0';
                            CE <= '0';
```

```
                        if ctr(2 downto 0) = "111" then
                              ctRst <= '1';
                              dataRST <= '1';
                              strNext <= strGetData;
                        else
                              ctRst <= '0';
                              dataRST <= '0';
                              strNext <= strEightDelay;
                        end if;

                  when strGetData =>
                        CE <= '0';
                        dataRst <= '0';
                        if ctr(3 downto 0) = "1111" then
                              ctRst <= '1';
                              dataIncr <= '1';
                              rShift <= '1';
                        else
                              ctRst <= '0';
                              dataIncr <= '0';
                              rShift <= '0';
                        end if;

                        if dataCtr = "1001" then
                              strNext <= strCheckStop;
                        else
                              strNext <= strGetData;
                        end if;

                  when strCheckStop =>
                        dataIncr <= '0';
                        rShift <= '0';
                        dataRst <= '0';
                        ctRst <= '0';
                        CE <= '1';
                        strNext <= strIdle;

            end case;

      end process;

-- Machine d'état de transfer

      process (tClk, RST)
            begin
                  if (tClk = '1' and tClk'Event) then
                        if RST = '1' then
                              sttCur <= sttIdle;
                        else
                              sttCur <= sttNext;
                        end if;
                  end if;
            end process;

-- Processes de gistion de la transmission

      process (sttCur, tfCtr, tfReg, tclk)
            begin
```

```
                case sttCur is

                      when sttIdle =>
                            tClkRST <= '0';
                            shift <= '0';
                            load <= '0';
                                  sttNext <= sttIdle;
                            else
                                  sttNext <= sttTransfer;
                            end if;

                      when sttTransfer =>
                            shift <= '0';
                            load <= '1';
                            tClkRST <= '1';
                            sttNext <= sttShift;


                      when sttShift =>
                            shift <= '1';
                            load <= '0';
                            tClkRST <= '0';
                            if tfCtr = "1011" then
                                  sttNext <= sttIdle;
                            else
                                  sttNext <= sttShift;
                            end if;


                end case;
          end process;

End Behavioral;
```

III.3.2.4.Simulation du fonctionnement de l'UART :

Les résultats de la simulation sont donnés ci-dessous :

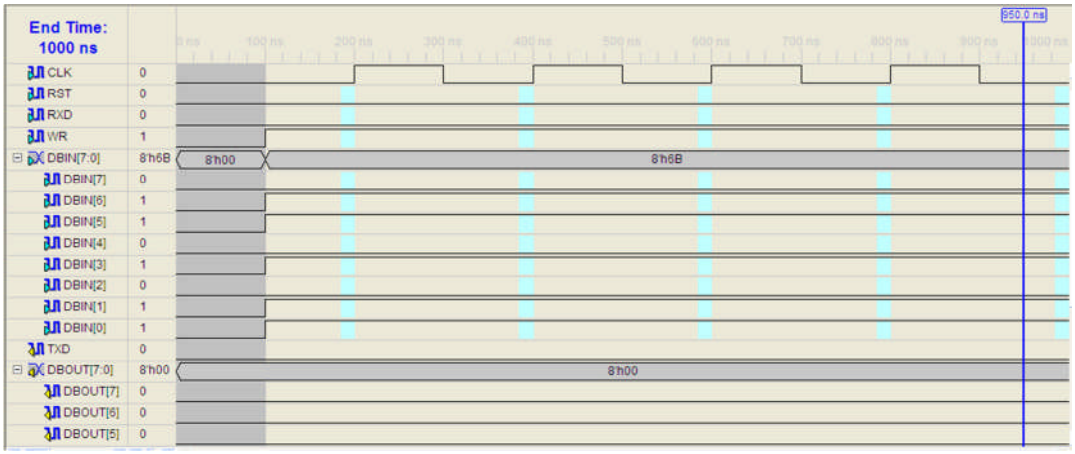


Figure III.14: Test Bench de l'UART.

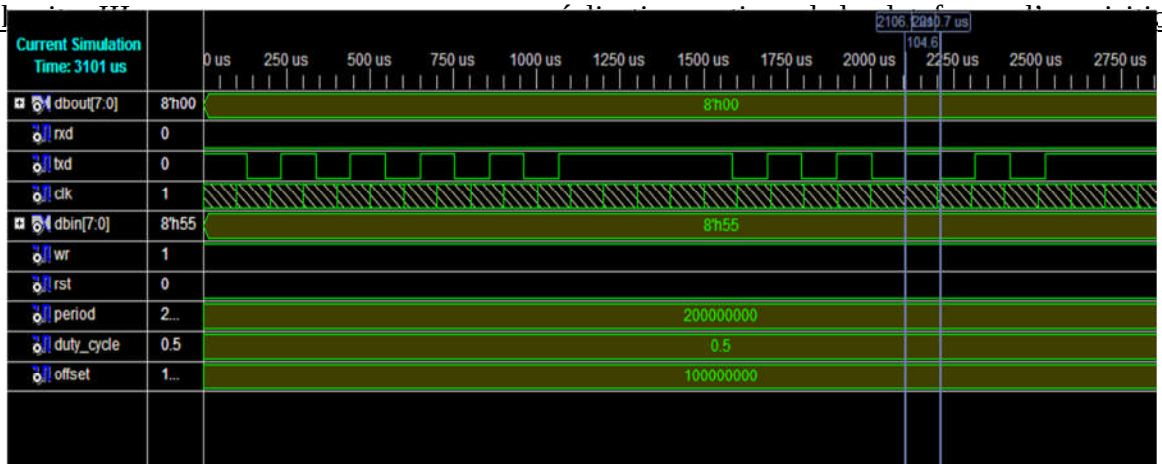


Figure III.15 : Simulation de l'UART **.**

III.2.5.Schéma de l'UART :

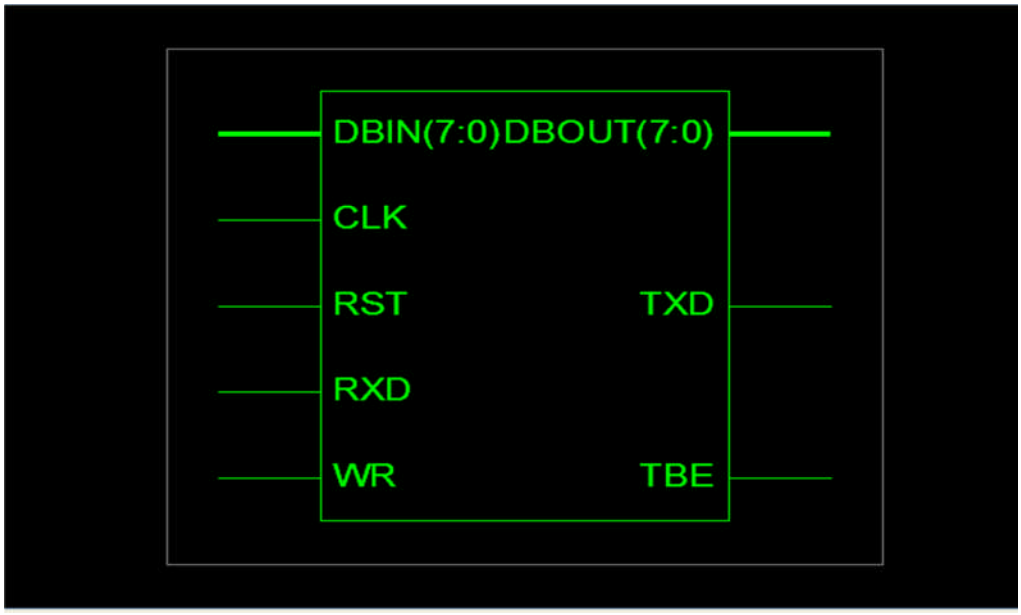


Figure III.16: Vue globale de l'UART.

III.3.2.6.Vu interne de l'UART :

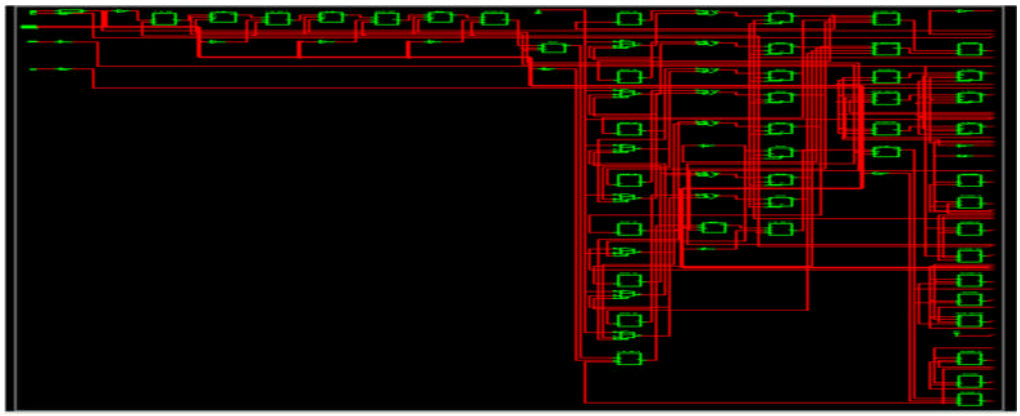


Figure III.16: Vue interne de l'UART.

Application de visualisation sur PC :

III.5.6.Transmission série dans LabView :

La transmission série utilise un émetteur pour envoyer les données les unes après les autres, bits par bit, sur une ligne de communication unique, à destination d'un récepteur.

Etant donné la simplicité de la liaison série, la bibliothèque pour le port série se résume à cinq instruments virtuels (VI) : Serial Port Init, Serial Port Write, Serial port Read, Bytes at Serial Port et Serial Port Close. Un programme de gestion de port série, réalisé en langage G dans l'environnement LabView consiste à mettre en séquence les déférents VI de base.

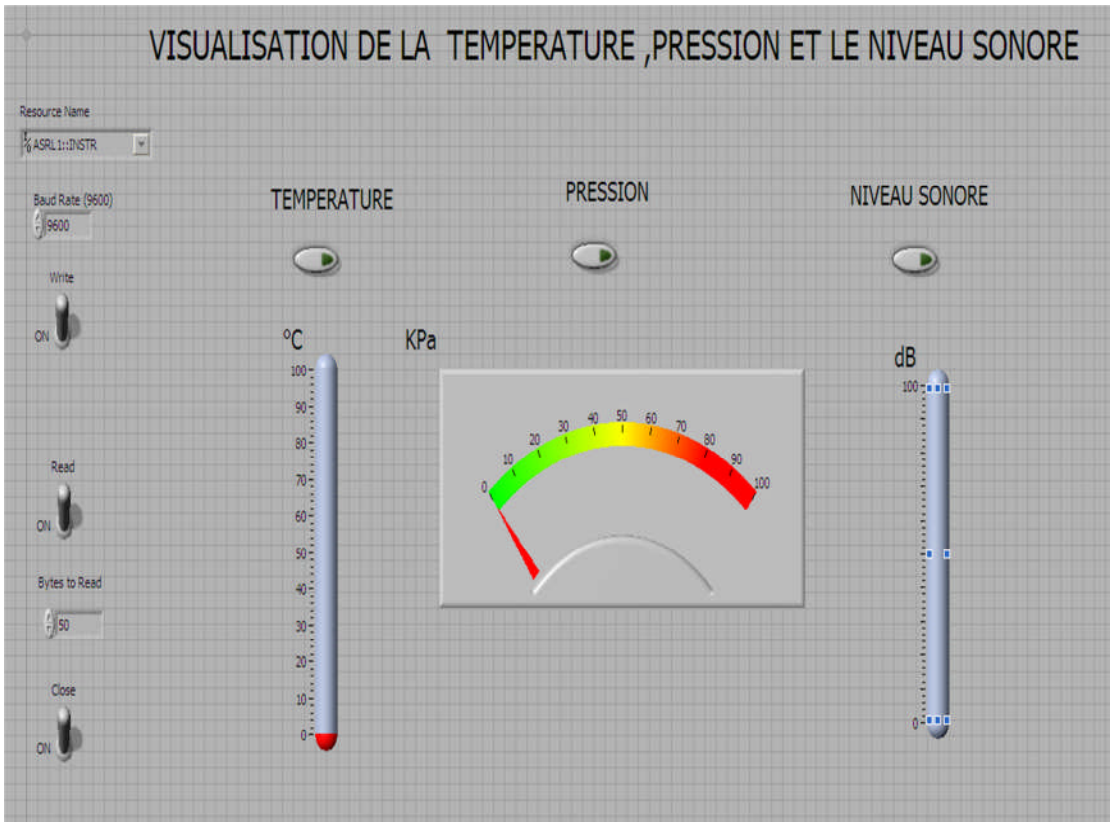


Figure III.17: Face avant de VI de visualisation.

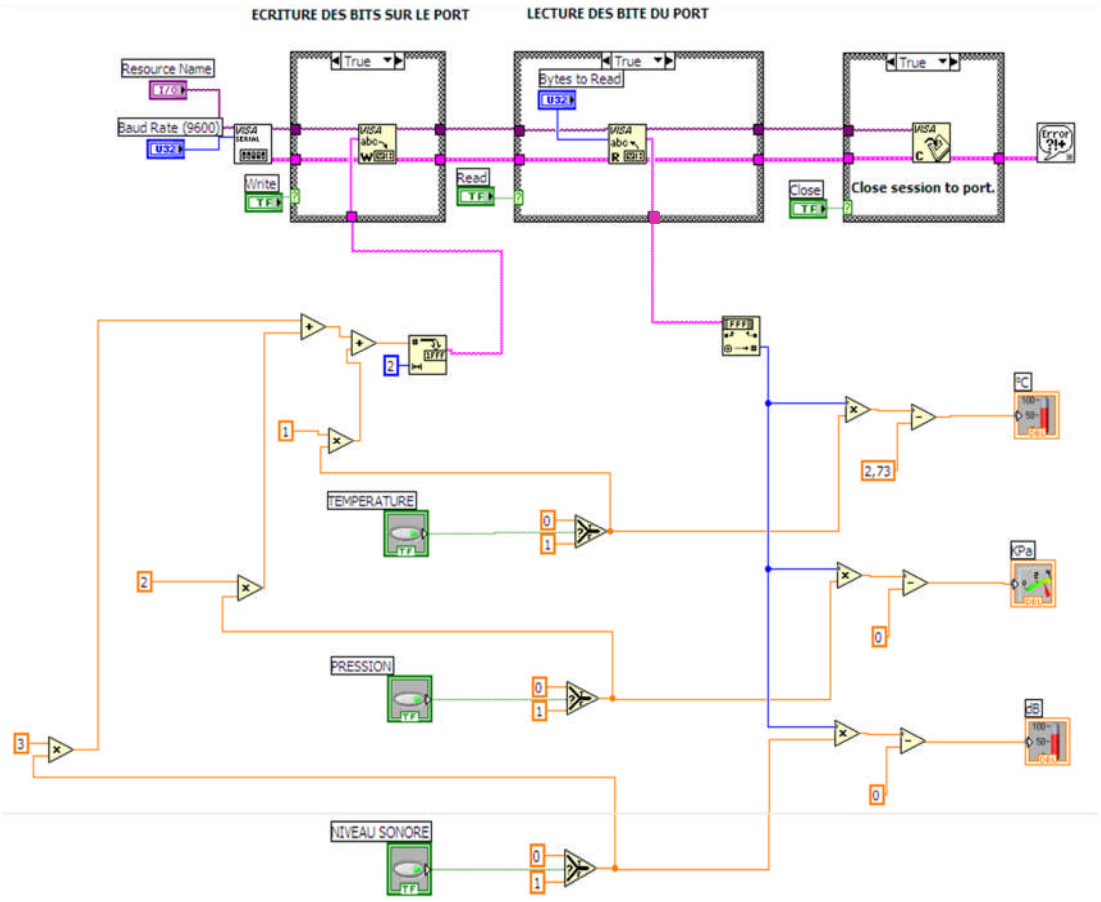


Figure III.18: Diagramme de VI de visualisation.

III.6.Conclusion :

Dans la première partie de ce chapitre on a présenté les caractéristiques et les schémas électriques des différentes cartes électroniques réalisées ainsi que le principe de fonctionnement de chaque une d'elles. Les cartes qu'on a réalisé sont la carte de conditionnement du capteur de pression MPX5100, la carte de conditionnement du capteur de température LM335, et celle du capteur de niveau sonore, ainsi que la carte de multiplexage et de conversion analogique numérique.

Dans la deuxième partie on a présenté le principe de fonctionnement et le code VHDL du module UART qu'on a implémenté dans un circuit FPGA de famille VIRTEX-II ainsi que les résultats de Testbench et de simulation, et pour l'affichage des résultats du mesure de pression ,température et de niveau sonore on a réalisé une application de visualisation sous environnement Labview .

Conclusion général :

Ce projet c'est accomplis par la conception d'un système d'acquisition de données, de transmission série asynchrone et d'affichage sur PC sous l'environnement Labview grâce à l'utilisation d'une carte de développement FPGA dont le principale avantage réside dans sa rapidité ce qui permis une acquisition en temps réel des données.

Ce travail nous a permis d'enrichir nos connaissances théoriques et pratiques acquises au cours de notre cursus d'études. En dépit des difficultés qu'on a rencontrées durant la concrétisation de ce projet, on a acquis une grande expérience en terme réalisation pratique des circuits électroniques et de programmation des circuits FPGAs.

On espère que cette thèse servira de base d'étude pour des systèmes plus performants d'une part et servir de moyen didactique pour les promotions à venir.

Bibiographie

Références bibliographiques :

- **Livres**

  - ✓ **G.ASCH et collaborateurs**. *Les capteurs en instrumentation industrielle.* (DUNOD, Paris, 1999).5eme édition.

  - ✓ **G.ASCH, E.chambirod, J.Gunther, P. Renard**. *Acquisition de données: Du capteur à l'ordinateur*. (DUNOD, Paris, 2003).2eme édition.

  - ✓ **Francis Cottet**. *LabView Programmation et application.* (DUNOD, Paris, 2001).

  - ✓ **Jacques Weber, Mourice Meaudre.** *Le langage VHDL Cours et exercices***.** (DUNOD, Paris, 2001).2eme édition.

  - ✓ **M.Laghrouche, M.berchiche.** *cour maquette*

- **Mémoire de fin d'études :**

  - ✓ **Etude et réalisation d'une chaine de mesure de CO_2,** par Zerioul Louis, promotion 2010, dirigé par Mr K.Bennamane.
  - ✓ **Implémentation d'un réseau de neurones d'un micro capteur sur FPGA,** par Mr Idir Mellal, promotion 2010, dirigé par Mr M.Laghrouche.
  - ✓ **Implémentation d'un variateur de vitesse à base d'un circuit FPGA,** Melle Naak Houria, Melle Hacid Hassina, Melle Houssouni Fadhila, promotion 2007, déridé par Mr Lekhlef.

- **Site internet :**

  - ✓ http://www.XILINX.com
  - ✓ http://www.digilent.com
  - ✓ http://www.xavier cotton.com

Annexe

ANNEXE 1 : Le langage VHDL

Introduction :

L'approche de conception de l'architecture à implémenter sur FPGA dans notre projet est par langage VHDL. le navigateur de projet ISE est utilisé comme outille de conception, cet outil de Xilinx permet de créer des projets comportant plusieurs types de fichiers (hdl, schématique,…etc.), de compiler ,d'effectuer des *design rule check*(drc) de créer des contrainte d'implémentation dont des contrainte de timing sur les horloges ,de déterminer l'emplacement des broches et de créer des bancs d'essai de simulation (testbench).

1. Présentation du langage VHDL:

L'abréviation *VHDL* signifie *VHSIC H*ardware *D*escription *L*anguage (*VHSI*: *V*ery *H*igh Speed *I*ntegrated *C*ircuit).ce langage a été écrit dans les années 70 pour réaliser la simulation des circuits électroniques. On l'a ensuite étendu pour permettre la conception (synthèse) de circuits logiques programmables (PLD. Programmable Logique Device).

2. Structure d'une description VHDL :

La syntaxe de langage VHDL est tirée du langage *Ada*, dont les mots clés ont été adaptés à la conception matérielle. L'une des particularités du VHDL provient de fait qu'il est possible d'exprimer facilement le parallélisme présent d'une structure d'un circuit.

3. Les différentes déclarations du langage VHDL :

3.1. Déclaration des bibliothèques.

La description VHDL utilisée pour la synthèse a besoin de bibliothèques. L'*IEEE* (*I*nstitut of *E*lectrical and *E*lectonics *E*ngineers).elles contiennent les définitions des types de signaux électroniques, des fonctions et sous-programme permettent de réaliser des opérations arithmétiques et logiques.

```
13  library IEEE;
14  use IEEE.STD_LOGIC_1164.ALL;
15  use IEEE.STD_LOGIC_ARITH.ALL;
16  use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

Figure 1 : déclaration des librairies

La directive **Use** permet de sélectionner les bibliothèques à utiliser.

3.2. Déclaration de l'entité et des entrées/sorties :

Elle permet de définir le **Nom** de la description **VHDL** ainsi que les entrées et sorties utilisées, l'instruction qui les défini est **Port.**

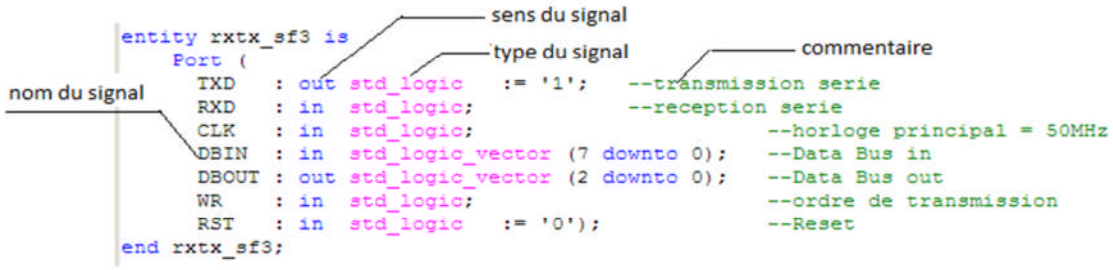


Figure 2 : déclaration de l'entité

3.2.1. Le nom du signal :

Il est composé de caractères, le premier caractère doit être une lettre, sa longueur est quelconque, mais elle ne doit pas dépasser une ligne de code.

3.2.2. Le **sens du signal** :

-*in* : pour un signal en entrée.

-*out* : pour un signal en sortie.

-*inout* : pour un signal en entrée sortie.

-*buffer* : pour un signal en sortie utilisé comme entrée dans la description.

3.2.3. Le type :

Le type utilisé pour les signaux d'entrées/sorties est :

-le *std_logic* : pour un signal.

-le *std_logic_vector* : pour un bus composé de plusieurs signaux.

3.3. Déclaration l'architecture :

L'architecture décrit le fonctionnement souhaité pour un circuit ou une partie du circuit. En effet le fonctionnement d'un circuit est généralement décrit par plusieurs modules VHDL.

L’architecture établit à travers les instructions les relations entre les entrées et les sorties. On peut avoir un fonctionnement purement combinatoire, séquentiel voire les deux séquentiel et combinatoire.

4. Les différents types de descriptions :

Une description de fonctionnement de circuit peut se faire de plusieurs manières :

4.1. Description structurelle « mapping » :

Dans ce type de description, il faut d’une part, spécifier les composants utilisés dans le circuit, et d’autre part, la manière dont ces composants sont connectés.

- **Partie déclarative**

Cette partie contient :

-la déclaration des signaux internes : ces signaux permettent de réaliser la connexion entre les différents composants. La déclaration ce fait par le mot clé **SIGNAL**, suivi du nom et le type du signal utilisé.

-la déclaration des composants utilisés : elle permet de lister les composant utiles pour la construction de l’entité et elle ce fait comme suite :

Component *nom_du_composant*

Port (…) ;

End component ;

- **Partie descriptive**

Représente la réalisation des différentes interconnexions des composants déclarés dans la partie déclarative. Chaque connexion est faite de la manière suivante :

nom_de_l_element : nom_du_composant **PORT MAP** (…) ;

4.2. Description comportementale

La description comportementale permet de décrire le fonctionnement de circuit à simuler par des équations logique qui relient ses sorties à ses entrées.

Le VHDL permet deux types de représentations comportementales :

a) Type DATAFLOW

Dans ce type de description, on modélise le circuit par un ensemble d'équations logiques on décrivant chaque sortie en fonction des entrées.

b) Type Algorithmique

Le contenue de l'entité dans ce type de description est décrit de façon algorithmique, en utilisant les structures de contrôle des langages de programmations.

- **la structure alternative**

IF *condition* **THEN**

déclarations séquentielles ;

ELSIF *condition* **THEN**

déclarations séquentielles ;

END IF;

- **la structure d'aiguillage**

CASE *entré_ de_command* **IS**

WHEN *etat_entree_ de _commande* => sortie **<=** *entrée_x;*

WHEN other =>*sortie***<=***entree_y;*

4.3. La description mixte

Elle regroupe les deux descriptions, structurel et comportementale.

ANNEXE 2 : Le navigateur de projet ISE 8.1.i :

Lors de la phase de synthèse de description VHDL chaque bloc sera matérialisés par des portes et /ou des bascules, suivi de la phase d'implémentation des portes et bascules dans le circuit logique.

Les différentes étapes de conception et de réalisation de circuits logiques est réalisée par les outils du navigateur de projet **ISE.**

1. Description du navigateur de projet ISE 8.1.i :

ISE 8.1.i est un environnement intégré de développement de systèmes numériques ayant pour but une implémentation matériel sur FPGA ou CPLD de la compagnie XILINX. Les désignes peuvent êtres décrit sous trois formes principales :

- Schéma.
- Langage de description matérielle (HDL) comme VHDL et Verilog …
- Diagramme d'états.

ISE intègre donc différant outils permettant de passer à travers tout le flot de conception d'un système numérique :

- Un éditeur de textes, de schémas et diagramme d'états.
- Un compilateur VHDL /Verilog.
- Un outil de simulation .des outils pour la gestion des contraintes temporelles.
- Des outils pour la synthèse.
- Des outils pour la vérification.
- Des outils pour l'implémentation sur FPGA.

2. Les étapes de simulation :

2.1.Lancement de l'ISE 8.1.i et création d'un projet :

1- le lancement *ISE 8.1.i* ce fait on cliquant deux fois sur l'icône 

sur le bureau ou en choisissons *Start* ⟶ *programme* ⟶ *Xilinx ISE 8.1.i* ⟶ *project navigator*.

3- On choisit *File* ⟶ *new project*

4- On donne un nom au projet dans la fenêtre apparue, puis on choisit *HDL* comme *Top_Level_source Type*

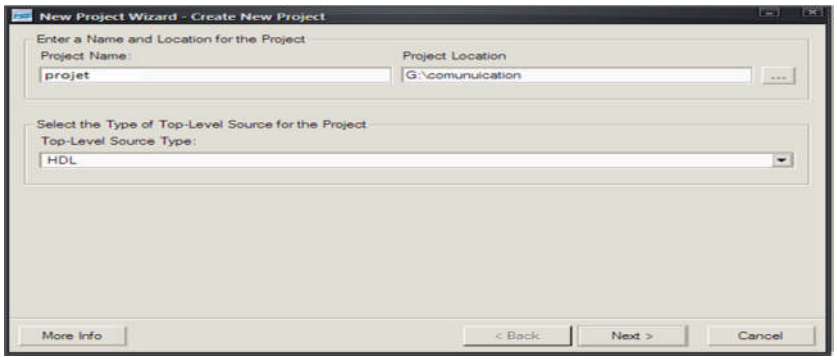


Figure 1 : création d'un nouveau projet.

5- Ensuite on clique sur *Next*.

6- On choisit les paramètres de FPGA utilisé :

Family : permet de choisir la famille du FPGA

Device : permet de sélectionner le sous-type (nombre de portes logiques).

Package : définit le type de boitier.

7- On appuie sur *Next* deux autre fois pour les deux autres fenêtres et sur *finish*.

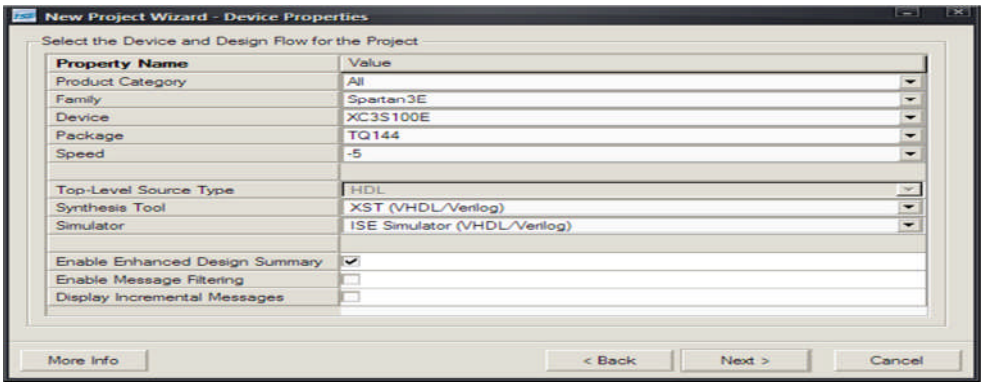


Figure 2 : spécification du type d'FPGA à utilisé

2.2. Description VHDL d'un circuit avec l'ISE :

1- On sélectionne le menu *Project* ⟶ *New Source***.**

2- On sélectionne *VHDL Module* comme source et on donne le nom du schéma à créer.

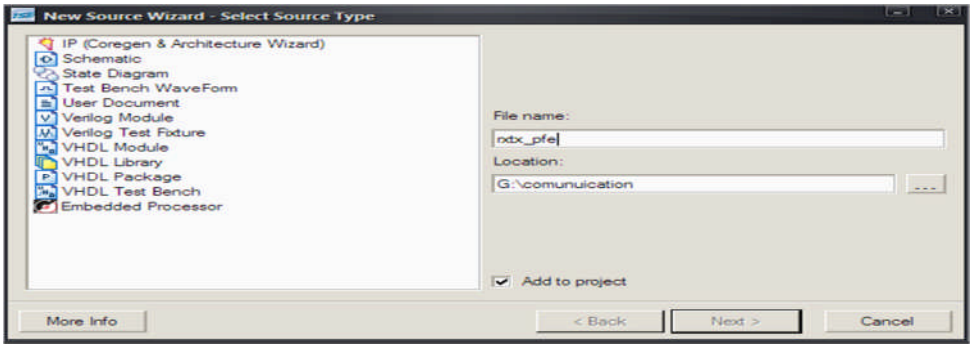


Figure 3 : choix de source de données.

3- On vérifie que *add to Project* est cochée.

4- On clique sur *Next***.**

5-On déclare les ports d'entrée / sortie du design comme définit ci –dessous.

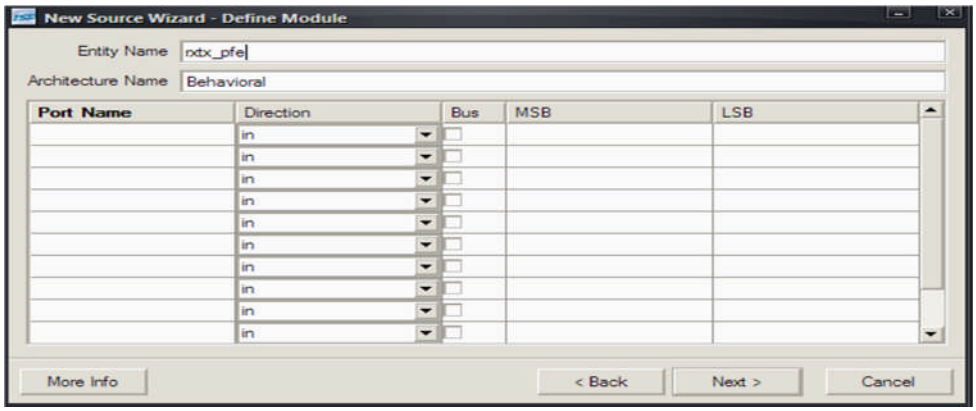


Figure 4 : définition des entrées/ sortie du module.

7- on clique sur *Next* en suite sur finish .une ébauche de fichier apparait avec la description de l'entité et de l'architecture.

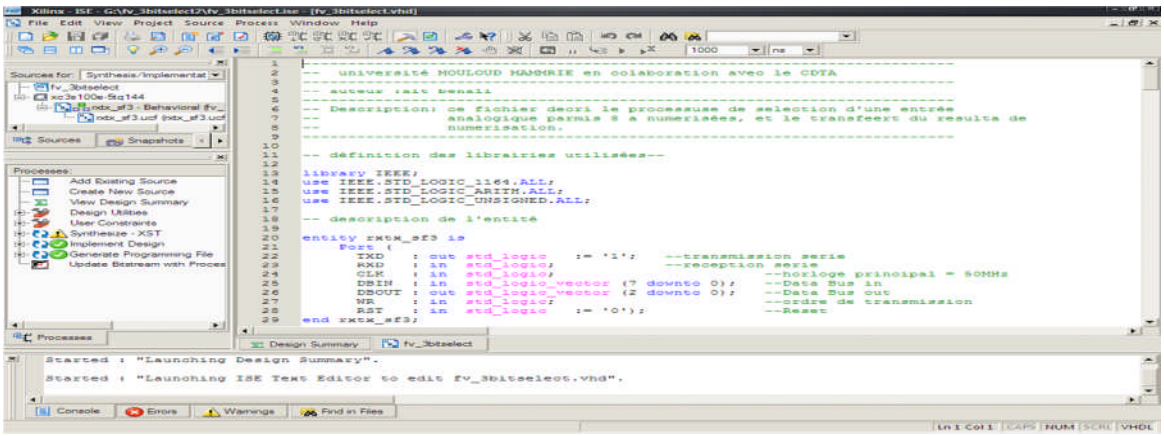


Figure 5 : édition du fichier VHDL.

2.3. Compilation :

Une fois que le fichier VHDL édité, il est impératif de vérifier la syntaxe du design afin de corriger des erreurs éventuelles :

a- on Vérifie que *synhesis /implémentation* est sélectionné dans la liste déroulante de la fenêtre *sources***.**

b- on sélectionne le fichier VHDL pour afficher les processus liés dans la fenêtre *sources*.

c- on Clique sur le « + » à côté de *synthezis-XST* dans la fenêtre *processes*

d- on Double –cliquez sur le process *check syntax*.si une erreur est détecter une croix rouge apparait à côté de l'onglet *check syntax* et un message en bas de l'écran indique la source de l'erreur si non, c'est un crochet vert qui apparait.

2.4. Simulation du design :

La simulation du design permet de savoir s'il fonctionne de la façon prévue par les spécifications. La simulation à faire dans cette étape est la simulation comportementale .pour simuler le design il faut créer un banc d'essai contenant les stimuli d'entrée.

Les étapes à suivre pour la simulation sont les suivantes :

1- on sélectionne le fichier à simuler dans la fenêtre *sources*.

2- on Crée un nouveau banc d'essai en sélectionnant *Project* puis *New source*.

3- Dans la fenêtre qui s'ouvre, on sélectionne *Test Bench Wave Forme***.**

4- on Clique sur *Next***.**

5- La page suivante montre quel fichier source est associe au banc d'essai ; il s'agit de notre fichier à simuler.

6- on Clique sur *Next*, puis *Finish***.**

7- Une fenêtre permettant d'effectuer une initialisation temporelle (détermination des paramètres d'horloge et de certaines contraintes temporelle) apparait .on coche *Single Clock***.**

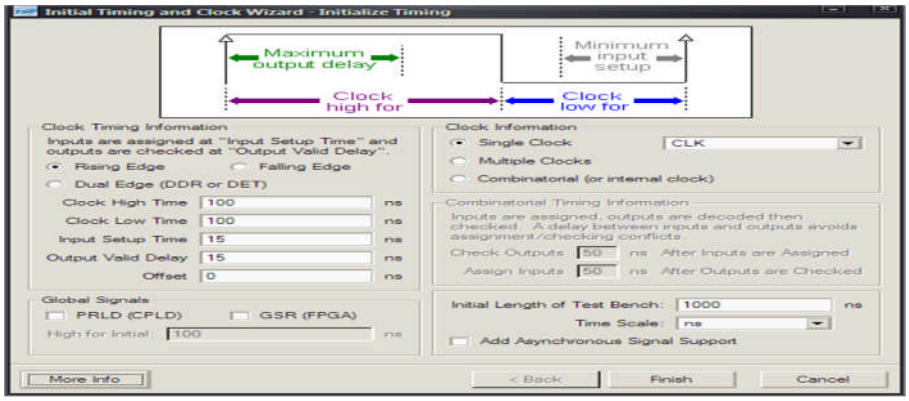


Figure 6 : initialisation temporelle.

8- on Modifie ensuite la durée de la simulation .puis on clique sur *finish* pour terminer l'initialisation temporelle.

9- Avec le bouton droit, on clique sur le début de la courbe (à coter de la zone grise) correspondant au signal ; et on sélectionne *Set Value***.**

10- Dans la fenêtre qui s'ouvre ,on choisit *Pattern Winzard*.

11- on Modifie les valeurs de *puls width et initial delay*.

12- on Clique sur **Ok**.la fenêtre suivante apparait.

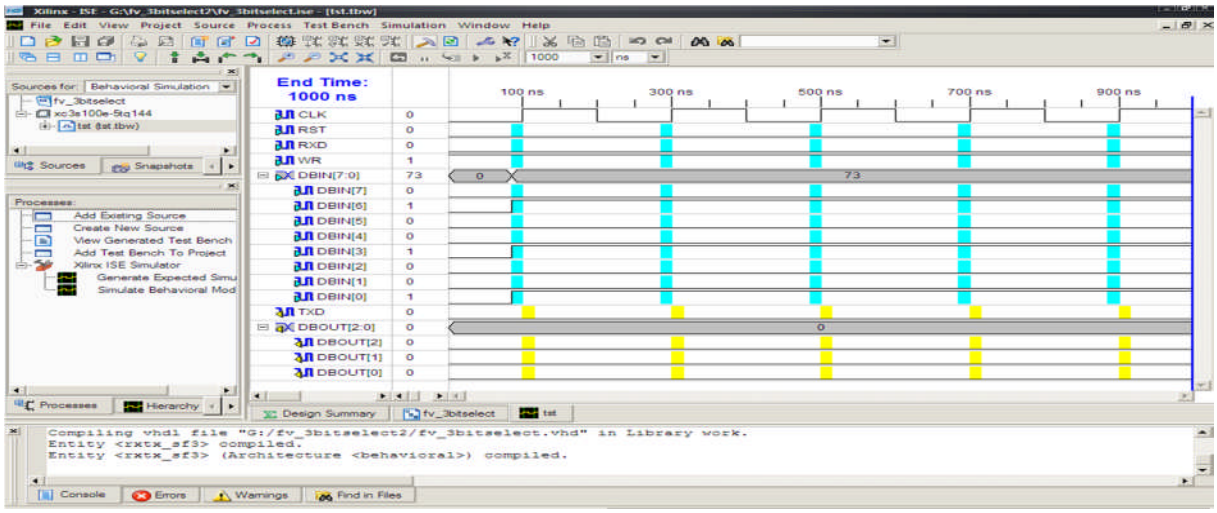


Figure 7 : génération d'un banc d'essai.

13- on Sauvegarde le banc d'essai avec *File >Save.*

14- Dans la fenêtre *sources* on sélectionne *behavioral simulation.*

15- on s'assure que le banc d'essai est bien sélectionné dans la fenêtre *sources.*

16- Dans la fenêtre *processes,* on déroule l'outil *Xilinx ISE simulator.*

17- on Double-clique sur *simulate behavioral model.*

18- on Corrige le design si on obtient des erreurs.

2.5. Implémentation du projet :

- **assignation des broches :**

1-dans la fenêtre process, on déroule le menu *User Contraints***,** on double clique ensuite sur *Assign Package Pins***.**

2-une fenêtre d'attribution de pins va suivre, dans la section *Loc.* saisir le numéro des broches du FPGA qu'on souhaite connecter aux E/S du désigne

Design Object List - I/O Pins

I/O Name	I/O Direction	Loc	Bank	I/O Std.
CLK	Input	a11	BANK0	
DBIN<0>	Input	b4	BANK0	
DBIN<1>	Input	a4	BANK0	
DBIN<2>	Input	c4	BANK0	
DBIN<3>	Input	c5	BANK0	
DBIN<4>	Input	b5	BANK0	
DBIN<5>	Input	a5	BANK0	
DBIN<6>	Input	d6	BANK0	
DBIN<7>	Input	c6	BANK0	
DBOUT<0>	Output	m2	BANK6	
DBOUT<1>	Output	n2	BANK6	
DBOUT<2>	Output	p2	BANK6	
DBOUT<3>	Output	r2	BANK6	
DBOUT<4>	Output	t2	BANK6	
DBOUT<5>	Output	u2	BANK6	
DBOUT<6>	Output	v2	BANK6	
DBOUT<7>	Output	w2	BANK6	
RST	Input	d7	BANK0	
RXD	Input	a7	BANK0	
TBE	InOut	f3	BANK7	
TXD	Output	b7	BANK0	
WR	Input	a6	BANK0	

Figure 8 : assignation des pins.

3-on sauvegarde les allocations avec la sélection de *File > Save*

- **génération du fichier de programmation :**

1- on sélectionne le fichier VHDL dans la fenêtre *sources* et on clique sur *Impliment Design*. C'est dans cette phase que le design est routé

3-dans la fenêtre *processes* on clique sur *Generat Programing File, un* symbole vert indique que les étapes de la synthèse sont bien déroulées.

- **programmation d'un FPGA avec le logiciel iMPACT :**

1-on connecte la maquette de développement au PC.

2-dans la fenêtre process on double-clique sur *Configure Device (iMPACT)*

3-on sélectionne le fichier de programmation (*nom_du_fichier.bit*)

4-on clique sur Ok dans la fenêtre *Programing properties*

5-l'ecran suivant indique que la programmation du FPGA s'est bien déroulée.

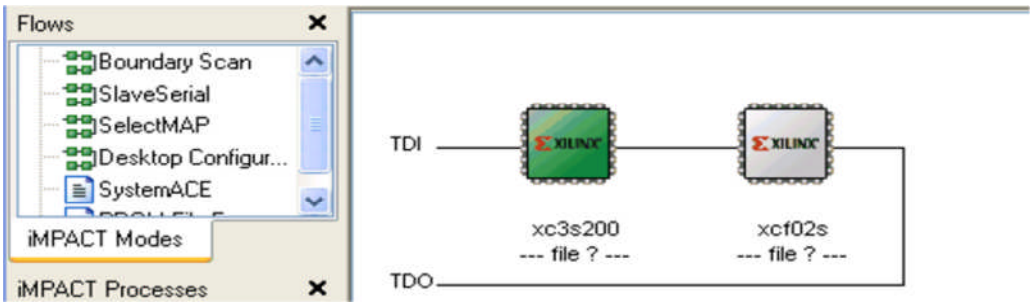


Figure 9 : programmation d'un FPGA avec iMPACT.

ANNEXE 3 : Acquisition de données avec LabView

1. Présentation du logiciel LabView :

LabView est un langage de programmation dédié au contrôle d'instrument et l'analyse de données. Contrairement à la nature séquentielle des langages textuels, LabView est basé sur un environnement de programmation graphique utilisant la notion de flot de données pour ordonnancer les opérations.

Le nom du logiciel LabView signifie « Laborararatory Virtual Instrument Engineering Workbench » ce langage de programmation graphique est un environnement de programmation à caractère universel adapté à la mesure , au test, à l'instrumentation et a l'automatisation il a été développé par la société National Instrument à partir de 1983 et c'est l'un des premier langage graphique destiné au développement d'applications d'instrumentation .Couplé à des cartes d'entrées /sorties, il permet de gérer des flux d'information numériques ou analogiques et de créer ou de simuler des instruments de mesure (oscilloscope , multimètre, etc…).

Les programmes LabView comportent des instruments virtuels dans l'abréviation (VI) signifie (Virtual instruments).leurs apparences et leur fonctionnement sont semblables à ceux d'instruments réels.

2. L'environnement de LabView :

Le démarrage de LabView ce fait on double cliquons sur l'icône National Instrument LabView de bureau .Au lancement de l'application, la boite de dialogue suivante apparait :

Figure 1: fenêtre de lancement de l'environnement LabView.

Pour créer un nouveau VI on clique sur l'icône *nouveau VI*, deux fenêtre vierge apparaissent à l'écran : une face avent ou *panel* (de fond gris) et un diagramme ou *Diagramme* (de fond blanc).

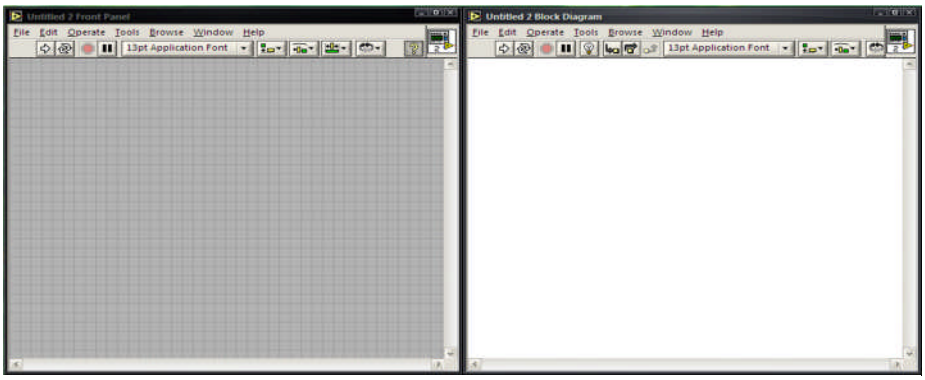


Figure 2 : nouveau VI.

2.1. La face avant :

La face avant est l'interface utilisateur graphique de *VI LabView*. Cette interface réceptionne les données entrées par l'utilisateur et affiche celles fournies, en sortie par le programme .cette face avant peut contenir des boutons rotatifs, des boutons poussoirs, des graphes, et bien d'autre commandes ou indicateurs.

Lorsque on construit la face avant, on intègre des commandes et des indicateurs, qui sont respectivement les terminaux interactifs d'entrées et de sortie du VI .Les commandes simulent les dispositifs d'entrées de l'instrument et fournissent des données au diagramme du VI .Les indicateurs simulent les dispositifs de sortie de l'instrument et affiche les données acquises ou générées par le diagramme.

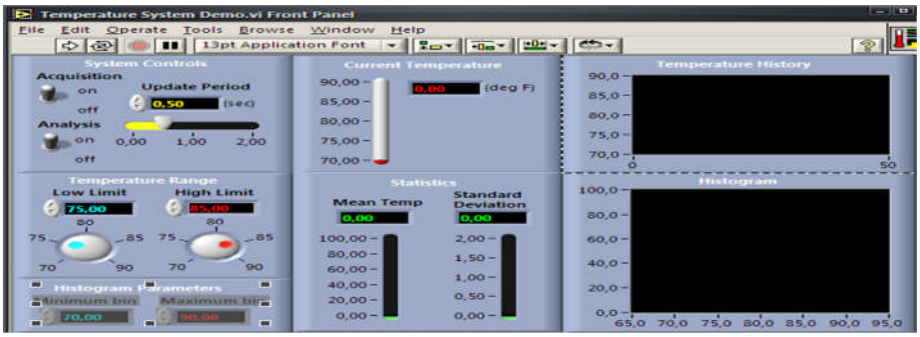


Figure 3 : face avent d'un VI.

2.2. Le diagramme :

Le diagramme contient le code source graphique de VI. On programme le VI pour contrôler et remplir des fonctions sur les entrées et sorties crées dans la face avant. Le diagramme peut contenir des fonctions et des structures issues des bibliothèques de VIs intégrées à LabView.il peut aussi contenir des terminaux associés à des indicateurs crées dans la face avant.

Les objets de face avant apparaissent sous la forme de terminaux. Le terminal ne disparait que lorsqu'on supprime les objets correspondants sur la face avant. Les objets du diagramme sont les terminaux, les sous-VIs, les fonctions, les constantes, les structures et les files de liaison qui transfèrent les données aux autres objets du diagramme.

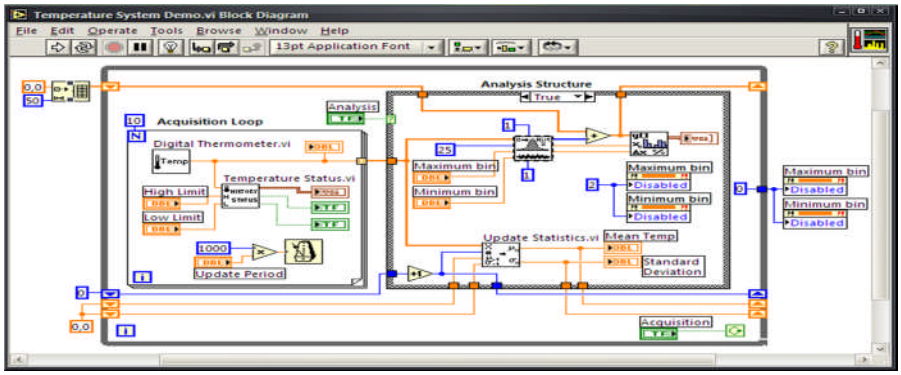


Figure 4 : diagramme d'un VI.

2.3. Palette d'outils :

Elle est disponible sur la face avant et sur le diagramme .elle contient les outils nécessaires pour faire fonctionner et modifier la face avant et les objets du diagramme.

Figure 5 : palette d'outils.

2.4. Palette de commandes :

Elle est disponible uniquement sur la face avant, elle contient les commandes et les indicateurs de la face avant nécessaire pour créer l'interface utilisateur.

Figure 6 : palette de commande :

2.5. Palette de fonctions :

Elle est disponible uniquement sur le diagramme .Elle contient les objets nécessaires pour la programmation graphique comme les opérations d'arithmétique, d'E/S d'instrument, d'E/S de fichier et d'acquisition de données.

Figure 7 : palette de fonction.

3. structure de données dans LabView :

LabView utilise un langage fortement typé et toutes données ne peuvent être manipulées qu'avec des fonctions admettant ce type .En fait ,dans LabView on trouve les types de base scalaire, les type entier s (signés ou non ,codés sur 8,16 ou 32bit),le type réel (code sur 16,32 ou 64bits),le type booléen et le type chaine de caractères .Il est important de noter que les éléments représentant ces données, ainsi que les liaisons issues de ces éléments, sont de forme et de couleur différente.

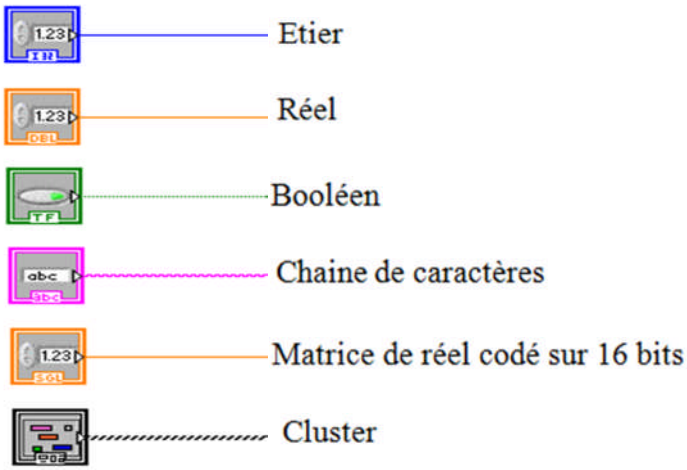


Figure 8 : structure de données.

Le langage permet aussi de créer des structures de données plus élaborées :

- **Le type tableau (structure de données homogène) :**

Comme tous les langages de programmation, la présence de notion de tableau est obligatoire, puisqu'elle définit un outille de base pour le stockage et le traitement des informations. La bibliothèque tableau dans LabView est riche en fonctions prédéfinit, tel que, l'initialisation, l'indexation, l'inversement d'un tableau, et plusieurs autres fonctions .La notion de tableau peut être élargie pour contenir les matrices.

- **Le type « clusters » (structure de données hétérogènes) :**

Les différents composants ou champs de ce groupe de données (cluster) peuvent être assemblés ou récupérés par des fonctions spécifiques.

4. Traitement numérique :

Les fonctions prédéfinies : LabView possède les instructions de base d'un langage de programmation permettant de traiter les différents types de données .Ainsi, on a des fonctions liées aux variables numériques (entiers, réels, et complexes), aux variables booléennes, aux chaines de caractères, et aux tableaux. On trouve aussi les opérateurs de tests et de comparaisons liées à ces différents types de données. A partir des structures de contrôle, des fonctions et des opérateurs, de base, il est alors possible de traduire un algorithme quelconque et d'enrichir la bibliothèque des fonctions en utilisant le mécanisme d'encapsulation. Un diagramme complet est alors réduit à un nœud qui peut être ensuite réutilisé.

Boites à outils mathématique : LabView contient aussi des boites de calcul mathématiques qui servent à introduire des commandes complexes telles que (sin, cos, etc…).

5. structure de programme :

LabView dispose d'un programme qui est enrichi de quatre types de structures : la séquence, deux structures d'itération (la boucle « pour » avec un nombre d'itérations fixe et la boucle « Tant Que »avec un nombre d'itérations soumis à une condition) et la structure de choix.

6. Transmission série dans LabView :

La transmission série utilise un émetteur pour envoyer les données les unes après les autres, bits par bit, sur une ligne de communication unique, à destination d'un récepteur.

Etant donné la simplicité de la liaison série, la bibliothèque pour le port série se résume à cinq instruments virtuels (VI) : Serial Port Init, Serial Port Write, Serial port Read, Bytes at Serial Port et Serial Port Close. Un programme de gestion de port série, réalisé en langage G dans l'environnement LabView consiste à mettre en séquence les déférents VI de base.

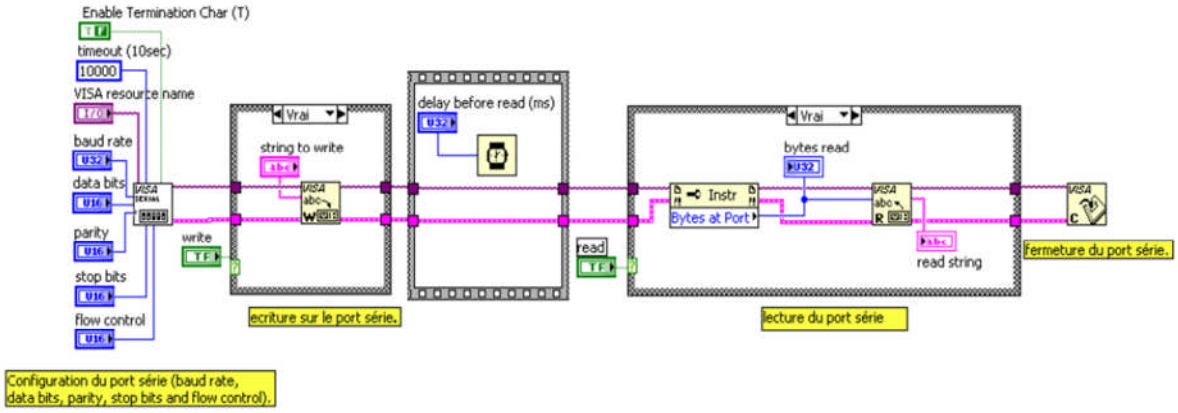


Figure 9: bibliothèque du port série sur LabView.

ANNEXE3 : **Datasheet**

CAPTEUR DE TEMPERATURE LM335

National Semiconductor

November 2000

LM135/LM235/LM335, LM135A/LM235A/LM335A Precision Temperature Sensors

General Description

The LM135 series are precision, easily-calibrated, integrated circuit temperature sensors. Operating as a 2-terminal zener, the LM135 has a breakdown voltage directly proportional to absolute temperature at +10 mV/°K. With less than 1Ω dynamic impedance the device operates over a current range of 400 µA to 5 mA with virtually no change in performance. When calibrated at 25°C the LM135 has typically less than 1°C error over a 100°C temperature range. Unlike other sensors the LM135 has a linear output.

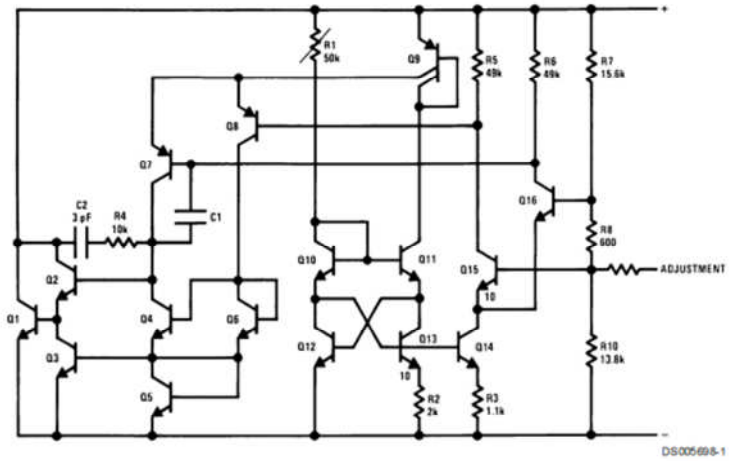
Applications for the LM135 include almost any type of temperature sensing over a −55°C to +150°C temperature range. The low impedance and linear output make interfacing to readout or control circuitry especially easy.

The LM135 operates over a −55°C to +150°C temperature range while the LM235 operates over a −40°C to +125°C temperature range. The LM335 operates from −40°C to +100°C. The LM135/LM235/LM335 are available packaged in hermetic TO-46 transistor packages while the LM335 is also available in plastic TO-92 packages.

Features

- Directly calibrated in °Kelvin
- 1°C initial accuracy available
- Operates from 400 µA to 5 mA
- Less than 1Ω dynamic impedance
- Easily calibrated
- Wide operating temperature range
- 200°C overrange
- Low cost

Schematic Diagram



MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document
by MPX5100/D

Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPX5100 SERIES

INTEGRATED PRESSURE SENSOR
0 to 100 kPa (0 to 14.5 psi)
15 to 115 kPa
(2.18 to 16.68 psi)
0.2 to 4.7 Volts Output

The MPX5100 series piezoresistive transducer is a state–of–the–art monolithic silicon pressure sensor designed for a wide range of applications, but particularly those employing a microcontroller or microprocessor with A/D inputs. This patented, single element transducer combines advanced micromachining techniques, thin–film metallization, and bipolar processing to provide an accurate, high level analog output signal that is proportional to the applied pressure.

Features

- 2.5% Maximum Error over 0° to 85°C
- Ideally suited for Microprocessor or Microcontroller–Based Systems
- Patented Silicon Shear Stress Strain Gauge
- Available in Absolute, Differential and Gauge Configurations
- Durable Epoxy Unibody Element
- Easy–to–Use Chip Carrier Option

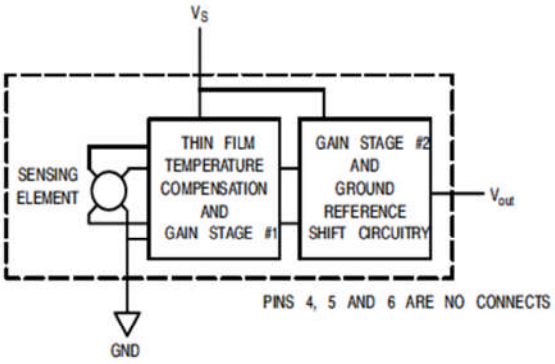
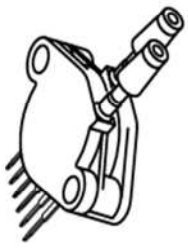
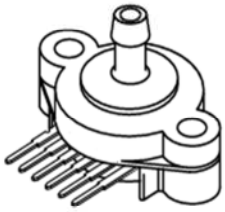


Figure 1. Fully Integrated Pressure Sensor Schematic

MPX5100D
CASE 867



MPX5100DP
CASE 867C



MPX5100GSX
CASE 867F

PIN NUMBER			
1	V_{out}	4	N/C
2	Gnd	5	N/C
3	V_S	6	N/C

ANALOG DEVICES

Low Cost, Low Power Instrumentation Amplifier

AD620

FEATURES
EASY TO USE
Gain Set with One External Resistor
(Gain Range 1 to 1000)
Wide Power Supply Range (±2.3 V to ±18 V)
Higher Performance than Three Op Amp IA Designs
Available in 8-Lead DIP and SOIC Packaging
Low Power, 1.3 mA max Supply Current

EXCELLENT DC PERFORMANCE ("B GRADE")
50 μV max, Input Offset Voltage
0.6 μV/°C max, Input Offset Drift
1.0 nA max, Input Bias Current
100 dB min Common-Mode Rejection Ratio (G = 10)

LOW NOISE
9 nV/√Hz, @ 1 kHz, Input Voltage Noise
0.28 μV p-p Noise (0.1 Hz to 10 Hz)

EXCELLENT AC SPECIFICATIONS
120 kHz Bandwidth (G = 100)
15 μs Settling Time to 0.01%

APPLICATIONS
Weigh Scales
ECG and Medical Instrumentation
Transducer Interface
Data Acquisition Systems
Industrial Process Controls
Battery Powered and Portable Equipment

PRODUCT DESCRIPTION

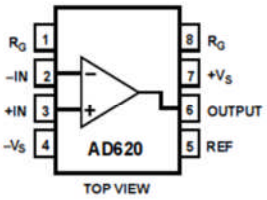
The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gains of 1 to 1000. Furthermore, the AD620 features 8-lead SOIC and DIP packaging that is smaller than discrete designs, and offers lower power (only 1.3 mA max supply current), making it a good fit for battery powered, portable (or remote) applications.

The AD620, with its high accuracy of 40 ppm maximum nonlinearity, low offset voltage of 50 μV max and offset drift of 0.6 μV/°C max, is ideal for use in precision data acquisition systems, such as weigh scales and transducer interfaces. Furthermore, the low noise, low input bias current, and low power of the AD620 make it well suited for medical applications such as ECG and noninvasive blood pressure monitors.

The low input bias current of 1.0 nA max is made possible with the use of Superβeta processing in the input stage. The AD620 works well as a preamplifier due to its low input voltage noise of 9 nV/√Hz at 1 kHz, 0.28 μV p-p in the 0.1 Hz to 10 Hz band, 0.1 pA/√Hz input current noise. Also, the AD620 is well suited for multiplexed applications with its settling time of 15 μs to 0.01% and its cost is low enough to enable designs with one in-amp per channel.

CONNECTION DIAGRAM

8-Lead Plastic Mini-DIP (N), Cerdip (Q) and SOIC (R) Packages



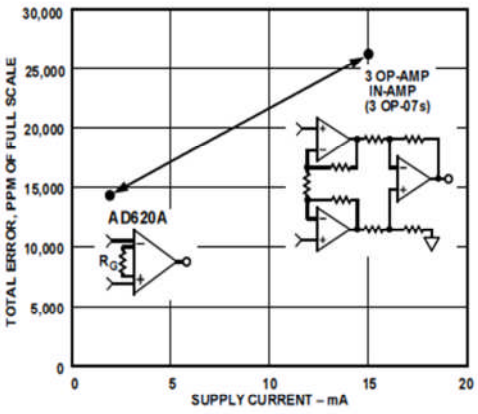


Figure 1. Three Op Amp IA Designs vs. AD620

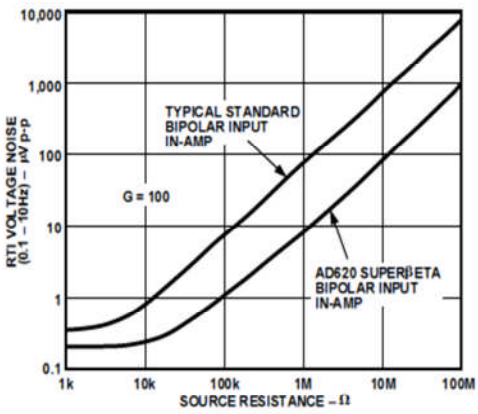


Figure 2. Total Voltage Noise vs. Source Resistance

National Semiconductor

November 1995

ADC0808/ADC0809 8-Bit μP Compatible A/D Converters with 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

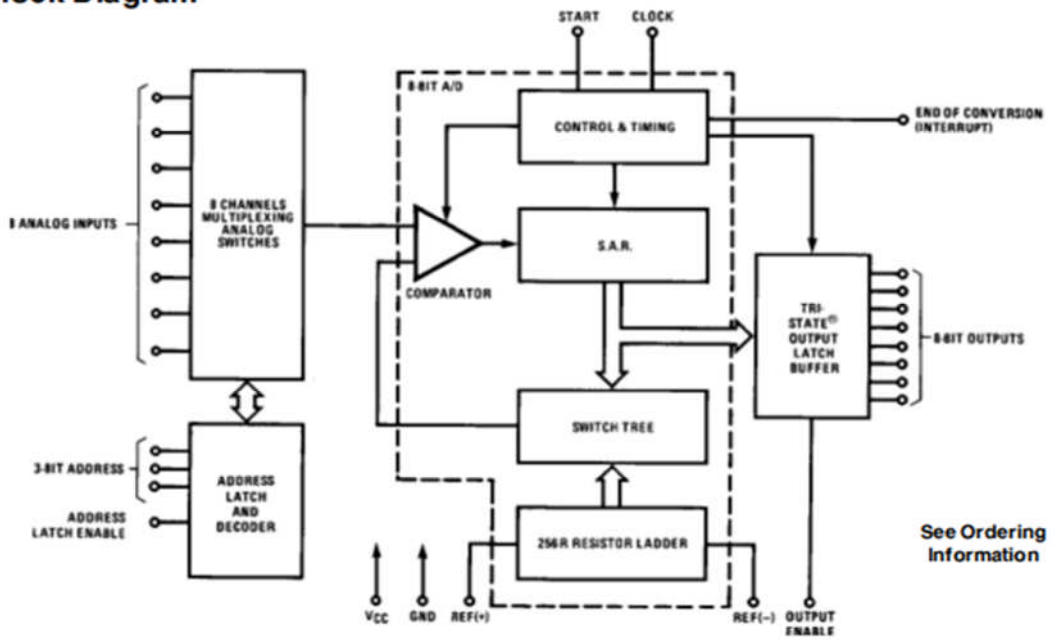
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 μs

Block Diagram



See Ordering Information

TL/H/5672–1

TRI-STATE® is a registered trademark of National Semiconductor Corp.

 TL/H/5672 RRD-B30M36/Printed in U. S. A. http://www.national.com

ADC0808/ADC0809 8-Bit μP Compatible A/D Converters with 8-Channel Multiplexer