PROJET SEMESTRE 6

NERF FANTOME

Réalisé par :

Daniel SOUZA

Jean-Baptiste VERROKEN

Ziwei LIAO

Alexandre CABROL

SOMMAIRE

1. Idées 3

1.1. En cas idéal 3

1.2. En cas réel 4

2. Connaissance en électronique acquise 5

3. Choix des composants électroniques 6

4. Réalisation de la carte électronique 9

5. Explication de code VHDL 13

5.1. Définition des portes 13

5.2. Machines d'état 13

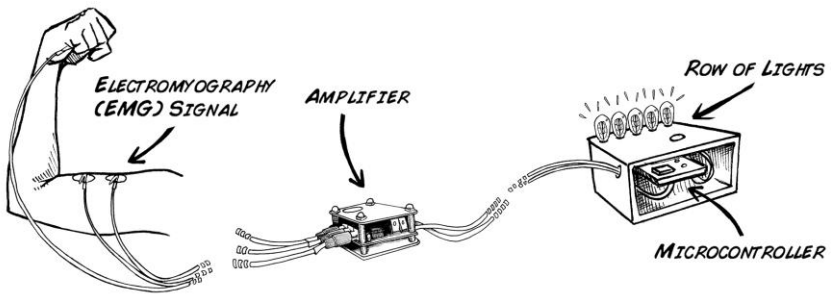
CONCLUSION 19

CONTEXTE

Dans le domaine médical, on demande de détecter les signaux des nerfs venant d'un muscle. Ces signaux pourront être transférés en signaux électriques (courant / électrode) ce qui nous permet de faire commander un robot médical afin d'aider les personnes incapables.

Il sera donc idéal d'avoir des dispositifs qui permettent de recevoir et analyser les informations. Néanmoins, en réalité, il est très difficile de réaliser ce type de dispositif. Les signaux venant d'un muscle pourront être très petits par apport aux bruits parasites.

Le but de projet nous amène au domaine scientifique d'Electromyographie (EMG). Les signaux peuvent être utilisés dans des applications biomédicales et cliniques, « Evolvable Hardware Chip Development » et interaction homme-ordinateur moderne par exemple. Les signaux EMG acquis à partir des muscles ont besoin des méthodes avancées pour la détection, décomposition, procession et classification.



En fait, ce qu'on peut mesurer après la réalisation de notre projet, sont le courant électrique, la source des signaux sont simulés par l'ordinateur ce qui représente l'activité neuromusculaire.

Il est très intéressent et il devient une exigence très importante dans biomédicale ingénierie. L'analyse des signaux est réalisée pour des diagnostics cliniques et applications biomédicale. Surtout dans le domaine de la gestion et de la réhabilitation de handicap moteur.

INTRODUCTION

Le but de ce projet est de développer la partie électronique des générateurs de courant, permettant de simuler le comportement d'axones sur quelques centimètres.

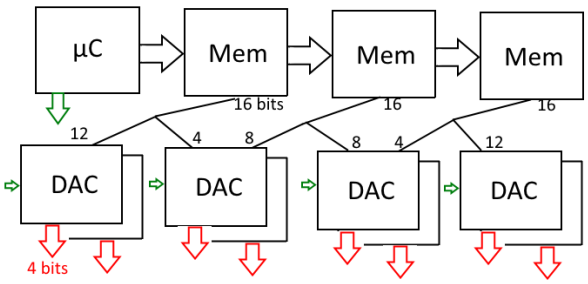
Le signal doit être échantillonné avec une période de 2,5 µs et on souhaite pouvoir simuler le fonctionnement de l'axone pendant une demi-seconde.

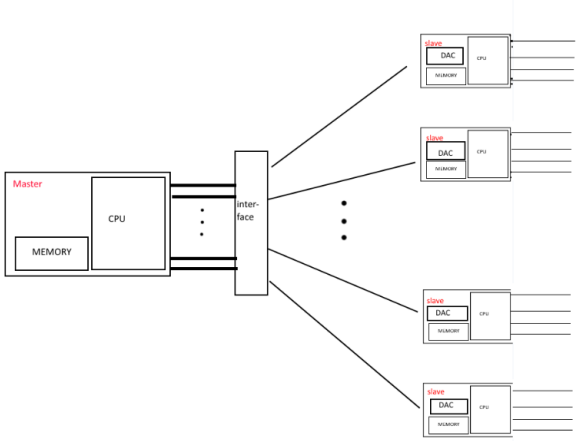
Nous avons besoin de 200 sorties, chacune pouvant délivrer un courant de -200 à +200 µA avec une résolution de 12 bits (100 nA) et échantillonné à 400kHz. La sortie doit être capable de passer de son courant maximum positif à son courant maximum négatif en 50 µs. Pour répondre à toutes ces contraintes, il vaut mieux utiliser une structure modulaire et hiérarchisée dû au fort débit de données nécessaire.

Du coup, pour tester notre projet, on réalise un module constitué de 16 sorties. Une fois la simulation est bonne on pourra rassembler plusieurs modules pour avoir 200 sorties.

1. Idées

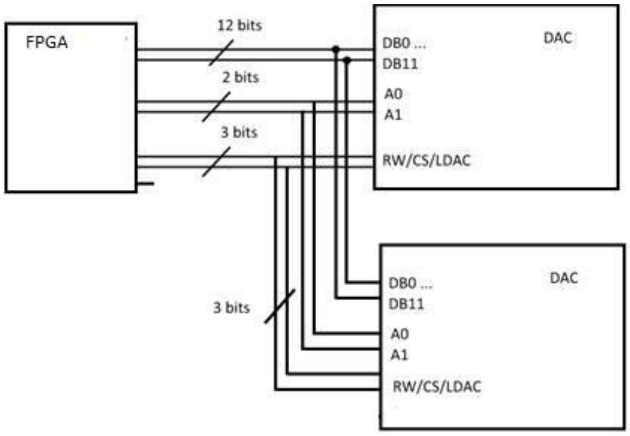
1.1. En cas idéal





Idéalement, on pourrait réaliser le maître avec un microcontrôleur que l'on peut programmer en langage C par nos connaissances. Et aussi, on pourrait avoir 32 sorties dans un module ce qui nous permet d'économiser les matériaux en fabriquant moins de modules. Cependant le processeur d'un microcontrôleur n'est pas assez puissant et aussi, le microcontrôleur lui-même n'est pas capable pour donner les flux de données voulues, si bien que l'on préfère utiliser un FPGA. On a choisi le modèle de Spartan-3E1600 qui a été conseillé par Mr. Cathébras. En effet, ce dispositif possède une capacité de vitesse et mémoire idéales pour le projet.

1.2. En cas réel



Un module du dispositif

La figure ci-dessus montre la connexion entre le maître (*la partie numérique*) et les esclaves (*la partie analogique*). On a 12 sorties de FPGA qui sont réservées par 12 bits de données désirées au début du projet. Les autres 7 bits permettent d'avoir d'autres fonctionnements.

- Partie numérique

La partie numérique est constituée de la carte FPGA qui permet de récupérer les signaux venant de la programmation en VHDL. Il va aussi contrôler tels signaux pour les renvoyer à la partie analogique.

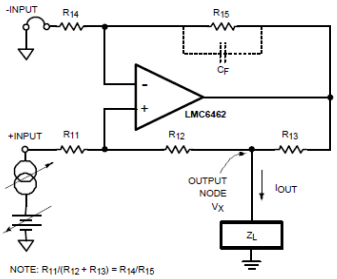
- Partie analogique

Dans cette partie, on réalise les cartes électriques de l'esclave ce qui nous permet d'avoir un courant de 200uA en sortie du dispositif.

2. Connaissance en électronique acquise

Source de courant Howland

La source de courant Howland possède beaucoup d'avantages que l'on peut profiter dans notre projet. La sortie du montage peut être des courants négatifs ou positif ou nul pour différentes charges. En plus, le gain est facilement réglable et c'est un circuit très efficace pour atteindre à une grande gamme de tension en sortie.



Dans ce circuit, le gain est facilement réglé par la résistance R_{13} dont la valeur est petite. Le gain peut être modifié par le rapport de $\frac{R_{14}}{R_{15}}$ qui est normalement égale à 1, c'est-à-dire que R_{14} et R_{15} ont la même valeur.
Contrairement à la valeur de résistance R_{13}, il faut que les valeurs de R14, R15, R11 et R12 soient grandes. Dans notre application la valeur qu'on a utilisée est 40kOhm.

On peut établir l'équation suivante selon ce montage :

$$\frac{R_{11}}{R_{12} + R_{13}} = \frac{R_{14}}{R_{15}}$$

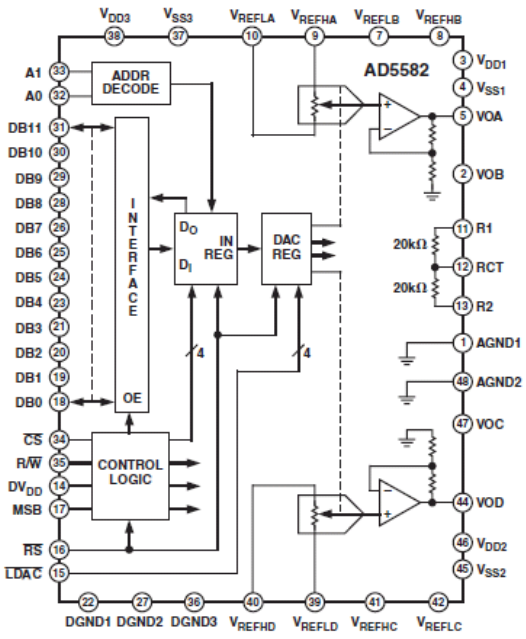
Dans cette condition, on peut avoir un courant de quelque milampère avec une tention de 10V en sortie.

En effet, la source de courant Howland nous permet d'avoir une gamme de tension en sortie la plus positive et la plus négative de l'alimentation appliquée. On voit bien l'efficacité de la source de courant Howland.

3. Choix des composants électroniques

A partir de structure conçue, on a commencé à choisir les composants électroniques. En effet, en sortie de la carte FPGA, on peut obtenir les signaux numériques, cependant, notre dispositif délivre les courants de 200uA, si bien que, tout d'abord, on a besoin d'un convertisseur numérique analogique.

➢ DAC *AD5288*



✓ Mots clés :

12 entrées numériques

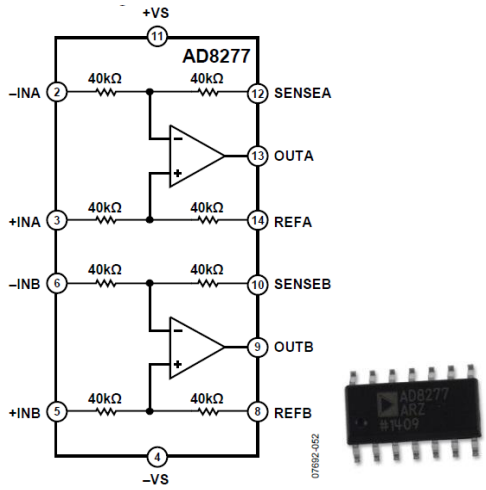
4 amplificateurs de suiveur intégré

Tension en sortie réglable

Le convertisseur numérique analogique AD5288 possède 12 entrées numériques ce qui correspond au nombre de sortie de FPGA. Chaque convertisseur a 4 sorties analogiques par lesquelles on peut avoir 4 sources de courants pour le dispositif. Aussi, on trouve 4 amplificateur de suiveurs, reliés aux 4 sorties du convertisseur, intégrés dans le boitier ce qui va faciliter le branchement du montage de la carte électronique.

Grâce aux tensions de référence appliquées, la tension de sortie du convertisseur est réglable.

➢ AOP *AD8277*



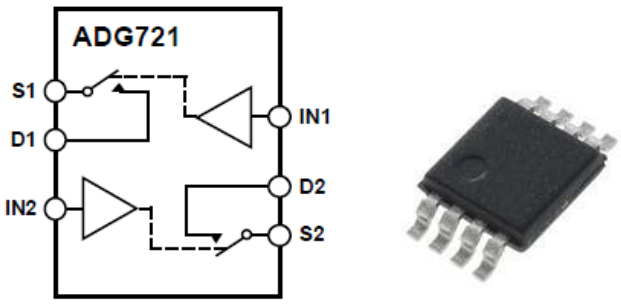
✓ Mots clés :

2 amplificateurs intégrés

Source de courant Howland

Le modèle AD8277 est constitué de 2 amplificateurs opérationnels dans un boitier. Ils sont aussi branché en source de courant Howland ce qui nous permet d'avoir une gamme de tension la plus positive et la plus négative de l'alimentation appliquée. Par conséquent, il possède plus de capacité pour avoir un courant élevé en sortie.

➢ Switch *ADG721*



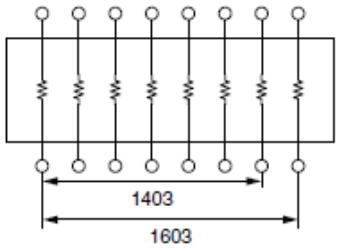
✓ Mots clés

Normalement fermé

2 interrupteurs intégrés

Afin de protéger la charge en sortie de notre dispositif, on peut bien brancher un interrupteur normalement fermé en parallèle avec la sortie de l'amplificateur. Quand le dispositif est en mode de fonctionnement, l'interrupteur sera en état ouvert.

➢ Réseau de résistance



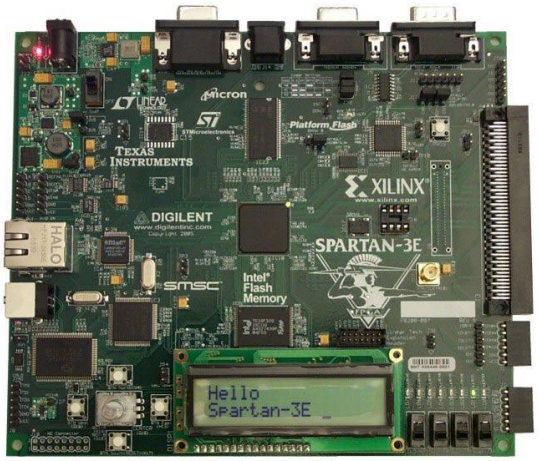
✓ Mots clés

Multiples résistances intégrés

Source de courant Howland

Pour améliorer la performance du modèle de source de courant Howland, il est nécessaire d'ajouter les résistances en sortie de l'amplificateur. On peut utiliser un réseau de résistance, c'est-à-dire 8 résistances intégrées dans un boitier.

➢ FPGA

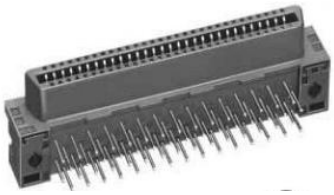


On utilise le FPGA qui joue le rôle du Master. Il contrôle les signaux qui sont renvoyé à la partie Analogique.

Le type de FPGA qu'on utilise est le Spartan-3E1600. La raison pour laquelle nous utilisons le FPGA est parce qu'elle est assez vite pour le but du projet, le mémoire est assez grand et le processeur correspond à nos conditions imposées. En plus, il y a encore beaucoup de fonctionnements supplémentaires qui pourraient être utiles pendant notre projet.

Pour programmer le FPGA nous utilisons le langage VHDL. Le VHDL est un langage très intéressant car c'est un nouveau paradigme. C'est-à-dire, une nouvelle façon de programmer. Une différence importante c'est qu'en VHDL, par exemple, toutes les commandes sont exécutées parallèlement, au lieu d'une façon synchronisée, ce qui prend (beaucoup) plus de temps (en considérant qu'on a assez de portes disponibles). En plus, le langage VHDL, c'est est un langage de description de matériel qui est plus dure d'apprendre dès qu'on connaissait plutôt des langages d'haute niveau. En plus c'est très intéressant de travailler avec un langage avec laquelle nous n'étions pas familiarisés.

- Connecteur FX2 100S

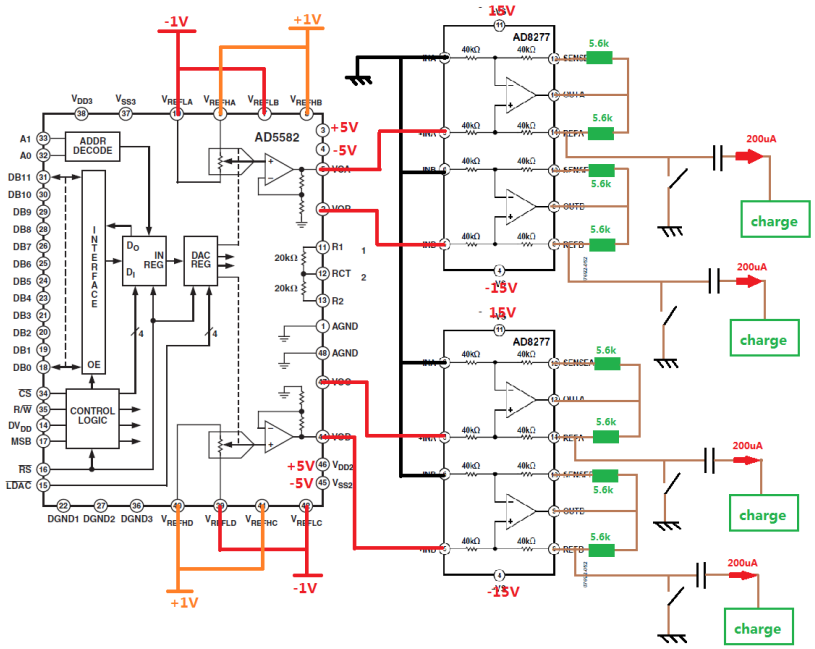


Pour communiquer la carte électronique et la carte FPGA, on a aussi besoin d'un connecteur FX2 100S ce qui est compatible à celui de FPGA

4. Réalisation de la carte électronique

Après le choix des composants, on a passé à la réalisation de la carte électronique pour un module.

➢ Schéma électrique à réaliser



❖ Calcul de la tension en sortie du convertisseur

Selon la formule indiquée dans le datasheet du convertisseur AD5582, où D représente la valeur des données en décimal. Ici, on veut utiliser les 12 entrées du convertisseur, donc la valeur de D est de 4095.

$$V_{OUT} = \left(\frac{2D}{4095} - 1\right) V_{REF} \qquad \text{(For AD5582)}$$

$$V_{REF} = V_{REFH} - V_{REFL}$$

Pour faciliter le calcul, on a pris $V_{REFH} = 1V$ et $V_{REFL} = -1V$, donc la valeur de V_{REF} est de 2V. Enfin, on a une tension en sortie du convertisseur qui est égale à 2V.

❖ Calcul des résistances branchées en sortie de l'amplificateur

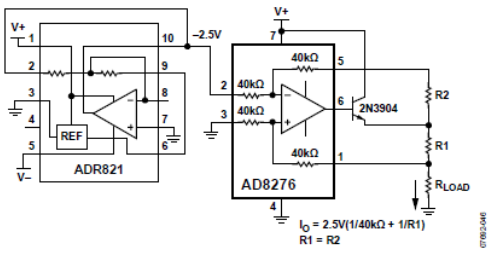


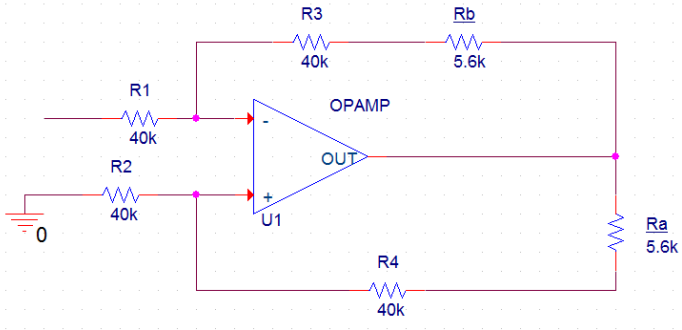
Figure 49. Constant Current Source

On a trouvé la formule proposé dans le document pour calculer les valeurs des résistances en sortie de l'amplificateur AD8277 :

$$I_o = V_{out}/({}^{1}/_{40k} + {}^{1}/_{R1}) \text{ avec } I_o = 200\mu A$$

On obtient $R_1 = R_2 = 5.6k\Omega$

❖ Calcul de la précision des résistances branchées en sortie de l'AOP



On doit calculer l'incertitude maximale que l'on peut avoir sur Ra et Rb.

On a $Rout\ min = \frac{R2+R4}{\frac{R4+Ra}{R3+Rb}} \cdot \frac{Ra}{R3+Rb} = 120\ M\Omega$

L'incertitude est :

$$\left|\frac{R4 + Ra}{R3 + Rb} - \frac{R2}{R1}\right| < \frac{1}{120\ M}$$

En supposant que R1, R2, R3 et R4 soient identiques, on obtient :

$$\left|\frac{R + Ra}{R + Rb}\right| - 1 < \frac{1}{120\ M}$$

En posant $Ra = Rb + \varepsilon$ on trouve :

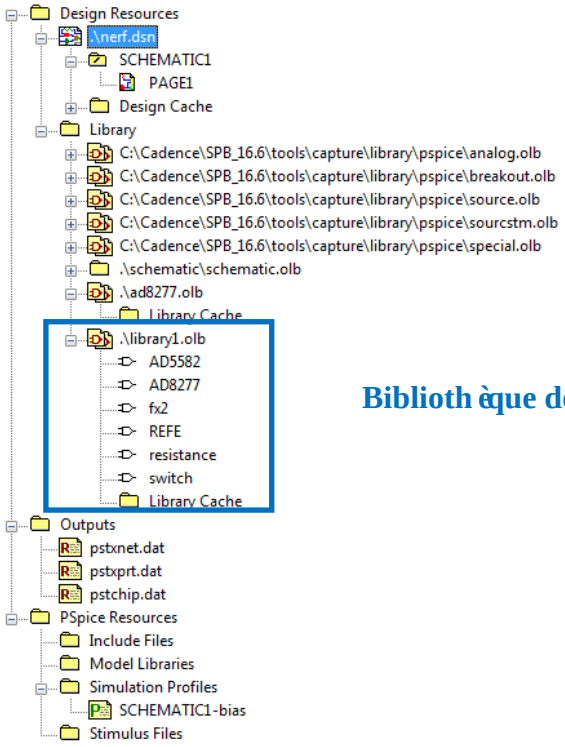
$$\varepsilon < \frac{40 + 5{,}6}{120\ M}\ k\Omega$$

ce qui équivaut à une incertitude maximale de 0,038%.

➢ Création du montage sous Cadence

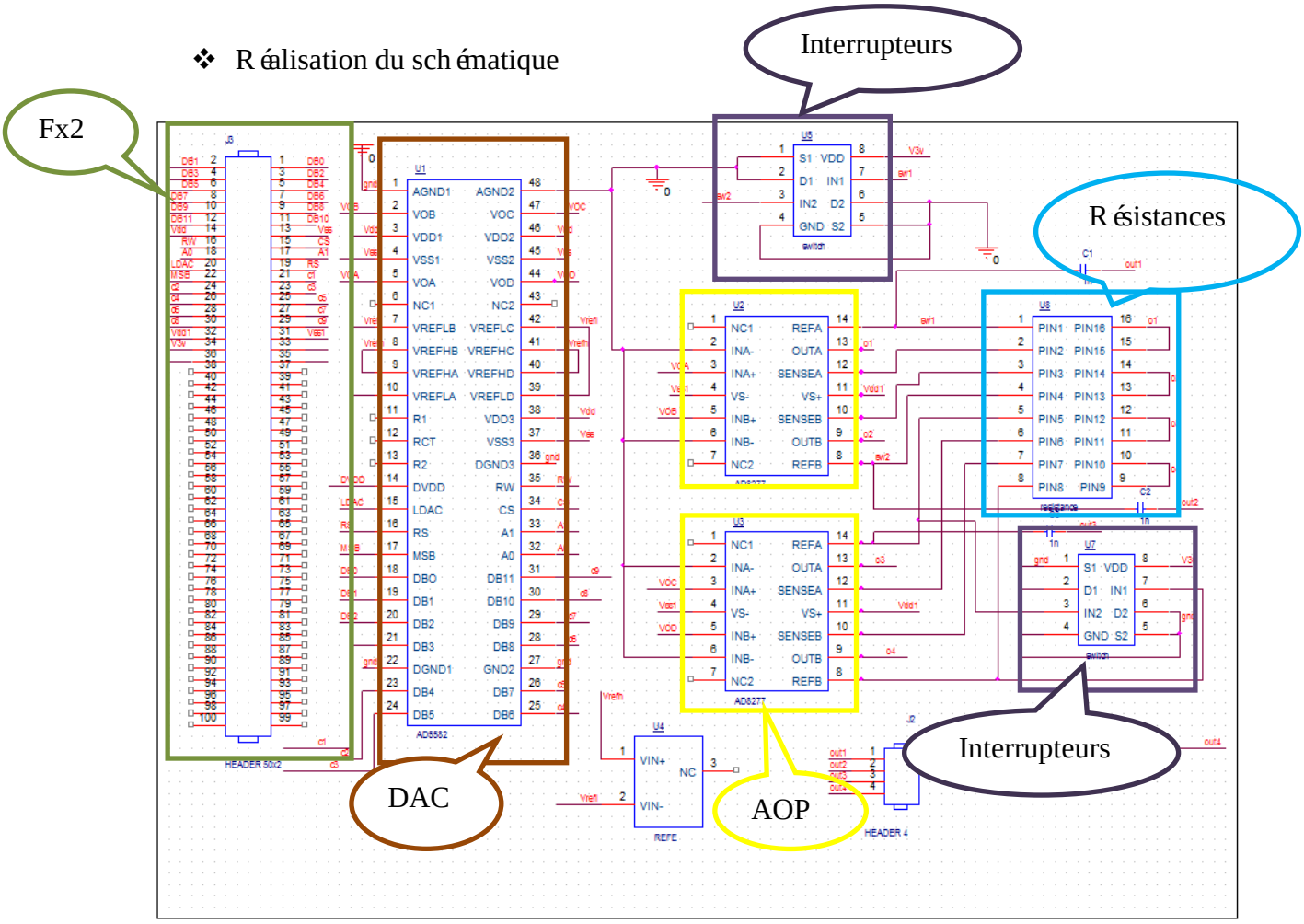
❖ Création de la bibliothèque pour le schématique

Les composants qu'on a choisis n'existent pas dans la bibliothèque de Cadence, on a donc crée notre propre bibliothèque du projet.



Biblioth èque de notre sch ématique

❖ R éalisation du sch ématique



5. Explication de code VHDL

5.1. Définition des portes

```
entity master is
   port(
          clk : in  std_logic;
          clke : in  std_logic;
          rst : in  std_logic;
          led : out std_logic_vector(7 downto 0);

        -- SDRAM pins out
          dram_clkp   : out   std_logic; -- 0 deg phase 100mhz clock going out to SDRAM chip
          dram_clkn   : out   std_logic; -- 180 deg phase version of dram_clkp
          dram_clke   : out   std_logic; -- clock enable, owned by the init module
          dram_cs     : out   std_logic; -- tied low upon powerup
          dram_cmd    : out   std_logic_vector(2 downto 0); -- this is the command vector <we_n,cas_n,ras_n>
          dram_bank   : out   std_logic_vector(1 downto 0); -- bank address
          dram_addr   : out   std_logic_vector(12 downto 0); -- row/col/mode register
          dram_dm     : out   std_logic_vector(1 downto 0); -- masks used for writing
          dram_dqs    : inout std_logic_vector(1 downto 0); -- strobes used for writing
          dram_dq     : inout std_logic_vector(15 downto 0); -- data lines

        -- debug signals
          debug_reg   : out std_logic_vector(7 downto 0);

        -- analog card signals/ 25 bits in total

        -- Inverted Signals
          analog_ldac         : out    std_logic; -- equivalent to ldac on the datasheet
          analog_reset_strobe : out    std_logic; -- Reset Strobe/ Corresponds to RS in the datasheet
          analog_chip_select  : out    std_logic; -- Signal to determine if the chip given by the current address will be used/ Corresponds to CS in the datasheet

        -- Non-inverted Signals
          analog_data         : inout     std_logic_vector(11 downto 0); -- Data to be converted/ Corresponds to DB0...DB11 in the datasheet
          analog_address      : out    std_logic_vector(1 downto 0); -- DAC addresses/ Corresponds to A0 and A1 in the datasheet
          analog_read_write   : out    std_logic; -- Read_write = 1 - Read Enabled/ Read_write = 0 - write enabled/ Corresponds to R/W in the datasheet
          analog_msb          : out    std_logic -- MSB = 0, Reset to 000h/ MSB = 1, Reset to 800h/ Corresponds to MSB in the datasheet

         );
end master;
```

Définition des portes

Ici, nous pouvons observer la définition des portes (input et output) que nous utilisons dans la partie « master ».

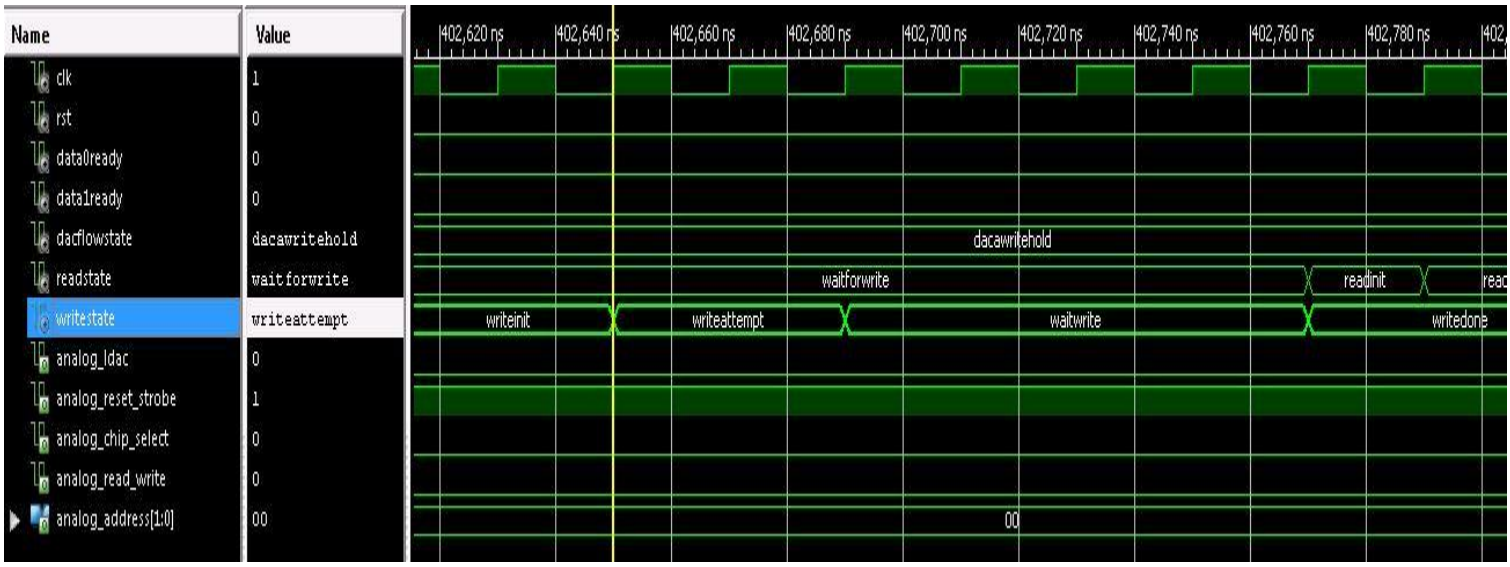
5.2. Machines d'état

```
case WriteState is
   when writeInit =>
      if (busy_n = '1') then
         writeState <= writeAttempt;
      end if;
   when writeAttempt =>
      op <= "10";
      address <= "0000000000"& x"6001";
      data_i <= "11110001";
      if (op_ack = '1') then
         writeState <= waitWrite;
      end if;
   when waitWrite =>
      if (busy_n = '1') then
         writeState <= writeDone;
         readState <= readInit;
      end if;
      op <= "00";
   when writeDone =>
end case;
```

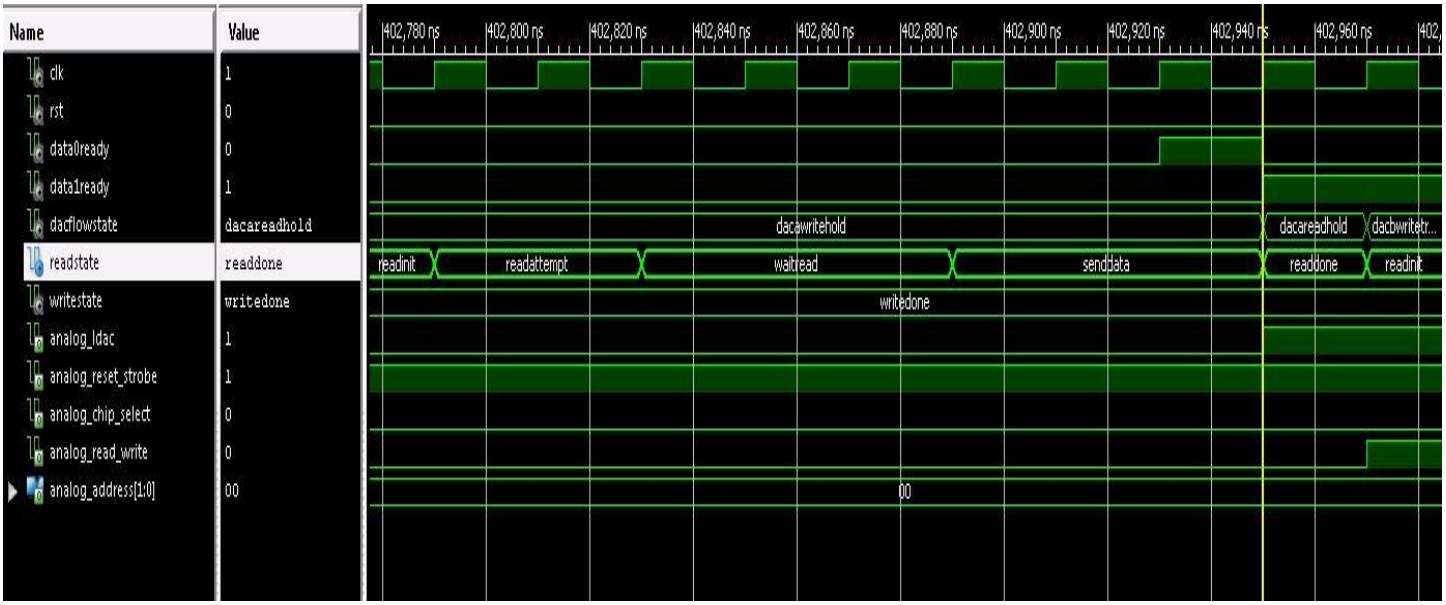
Machine d'état pour faire l'écriture

Cette machine d'état est utile que pour la simulation car normalement les données sont déjà écrites sur la mémoire.

Dans cette machine d'état nous réalisons l'écriture d'une donnée sur une adresse choisie de façon que nous puissions vérifier le bon fonctionnement de notre système.



Simulation d'une écriture sur la mémoire



Simulation d'une opération de lecture et envoie de données

La machine d'état montré ci-dessous est responsable pour fournir les données, lues de la mémoire, aux DACs quand ils en ont besoin.

En plus, comme nous pouvons lire seulement 8 bit de la mémoire à chaque fois, cette machine fait aussi la gestion des valeurs pour que nous puissions envoyer aux DACS 12 bits.

```
case readState is
   when waitForWrite =>

   when readInit =>
      if (busy_n = '1') then
         readState <= readAttempt;
      end if;
   when readAttempt =>
      op <= "01";
      address <= "0000000000"& x"6001";
      if (op_ack = '1') then
         readState <= waitRead;
      end if;
   when waitRead =>
      if (busy_n = '1') then
         readState <= sendData;
      end if;
      op <= "00";
   when sendData => --choose which DAC to send data to
      if(dacCount = 0) then
         output8bits <= data_o(7 downto 0);
         dacCount <= dacCount +1;
      elsif(dacCount = 1) then
         output4bits0 <= data_o(3 downto 0);
         analog_data <= output8bits&output4bits0;
         output4bits1 <= data_o(7 downto 4);
         dacCount <= dacCount +1;
         data0Ready <= '1';
         led <= analog_data(7 downto 0);
      elsif(dataSent <= '1') then
         output8bits <= data_o(7 downto 0);
         analog_data <= output4bits1&output8bits;
         dacCount <= 0;
         data1Ready <= '1';
         led <= analog_data(7 downto 0);
         readState <= readDone;
      end if;
   when readDone =>
      if(readRequired = '1') then
         readRequired <= '0';
         readState <= readInit;
      end if;
end case;
```

Machin d'état pour la lecture

```
if(data0Ready = '1' and dataNeeded = '1') then
   data0Ready <= '0';
end if;
if(data1Ready = '1' and dataNeeded = '1') then
   data1Ready <= '0';
end if;
if(dataNeeded = '1' and data0Ready = '0' and data1Ready = '0') then
   readRequired <= '1';
else
```

Contrôle dd'usage de data

Ici nous faisons le contrôle des signaux qui étaient déjà lus et envoyés par la machine d'état de contrôle de la lecture. Cette contrôle consiste à vérifier si le signal reçu était déjà utilisé ou pas.

```
case DACFlowState is
   --DAC A Selected--
   when DACAWriteTrans =>
      --Transparent--
      analog_read_write <= '0';
      analog_chip_select <= '0';
      analog_ldac <= '0';
      analog_reset_strobe <= '1';
      dataNeeded <= '1';
      analog_address <= "00";
      DACFlowState <= DACAWriteHold;
   when DACAWriteHold =>
      --Hold Input--
      dataNeeded <= '0';
      analog_ldac <= '1';
      DACFlowState <= DACAReadHold;
   when DACAReadHold =>
      --ReadBack data--
      analog_read_write <= '1';
      DACFlowState <= DACBWriteTrans;
```

Mode de processus réalisée par chaque DAC

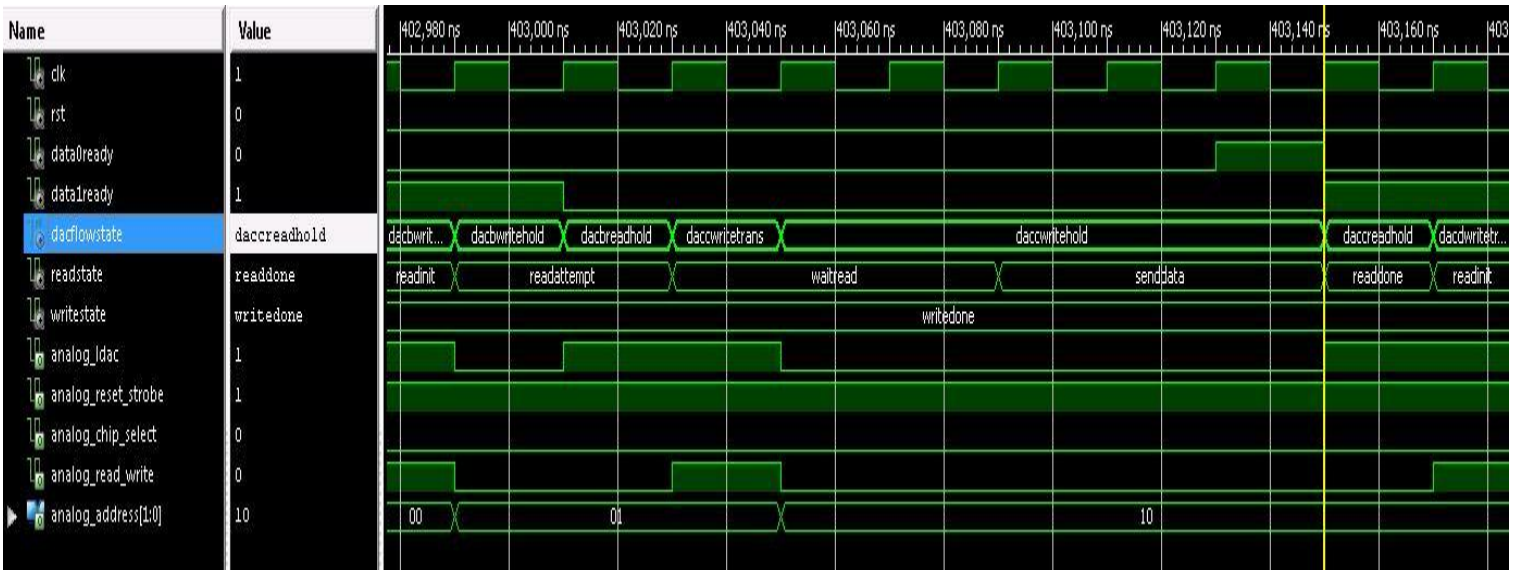
Dans cette machine d'état nous faisons le contrôle de l'envoie des signaux venant de master aux DACs. Le processus que nous réalisons ici consiste d'écriture en mode « Transparent », après en mode «Hold » et finalement nous faisons le «Readback » afin de vérifier si les données envoyées étaient bien reçus.

La figure ci-dessous montre le même processus appliqué aux autres DACs. Puis nous faisons le mise-à-jour des DACs et finalement le reset.

```
--DAC B Selected--
when DACBWriteTrans =>
   --Transparent--
   analog_read_write <= '0';
   analog_chip_select <= '0';
   analog_ldac <= '0';
   analog_reset_strobe <= '1';
   dataNeeded <= '1';
   analog_address <= "01";
   DACFlowState <= DACBWriteHold;
when DACBWriteHold =>
   --Hold Input--
   dataNeeded <= '0';
   analog_ldac <= '1';
   DACFlowState <= DACBReadHold;
when DACBReadHold =>
   --ReadBack data--
   analog_read_write <= '1';
   DACFlowState <= DACCWriteTrans;


--DAC C Selected--
when DACCWriteTrans =>
   --Transparent--
   analog_read_write <= '0';
   analog_chip_select <= '0';
   analog_ldac <= '0';
   analog_reset_strobe <= '1';
   dataNeeded <= '1';
   analog_address <= "10";
   DACFlowState <= DACCWriteHold;
when DACCWriteHold =>
   --Hold Input--
   dataNeeded <= '0';
   analog_ldac <= '1';
   DACFlowState <= DACCReadHold;
when DACCReadHold =>
   --ReadBack data--
   analog_read_write <= '1';
   DACFlowState <= DACDWriteTrans;


--DAC D Selected--
when DACDWriteTrans =>
   --Transparent--
   analog_read_write <= '0';
   analog_chip_select <= '0';
   analog_ldac <= '0';
   analog_reset_strobe <= '1';
   dataNeeded <= '1';
   analog_address <= "11";
   DACFlowState <= DACDWriteHold;
when DACDWriteHold =>
   --Hold Input--
   dataNeeded <= '0';
   analog_ldac <= '1';
   DACFlowState <= DACDReadHold;
when DACDReadHold =>
   --ReadBack data--
   analog_read_write <= '1';
   DACFlowState <= DACUpdateAll;

--Update All DACs--
when DACUpdateAll =>
   analog_chip_select <= '1';
   analog_ldac <= '0';
   analog_reset_strobe <= '1';
   DACFlowState <= DACHoldAll;
```

```
--Hold All--
when DACHoldAll =>
   analog_ldac <= '1';
   DACFlowState <= DACResetAll;


--Reset All--
when DACResetAll =>
   analog_reset_strobe <= '0';
   DACFlowState <= DACFlowOver;
```

Machine d'état de contrôle du système



Simulation du controle de la fluxe des données

CONCLUSION

- Problèmes rencontrés

  Pour la partie analogique, après avoir fini la schématique sous Cadence, nous n'avons pas pu créer la netlist à partir du montage.

- Connaissances acquises

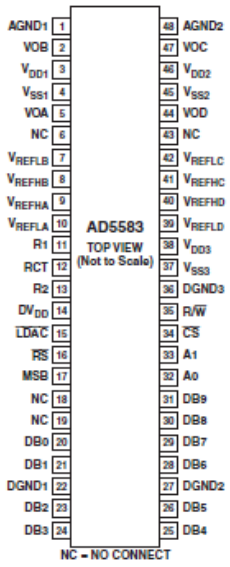
  Pendant notre projet, nous avons appris le principe et l'application de la source de courant Howland. Afin de réaliser la carte électronique, nous avons amélioré nos compétences en utilisations de logiciel Cadence.

- Amélioration

  Ce projet n'a pas été fini à cause de manque de connaissance en utilisation de logiciel Cadence ainsi que la gestion du temps.

Annexe

AD5583 PIN CONFIGURATION

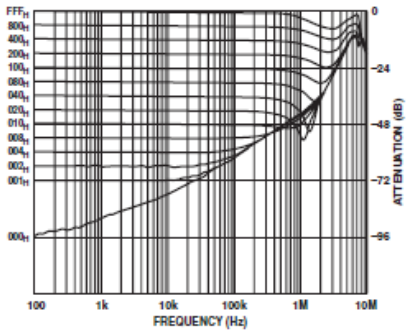


AD5583 PIN FUNCTION DESCRIPTIONS*

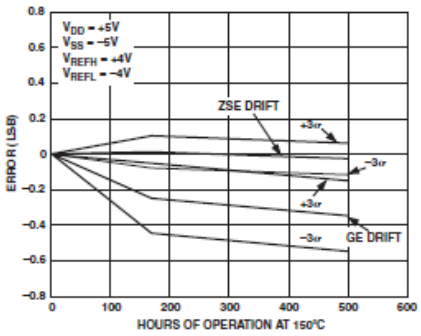
Pin No.	Mnemonic	Description
1	AGND1	Analog Ground for DAC A and B
2	VOB	DAC B Output
3	V_{DD1}	Positive Power Supply for DAC A and B
4	V_{SS1}	Negative Power Supply for DAC A and B
5	VOA	DAC A Output
6	NC	No Connect (Do Not Connect Anything other than Dummy Pad)
7	V_{REFLB}	DAC B Voltage Reference Low Terminal
8	V_{REFHB}	DAC B Voltage Reference High Terminal
9	V_{REFHA}	DAC A Voltage Reference High Terminal
10	V_{REFLA}	DAC A Voltage Reference Low Terminal
11	R1	R1 Terminal (for Negative Reference)
12	RCT	Center Tap Terminal (for Negative Reference)
13	R2	R2 Terminal (for Negative Reference)
14	DV_{DD}	Power Supply for Digital Circuits
15	$\overline{LDAC}$	DAC Register Load, Active Low Level Sensitive
16	$\overline{RS}$	Reset Strobe
17	MSB	MSB = 0, Reset to 000_H. MSB = 1, Reset to 200_H.
18	NC	No Connect (Do Not Connect Anything other than Dummy Pad)
19	NC	No Connect (Do Not Connect Anything other than Dummy Pad)
20	DB0	Data Bit 0
21	DB1	Data Bit 1
22	DGND1	Digital Ground 1
23	DB2	Data Bit 2
24	DB3	Data Bit 3
25	DB4	Data Bit 4
26	DB5	Data Bit 5
27	DGND2	Digital Ground 2
28	DB6	Data Bit 6
29	DB7	Data Bit 7
30	DB8	Data Bit 8
31	DB9	Data Bit 9
32	A0	Address Input 0
33	A1	Address Input 1
34	$\overline{CS}$	Chip Select, Active Low
35	R/$\overline{W}$	Read/Write Mode Select
36	DGND3	Digital Ground 3
37	V_{SS3}	Negative Power Supply for Analog Switches
38	V_{DD3}	Positive Power Supply for Analog Switches
39	V_{REFLD}	DAC D Voltage Reference Low Terminal
40	V_{REFHD}	DAC D Voltage Reference High Terminal
41	V_{REFHC}	DAC C Voltage Reference High Terminal
42	V_{REFLC}	DAC C Voltage Reference Low Terminal
43	NC	No Connect (Do Not Connect Anything other than Dummy Pad)
44	VOD	DAC D Output
45	V_{SS2}	Negative Power Supply for DAC C and D
46	V_{DD2}	Positive Power Supply for DAC C and D
47	VOC	DAC C Output
48	AGND2	Analog Ground for DAC C and D

*AD5583 optimizes internal layout design to reduce die area so that all supply voltage pins are required to be connected externally. See Figure 5.

Description des pins du convertisseur AD5582

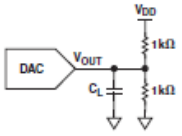


TPC 25. AD5582 Multiplying Bandwidth



TPC 26. AD5582 Long-Term Drift

Test Circuit



Test Circuit 1

THEORY OF OPERATION

The AD5582/AD5583 are quad, voltage output, 12-/10-bit parallel input DACs in compact TSSOP-48 packages.

Each DAC is a voltage switching, high impedance (R = 20 kΩ), R-2R ladder configuration with segmentation to optimize die area and precision. Figure 3 shows a simplified R-2R structure without the segmentation. The 2R resistances are switched between V_{REFH} and V_{REFL}, and the output is obtained from the rightmost ladder node. As the code is sequenced through all possible states, the voltage of this node changes in steps of $(2/3\ V_{REFH} - V_{REFL})/(2^N - 1)$ starting from the lowest V_{REFL} and going to the highest V_{REFH} – DUTLSB. Buffering it with an amplifier with a gain of 1.5 brings the output to:

$$V_{OUT} = \frac{D}{2^N - 1}\left(V_{REFH} - V_{REFL}\right) + \left(V_{REFL}\right) \quad (1)$$

where D is the decimal equivalent of the data bits and N is the numbers of bits.

If $-V_{REFL}$ is equal to V_{REFH} as V_{REF}, V_{OUT} is simplified to:

$$V_{OUT} = \left(\frac{2D}{4095} - 1\right)V_{REF} \quad \text{(For AD5582)} \quad (2)$$

$$V_{OUT} = \left(\frac{2D}{1023} - 1\right)V_{REF} \quad \text{(For AD5583)} \quad (3)$$

The advantage of this scheme is that it allows the DAC to interpolate between two voltages for differential references or single-ended reference.

These DACs feature double buffers, which allow both synchronous and asynchronous channels update with additional data readback capability. These parts can be reset to zero scale or midscale controlled by the $\overline{RS}$ and MSB pins. When $\overline{RS}$ is activated, the MSB of 0 resets the DACs to zero scale and the MSB of 1 resets the DACs to midscale. The ability to operate from wide supply voltages, +5 V to +15 V or ±5 V, with multiplying bipolar references is another key feature of these DACs.

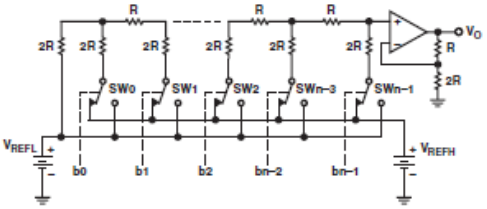


Figure 3. Simplified R-2R Architecture (Segmentation Not Shown)

Power Supplies

There are three separate power supplies needed for the operation of the DACs. For dual supply, V_{SS} can be set from –6.5 V to –2.7 V and V_{DD} can be set from +2.7 V to +6.5 V. For single supply, V_{SS} should be set at 0 V while V_{DD} is set from 3 V to 16.5 V. However, setting the single supply of V_{DD} below 4.5 V can impact the overall accuracy of the device.

Calcul de la tension en sortie du convertisseur

The differential output voltage and common-mode voltage of the AD8226 is shown in the following equations:

$$V_{DIFF\_OUT} = V_{+OUT} - V_{-OUT} = Gain_{AD8226} \times (V_{+IN} - V_{-IN})$$

$$V_{CM} = (V_{S+} - V_{S-})/2 = V_{BIAS}$$

Refer to the AD8226 data sheet for additional information.

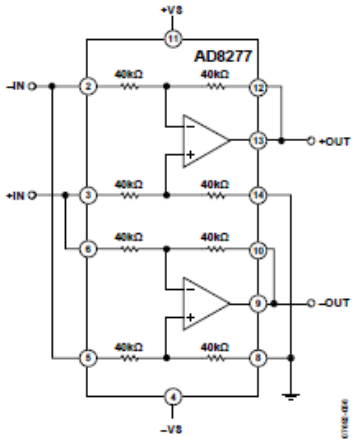


Figure 48. AD8277 Differential Output Configuration

The two difference amplifiers of the AD8277 can be configured to provide a differential output, as shown in Figure 48. This differential output configuration is suitable for various applications, such as strain gage excitation and single-ended-to-differential conversion. The differential output voltage has a gain of 2 as shown in the following equation:

$$V_{DIFF\_OUT} = V_{+OUT} - V_{-OUT} = 2 \times (V_{+IN} - V_{-IN})$$

CURRENT SOURCE

The AD8276 difference amplifier can be implemented as part of a voltage-to-current converter or a precision constant current source as shown in Figure 49. Using an integrated precision solution such as the AD8276 provides several advantages over a discrete solution, including space-saving, improved gain accuracy, and temperature drift. The internal resistors are tightly matched to minimize error and temperature drift. If the external resistors, R1 and R2, are not well-matched, they become a significant source of error in the system, so precision resistors are recommended to maintain performance. The ADR821 provides a precision voltage reference and integrated op amp that also reduces error in the signal chain.

The AD8276 has rail-to-rail output capability that allows higher current outputs.

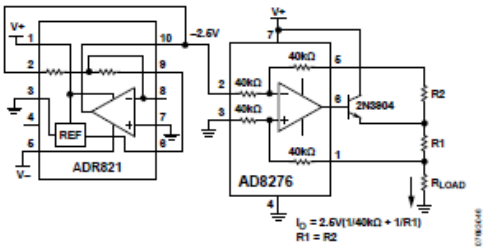


Figure 49. Constant Current Source

VOLTAGE AND CURRENT MONITORING

Voltage and current monitoring is critical in the following applications: power line metering, power line protection, motor control applications, and battery monitoring. The AD8276/AD8277 can be used to monitor voltages and currents in a system, as shown in Figure 50. As the signals monitored by the AD8276/AD8277 rise above or drop below critical levels, a circuit event can be triggered to correct the situation or raise a warning.

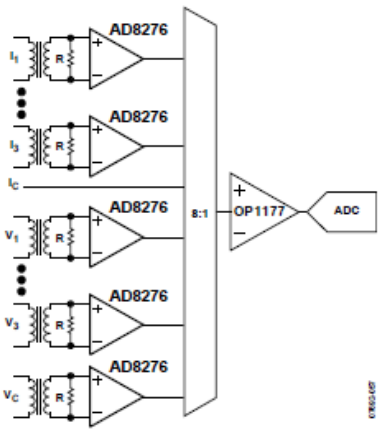


Figure 50. Voltage and Current Monitoring in 3-Phase Power Line Protection Using the AD8276

Figure 50 shows an example of how the AD8276 can be used to monitor voltage and current on a 3-phase power supply. I_1 through I_3 are the currents to be monitored, and V_1 through V_3 are the voltages to be monitored on each phase. I_C and V_C are the common or zero lines. Couplers or transformers interface the power lines to the front-end circuitry and provide attenuation, isolation, and protection.

On the current monitoring side, current transformers (CTs) step down the power-line current and isolate the front-end circuitry from the high voltage and high current lines. Across the inputs of each difference amplifier is a shunt resistor that converts the coupled current into a voltage. The value of the

Calcul des résistances en sorties de l'amplificateur