Projet Conception de Microsystèmes
Altimètre intégré en VHDL-AMS

Mohamed Hage Hassan
Clément Cheung

8 Decembre 2017

Prémabule

La conception des microsystèmes microélectromécaniques constitue une compétance importante dans le domaine de la microélectronique : L'intégration continue des élements de micro-nano dimensions assure la réduction de la taille des circuits mixtes microélectronique, qui peuvent comprendre une multitude de capteurs ainsi que la réalisations des fonctions de plus en plus complexes.
Le document présente les différentes méthodologies suivies ainsi que les architectures des blocs fonctionels décrits en VHDL-AMS, ainsi que l'incorporation et la simulation de modèle de capteur physique et les circuits de traitement. On finalise en effectuant une simulation complète du modèle pour caractériser ses performances vis-à-vis du cahier des charges.

Table des matières

1 Introduction

2 Architecture Générale de l'altimètre

3 Modélisation et simulation des blocs

3.1 Transducteurs comme élements de tests

3.2 Capteur de température

3.3 Architecture de l'amplificateur différentiel

3.4 Convertisseur Analogique-Numérique en VHDL-AMS

3.4.1 Convertiseur Analogique-Numérique 1-bit

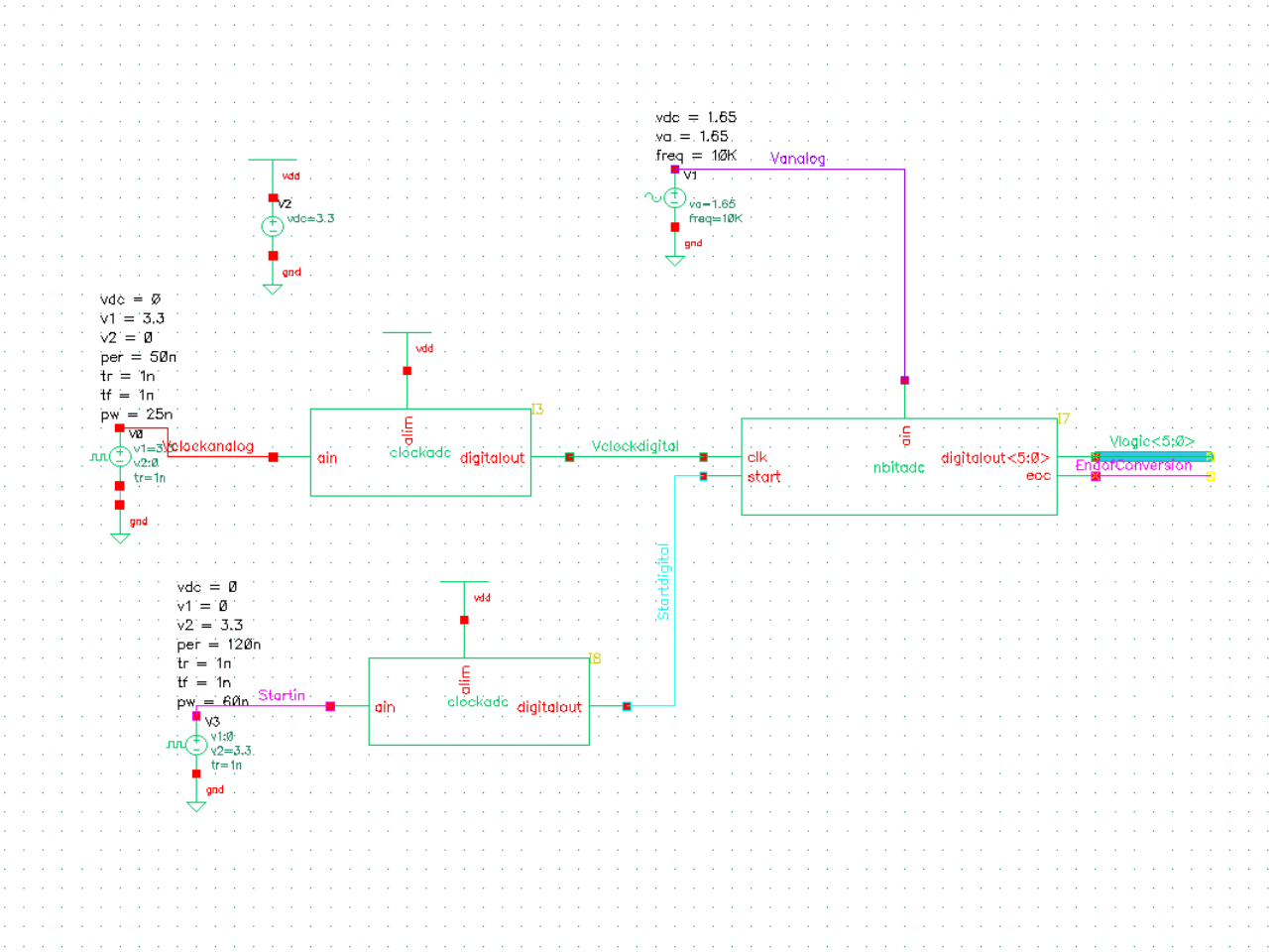

Figure 1: Schéma général pour le test de l'ADC 6-bits

Utilisation des bibiliothèques standards pour les éléments AMS

```
library ieee, std;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
```

Définitions des terminaux principaux : l'alimentation ainsi que le signal de la clock est à convertir vers une version digitale.

```
entity clockadc is
  port (
    terminal ain : electrical;  --Analog input terminal
    terminal alim : electrical; --Alim input terminal
    signal Digitalout : out std_ulogic -- Digital 1 bit output
    );
end clockadc;
```

L'architecture Conversion_clock principale de l'ADC, on définit les quantités physiques Vin et Vdd pour les tensions d'alimentation et l'entrée du signal à convertir. *lin* et *lalim* seront définies à 0 pour modéliser l'alimentation et la clock comme sources de tensions idéaux.

```
architecture Conversion_clock of clockadc is

  quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
  quantity Vdd across lalim through alim to electrical_ref; -- Alim Vdd input

  begin

  -- Process core --

    end process conversion;

  lin == 0.0; -- ideal input
  lalim == 0.0; -- ideal input

end Conversion_clock;
```

Le coeur de l'architecture comporte un modèle simplifier pour convertir un signal analogique dépassant une tension de seuil définie (Vdd) vers un bit logique "1", "0" sinon.

```
conversion:process(Vin'above(Vdd/2.0))

begin
if Vin'above(Vdd/2.0) then
  Digitalout<= '1';
else
  Digitalout <='0';
end if;
```

La simulation d'un tel convertisseur nous remporte des valeurs digitals à partir de celles analogiques.

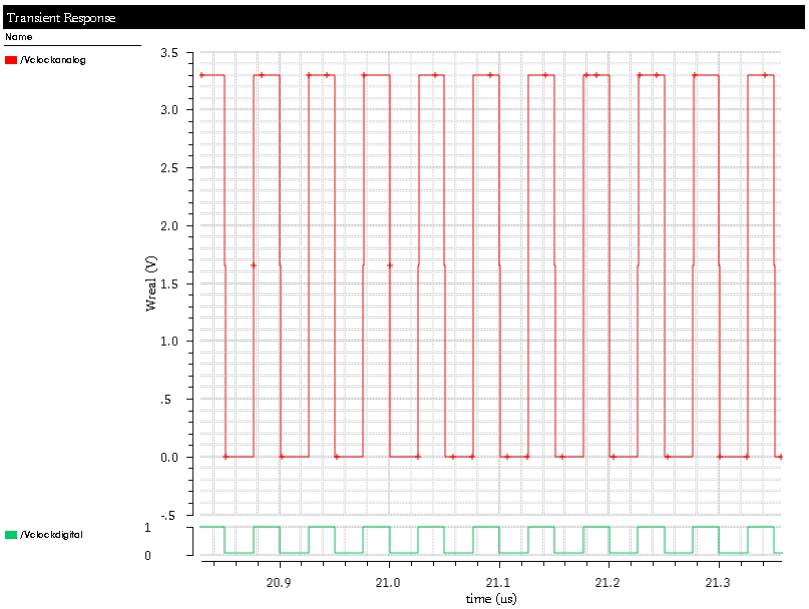

Figure 2: Simulation de l'ADC 1-bit

3.4.2 Convertiseur Analogique-Numérique 6-bits

Définition des bibilothèques principaux :

```
library ieee, std;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
```

Définition de l'entité : L'architecture de l'ADC nécessite un signal *start* pour le déclanchement de l'ADC, une horloge de candancement *clck*, l'entrée analogique à convertire *ain*, le signal de la fin de conversion *eoc* ainsi qu'un bus en *std_ulogic_vector* qui nous rends les valeurs digitales.

```
entity nbitadc is
  port (
    signal start: in std_ulogic; -- Start signal from the command logic
    signal clk : in std_ulogic; --clock signal
    terminal ain : electrical;  --Analog input terminal
    signal eoc : out std_ulogic:='0'; -- End of conversion status, initialized on default and
    used by the command logic
    signal Digitalout : out std_ulogic_vector(5 downto 0) -- Digital 6 bits output
    );
end nbitadc;
```

La conversion de l'ADC se déroule dans 2 états différents *initial, conversion* : On débute par l'intitalisation des valeurs de l'ADS,

```
architecture Conversion_alpha of nbitadc is
  constant delay:time:=1 ns; -- Conversion time, might be unnecessary
  type adcstates is (initial, conversion); -- Conversion status both initial and on time
  constant bit_range:integer:=5; -- bit range for Digitalout
  quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
  constant Vmax:real:=3.3;

  begin
    convertion_adc:process is
      variable thresh:real:= Vmax;  -- Threshold to test the input voltage against
      variable Vtmp:real; -- Temporary storage of Vin
      variable digital_tmp : std_ulogic_vector(bit_range downto 0); -- Temporary digital
    output data
      variable actual_status: adcstates:=initial; -- Begin the ADC states with the initial
    state
      variable bit_cnt:integer:=bit_range;

          -- Process core --

      lin == 0.0; -- ideal input
    end Conversion_alpha;
```

```
      begin
      case actual_status is

        when initial =>
        wait on start until start='1' or start='H';
          bit_cnt:=bit_range;
          thresh := Vmax;
          Vtmp := Vin;
          eoc<='0';
          actual_status:= conversion; -- Jump to conversion state

        when conversion =>
          wait on clk until clk='1' or clk='H';
          thresh:= thresh/2.0; -- MSB value
          if Vtmp > thresh then
            digital_tmp(bit_cnt):= '1';
            Vtmp := Vtmp - thresh;
          else
            digital_tmp(bit_cnt):='0';
          end if;
          if bit_cnt > 0 then
            bit_cnt := bit_cnt -1;
          else
```

```
            Digitalout <= digital_tmp;
            eoc <= '1' after delay; -- End of conversion after a delay
            actual_status := initial;
          end if;
      end case;
    end process convertion_adc;
```

Listing 1: Coeur du process pour le convertisseur

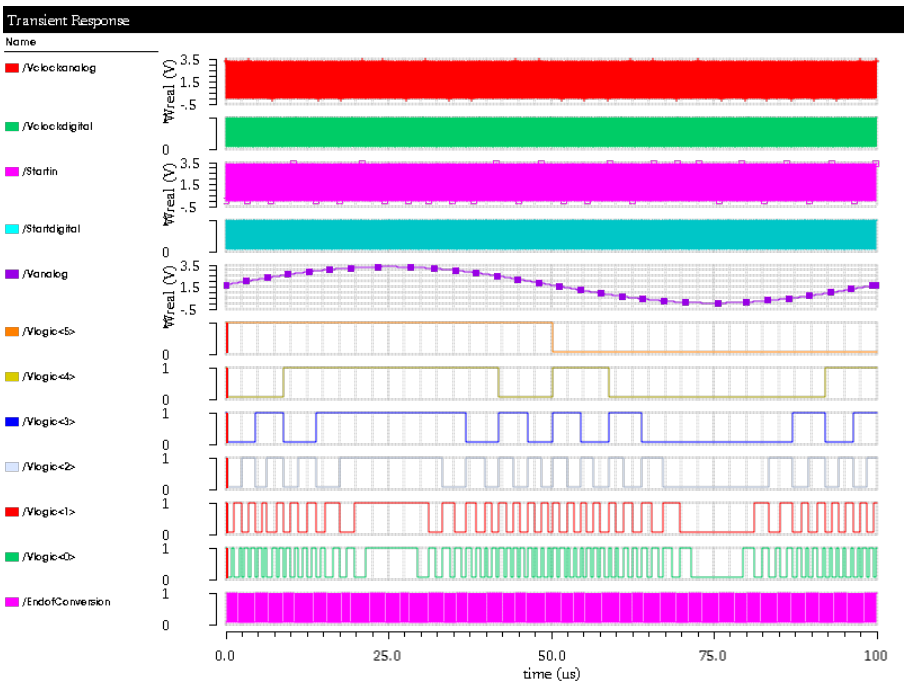

Figure 3: Simulation de totale de l'ADC

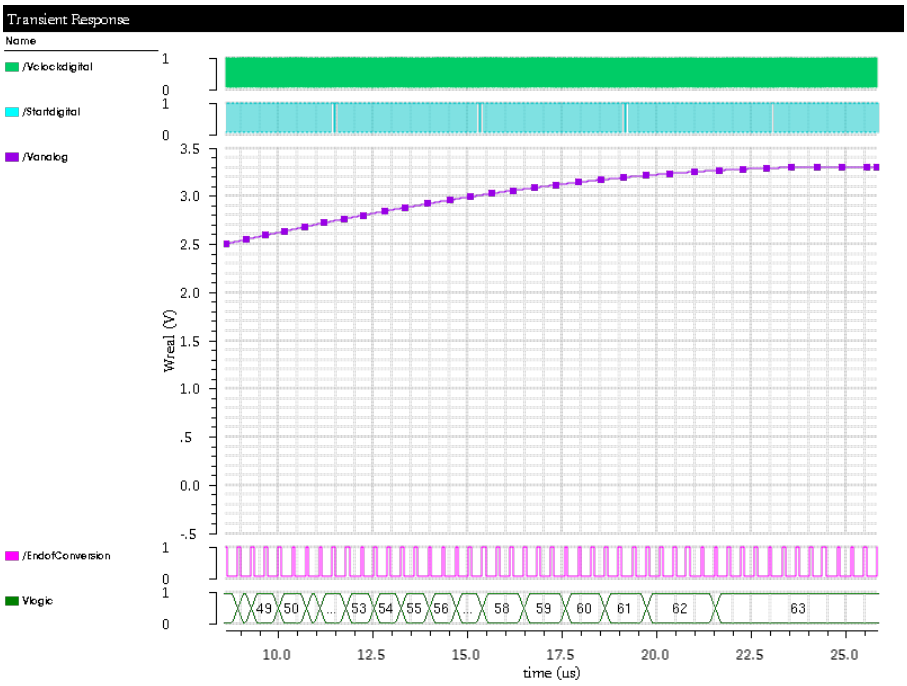

Figure 4: Simulation de l'ADC

3.5 Logique de commande

4 Conclusion

5 Annexes

5.1 Code complet des blocs VHDL-AMS

5.1.1 Convertisseur nbitsADC 6-bits

```
library ieee, std;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;

entity nbitadc is
  port (
    signal start: in std_ulogic; -- Start signal from the command logic
    signal clk : in std_ulogic; --clock signal
    terminal ain : electrical;  --Analog input terminal
    signal eoc : out std_ulogic:='0'; -- End of conversion status, initialized on default and
    used by the command logic
    signal Digitalout : out std_ulogic_vector(5 downto 0) -- Digital 6 bits output
    );
end nbitadc;

architecture Conversion_alpha of nbitadc is
  constant delay:time:=1 ns; -- Conversion time, might be unnecessary
  type adcstates is (initial, conversion); -- Conversion status both initial and on time
  constant bit_range:integer:=5; -- bit range for Digitalout
  quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
  constant Vmax:real:=3.3;

  begin
    convertion_adc:process is
      variable thresh:real:= Vmax;  -- Threshold to test the input voltage against
      variable Vtmp:real; -- Temporary storage of Vin
      variable digital_tmp : std_ulogic_vector(bit_range downto 0); -- Temporary digital
    output data
      variable actual_status: adcstates:=initial; -- Begin the ADC states with the initial
    state
      variable bit_cnt:integer:=bit_range;

      begin
      case actual_status is

        when initial =>
        wait on start until start='1' or start='H';
          bit_cnt:=bit_range;
          thresh := Vmax;
          Vtmp := Vin;
          eoc<='0';

          actual_status:= conversion; -- Jump to conversion state

        when conversion =>
          wait on clk until clk='1' or clk='H';
          thresh:= thresh/2.0; -- MSB value
          if Vtmp > thresh then
            digital_tmp(bit_cnt):= '1';
            Vtmp := Vtmp - thresh;
          else
            digital_tmp(bit_cnt):='0';
          end if;
          if bit_cnt > 0 then
            bit_cnt := bit_cnt -1;
          else
            Digitalout <= digital_tmp;
            eoc <= '1' after delay; -- End of conversion after a delay
            actual_status := initial;
          end if;
      end case;
    end process convertion_adc;

  lin == 0.0; -- ideal input
  -- lalim == 0.0; --ideal power input

end Conversion_alpha;
```

5.1.2 Convertiseur 1bit clockadc

```
library ieee, std;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;

entity clockadc is
  port (
    terminal ain : electrical;  --Analog input terminal
    terminal alim : electrical; --Alim input terminal
    signal Digitalout : out std_ulogic -- Digital 1 bit output
    );
end clockadc;

architecture Conversion_clock of clockadc is

  quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
  quantity Vdd across lalim through alim to electrical_ref; -- Alim Vdd input

  begin

      conversion:process(Vin'above(Vdd/2.0))

      begin
      if Vin'above(Vdd/2.0) then
        Digitalout<= '1';
      else
        Digitalout <='0';
      end if;

    end process conversion;

  lin == 0.0; -- ideal input
  lalim == 0.0; -- ideal input

end Conversion_clock;
```

Références

[1] *The System Designers Guide to VHDL-AMS Analog, Mixed-Signal, and Mixed-Technology Modeling,* Peter J. Ashenden, Gregory D. Peterson and Darrell A. Teegarden, Morgan Kaufmann Publishers Inc