

# 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 compétence importante dans le domaine de la microélectronique : L'intégration continue des éléments 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Architecture Générale de l'altimètre</b>	<b>2</b>
<b>3</b>	<b>Modélisation et simulation des blocs</b>	<b>2</b>
3.1	Transducteurs comme éléments de tests . . . . .	2
3.2	Capteur de température . . . . .	2
3.3	Architecture de l'amplificateur différentiel . . . . .	2
3.4	Convertisseur Analogique-Numérique en VHDL-AMS . . . . .	3
3.4.1	Convertisseur Analogique-Numérique 1-bit . . . . .	3
3.4.2	Convertisseur Analogique-Numérique 6-bits . . . . .	5
3.5	Logique de commande . . . . .	8
<b>4</b>	<b>Conclusion</b>	<b>9</b>
<b>5</b>	<b>Annexes</b>	<b>10</b>
5.1	Code complet des blocs VHDL-AMS . . . . .	10
5.1.1	Convertisseur nbitsADC 6-bits . . . . .	10
5.1.2	Convertisseur 1bit clockadc . . . . .	11
	<b>Références</b>	<b>12</b>

- 1 Introduction**
- 2 Architecture Générale de l’altimètre**
- 3 Modélisation et simulation des blocs**
  - 3.1 Transducteurs comme éléments 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 Convertisseur Analogique-Numérique 1-bit

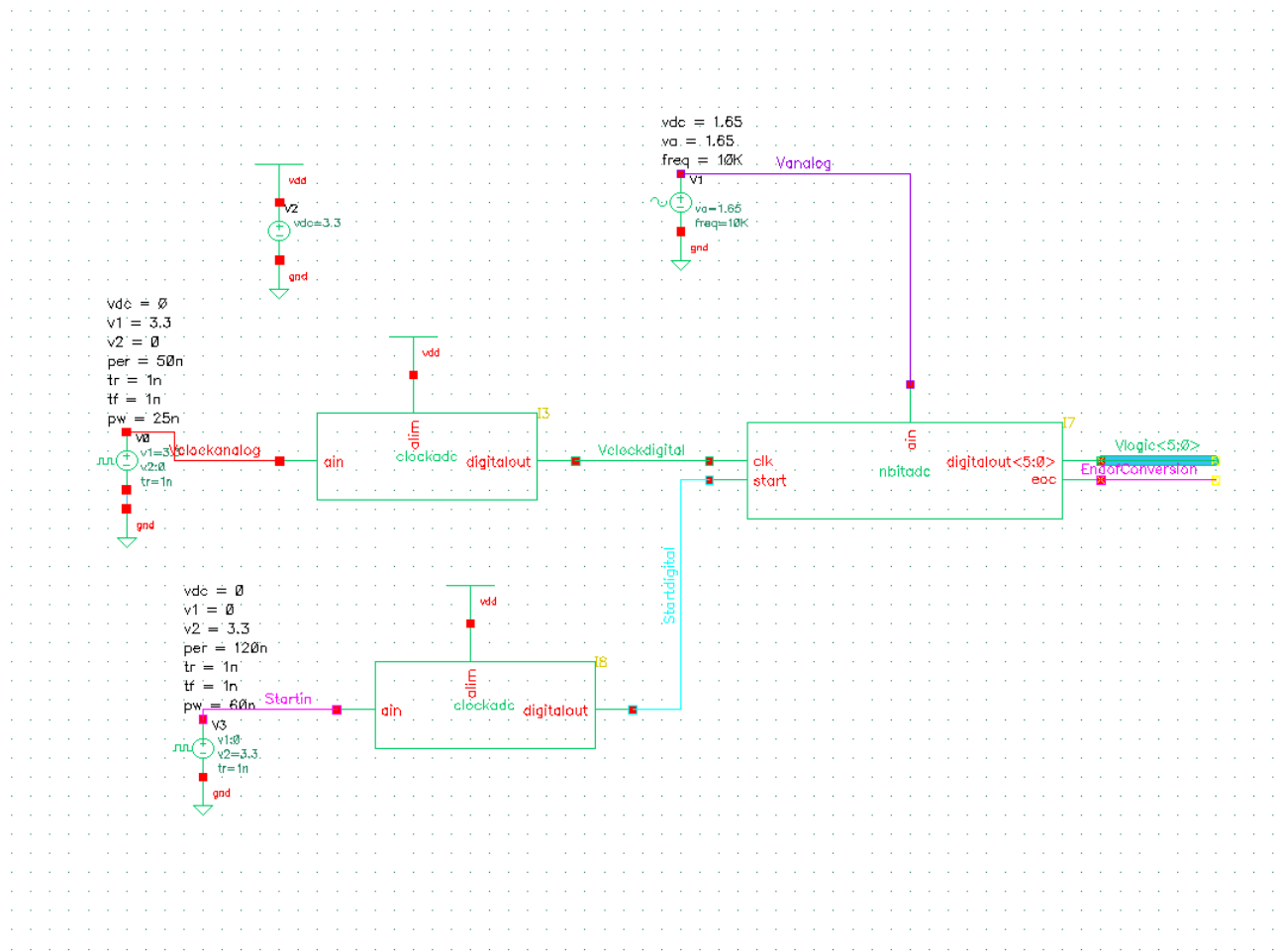


Figure 1: Schéma général pour le test de l'ADC 6-bits

Utilisation des bibliothèques standards pour les éléments AMS

```
1 library ieee, std;
2 use ieee.std_logic_1164.all;
3 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.

```
1 entity clockadc is
2   port (
3     terminal ain : electrical; -- Analog input terminal
4     terminal alim : electrical; -- Alim input terminal
5     signal Digitalout : out std_ulogic -- Digital 1 bit output
6   );
7 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.

```

1 architecture Conversion_clock of clockadc is
3     quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
     quantity Vdd across lalim through alim to electrical_ref; -- Alim Vdd input
5
     begin
7
         -- Process core --
9
         end process conversion;
11
         lin == 0.0; -- ideal input
13         lalim == 0.0; -- ideal input
15 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.

```

1 conversion : process (Vin' above(Vdd/2.0))
3 begin
4 if Vin' above(Vdd/2.0) then
5     Digitalout <= '1';
6 else
7     Digitalout <= '0';
8 end if;

```

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

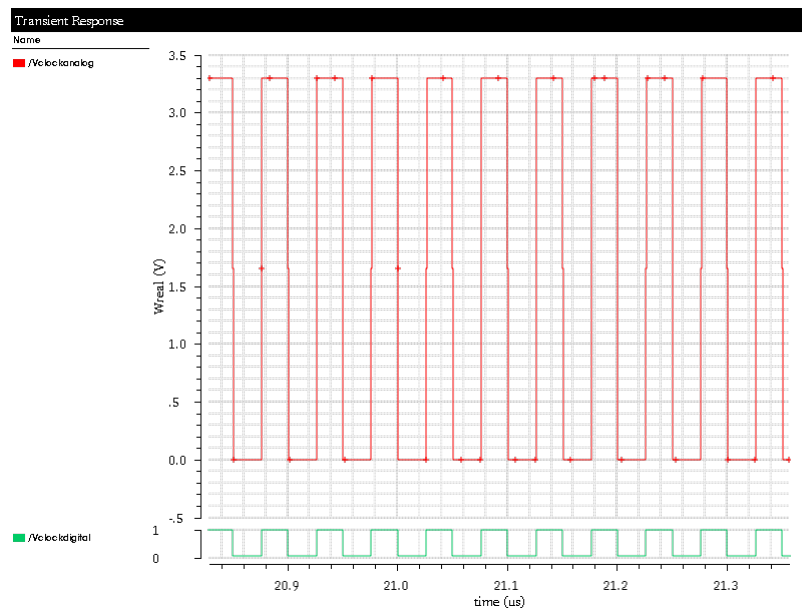


Figure 2: Simulation de l'ADC 1-bit

### 3.4.2 Convertisseur Analogique-Numérique 6-bits

Définition des bibliothèques principaux :

```

1 library ieee, std;
2 use ieee.std_logic_1164.all;
3 use ieee.electrical_systems.all;

```

Définition de l'entité : L'architecture de l'ADC nécessite un signal *start* pour le déclenchement de l'ADC, une horloge de cadencement *clk*, l'entrée analogique à convertir *ain*, le signal de la fin de conversion *eoc* ainsi qu'un bus en *std\_ulogic\_vector* qui nous rends les valeurs digitales.

```

1 entity nbitadc is
2   port (
3     signal start: in std_ulogic; -- Start signal from the command logic
4     signal clk : in std_ulogic; --clock signal
5     terminal ain : electrical; --Analog input terminal
6     signal eoc : out std_ulogic:= '0'; -- End of conversion status, initialized on default and
7       used by the command logic
8     signal Digitalout : out std_ulogic_vector(5 downto 0) -- Digital 6 bits output
9   );
10 end nbitadc;

```

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

```

1 architecture Conversion_alpha of nbitadc is
2   constant delay:time:=1 ns; -- Conversion time, might be unnecessary
3   type adcstates is (initial, conversion); -- Conversion status both initial and on time
4   constant bit_range:integer:=5; -- bit range for Digitalout
5   quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
6   constant Vmax:real:=3.3;
7
8   begin
9     conversion_adc:process is
10      variable thresh:real:= Vmax; -- Threshold to test the input voltage against
11      variable Vtmp:real; -- Temporary storage of Vin
12      variable digital_tmp : std_ulogic_vector(bit_range downto 0); -- Temporary digital
13      output data
14      variable actual_status: adcstates:=initial; -- Begin the ADC states with the initial
15      state
16      variable bit_cnt:integer:=bit_range;
17
18      -- Process core --
19
20      lin == 0.0; -- ideal input
21    end Conversion_alpha;

```

```

1   begin
2     case actual_status is
3
4       when initial =>
5         wait on start until start='1' or start='H';
6         bit_cnt:=bit_range;
7         thresh := Vmax;
8         Vtmp := Vin;
9         eoc<='0';
10        actual_status:= conversion; -- Jump to conversion state
11
12      when conversion =>
13        wait on clk until clk='1' or clk='H';
14        thresh:= thresh/2.0; -- MSB value
15        if Vtmp > thresh then
16          digital_tmp(bit_cnt):= '1';
17          Vtmp := Vtmp - thresh;
18        else
19          digital_tmp(bit_cnt):= '0';
20        end if;
21        if bit_cnt > 0 then
22          bit_cnt := bit_cnt -1;
23        else

```

```

25         Digitalout <= digital_tmp;
        eoc <= '1' after delay; -- End of conversion after a delay
        actual_status := initial;
27     end if;
    end case;
29 end process conversion_adc;

```

Listing 1: Coeur du process pour le convertisseur

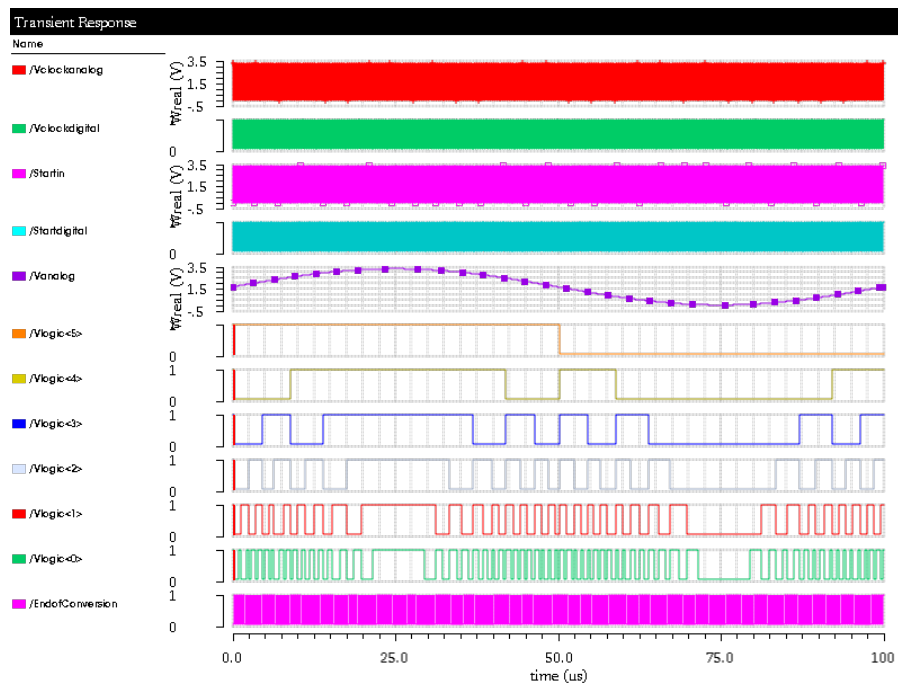


Figure 3: Simulation de totale de l'ADC

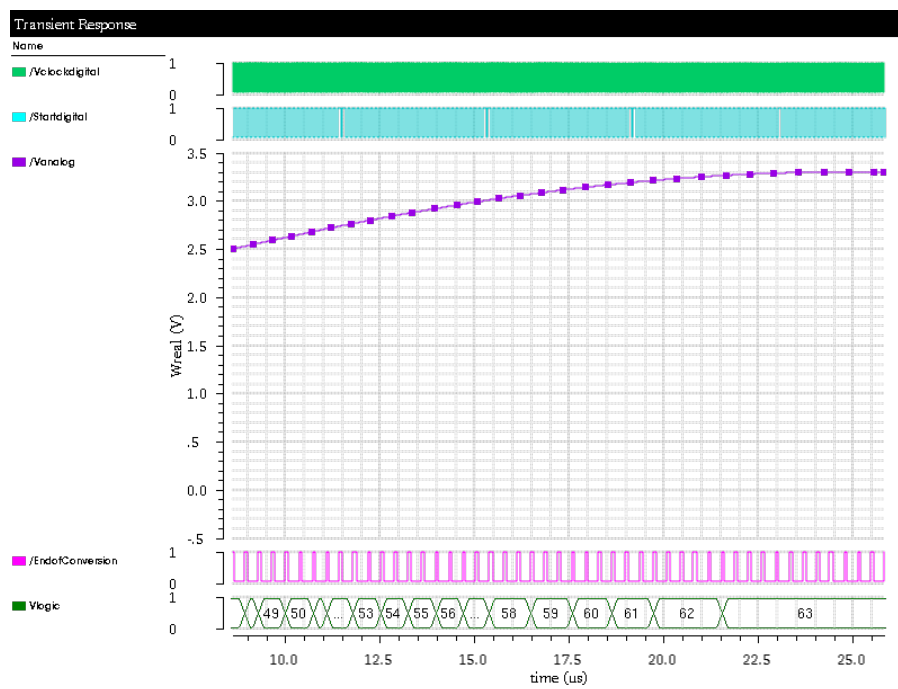


Figure 4: Simulation de l'ADC

### 3.5 Logique de commande



## 4 Conclusion

## 5 Annexes

### 5.1 Code complet des blocs VHDL-AMS

#### 5.1.1 Convertisseur nbitsADC 6-bits

```
1 library ieee, std;
2 use ieee.std_logic_1164.all;
3 use ieee.electrical-systems.all;
4
5 entity nbitadc is
6     port (
7         signal start: in std_ulogic; -- Start signal from the command logic
8         signal clk : in std_ulogic; --clock signal
9         terminal ain : electrical; --Analog input terminal
10        signal eoc : out std_ulogic:= '0'; -- End of conversion status, initialized on default and
11        used by the command logic
12        signal Digitalout : out std_ulogic_vector(5 downto 0) -- Digital 6 bits output
13    );
14 end nbitadc;
15
16 architecture Conversion_alpha of nbitadc is
17     constant delay:time:=1 ns; -- Conversion time, might be unnecessary
18     type adcstates is (initial, conversion); -- Conversion status both initial and on time
19     constant bit_range:integer:=5; -- bit range for Digitalout
20     quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
21     constant Vmax:real:=3.3;
22
23     begin
24         conversion_adc:process is
25             variable thresh:real:= Vmax; -- Threshold to test the input voltage against
26             variable Vtmp:real; -- Temporary storage of Vin
27             variable digital_tmp : std_ulogic_vector(bit_range downto 0); -- Temporary digital
28             output data
29             variable actual_status: adcstates:=initial; -- Begin the ADC states with the initial
30             state
31             variable bit_cnt:integer:=bit_range;
32
33             begin
34                 case actual_status is
35
36                     when initial =>
37                         wait on start until start='1' or start='H';
38                         bit_cnt:=bit_range;
39                         thresh := Vmax;
40                         Vtmp := Vin;
41                         eoc<='0';
42
43                         actual_status:= conversion; -- Jump to conversion state
44
45                     when conversion =>
46                         wait on clk until clk='1' or clk='H';
47                         thresh:= thresh/2.0; -- MSB value
48                         if Vtmp > thresh then
49                             digital_tmp(bit_cnt):= '1';
50                             Vtmp := Vtmp - thresh;
51                         else
52                             digital_tmp(bit_cnt):='0';
53                         end if;
54                         if bit_cnt > 0 then
55                             bit_cnt := bit_cnt -1;
56                         else
57                             Digitalout <= digital_tmp;
58                             eoc <= '1' after delay; -- End of conversion after a delay
59                             actual_status := initial;
60                         end if;
61                     end case;
62                 end process conversion_adc;
63
64                 lin == 0.0; -- ideal input
65                 -- lalim == 0.0; --ideal power input
66 end Conversion_alpha;
```

### 5.1.2 Convertiseur 1bit clockadc

```
library ieee, std;
2 use ieee.std_logic_1164.all;
3 use ieee.electrical_systems.all;
4
5 entity clockadc is
6   port (
7     terminal ain : electrical; --Analog input terminal
8     terminal alim : electrical; --Alim input terminal
9     signal Digitalout : out std_ulogic -- Digital 1 bit output
10    );
11 end clockadc;
12
13 architecture Conversion_clock of clockadc is
14
15   quantity Vin across lin through ain to electrical_ref; -- ADC Analog input
16   quantity Vdd across lalim through alim to electrical_ref; -- Alim Vdd input
17
18   begin
19
20     conversion : process (Vin'above(Vdd/2.0))
21
22     begin
23       if Vin'above(Vdd/2.0) then
24         Digitalout <= '1';
25       else
26         Digitalout <= '0';
27       end if;
28
29     end process conversion;
30
31     lin == 0.0; -- ideal input
32     lalim == 0.0; -- ideal input
33
34 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