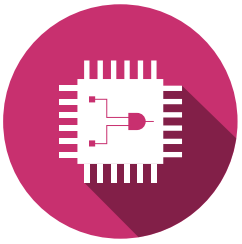


VHSIC Hardware Design Language (ex. VHDL)

Exercises systèmes embarqués

1 Introduction

2 Description structurelle

3 Description comportementale

3.1 Assignements dans un processus

Le processus suivant contient deux assignements. Au moment du changement d'état d'un des signaux de la liste de sensibilité, il est évalué à nouveau.

```
1 process(in1, in2, in3)
2 begin
3   or12  <= in1 or in2 after 1 ns;
4   or123 <= or12 or in3 after 1 ns;
5 end process;
```

Listing 1 – Processus OU

Dessiner le déroulement temporel des signaux `or12` et `or123` dans le chronogramme de la figure 1.

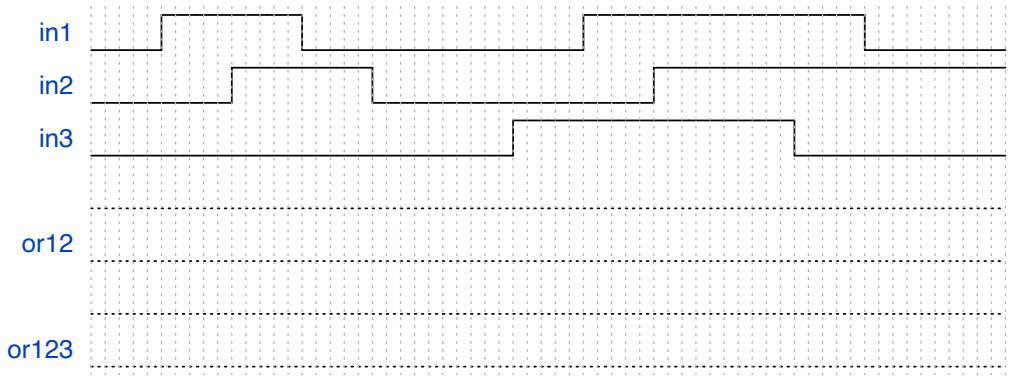
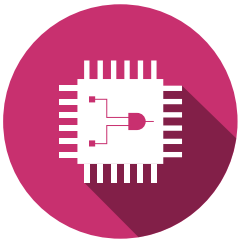


FIGURE 1 – Chronogramme

La grille verticale de la figure 1 un pas de $1ns$.

3.2 Multiplexeur

Ecrire l'architecture du multiplexeur dont l'entité est donnée ci-dessous.



```vhdl
library ieee;
  use ieee.std_logic_1164.all;

entity multiplexer is
  port(
    sel    : in  std_ulogic;
    in0    : in  std_ulogic;
    in1    : in  std_ulogic;
    muxOut : out std_ulogic
  );
end multiplexer;
```

Listing 2 – Multiplexeur **entity**

Le signal de commande `sel` peut prendre 9 valeurs distinctes. Assigner la valeur `'X'` à la sortie lorsque sa valeur ne peut être déterminée. Tenir compte du fait que si les deux entrées `in0` et `in1` sont identiques, la sortie prend la valeur de ces entrées, même si le signal de commande a une valeur indéterminée.

4 Types usuels et opérations

4.1 Transformation de non-signé à signé

Ecrire l'architecture du circuit qui transforme un nombre non-signé en un nombre signé et dont l'entité est donnée ci-dessous.

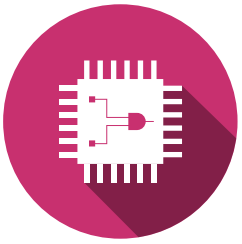
```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity transformToSigned is
  generic(
    dataBitNb : positive := 8
  );
  port(
    signedIn  : in signed(dataBitNb-1 downto 0);
    unsignedOut : out unsigned(dataBitNb-1 downto 0)
  );
end transformToSigned;
```

Listing 3 – Transform to signed **entity**

La gamme du signal d'entrée est de 0 à $2^{dataBitNb} - 1$. Effectuer la transformation de manière à ce que la valeur 0, la valeur minimale du signal d'entrée, soit transformée en $-2^{dataBitNb-1}$, la valuer minimale du signal de sortie et que la valeur maximale $2^{dataBitNb} - 1$ de l'entrée donnera $2^{dataBitNb-1} - 1$, la valeur maximal de la sortie. De cette manière, la valeur $2^{dataBitNb}$ de l'entrée donnera 0.

4.2 Réduction de la gamme d'un signal

Ecrire l'architecture du circuit qui réduit la gamme d'un signal et dont l'entité est donnée ci-dessous.



```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity rangeReducer is
  generic(
    signalBitNb     : positive := 16;
    saturationBitNb : positive := 8
  );
  port(
    inputSignal  : in  unsigned(signalBitNb-1 downto 0);
    outputSignal : out unsigned(signalBitNb-1 downto 0)
  );
end rangeReducer;
```

Listing 4 – Range reducer `entity`

L'opération réduit l'amplitude maximale du signal de sortie à $2^{saturationBitNb} - 1$. Elle s'effectue avec saturation : si l'amplitude du signal d'entrée dépasse la valeur maximale, alors la sortie prendra cette valeur maximale.

Considérer que `saturationBitNb` est toujours strictement plus petit que `signalBitNb`.

4.3 Compteur avec remise à zéro synchrone

Ecrire l'architecture du compteur avec une commande remise à zéro synchrone, `signalBitNb`, dont l'entité est donnée ci-dessous.

```vhdl
library ieee;
  use ieee.std_logic_1164.all;

entity counter is
  port(
    reset      : in  std_ulogic;
    clock      : in  std_ulogic;
    synchReset : in  std_ulogic;
    countOut   : out unsigned(7 downto 0)
  );
end counter;
```

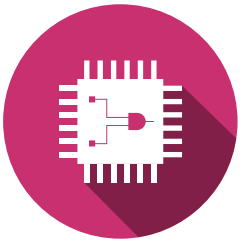
Listing 5 – Compteur `entity`

Le langage VHDL est très strict : il est seulement permis d'assigner une valeur à la sortie, et il est interdit d'en relire la valeur à l'intérieur de l'architecture.

4.4 Délai non redémarrable

Ecrire l'architecture du compteur qui génère une impulsion de durée égale à une période d'horloge et aparaissant 100 périodes d'horloge après le passage à `'1'` du signal d'entrée. Son entité est donnée ci-dessous.

```vhdl
library ieee;
  use ieee.std_logic_1164.all;

```



```vhdl
entity counter is
  port(
    reset    : in  std_ulogic;
    clock    : in  std_ulogic;
    trigger  : in  std_ulogic;
    pulseOut : out std_ulogic
  );
end counter;
```

Listing 6 – Compteur `entity`

Si l'entrée trigger passe à `'0'` puis à `'1'` avant que l'impulsion de sortie n'ait apparu, le circuit ignorera cette impulsion supplémentaire. Pour ce circuit, on parle de non-retriggerable one-shot.

Considérer que le signal d'entrée aura repassé à `'0'` avant que l'impulsion de sortie n'ait apparu.

Le langage VHDL est très strict : il est seulement permis d'assigner une valeur à la sortie, et il est interdit d'en relire la valeur à l'intérieur de l'architecture.

5 Paramètres génériques

5.1 Délai redémarrable

Ecrire l'entité et l'architecture du compteur qui génère une impulsion de durée égale à une période d'horloge et aparaissant `delay` périodes d'horloge après le passage à `'1'` du signal d'entrée. Le retard `delay` est configurable entre 1 et 100.

Si l'entrée `trigger` passe à `'0'` puis à `'1'` avant que l'impulsion de sortie n'ait apparu, le compteur interne redémarrera. Pour ce circuit, on parle de retriggerable one-shot.

Considérer que le signal d'entrée aura repassé à `'0'` avant que l'impulsion de sortie n'ait apparu.

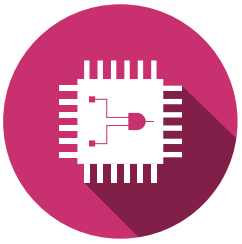
5.2 Additionneur

Ecrire l'architecture de l'additionneur itératif dont l'entité est donnée ci-après :

```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity iterativeAdder is
  port(
    Cin  : in  std_ulogic;
    Ai   : in  std_ulogic;
    Bi   : in  std_ulogic;
    Si   : out std_ulogic;
    Cout : out std_ulogic
  );
end iterativeAdder;
```

Listing 7 – iterativeAdder `entity`

A l'aide de ce composant, écrire l'architecture de l'additionneur à propagation de report dont



l'entité est donnée ci-après :

```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity adder is
  generic(
    bitNb : positive := 8
  );
  port(
    A        : in  unsigned (bitNb-1 downto 0);
    B        : in  unsigned (bitNb-1 downto 0);
    overflow : out std_ulogic;
    S        : out unsigned (bitNb-1 downto 0)
  );
end adder;
```

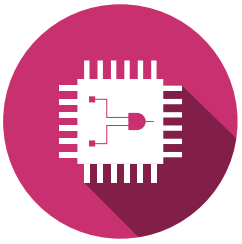
Listing 8 – adder `entity`

6 Signaux et variables

6.1 Compteurs

Donner la séquence des 3 compteurs de l'architecture VHDL suivante.

```vhdl
architecture RTL of threeCounters is
  signal count1_int: unsigned(count1'range);
  signal count2_int: unsigned(count2'range);
begin

  cnt1: process(reset, clock)
  begin
    if reset = '1' then
      count1_int <= (others => '0');
    elsif rising_edge(clock) then
      if count1_int = 6 then
        count1_int <= (others => '0');
      else
        count1_int <= count1_int + 1;
      end if;
    end if;
  end process cnt1;

  count1 <= count1_int;

  cnt2: process(reset, clock)
  begin
    if reset = '1' then
      count2_int <= (others => '0');
    elsif rising_edge(clock) then
      count2_int <= count2_int + 1;
      if count2_int = 6 then
        count2_int <= (others => '0');
      end if;
    end if;
```



```
31    end process cnt2;
32
33    count2 <= count2_int;
34
35    cnt3: process(reset, clock)
36      variable count3_int: unsigned(count3'range);
37    begin
38      if reset = '1' then
39        count3_int := (others => '0');
40      elsif rising_edge(clock) then
41        count3_int := count3_int + 1;
42        if count3_int = 6 then
43          count3_int := (others => '0');
44        end if;
45      end if;
46      count3 <= count3_int;
47    end process cnt3;
48
49  end RTL;
```

Listing 9 – 3 petits compteurs

6.2 Parité

Donner l'architecture VHDL du bloc qui la parité paire d'un vecteur de bits et dont l'entité est donnée ci-dessous.

```
1   library ieee;
2     use ieee.std_logic_1164.all;
3     use ieee.numeric_std.all;
4
5   entity parityCalc is
6     generic(
7       dataBitNb : positive := 8
8     );
9     port(
10      vectorIn  : in  std_ulogic_vector(dataBitNb-1 downto 0);
11      vectorOut : out std_ulogic_vector(dataBitNb-1 downto 0)
12    );
13  end parityCalc;
```

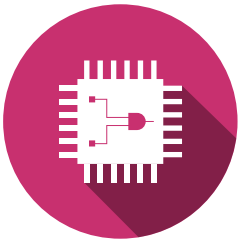
Listing 10 – parityCalc `entity`

Dans le cas d'une parité paire, le nombre total de bits à `'1'`, bit de parité inclus, est pair.

6.3 Majorité

Ecrire l'architecture du circuit qui détermine la majorité (`'0'` ou `'1'`) des bits d'un tableau avec un nombre impair d'éléments.

```
1   library ieee;
2     use ieee.std_logic_1164.all;
3     use ieee.numeric_std.all;
4
5   entity majorityFinder is
6     port(
7       poll     : in  std_uLogic_vector (1 to 7);
```



```
8       majority : out std_ulogic
9     );
10  end majorityFinder;
```

Listing 11 – majorityFinder `entity`

En règle générale, le calcul de la majorité se fait par le comptage des voix.

Acronymes

VHDL **V**HSIC **H**ardware **D**esign **L**anguage. 1–7