Electronique numérique

Initiation à VHDL (2/3)

Circuits séquentiels. Automates.

CORRECTIONS

1 Descriptions de chemin de données séquentiel

Questions

1. Coder en VHDL au moins deux architectures numériques qui réalisent un compteur, qui compte de 0 à 255, en respectant l'entité donnée ci-dessous. On note que les opérandes sont codés en **unsigned(7 downto 0)** (octets non-signés) ; On prendra soin de dessiner le circuit avant de coder l'architecture.
2. Tester le circuit à l'aide du testbench fourni.

Solutions On présente 2 architecture, *in fine* identiques, mais codées différemment. L'architecture 2 semble plus naturelle ; l'architecture 1 sépare explicitement partie combinatoire et partie séquentielle.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity compteur is
6    port(
7      reset_n : in std_logic;
8      clk : in std_logic;
9      value : out unsigned(7 downto 0)
10   );
11 end compteur;
12
13 architecture arch1 of compteur is
14   signal counter : unsigned(7 downto 0);
15   signal counter_comb : unsigned(7 downto 0);
16 begin
17
18   counter_p : process(reset_n,clk)
19   begin
```

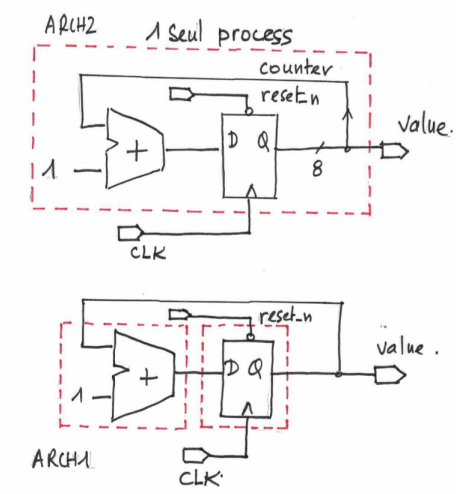


FIGURE 1 – Architecture du compteur, deux organisations de code possibles : soit 1 seul processus, soit 2 processus (1 processus + 1 assignation concurrente)

```
20      if reset_n='0' then
21        counter <= "00000000";
22      elsif rising_edge(clk) then
23        counter <= counter_comb;
24      end if;
25    end process;
26
27    counter_comb <= counter + 1;
28
29    value <= counter;
30  end arch1;
31
32  architecture arch2 of compteur is
33    signal counter : unsigned(7 downto 0);
34  begin
35    counter_p : process(reset_n,clk)
36    begin
37      if reset_n='0' then
38        counter <= "00000000";
39      elsif rising_edge(clk) then
40        counter <= counter + 1;
41      end if;
42    end process;
43
44    value <= counter;
45  end arch2;
```

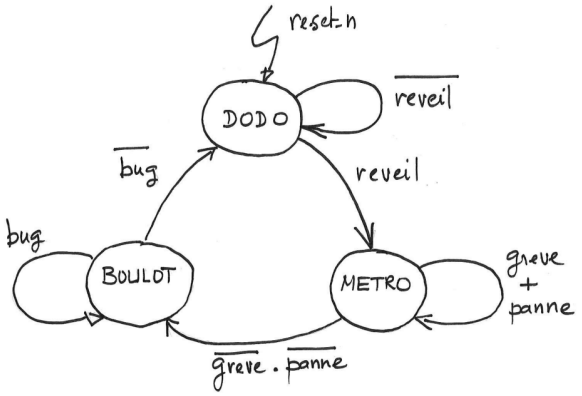


FIGURE 2 – Diagramme états-transitions de l'automate "I love Paris"

2 Descriptions d'automates au niveau logique

Soit un automate bien connu des informaticiens parisiens, sur la figure 2.

Questions

1. Vérifier que, dans le cas d'un encodage one-hot, les équations d'état suivant sont bien données par : $\begin{cases} D_0 &= Q_0.\overline{reveil} + Q_2.\overline{bug} \\ D_1 &= Q_0.reveil + Q_1.(greve + panne) \\ D_2 &= Q_1.\overline{greve}.\overline{panne} + Q_2.bug \end{cases}$
2. Coder l'automate "I_love_Paris" en VHDL, au niveau logique, en prenant soin de bien coder les bascules D à l'aide de processus. Respectez l'entité fournie ici. On utilisera une assignation conditionnelle (when) pour coder le signal de sortie "up_and_running".

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity i_love_paris is
6    port(
7      reset_n : in std_logic;
8      clk : in std_logic;
9      reveil : in std_logic;
10     panne : in std_logic;
11     greve : in std_logic;
12     bug : in std_logic;
13     up_and_running : out std_logic
14   );
15 end i_love_paris;
```

3. Utilisez le testbench donné sous Moodle pour tester votre automate. Observez le résultat.

Solutions

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity i_love_paris is
6    port(
7      reset_n : in std_logic;
8      clk : in std_logic;
9      reveil : in std_logic;
10     panne : in std_logic;
11     greve : in std_logic;
12     bug : in std_logic;
13     up_and_running : out std_logic
14   );
15 end i_love_paris;
16
17 architecture logic of i_love_paris is
18   signal state, next_state : std_logic_vector(2 downto 0);
19   --preferable to :
20   --signal d, q : std_logic_vector(2 downto 0);
21 begin
22   process(reset_n,clk)
23   begin
24     if reset_n='0' then
25       state <= "001";--dodo
26     elsif rising_edge(clk) then
27       state <= next_state;
28     end if;
29   end process;
30
31   next_state(0) <= (state(0) and not(reveil)) or (state(2) and not(bug));
32   next_state(1) <= (state(0) and reveil) or (state(1) and (greve or panne));
33   next_state(2) <= (state(2) and bug) or (state(1) and (not(greve) and not(panne));
34
35 end logic;
```

3 Descriptions d'automates au niveau RTL

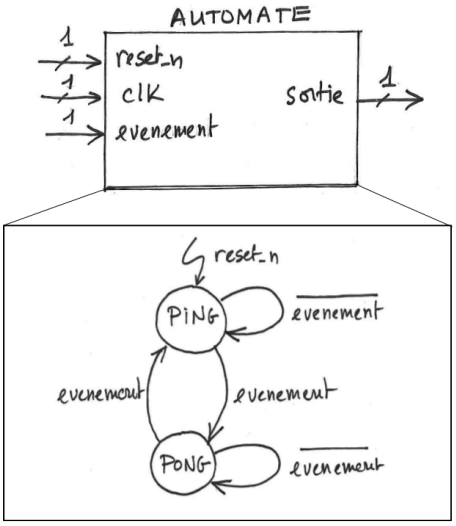
Ce niveau d'abstraction RTL est le plus couramment utilisé, car il permet de s'affranchir des détails des équations logiques, et décrire des automates (et *micro-architectures*) de manière plus naturelle. Le niveau RTL repose sur la notion d'*inférence matérielle* : l'Electronicien doit connaître certains *motifs*

de conception[1], afin de permettre à l'outil de synthèse d'établir automatiquement les équations logiques sous-jacentes. Le cas des automates d'états finis est instructif en ce sens. Voici un exemple de codage VHDL, qui décrit un automate, sans expliciter les équations logiques sous-jacentes, ni l'encodage des états. Pour information, le synthétiseur peut choisir de lui-même un encodage qui maximise la fréquence de fonctionnement du circuit (encodage one-hot généralement), ou tout autre type de contraintes imposées par le concepteur.

Exemple d'automate de niveau RTL Nous donnons ici à titre d'exemple un diagramme états-transitions, ainsi que son code VHDL de niveau RTL. Notez que nous avons séparé le processus ici appelé "next_state_p", qui code la fonction de transition (calcul de l'état suivant), et la sortie de l'automate.

1. "Design patterns", en anglais.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity automate is
6    port(
7      reset_n : in std_logic;
8      clk : in std_logic;
9      evenement : in std_logic;
10     sortie : out std_logic
11   );
12 end automate;
13
14 architecture rtl of automate is
15
16   type state_type is (PING,PONG);
17   signal state, next_state : state_type;
18
19 begin
20   process(reset_n,clk)
21   begin
22     if reset_n='0' then
23       state <= PING;
24     elsif rising_edge(clk) then
25       state <= next_state;
26     end if;
27   end process;
28
29   -- logique d'etat suivant
30   next_state_p: process(state,evenement)
31   begin
32     next_state <= state;--default
33     case state is
34       when PING =>
35        if evenement='1' then
36          next_state <= PONG;
37        end if;
38       when PONG =>
39        if evenement='1' then
40          next_state <= PING;
41        end if;
42       when others =>
43         null;
44     end case;
45   end process;
46
47   -- logique de sortie
48   sortie <='1' when state=PONG else '0';
49
50 end rtl;
```



Questions

1. Prenez le temps de comprendre le codage utilisé ici.
2. Coder l'automate "I_love_Paris" en VHDL, au niveau RTL, en vous inspirant de l'exemple donné.
3. Utilisez le même testbench que précédemment pour tester votre automate. Modifier uniquement le nom de l'architecture associée à l'entité instanciée.

Solutions

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity i_love_paris is
6    port(
7      reset_n : in std_logic;
8      clk : in std_logic;
9      reveil : in std_logic;
10     panne : in std_logic;
11     greve : in std_logic;
12     bug : in std_logic;
13     up_and_running : out std_logic
14   );
15 end i_love_paris;
16
17 architecture rtl of i_love_paris is
18
19   type state_type is (DODO,METRO,BOULOT);
20   signal state, next_state : state_type;
21
22 begin
23   process(reset_n,clk)
24   begin
25     if reset_n='0' then
26       state <= DODO;
27     elsif rising_edge(clk) then
28       state <= next_state;
29     end if;
30   end process;
31
32   next_state_p: process(state,reveil,greve,panne,bug)
33   begin
34     next_state <= state;--default
35     case state is
36       when DODO =>
37        if reveil='1' then
38          next_state <= METRO;
39        end if;
```

```
40          when METRO =>
41           if greve='0' and panne='0' then
42             next_state <= BOULOT;
43           end if;
44          when BOULOT =>
45            if bug='0' then
46              next_state <= DODO;
47            end if;
48          when others =>
49            null;
50       end case;
51    end process;
52
53    -- output logic :
54    up_and_running <= '0' when state=DODO else '1';
55
56 end rtl;
```