Electronique numérique
Initiation à VHDL (1/3)
CORRECTIONS

1. Dessiner l'enveloppe externe du composant décrit par cette entité. Voir figure 1.
2. Dessiner l'entité d'un demi-additionneur, puis le code VHDL de cette entité. Voir figure 4
3. Rappeler la constitution interne du demi-additionneur. Voir figure 4
4. Coder l'architecture de ce demi-additionneur.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity half_adder is
5    port(
6      a,b : in std_logic;
7      sum,cout : out std_logic
8    );
9  end half_adder;
10
11 architecture logic of half_adder is
12 begin
13   sum <= a xor b;
14   cout <= a and b;
15 end logic;
```

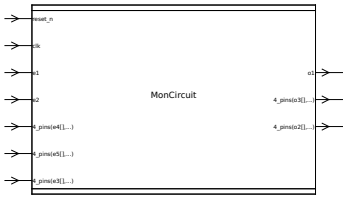

FIGURE 1 – Entity du composant MonCircuit

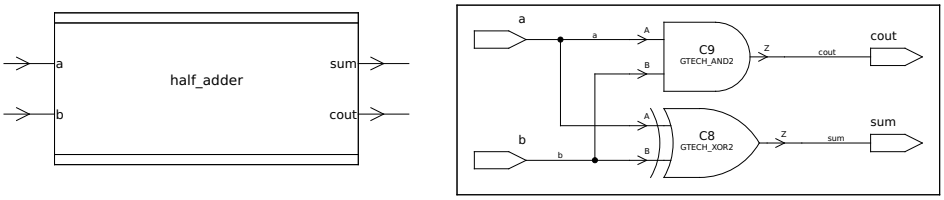

FIGURE 2 – Entité et architecture du demi-additionneur 1 bit

5. Dessinez la constitution interne d'un additionneur 1 bit complet, à partir du demi-additionneur.
6. Coder en VHDL l'architecture de cet additionneur 1 bit.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity full_adder is
5    port(
6      a,b : in std_logic;
7      cin : in std_logic;
8      sum,cout : out std_logic
9    );
10 end full_adder;
11
12 architecture logic of full_adder is
13   signal s_1, cout_1,cout_2 : std_logic;
14 begin
15
16   ha_1 : entity work.half_adder(logic)
17   port map(
18     a => a,
19     b => b,
20     sum => s_1,
21     cout => cout_1
22   );
23
24   ha_2 : entity work.half_adder(logic)
25   port map(
26     a => cin,
27     b => s_1,
28     sum => sum,
29     cout => cout_2
```

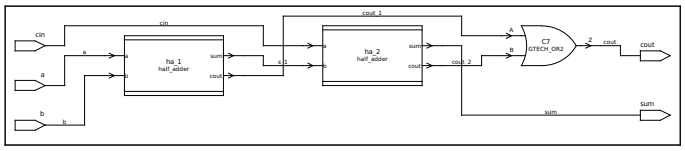

FIGURE 3 – Architecture du demi-additionneur 1 bit

```
30    );
31
32    cout <= cout_1 or cout_2;
33
34  end logic;
```

7. Coder en VHDL l'entité et l'architecture d'un additionneur 8 bits.

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity adder8b is
5     port(
6       a,b : in std_logic_vector(7 downto 0);
7       sum : out std_logic_vector(7 downto 0);
8       cout : out std_logic
9     );
10  end adder8b;
11
12  architecture structural of adder8b is
13    signal dummy_carry : std_logic;
14    signal carry : std_logic_vector(7 downto 0);
15  begin
16
17    dummy_carry <= '0';
18
19    fa_0 : entity work.full_adder(logic)
20    port map(
```

```
21      a => a(0),
22      b => b(0),
23      cin => dummy_carry,
24      sum => sum(0),
25      cout => carry(0)
26    );
27
28    fa_1 : entity work.full_adder(logic)
29    port map(
30      a => a(1),
31      b => b(1),
32      cin => carry(0),
33      sum => sum(1),
34      cout => carry(1)
35    );
36
37    fa_2 : entity work.full_adder(logic)
38    port map(
39      a => a(2),
40      b => b(2),
41      cin => carry(1),
42      sum => sum(2),
43      cout => carry(2)
44    );
45
46    fa_3 : entity work.full_adder(logic)
47    port map(
48      a => a(3),
49      b => b(3),
50      cin => carry(2),
51      sum => sum(3),
52      cout => carry(3)
53    );
54
55    fa_4 : entity work.full_adder(logic)
56    port map(
57      a => a(4),
58      b => b(4),
59      cin => carry(3),
60      sum => sum(4),
61      cout => carry(4)
62    );
63
64    fa_5 : entity work.full_adder(logic)
65    port map(
66      a => a(5),
67      b => b(5),
68      cin => carry(4),
69      sum => sum(5),
```

```
70        cout => carry(5)
71    );
72
73    fa_6 : entity work.full_adder(logic)
74    port map(
75        a => a(6),
76        b => b(6),
77        cin => carry(5),
78        sum => sum(6),
79        cout => carry(6)
80    );
81
82    fa_7 : entity work.full_adder(logic)
83    port map(
84        a => a(7),
85        b => b(7),
86        cin => carry(6),
87        sum => sum(7),
88        cout => carry(7)
89    );
90
91    cout <= carry(7);
92
93  end structural;
```

8. Le testbench présente également une entité. Localisez cette entité. En quoi est-elle particulière ? Comment peut-on l'expliquer ?

   Un tel banc de test possède une entité *vide* : elle ne possède aucune entrée, ni sortie. C'est le laboratoire, portes closes.

9. Un générateur d'horloge est contenu dans le banc de test. Combien de lignes sont nécessaires ? Dessinez le chronogramme de cette horloge.

   Une seule ligne suffit. On inverse le signal tous les demi-période d'horloge, à l'aide d'une assignation conditionnée par un signal "running", qui nous permettra de contrôler l'arrêt de la simulation

10. Comment notre banc de test provoque t-il concrètement cet arrêt ? Le signal running est affecté à "false" à la fin du processus de stimulation. Plus aucun événement n'est crée par le simulateur.

11. Dessiner le chronogramme des stimuli.

12. L'additionneur fonctionne-t-il avec des nombres signés ? Oui ! En tous les cas, un test semble aller en ce sens ! Voir figure **??**.

1 Script de compilation

Script de compilation GHDL (donné sous Moodle) :

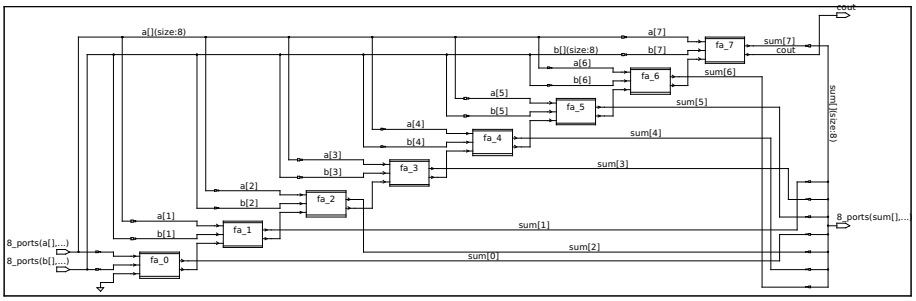

FIGURE 4 – Architecture de l'additionneur 8 bits

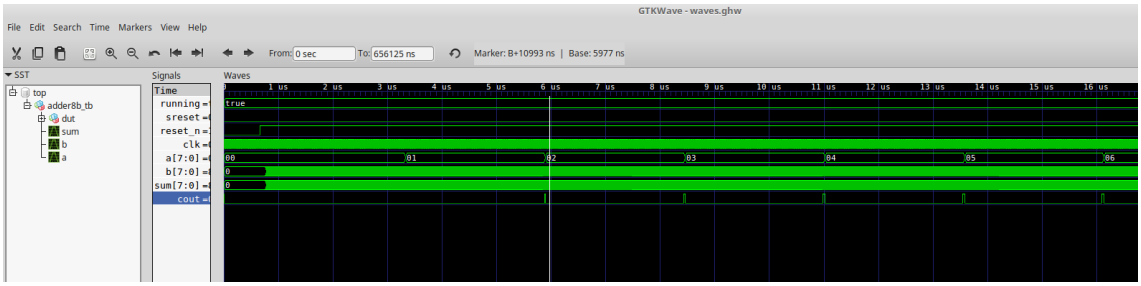

FIGURE 5 – Chronogramme (overview)

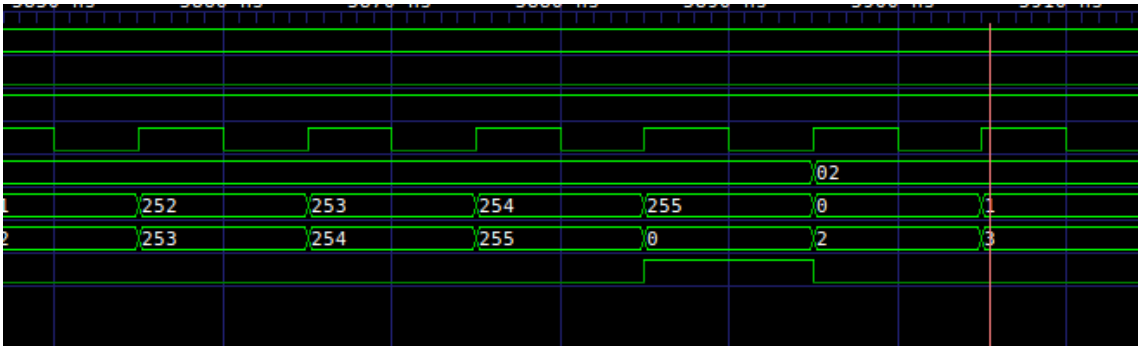

FIGURE 6 – Chronogramme : cas de $1 + 255$

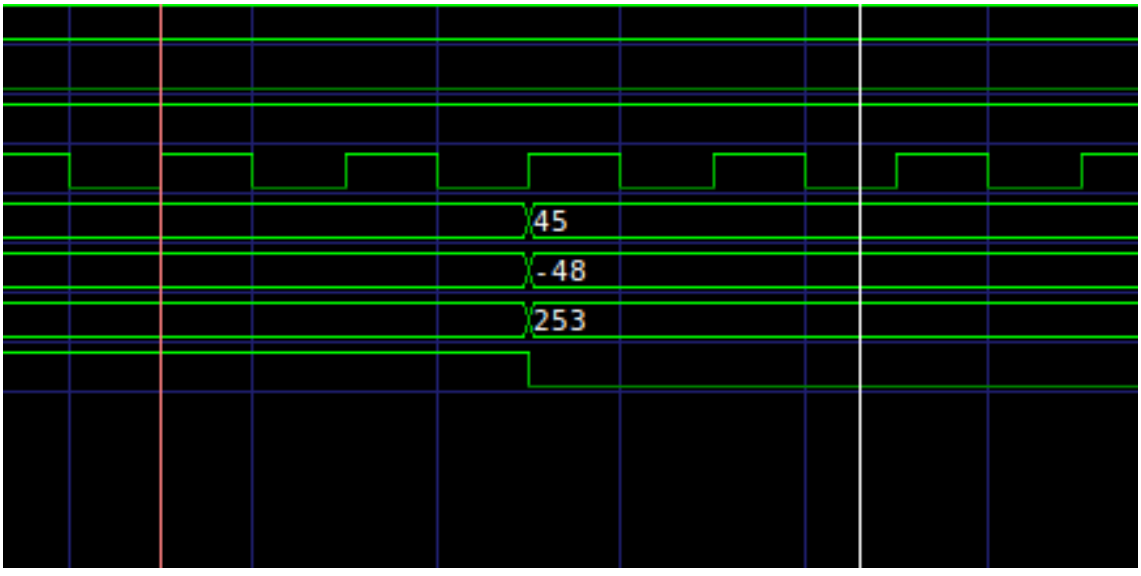

FIGURE 7 – L'additionneur fonctionne pour les nombres signés.

```
echo "=> cleaning..."
rm -rf waves.ghw *.o
echo "=> analyzing VHDL files..."
ghdl -a half_adder.vhd
ghdl -a full_adder.vhd
ghdl -a adder8b.vhd
ghdl -a adder8b_tb.vhd
echo "=> elaboration..."
ghdl -e adder8b_tb
echo "=> running simulation"
ghdl -r adder8b_tb --wave=waves.ghw
echo "=> starting waveform viewer"
gtkwave waves.ghw chrono.sav
```

Testbench (donné sous Moodle) :

```
1  ------------------------------------------------------------------------
2  -- This file was generated automatically by vhdl_tb Ruby utility
3  -- date : (d/m/y) 05/11/2018 15:40
4  -- Author : Jean-Christophe Le Lann - 2014
5  ------------------------------------------------------------------------
6  library ieee;
7  use ieee.std_logic_1164.all;
8  use ieee.numeric_std.all;
9
10 entity adder8b_tb is
11 end entity;
12
13 architecture bhv of adder8b_tb is
14
15   constant HALF_PERIOD : time := 5 ns;
16
17   signal clk : std_logic := '0';
18   signal reset_n : std_logic := '0';
19   signal sreset : std_logic := '0';
20   signal running : boolean := true;
21
22   procedure wait_cycles(n : natural) is
23    begin
24      for i in 1 to n loop
25        wait until rising_edge(clk);
26      end loop;
27    end procedure;
28
29   signal a : std_logic_vector(7 downto 0);
30   signal b : std_logic_vector(7 downto 0);
31   signal sum : std_logic_vector(7 downto 0);
32   signal cout : std_logic;
33
```

```
34  begin
35    ------------------------------------------------------------------------
36    -- clock and reset
37    ------------------------------------------------------------------------
38    reset_n <= '0','1' after 666 ns;
39
40    clk <= not(clk) after HALF_PERIOD when running else clk;
41
42    ------------------------------------------------------------------------
43    -- Design Under Test
44    ------------------------------------------------------------------------
45    dut : entity work.adder8b(using_generate)
46
47          port map (
48            a => a ,
49            b => b ,
50            sum => sum ,
51            cout => cout
52          );
53
54    ------------------------------------------------------------------------
55    -- sequential stimuli
56    ------------------------------------------------------------------------
57    stim : process
58      variable expected_sum : signed(7 downto 0);
59      variable nb_errors : natural := 0;
60      function str(v : std_logic_vector(7 downto 0)) return String is
61      begin
62        return integer'image(to_integer(unsigned(v)));
63      end ;
64
65     begin
66       a <= (others=>'0');
67       b <= (others=>'0');
68       report "running testbench for adder8b(structural)";
69       report "waiting for asynchronous reset";
70       wait until reset_n='1';
71       wait_cycles(10);
72       report "applying stimuli...";
73       for i in 0 to 255 loop
74         for j in 0 to 255 loop
75           wait until rising_edge(clk);
76           a <= std_logic_vector(to_unsigned(i,8));
77           b <= std_logic_vector(to_unsigned(j,8));
78           expected_sum := signed(a)+signed(b);
79           if expected_sum /= signed(sum) then
80             report "ERROR for sum : " & str(a) & "+" & str(b);
81             report "expecting " & str(std_logic_vector(expected_sum)) & ". Got " & str(sum);
82             nb_errors:=nb_errors+1;
```

```
83              else
84                report "OK for sum : " & str(a) & "+" & str(b);
85              end if;
86            end loop;
87          end loop;
88
89          wait_cycles(100);
90          a <= std_logic_vector(to_signed(45,8));
91          b <= std_logic_vector(to_signed(-48,8));
92          wait_cycles(100);
93          report "end of simulation";
94          report "number of errors detected : " & integer'image(nb_errors);
95          running <=false;
96          wait;
97        end process;
98
99  end bhv;
```

2 Using generate

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   architecture using_generate of adder8b is
5     signal dummy_carry : std_logic;
6     signal carry : std_logic_vector(7 downto 0);
7   begin
8
9     dummy_carry <= '0';
10
11    Loop_index: for i in 0 to 7 generate
12
13      first_stage: if i=0 generate
14        stage_0: entity work.full_adder(logic)
15        port map(
16          a => a(i),
17          b => b(i),
18          cin => dummy_carry,
19          sum => sum(i),
20          cout => carry(i)
21        );
22      end generate;
23
24      other_stages: if i>0 generate
25        state_i: entity work.full_adder(logic)
26        port map(
27          a => a(i),
```

```
28            b => b(i),
29            cin => carry(i-1),
30            sum => sum(i),
31            cout => carry(i)
32          );
33      end generate;
34
35    end generate;
36
37    cout <= carry(7);
38
39  end using_generate;
```