ECEN302 Lab 3 - Functions, Procedures, and Test Benches

Joshua Benfell - 300433229

August 29, 2020

1 Objectives

The primary objectives of this lab was to learn to develop reusable VHDL code with the inbuilt procedures and functions. We are to learn the differences between both constructs in VHDL and this will help us develop our thinking and understanding about sequential and procedural programming. We will be able to verify this learning when we have developed models for combinatorial logic using both functions and procedures and when we develop test benches to verify their functionality.

2 Methodology

2.1 Introduction

In this lab we will be implementing models for various combinatorial logic circuits in VHDL. This will be done using the inbuilt procedures and functions. The functionality of each of the implemented models will be verified through the use of test benches. Finally a test bench will be used to generate a waveform. This will provide a deeper understanding of how each of the constructs work and the features that are provided with them.

2.2 Procedures

The first procedural model that was created was one that would add two 4 bit values into a sum nybble and a carry bit. To do this a procedure was created that took the two numbers as input and outputted a 5 bit number. Inside this procedure is a single line that adds the numbers together.

2.3 Functions

2.4 Testbench

2.5 Code

2.5.1 1-1 Procedures Add two Values

```
1 ----------------------------------------------------------------------------------
2 -- Module Name: add_two_values_procedure
3 ----------------------------------------------------------------------------------
```

```
4
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7  use IEEE.STD_LOGIC_ARITH.ALL;
8  use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10 library UNISIM;
11 use UNISIM.VComponents.all;
12
13 Entity add_two_values_procedure Is Port (
14     Signal ain : in STD_LOGIC_VECTOR (3 downto 0);
15     Signal bin : in STD_LOGIC_VECTOR (3 downto 0);
16     Signal sum : out STD_LOGIC_VECTOR (3 downto 0);
17     Signal cout : out STD_LOGIC
18 );
19 end add_two_values_procedure;
20
21 Architecture behavior of add_two_values_procedure Is
22
23     Signal total_out : STD_LOGIC_VECTOR (4 downto 0);
24
25     procedure add_two_values (
26         Signal ain_int : in STD_LOGIC_VECTOR (3 downto 0);
27         Signal bin_int : in STD_LOGIC_VECTOR (3 downto 0);
28         Signal total_out_int : out STD_LOGIC_VECTOR (4 downto 0)
29     ) is
30
31     begin
32         total_out_int(4 downto 0) <= ('0' & ain_int) + ('0' & bin_int);
33     end add_two_values;
34
35 begin
36     cout <= total_out(4);
37     sum <= total_out(3 downto 0);
38
39     process (ain, bin)
40     begin
41           add_two_values (ain, bin, total_out);
42     end process;
43 end behavior;
```

code/lab311.vhd

```
1  ----------------------------------------------
2  -- Module Name: add_two_values_procedure_tb
3  ----------------------------------------------
4
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7  use IEEE.STD_LOGIC_ARITH.ALL;
8  use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10 use STD.textio.all;
11 use IEEE.std_logic_textio.all;
12
13 library UNISIM;
14 use UNISIM.VComponents.all;
15
16 Entity add_two_values_procedure_tb Is
17 end add_two_values_procedure_tb;
```

```
18
19 Architecture behavior of add_two_values_procedure_tb Is
20     Component add_two_values_procedure
21     port(
22         Signal ain : in STD_LOGIC_VECTOR (3 downto 0);
23         Signal bin : in STD_LOGIC_VECTOR (3 downto 0);
24         Signal sum : out STD_LOGIC_VECTOR (3 downto 0);
25         Signal cout : out STD_LOGIC
26     );
27     End Component;
28
29     Signal count : STD_LOGIC_VECTOR (2 downto 0) := "000";
30
31     Signal ain : STD_LOGIC_VECTOR (3 downto 0) := "0000";
32     Signal bin : STD_LOGIC_VECTOR (3 downto 0) := "0000";
33     Signal sum : STD_LOGIC_VECTOR (3 downto 0) := "0000";
34     Signal cout : STD_LOGIC := '0';
35
36 begin
37     DUT:  add_two_values_procedure PORT MAP (
38             ain => ain,
39             bin => bin,
40             sum => sum,
41             cout => cout
42           );
43
44     process
45         variable k : integer := 0;
46         --variable s : line;
47
48     begin
49
50         ain <= "0110"; bin <= "1010";
51         --write (s, "ain = ");  write (s, ain); write (s, "bin = "); write
               (s, bin); write (s, "cout = "); write (s, cout); write (s, "sum =
               "); write (s, sum);
52         --writeline (output, s);
53
54         for k in 0 to 4 loop
55
56           wait for 5 ns;
57           count <= count + "1";
58           ain <= ain + count;
59           bin <= bin + count;
60           --write (s, "ain = ");  write (s, ain); write (s, "bin = "); write
                 (s, bin); write (s, "cout = "); write (s, cout); write (s, "sum
                 = "); write (s, sum);
61           --writeline (output, s);
62
63         end loop;
64
65         wait;
66
67         --write (s, "Simulation Ends");
68         --writeline (output, s);
69
70     end process;
71 end behavior;
```

code/lab311_tb.vhd

2.5.2 1-2 Calc Even Parity

```
----------------------------------------------
-- Module Name: calc_even_parity_procedure
----------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

Entity calc_even_parity_procedure  Is Port (
    Signal ain : in STD_LOGIC_VECTOR (7 downto 0);
    Signal parity : out STD_LOGIC
);
end calc_even_parity_procedure ;



Architecture behavior of calc_even_parity_procedure  Is

    procedure calc_even_parity (
        signal input : in STD_LOGIC_VECTOR (7 downto 0);
        signal output : out std_logic
    ) is
    variable par_bit : STD_LOGIC := '0';
    begin
        for k in 0 to input'length-1 loop
            par_bit := par_bit xor ain(k);
        end loop;
        output <= par_bit;
    end calc_even_parity;

begin

    calc_even_parity(ain, parity);

end behavior;
```

code/lab312.vhd

```
----------------------------------------------
-- Module Name: calc_even_parity_procedure_tb
----------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

use STD.textio.all;
use IEEE.std_logic_textio.all;

library UNISIM;
use UNISIM.VComponents.all;

Entity calc_even_parity_procedure_tb Is
```

```
end calc_even_parity_procedure_tb;

Architecture behavior of calc_even_parity_procedure_tb Is
    Component calc_even_parity_procedure
    port(
        Signal ain : in STD_LOGIC_VECTOR (7 downto 0);
        Signal parity : out STD_LOGIC
    );
    End Component;

    Signal count : STD_LOGIC_VECTOR (2 downto 0) := "000";

    Signal ain : STD_LOGIC_VECTOR (7 downto 0);
    Signal parity : STD_LOGIC;

begin
    DUT:  calc_even_parity_procedure PORT MAP (
            ain => ain,
            parity => parity
         );

    process
        variable k : integer := 0;
        --variable s : line;

    begin

        ain <= "10101000";
        --write (s, "ain = ");  write (s, ain); write (s, "parity = ");
            write (s, parity);
        --writeline (output, s);

        for k in 0 to 4 loop

          wait for 5 ns;
          count <= count + "1";
          ain <= ain + count;
          --write (s, "ain = ");  write (s, ain); write (s, "parity = ");
              write (s, parity);
          --writeline (output, s);

        end loop;

        wait;

        --write (s, "Simulation Ends");
        --writeline (output, s);

    end process;
end behavior;
```

code/lab312_tb.vhd

2.5.3 2-1 Functions Add Two Values

```
----------------------------------------------
-- Module Name: add_two_values_function
----------------------------------------------

library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

Entity add_two_values_function  Is Port (
    Signal ain : in STD_LOGIC_VECTOR (3 downto 0);
    Signal bin : in STD_LOGIC_VECTOR (3 downto 0);
    Signal sum : out STD_LOGIC_VECTOR (4 downto 0)
);
end add_two_values_function ;

Architecture behavior of add_two_values_function  Is

    signal ain_int : STD_LOGIC_VECTOR(4 downto 0);
    signal bin_int : STD_LOGIC_VECTOR(4 downto 0);

    function add_two_values  (signal ain, bin : in STD_LOGIC_VECTOR)
    return std_logic_vector is
        variable result : STD_LOGIC_VECTOR(4 downto 0);
    begin
        result := ain + bin;
        return result;
    end add_two_values;

begin
    ain_int <= "0" & ain;
    bin_int <= "0" & bin;

    process (ain_int, bin_int) begin
        sum <= add_two_values(ain_int, bin_int);
    end process;
end behavior;
```

code/lab321.vhd

```
----------------------------------------------------
-- Module Name: add_two_values_function_tb
----------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

use STD.textio.all;
use IEEE.std_logic_textio.all;

library UNISIM;
use UNISIM.VComponents.all;

Entity add_two_values_function_tb Is
end add_two_values_function_tb;

Architecture behavior of add_two_values_function_tb Is
    Component add_two_values_function
    port(
        Signal ain : in STD_LOGIC_VECTOR (3 downto 0);
```

```
        Signal bin : in STD_LOGIC_VECTOR (3 downto 0);
        Signal sum : out STD_LOGIC_VECTOR (4 downto 0)
    );
    End Component;

    Signal count : STD_LOGIC_VECTOR (2 downto 0) := "000";

    Signal ain : STD_LOGIC_VECTOR (3 downto 0);
    Signal bin : STD_LOGIC_VECTOR (3 downto 0);
    Signal sum : STD_LOGIC_VECTOR (4 downto 0);

begin
    DUT:  add_two_values_function PORT MAP (
            ain => ain,
            bin => bin,
            sum => sum
         );

    process
        variable k : integer := 0;
        --variable s : line;

    begin

        ain <= "0110"; bin <= "1010";
        --write (s, "ain = ");  write (s, ain); write (s, "bin = "); write
            (s, bin); write (s, "sum = "); write (s, sum);
        --writeline (output, s);

        for k in 0 to 4 loop

          wait for 5 ns;
          count <= count + "1";
          ain <= ain + count;
          bin <= bin + count;
          --write (s, "ain = ");  write (s, ain); write (s, "bin = "); write
              (s, bin); write (s, "sum = "); write (s, sum);
          --writeline (output, s);

        end loop;

        wait;

        --write (s, "Simulation Ends");
        --writeline (output, s);

    end process;
end behavior;
```

code/lab321_tb.vhd

2.5.4 2-2 Calc Ones

```
----------------------------------------------
-- Module Name: calc_ones_function
----------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

Entity calc_ones_function  Is Port (
    Signal ain : in STD_LOGIC_VECTOR (7 downto 0);
    Signal number_of_ones : out STD_LOGIC_VECTOR (2 downto 0)
);
end calc_ones_function ;

Architecture behavior of calc_ones_function  Is

    function count_ones(
        signal input : std_logic_vector (7 downto 0)
    )
    return std_logic_vector is
       variable count : std_logic_vector (2 downto 0) := "000";
    begin
       for k in 0 to input'length-1 loop
           if (input(k) = '1') then
              count := count + 1;
           end if;
       end loop;
    return count;
    end count_ones;


begin
    number_of_ones <= count_ones(ain);


end behavior;
```

code/lab322.vhd

```
----------------------------------------------------------------------------------
-- Module Name: calc_ones_function_tb
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

use STD.textio.all;
use IEEE.std_logic_textio.all;

library UNISIM;
use UNISIM.VComponents.all;

Entity calc_ones_function_tb Is
end calc_ones_function_tb;

Architecture behavior of calc_ones_function_tb Is
    Component calc_ones_function
    port(
        Signal ain : in STD_LOGIC_VECTOR (7 downto 0);
        Signal number_of_ones : out STD_LOGIC_VECTOR (2 downto 0)
    );
```

```
25      End Component;
26
27      Signal count : STD_LOGIC_VECTOR (2 downto 0) := "000";
28
29      Signal ain : STD_LOGIC_VECTOR (7 downto 0);
30      Signal number_of_ones : STD_LOGIC_VECTOR (2 downto 0);
31
32  begin
33      DUT:  calc_ones_function PORT MAP (
34              ain => ain,
35              number_of_ones => number_of_ones
36            );
37
38      process
39          variable k : integer := 0;
40          --variable s : line;
41
42      begin
43
44          ain <= "01001010";
45          wait for 5 ns;
46          --write (s, "ain = ");  write (s, ain); write (s, "number of ones =
                "); write (s, number_of_ones);
47          --writeline (output, s);
48
49          for k in 0 to 4 loop
50
51            wait for 5 ns;
52            count <= count + "1";
53            ain <= ain + count;
54            --write (s, "ain = ");  write (s, ain); write (s, "number of ones
                 = "); write (s, number_of_ones);
55            --writeline (output, s);
56
57          end loop;
58
59          wait;
60
61          --write (s, "Simulation Ends");
62          --writeline (output, s);
63
64      end process;
65  end behavior;
```

code/lab322_tb.vhd

2.5.5 3-1 RCA Test Bench

```
1   ----------------------------------------------------
2   -- Module Name: rca_dataflow_tb
3   ----------------------------------------------------
4
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  use STD.textio.all;
11  use IEEE.std_logic_textio.all;
12
```

```
13 library UNISIM;
14 use UNISIM.VComponents.all;
15
16 Entity rca_dataflow_tb Is
17 end rca_dataflow_tb;
18
19 Architecture behavior of rca_dataflow_tb Is
20     Component rca_dataflow
21     port(
22         a    : in STD_LOGIC_VECTOR (3 downto 0);
23         b    : in STD_LOGIC_VECTOR (3 downto 0);
24         cin  : in STD_LOGIC;
25         s    : out STD_LOGIC_VECTOR (3 downto 0);
26         cout : out STD_LOGIC
27     );
28     End Component;
29
30     Signal count : STD_LOGIC_VECTOR (2 downto 0) := "000";
31     Signal cout_expected: STD_LOGIC;
32
33     Signal a : STD_LOGIC_VECTOR (3 downto 0);
34     Signal b : STD_LOGIC_VECTOR (3 downto 0);
35     Signal c : STD_LOGIC;
36     Signal test_failed : STD_LOGIC;
37     Signal cout_compare1 : STD_LOGIC;
38     Signal s_compare1 : STD_LOGIC;
39     Signal s_compare2 : STD_LOGIC;
40     Signal s_int : STD_LOGIC_VECTOR (3 downto 0);
41     Signal cout : STD_LOGIC;
42
43     procedure add_two_values (
44         a_in : in std_logic_vector(3 downto 0);
45         b_in : in std_logic_vector(3 downto 0);
46         c_in : in std_logic;
47
48         sum : out std_logic_vector(4 downto 0)
49     ) is
50
51     begin
52         sum := ("0" & a_in) + ("0" & b_in) + ("0" & c_in);
53     end add_two_values;
54
55 begin
56     DUT:  rca_dataflow PORT MAP (
57             a => a,
58             b => b,
59             cin => c,
60             s => s_int,
61             cout => cout
62          );
63
64     process
65         variable k : integer := 0;
66         variable s : line;
67         variable sum_out : STD_LOGIC_VECTOR (4 downto 0);
68
69     begin
70
71         a <= "0100"; b <= "1010"; c <= '0'; count <= "000"; test_failed <=
```

```
            '0';
        wait for 10 ns;

        add_two_values(a, b, c, sum_out);
        cout_expected <= sum_out(4);

        wait for 10 ns;
        write (s, string'("a = "));
        write (s, a);
        write (s,  string'("b = "));
        write (s, b);
        write (s, string'(" cin = "));
        write (s, c);
        write (s, string'(" sum expected = "));
        write (s, sum_out);
        write (s, string'(" cout expected = "));
        write (s, cout_expected);
        write (s, string'(" actual sum = "));
        write (s, s_int);
        write (s, string'(" actual cout = "));
        write (s, cout);
        writeline (output, s);

        cout_compare1 <= cout_expected XOR cout;
        s_compare2 <= (sum_out(3) XOR s_int(3)) OR (sum_out(2) XOR s_int(2))
           OR (sum_out(1) XOR s_int(1)) OR (sum_out(0) XOR s_int(0)) ;

        if (cout_compare1 = '1') or (s_compare2 = '1') then
            test_failed <= '1';
        end if;

        for k in 1 to 4 loop
          count <= count + "1";

          wait for 10 ns;
          a <= a + count; b <=  b - count;
          add_two_values(a, b, c, sum_out);
          cout_expected <= sum_out(4);

          wait for 10 ns;
            write (s, string'("a = "));
            write (s, a);
            write (s,  string'("b = "));
            write (s, b);
            write (s, string'(" cin = "));
            write (s, c);
            write (s, string'(" sum expected = "));
            write (s, sum_out);
            write (s, string'(" cout expected = "));
            write (s, cout_expected);
            write (s, string'(" actual sum = "));
            write (s, s_int);
            write (s, string'(" actual cout = "));
            write (s, cout);
            writeline (output, s);

          cout_compare1 <= cout_expected XOR cout;
          s_compare1 <= (sum_out(3) XOR s_int(3)) OR (sum_out(2) XOR
             s_int(2)) OR (sum_out(1) XOR s_int(1)) OR (sum_out(0) XOR
```

```
            s_int(0));
128
129           if (cout_compare1 = '1') or (s_compare2 = '1') then
130             test_failed <= '1';
131           end if;
132         end loop;
133
134
135         a <= "1000"; b <= "0011"; c <= '1'; count <= "000";
136         wait for 10 ns;
137
138         add_two_values(a, b, c, sum_out);
139         cout_expected <= sum_out(4);
140
141         wait for 10 ns;
142         write (s, string'("a = "));
143         write (s, a);
144         write (s,  string'("b = "));
145         write (s, b);
146         write (s, string'(" cin = "));
147         write (s, c);
148         write (s, string'(" sum expected = "));
149         write (s, sum_out);
150         write (s, string'(" cout expected = "));
151         write (s, cout_expected);
152         write (s, string'(" actual sum = "));
153         write (s, s_int);
154         write (s, string'(" actual cout = "));
155         write (s, cout);
156         writeline (output, s);
157
158         cout_compare1 <= cout_expected XOR cout;
159         s_compare2 <= (sum_out(3) XOR s_int(3)) OR (sum_out(2) XOR s_int(2))
               OR (sum_out(1) XOR s_int(1)) OR (sum_out(0) XOR s_int(0)) ;
160
161         if (cout_compare1 = '1') or (s_compare2 = '1') then
162             test_failed <= '1';
163         end if;
164
165         for k in 2 to 4 loop
166           count <= count + '1';
167
168           wait for 10 ns;
169           a <= a - count; b <=  b + count;
170           add_two_values(a, b, c, sum_out);
171           cout_expected <= sum_out(4);
172
173           wait for 10 ns;
174             write (s, string'("a = "));
175             write (s, a);
176             write (s,  string'("b = "));
177             write (s, b);
178             write (s, string'(" cin = "));
179             write (s, c);
180             write (s, string'(" sum expected = "));
181             write (s, sum_out);
182             write (s, string'(" cout expected = "));
183             write (s, cout_expected);
184             write (s, string'(" actual sum = "));
```

```
185             write (s, s_int);
186             write (s, string'(" actual cout = "));
187             write (s, cout);
188             writeline (output, s);
189 
190           cout_compare1 <= cout_expected XOR cout;
191           s_compare1 <= (sum_out(3) XOR s_int(3)) OR (sum_out(2) XOR
                  s_int(2)) OR (sum_out(1) XOR s_int(1)) OR (sum_out(0) XOR
                  s_int(0));
192 
193           if (cout_compare1 = '1') or (s_compare1 = '1') then
194             test_failed <= '1';
195           end if;
196         end loop;
197 
198         wait for 10 ns;
199 
200         if (test_failed = '1') then
201             write (s, string'("Simulation DONE"));
202             write (s, string'(" Test Failed"));
203             writeline (output, s);
204         else
205             write (s, string'("Simulation DONE"));
206             write (s, string'(" Test Passed"));
207             writeline (output, s);
208         end if;
209 
210         wait;
211     end process;
212 end behavior;
```

code/lab331_tb.vhd

2.5.6 3-2 Waveform Generator Testbench

```
1  ----------------------------------------------------------------------------------
2  -- Module Name: waveform_generation_tb
3  ----------------------------------------------------------------------------------
4 
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7 
8  library UNISIM;
9  use UNISIM.VComponents.all;
10 
11 Entity waveform_generation_tb Is
12 end waveform_generation_tb;
13 
14 Architecture behavior of waveform_generation_tb Is
15     Signal a : STD_LOGIC := '0';
16     Signal g1 : STD_LOGIC := '0';
17     Signal g2 : STD_LOGIC := '1';
18 
19 begin
20 
21     process
22     begin
23         wait for 40ns;
24         a <= '1';
25         wait for 20ns;
```

```
26          g1 <= '1';
27          wait for 20ns;
28          g2 <= '0';
29          wait for 20ns;
30          a <= '0';
31          wait for 20ns;
32          g1 <= '0';
33          wait for 20ns;
34          g2 <= '1';
35      end process;
36
37
38  end behavior;
```

code/lab332_tb.vhd

3 Questions