Dejan Rasuo
Hassan Allabad
Gabriel Van Dreel
October 16, 2019
ECGR 2181-002

Project 1 – Fall 2019

Part 1:

The first part of the project involved developing and simulating several Boolean functions using Very High-Speed Integrated Circuit (VHSIC) Hardware Description Language (HDL), commonly described with the acronym, VHDL. These Boolean functions represented combinational logic input and output for simple circuit components that would need to be developed as entities and simulated using testbenches in the VHDL programming language. The Vivado 2018.3 Webpack Integrated Development Environment (IDE) was required for synthesizing and simulating the developed VHDL files. The VHDL project requirements stated that the XC7A35TCPG236-1 Field-Programmable Gate Array (FPGA) was to be used as the given Basys3 constraints file would need to use the FPGA that is in the Artix-7 family, has the CPG236 package, operates at speed -1, and contains 41600 flip flops. Since this project did not involve an actual circuit implementation, however, the Basys3 constraint file was not used.

This part of the project was organized into two general tasks and, because of the tasks' similar use of combinational logic, grouped into one Vivado project titled "computer_assignment_1." The first of which was to develop and simulate a set of two Boolean functions that each took inputs A, B, and C_{in} and individually yielded outputs Sum and C_{out}. The circuit component that is formed from this combinational logic is called a one-bit full adder, which can be used to perform binary addition. The Boolean equations that represent these functions state,

$$Sum = A \oplus B \oplus C_{in} \tag{1}$$

$$C_{out} = AB + AC_{in} + BC_{in} \tag{2}$$

In implementing Boolean equations (1) and (2), the design process was used and involved first establishing the given problem with its constraints and deliverables. The brainstorming step of the design process was then used to write sections of the VHDL code that were already understood. Since, many aspects of basic VHDL were still unknown, Dr. Mihail Cutitaru's "ECGR 2181 - Vivado Walkthrough" was used as a reference for using the Vivado IDE to produce VHDL code. There was frequent interchange between brainstorming and development in the project for creating the circuit entity files and their respective test bench files. The first aspects of development involved simply writing the Boolean functions using VHDL and defining the inputs and outputs as well as their object type (std_logic) for the one-bit full adder entity. The one-bit full adder was successfully developed as the file task1.vhd shown in Figure 1 with its testbench file task1_tb.vhd shown in Figure 2. The simulation for the first task was also successful in that it produced the anticipated output as given in Figure 3.

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 01:11:42 PM
-- Design Name: One Bit Adder
-- Module Name: task1 - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task defines the functionality of a one bit adder that may be used to form a
     multibit adder
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task1 is
    port(   a: in std_logic;
            b: in std_logic;
            cin: in std_logic;
            sum: out std_logic;
            cout: out std_logic);
end task1;
```

```
architecture Behavioral of task1 is

begin

    sum <= a xor b xor cin;
    cout <= (a and b) or (a and cin) or (b and cin);

end Behavioral;
```

Figure 1: Task 1 (task1.vhd) Code

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 01:19:46 PM
-- Design Name: One Bit Adder Test Bench
-- Module Name: task1_tb - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task defines the simulation of a one bit adder that may be used to form a
      multibit adder
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL; --Uncommented for supporting unsigned values
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task1_tb is
--  Port ( );
end task1_tb;

architecture Behavioral of task1_tb is

component task1
port(   a, b, cin: in std_logic;
        sum, cout: out std_logic);
end component;

signal a, b, cin, sum, cout: std_logic;

signal counter: unsigned(2 downto 0) := "000";

signal sw: std_logic_vector(2 downto 0);

begin

uut: task1 port map(a => a, b => b, cin => cin, sum => sum, cout => cout);

sw <= std_logic_vector(counter);

a <= sw(2);
b <= sw(1);
cin <= sw(0);

tb: process
begin
    wait for 20 ns;
    counter <= counter + 1;
end process tb;

end Behavioral;
```

Figure 2: Task 1 Test Bench (task1_tb.vhd) Code

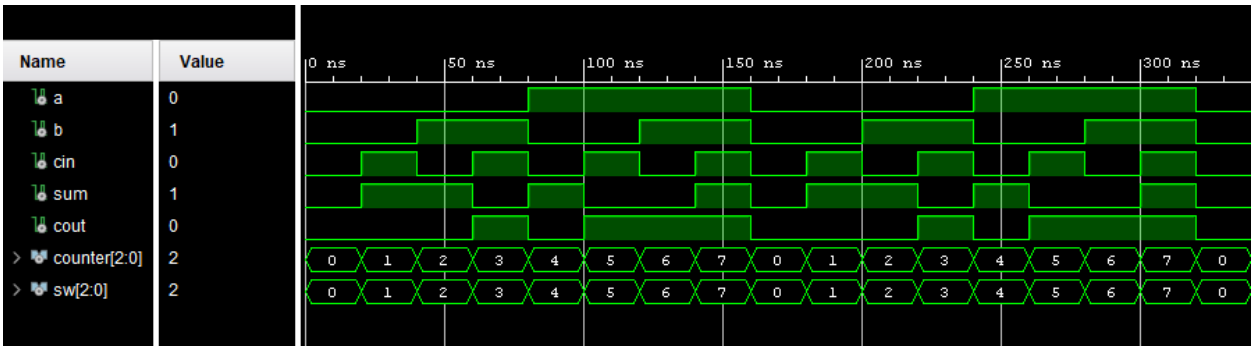


Figure 3: Task 1 Simulation Results

The second task for the first part of the project involved implementing a Boolean function of inputs A, B, C, and D that yields output F as given in the equation,

$$F(A,B,C,D) = \overline{\overline{(A\,\overline{B}\,\overline{D}) \oplus (\overline{A}B + \overline{B}\overline{C})} + \overline{A}CD(B\overline{C} + A\overline{B}C\overline{D} + \overline{A}C\overline{D})} \qquad (3)$$

Three equivalent forms of the function were used, and, in the VHDL code, these forms were denoted with Boolean outputs F, G, and H. F in the VHDL implementation was developed and typed as the lexicographically-equivalent form of equation (3). G was developed and typed in VHDL as the algebraically-simplified form of equation (3). Lastly, H was developed and typed in VHDL as the canonical Sum-of-Products (SOP) form of equation (3). These alternate forms of equation (3) were obtained with the hand-written work shown in Figure 4.

The design process for the second task was identical to the first since many of the same demands were made to implement some form of combinational logic. No issues occurred during the development of the task2.vhd entity file. During the development of the task2_tb.vhd test bench file, however, difficulty was encountered when the port map for the test bench was not properly defined for the VHDL where the Device Under Test (dut) keyword was used instead of the Unit Under Test (uut) keyword, which represented an incorrect object template in the Institute for Electrical and Electronics Engineers (IEEE) VHDL library for the test bench. Once the project was updated with this information, the project state could not be undone and therefore, the task 2 test bench would not run. To resolve this error, the entire computer_assignment_1 project was redeveloped and resynthesized.

Upon the correction of this significant error, the task 2 test bench was completed as normal using a test bench process and a counter that sequentially reset all the input signals for the combinational logic of task 2. The combinational logic for task 2 was successfully developed as the file task2.vhd shown in Figure 5 with its testbench file task2.vhd shown in Figure 6. The simulation for the second task was also successful in that it produced the anticipated output as given in Figure 7.

$$F = \overline{A}\overline{B}\overline{D} \oplus (\overline{A}B + \overline{B}\overline{C}) + \overline{A}CD(B\overline{C} + AB\overline{C}\overline{D} + \overline{A}C\overline{D})$$

$$= A\overline{B}\overline{D} \oplus (\overline{A}B + \overline{B} + \overline{C}) + \overline{A}CD(B\overline{C} + A\overline{B}C\overline{D} + \overline{A}C\overline{D})$$

$$= A\overline{B}\overline{D} \oplus (\overline{A} + \overline{B} + \overline{C}) + \cancel{AB\overline{C}CD}^{\,0} + \cancel{A\overline{A}BCDD}^{\,0} + \cancel{\overline{A}C\overline{D}D}^{\,0}$$

$$= A\overline{B}\overline{D} \oplus (\overline{A} + \overline{B} + \overline{C})$$

$$= \cancel{(A\overline{B}\overline{D})(ABC)}^{\,0} + (\overline{A} + B + D)(\overline{A} + \overline{B} + \overline{C})$$

$$= \overline{A} + \cancel{\overline{A}B}^{\,0} + \cancel{\overline{A}C}^{\,0} + \cancel{\overline{A}B}^{\,0} + \cancel{B\overline{B}}^{\,0} + B\overline{C} + \cancel{\overline{A}D}^{\,0} + \overline{B}D + \overline{C}D$$

$$= \overline{A} + B\overline{C} + \overline{B}D + \cancel{\overline{C}D}^{\,0}$$

$$\boxed{F = \overline{A} + B\overline{C} + \overline{B}D}$$

$$= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + B\overline{C}(A+\overline{A})(D+\overline{D}) + \overline{B}D(A+\overline{A})(C+\overline{C})$$

$$= \overline{A}\overline{B}_0\overline{C}\overline{D} + \overline{A}\overline{B}_1\overline{C}D + \overline{A}\overline{B}_2C\overline{D} + \overline{A}\overline{B}_3CD + \overline{A}\overline{B}_4\overline{C}\overline{D} + \overline{A}\overline{B}_5\overline{C}D + \overline{A}\overline{B}_6C\overline{D} + \overline{A}\overline{B}_7CD + \cancel{AB\overline{C}\overline{D}}^{\,0} + \overline{A}B_{12}\overline{C}\overline{D} + \cancel{AB\overline{C}D}^{\,0} + \overline{A}B_{13}\overline{C}D + \cancel{A\overline{B}C\overline{D}}^{\,0} + \overline{A}B_9\overline{C}D + \cancel{AB\overline{C}D}^{\,0} + \overline{A}B_{11}\overline{C}D$$

$$\boxed{F = \sum m(0, 1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 13)}$$

$$= \prod M(8, 10, 14, 15)$$

Figure 4: Task 2 Boolean Simplifications

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 02:15:15 PM
-- Design Name: Combinational Logic
-- Module Name: task2 - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task acts as a combinational logic system that takes four binary inputs
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task2 is
    port(   a: in std_logic;
            b: in std_logic;
            c: in std_logic;
            d: in std_logic;
            f: out std_logic;
            g: out std_logic;
            h: out std_logic);
```

```
end task2;

architecture Behavioral of task2 is

begin

    f <= not(not((a and not(b) and not(d)) xor ((not(a) and b) or not(b and c))) or (not(a) and c and
      d and ((b and not(c)) or (a and not(b) and c and not(d)) or (not(a) and c and not(d)))));
    g <= not(a) or (b and not(c)) or (not(b) and d);
    h <= (not(a) and not(b) and not(c) and not(d)) or (not(a) and not(b) and not(c) and d) or (not(a)
      and not(b) and c and not(d)) or (not(a) and not(b) and c and d) or (not(a) and b and not(c) and
      not(d)) or (not(a) and b and not(c) and d) or (not(a) and b and c and not(d)) or (not(a) and b
      and c and d) or (a and b and not(c) and not(d)) or (a and b and not(c) and d) or (a and not(b)
      and not(c) and d) or (a and not(b) and c and d);

end Behavioral;
```

Figure 5: Task 2 (task2.vhd) Code

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 02:59:15 PM
-- Design Name: Combinational Logic Test Bench
-- Module Name: task2_tb - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task defines the simulation of a combinational logic system that takes four
     binary inputs
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task2_tb is
--  Port ( );
end task2_tb;

architecture Behavioral of task2_tb is

component task2
port(   a, b, c, d: in std_logic;
        f, g, h: out std_logic);
end component;

signal a, b, c, d, f, g, h: std_logic;

signal counter: unsigned(3 downto 0) := "0000";

signal sw: std_logic_vector(3 downto 0);

begin

uut: task2 port map(a => a, b => b, c => c, d => d, f => f, g => g, h => h);

sw <= std_logic_vector(counter);

a <= sw(3);
b <= sw(2);
c <= sw(1);
d <= sw(0);

tb: process
begin
```

```
    wait for 20 ns;
    counter <= counter + 1;
end process tb;

end Behavioral;
```

Figure 6: Task 2 Test Bench (task2_tb.vhd) Code

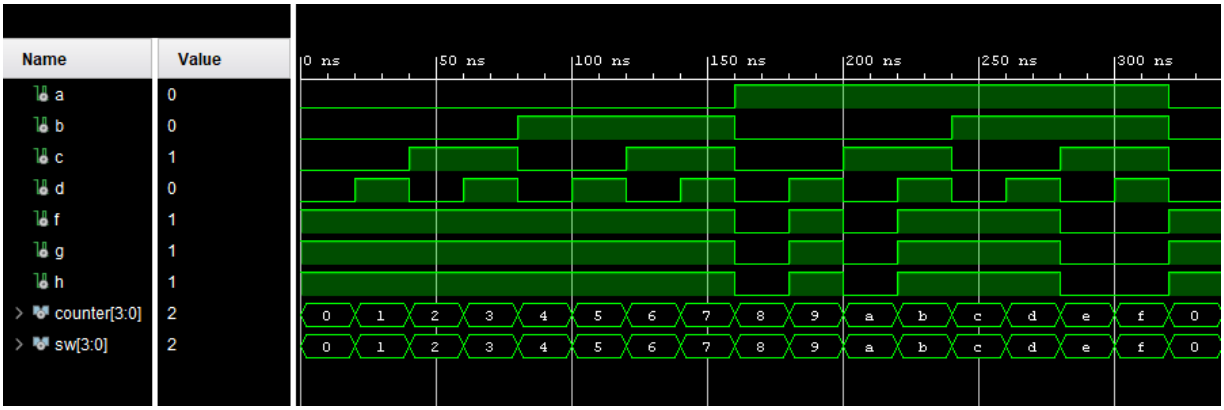


Figure 7: Task 2 Simulation Results

Part 2:

For the second part of the project, one task was completed under the same constraints and as that of the first part of the project where a seven-segment display needed to be modeled. The seven-segment display needed to take a four-bit binary value as an input and output (display) the hexadecimal representation of the inputted binary value. The segments of the display that would need to be logic highs were given for the project in Figures 8 and 9. Since the output of the seven-segment display may be interpreted as a seven-bit signal, the truth table shown as Table 10 was derived for showing all seven of the possible seven-segment display outputs given any four-bit binary value. Because of the new technical nature of this part of the project, part 2 was organized in a separate Vivado project titled, “computer_assignment_2.”

When the seven-segment display combinational logic was developed in Vivado using VHDL, a similar brainstorming and development process was followed to that of part 1. The development process differed, however, since normal Boolean output value assignments were found to be impractical and too time-consuming. A new type of conditional statement was therefore used with this part of the project that is known as a when-else statement from the VHDL Application Programming Interface (API). This was found during the VHDL research process with Cutitaru’s prime number detector from his document. It was also found that, given the multibit inputs and outputs for the seven-segment display, the std_logic_vector port and signal type was practical and used more frequently than in part 1. The seven-segment display combinational logic was successfully developed as the file task3.vhd shown in Figure 11 with its testbench file task3_tb.vhd shown in Figure 12. The simulation for the first task was also successful in that it produced the anticipated output for each of the display’s segments as given in Figure 13.

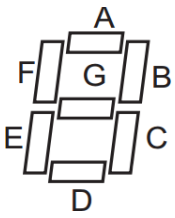
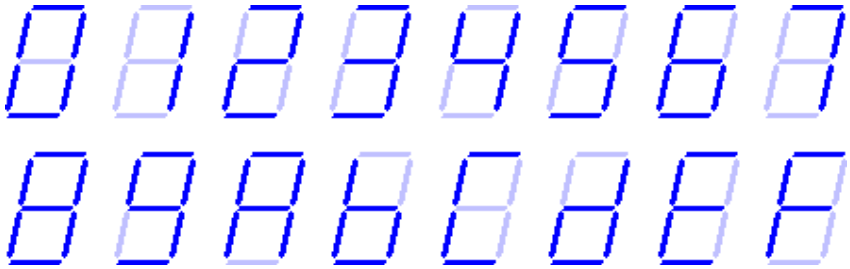


Figure 8: Seven-Segment Display



Layout

Figure 9: Hex Encoding

in(3)	in(2)	in(1)	in(0)	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

Table 10: Truth Table of the Encoder

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 04:12:54 PM
-- Design Name: Encoder
-- Module Name: task3 - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task acts as an encoder for a seven segment display that takes a four bit
     unsigned binary value
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task3 is
    port(   hex_in: in std_logic_vector(3 downto 0);
            a: out std_logic;
            b: out std_logic;
            c: out std_logic;
            d: out std_logic;
            e: out std_logic;
```

```
        f: out std_logic;
        g: out std_logic);
end task3;

architecture Behavioral of task3 is

signal output: std_logic_vector(6 downto 0);

begin

process(hex_in)
begin
   case hex_in is
      when "0000" => output <= "1111110";
      when "0001" => output <= "0110000";
      when "0010" => output <= "1101101";
      when "0011" => output <= "1111001";
      when "0100" => output <= "0110011";
      when "0101" => output <= "1011011";
      when "0110" => output <= "1011111";
      when "0111" => output <= "1110000";
      when "1000" => output <= "1111111";
      when "1001" => output <= "1111011";
      when "1010" => output <= "1110111";
      when "1011" => output <= "0011111";
      when "1100" => output <= "1001110";
      when "1101" => output <= "0111101";
      when "1110" => output <= "1001111";
      when "1111" => output <= "1000111";
      when others => output <= "0000000";
   end case;
end process;

a <= output(6);
b <= output(5);
c <= output(4);
d <= output(3);
e <= output(2);
f <= output(1);
g <= output(0);
```

```
end Behavioral;
```

Figure 11: Task 3 (task3.vhd) Code

```
----------------------------------------------------------------------------------
-- Company: UNC Charlotte
-- Engineer: Dejan Rasuo, Gabriel Van Dreel, & Hassan Allabad
--
-- Create Date: 08/25/2019 05:15:16 PM
-- Design Name: Encoder Test Bench
-- Module Name: task3_tb - Behavioral
-- Project Name: Project 1
-- Target Devices:
-- Tool Versions:
-- Description: This task defines the simulation of an encoder for a seven segment display that
     takes a four bit unsigned binary value
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity task3_tb is
--  Port ( );
end task3_tb;
```

```
architecture Behavioral of task3_tb is

component task3
port(   hex_in: in std_logic_vector(3 downto 0);
        a, b, c, d, e, f, g: out std_logic);
end component;

signal hex_in: std_logic_vector(3 downto 0);
signal a, b, c, d, e, f, g: std_logic;

signal counter: unsigned(3 downto 0) := "0000";

signal sw: std_logic_vector(3 downto 0);

begin

uut: task3 port map(hex_in => hex_in, a => a, b => b, c => c, d => d, e => e, f => f, g => g);

sw <= std_logic_vector(counter);

hex_in <= sw;

tb: process
begin
    wait for 20 ns;
    counter <= counter + 1;
end process tb;

end Behavioral;
```

Figure 12: Task 3 Test Bench (task3_tb.vhd) Code

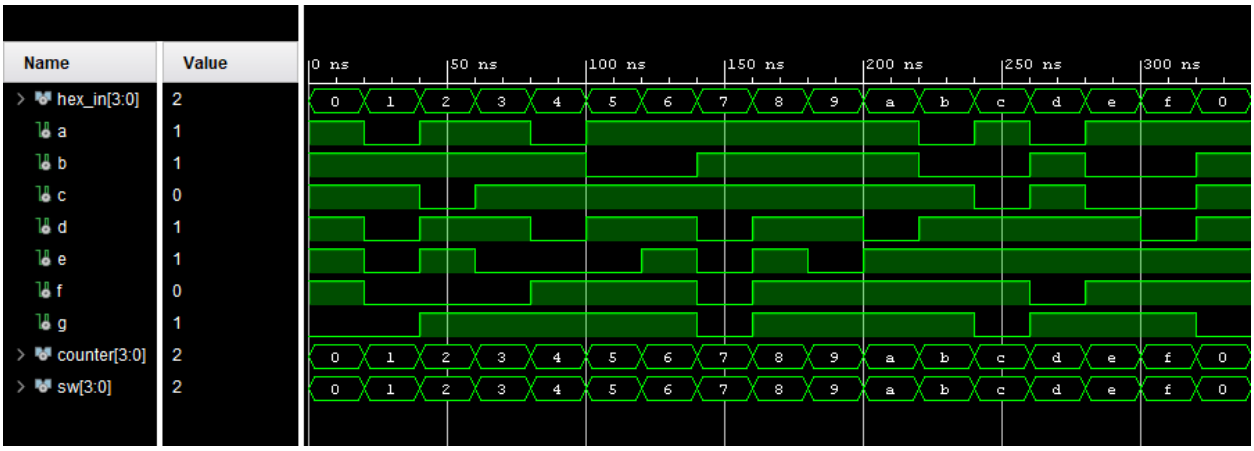


Figure 13: Task 3 Simulation Results

References

1) Cutitaru, M. (2019, August). Vivado Walkthrough. Retrieved from https://uncc.instructure.com/courses/111351/files/folder/Vivado?preview=6453261