Luis Umana
801018765

ECGR 4146: Introduction to VHDL
Lab 5: 8-bit Ripple Adder

Note: I wasn't sure if everything in our code needed to be a structural model, or just the 8-bit ripple adder, but just in case the half adder and full adder were both made in structural modeling.

You may notice that I named the project folder "half_adder", initially I was gonna create the half adder, full adder, and ripple adder into separate folders like I did for the different model styles. However, I found it much easier to keep everything in one folder once I learned about the top module to specify what design and simulation source to run synthesis.

VHDL Code for Half Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity half_adder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end half_adder;
architecture Behavioral of half_adder is
  component XOR1 is
    port (A, B: in std_logic;
      SUM: out std_logic);
  end component;
  component AND1 is
    port (A, B: in std_logic;
      CARRY: out std_logic);
  end component;
begin
 u1: XOR1 port map (A => a, B => b, SUM => sum);
 u2: AND1 port map (A => a, B => b, CARRY => carry);
end Behavioral;
```

VHDL Code for XOR1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR1 is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           SUM : out STD_LOGIC);
end XOR1;
architecture Behavioral of XOR1 is
begin
SUM <= A XOR B;
end Behavioral;
```

Luis Umana
801018765

VHDL Code for AND1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND1 is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           CARRY : out STD_LOGIC);
end AND1;
architecture Behavioral of AND1 is
begin
CARRY <= A AND B;
end Behavioral;
```

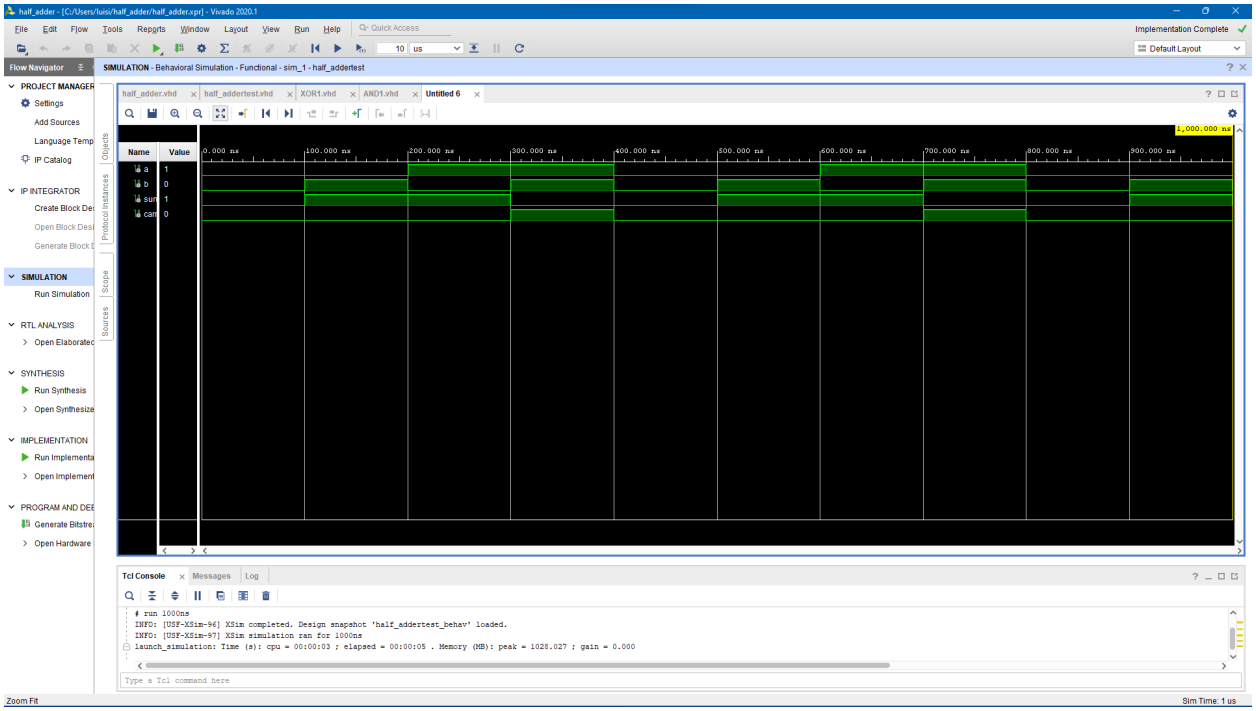
VHDL Testbench Code for Half Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity half_addertest is
--(Port)
end half_addertest;

architecture Behavioral of half_addertest is
component half_adder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end component;
signal a :std_logic :='0';
signal b :std_logic :='0';
signal sum :std_logic ;
signal carry :std_logic ;

begin
uut:half_adder port map
    (a=>a,b=>b,sum=>sum,carry=>carry);
stim_proc: process
    begin
        a <= '0';
        b <= '0';
        wait for 100ns;
        a <= '0';
        b <= '1';
         wait for 100ns;
        a <= '1';
        b <= '0';
         wait for 100ns;
        a <= '1';
```

Luis Umana
801018765

```
        b <= '1';
         wait for 100ns;
   end process;
end Behavioral;
```

Simulation Waveform of Half Adder



VHDL Code for Full Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_adder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           Cin : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end full_adder;
architecture Behavioral of full_adder is
component half_adder is
port(a,b : in std_logic;
sum,carry : out std_logic);
end component;
component OR1 is
port(X,Y : in std_logic;
Z : out std_logic);
end component ;
signal S0,S1,S2:std_logic;
```

```
begin
U1:half_adder port map(a=>a,b=>b,sum=>S0,carry=>S1);
U2:half_adder port map(a=>S0,b=>Cin,sum=>sum,carry=>S2);
U3:OR1 port map(X=>S2,Y=>S1,Z=>carry);
end Behavioral;
```

VHDL Code for OR1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR1 is
    Port ( X : in STD_LOGIC;
           Y : in STD_LOGIC;
           Z : out STD_LOGIC);
end OR1;
architecture Behavioral of OR1 is
begin
Z <= X OR Y;
end Behavioral;
```

VHDL Testbench Code for Full Adder;

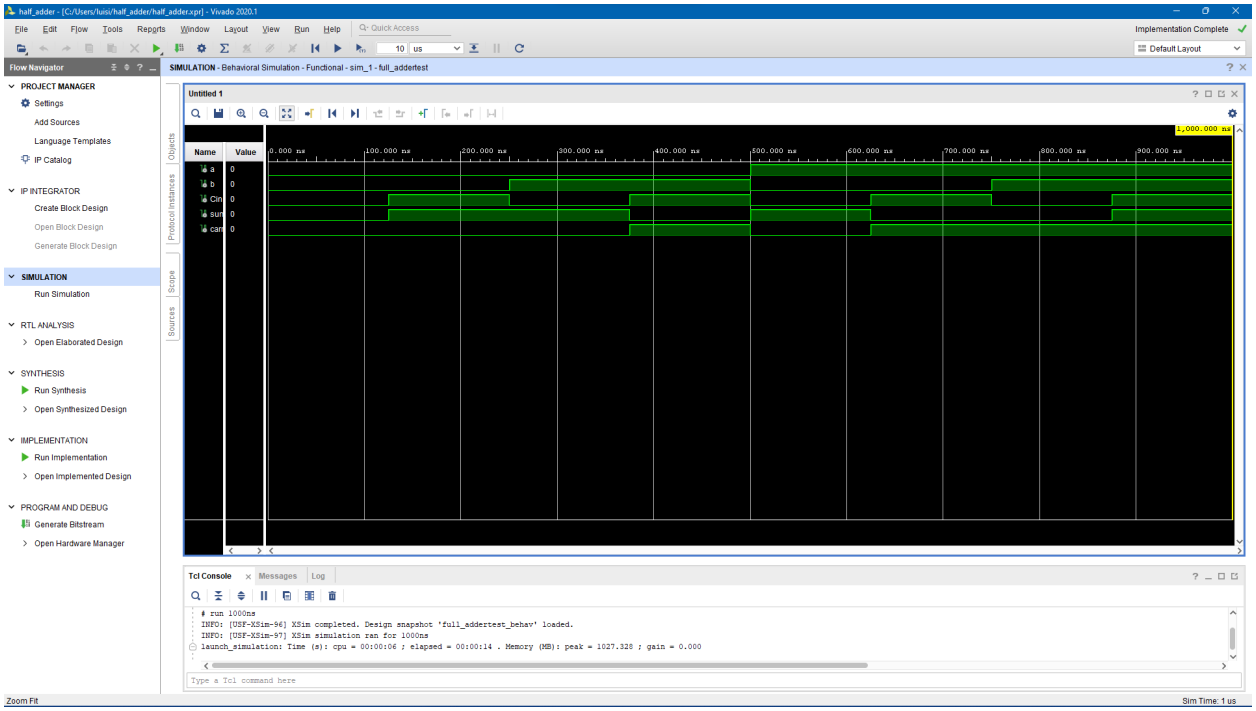
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_addertest is
--Port();
end full_addertest;
architecture Behavioral of full_addertest is
component full_adder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           Cin : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end component;
signal a :std_logic :='0';
signal b :std_logic :='0';
signal Cin:std_logic :='0';
signal sum :std_logic;
signal carry :std_logic ;
begin
uut:full_adder port map
    (a=>a,b=>b,Cin=>Cin,sum=>sum,carry=>carry);
stim_proc: process
begin
a <= '0';
b <= '0';
Cin <= '0';
wait for 125 ns;

a <= '0';
b <= '0';
Cin <= '1';
```

Luis Umana
801018765

```
wait for 125 ns;

a <= '0';
b <= '1';
Cin <= '0';
wait for 125 ns;

a <= '0';
b <= '1';
Cin <= '1';
wait for 125 ns;

a <= '1';
b <= '0';
Cin <= '0';
wait for 125 ns;

a <= '1';
b <= '0';
Cin <= '1';
wait for 125 ns;

a <= '1';
b <= '1';
Cin <= '0';
wait for 125 ns;

a <= '1';
b <= '1';
Cin <= '1';
wait for 125 ns;
end process;
end Behavioral;
```

Luis Umana
801018765

Simulation Waveform of Full Adder



VHDL Code for 8-bit Ripple Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity R_adder is
    Port ( a : in STD_LOGIC_VECTOR(7 downto 0);
           b : in STD_LOGIC_VECTOR(7 downto 0);
           Cin : in STD_LOGIC;
           sum : out STD_LOGIC_VECTOR(7 downto 0);
           carry : out STD_LOGIC);
end R_adder;
architecture Behavioral of R_adder is
component half_adder is
port(a,b : in std_logic;
sum,carry : out std_logic);
end component;
component full_adder is
port(a,b,Cin : in std_logic;
sum,carry : out std_logic);
end component;
signal output :std_logic_vector(6 downto 0);
begin
U1:half_adder port map(a=>a(0),b=>b(0),sum=>sum(0),carry=>output(0));
U2:full_adder port map(a=>a(1),b=>b(1),Cin=>output(0),sum=>sum(1),carry=>output(1));
U3:full_adder port map(a=>a(2),b=>b(2),Cin=>output(1),sum=>sum(2),carry=>output(2));
U4:full_adder port map(a=>a(3),b=>b(3),Cin=>output(2),sum=>sum(3),carry=>output(3));
```

```
U5:full_adder port map(a=>a(4),b=>b(4),Cin=>output(3),sum=>sum(4),carry=>output(4));
U6:full_adder port map(a=>a(5),b=>b(5),Cin=>output(4),sum=>sum(5),carry=>output(5));
U7:full_adder port map(a=>a(6),b=>b(6),Cin=>output(5),sum=>sum(6),carry=>output(6));
U8:full_adder port map(a=>a(7),b=>b(7),Cin=>output(6),sum=>sum(7),carry=>carry);
end Behavioral;
```

VHDL Testbench Code for 8-bit Ripple Adder

Note: I used a random number generator for every 8-bit value of a and b.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity R_addertest is
--Port();
end R_addertest;
architecture Behavioral of R_addertest is
component R_adder is
    Port ( a : in STD_LOGIC_VECTOR(7 downto 0);
           b : in STD_LOGIC_VECTOR(7 downto 0);
           Cin : in STD_LOGIC;
           sum : out STD_LOGIC_VECTOR(7 downto 0);
           carry : out STD_LOGIC);
end component;

signal a : std_logic_vector(7 downto 0);
signal b : std_logic_vector(7 downto 0);
signal Cin: std_logic;
signal sum : std_logic_vector(7 downto 0);
signal carry : std_logic ;
begin
uut:R_adder port map
    (a=>a,b=>b,Cin=>Cin,sum=>sum,carry=>carry);
stim_proc: process
begin
  a <= "01001010";
  b <= "11110100";
  Cin <= '0';
  wait for 100 ns;

  a <= "10010001";
  b <= "01110001";
  Cin <= '1';
  wait for 100 ns;

  a <= "11111000";
  b <= "11011000";
  Cin <= '0';
  wait for 100 ns;

  a <= "11000101";
  b <= "00100101";
  Cin <= '1';
  wait for 100 ns;

  a <= "11110001";
  b <= "10111000";
```

Luis Umana
801018765

```
  Cin <= '0';
  wait for 100 ns;

  a <= "01111100";
  b <= "10110001";
  Cin <= '1';
  wait for 100 ns;

  a <= "11010011";
  b <= "10110000";
  Cin <= '0';
  wait for 100 ns;

  a <= "10110001";
  b <= "01010010";
  Cin <= '1';
  wait for 100 ns;

  a <= "01010000";
  b <= "00000110";
  Cin <= '0';
  wait for 100 ns;

  a <= "10111101";
  b <= "11111111";
  Cin <= '1';
  wait for 100 ns;
end process;
end Behavioral;
```

Simulation Waveform of 8-bit Ripple Adder