****

**EE 4550L**

**IC Hardware Security and Trust LAB**

**SPRING 2024**

**TA: Kanchan Vissamsetty**

**Lab section: 01**

**Name: Alex Yeoh**

**“I have neither given nor received aid on this assignment, nor have I observed any violation of the Honor code”**

**Signature: Alex Yeoh Date: 23rd January 2024**

**Report due date: 25th January 2024**

1. **OBJECTIVE**

To learn how to use Vivado to write and simulate VHDL code.

1. **PROCEDURE**

Design a 1-bit adder, make the necessary basic gates in VHDL, make the 1-bit adder in VHDL, make a 4-bit adder in VHDL, make a 4-bit adder subtractor in VHDL, and simulate them all with testbenches in VHDL.

1. **RESULT**

A table with numbers and symbols

Description automatically generated

The truth table for a 1-bit adder that shows the output values of Cout and Sum.

|  |
| --- |
| 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 AND2 is  Port (A: in std\_logic; --AND gate input  B: in std\_logic; --AND gate input  C: out std\_logic); --AND gate output  end AND2;  architecture Behavioral of AND2 is  begin  C <= A and B; -- 2 input AND gate  end Behavioral; |

VHDL code for AND2

|  |
| --- |
| 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 AND2Testbench is  -- Port ( );  end AND2Testbench;  architecture Behavioral of AND2Testbench is  Component AND2 is  Port (A,B:in std\_logic;  C: out std\_logic);  End component;  --inputs  Signal a: std\_logic:= '0';  Signal b: std\_logic:= '0';  --outputs  Signal c: std\_logic;  Constant period: time := 1ns;  begin  Uut: AND2 PORT MAP (a =>A, b => B, c => C);  a<= not a after period;  b<= not b after period\*2;  end Behavioral; |

Testbench for AND2

|  |
| --- |
| 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 OR2 is  Port ( A : in STD\_LOGIC;  B : in STD\_LOGIC;  C : out STD\_LOGIC);  end OR2;  architecture Behavioral of OR2 is  begin  C <= A or B; -- 2 input OR gate  end Behavioral; |

VHDL code for OR2

|  |
| --- |
| 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 OR2Testbench is  -- Port ( );  end OR2Testbench;  architecture Behavioral of OR2Testbench is  Component OR2 is  Port (A,B:in std\_logic;  C: out std\_logic);  End component;  --inputs  Signal a: std\_logic:= '0';  Signal b: std\_logic:= '0';  --outputs  Signal c: std\_logic;  Constant period: time := 1ns;  begin  Uut: OR2 PORT MAP (a =>A, b => B, c => C);  a<= not a after period;  b<= not b after period\*2;  end Behavioral; |

Testbench for OR2

|  |
| --- |
| 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 XOR2 is  Port ( A : in STD\_LOGIC;  B : in STD\_LOGIC;  C : out STD\_LOGIC);  end XOR2;  architecture Behavioral of XOR2 is  begin  C <= A xor B; -- 2 input XOR gate  end Behavioral; |

VHDL code for XOR2

|  |
| --- |
| 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 XOR2Testbench is  -- Port ( );  end XOR2Testbench;  architecture Behavioral of XOR2Testbench is  Component XOR2 is  Port (A,B:in std\_logic;  C: out std\_logic);  End component;  --inputs  Signal a: std\_logic:= '0';  Signal b: std\_logic:= '0';  --outputs  Signal c: std\_logic;  Constant period: time := 1ns;  begin  Uut: XOR2 PORT MAP (a =>A, b => B, c => C);  a<= not a after period;  b<= not b after period\*2;  end Behavioral; |

Testbench for XOR2

|  |
| --- |
| 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 OneBitFullAdder is  Port ( A : in STD\_LOGIC;  B : in STD\_LOGIC;  Cin : in STD\_LOGIC;  S : out STD\_LOGIC;  Cout : out STD\_LOGIC);  end OneBitFullAdder;  architecture Behavioral of OneBitFullAdder is  begin  S <= A xor B xor Cin;  Cout<= (((a xor b) and cin) or (a and b));  end Behavioral; |

VHDL code for 1-bit full adder

|  |
| --- |
| 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 OneBitFullAdderTestbench is  -- Port ( );  end OneBitFullAdderTestbench;  architecture Behavioral of OneBitFullAdderTestbench is  Component OneBitFullAdder is  Port (A,B,Cin:in std\_logic;  S,Cout: out std\_logic);  End component;  --inputs  Signal a: std\_logic:= '0';  Signal b: std\_logic:= '0';  Signal cin: std\_logic:= '0';  --outputs  Signal s: std\_logic;  Signal cout: std\_logic;  Constant period: time := 1ns;  begin  Uut: OneBitFullAdder PORT MAP (a =>A, b => B, cin => Cin, s => S, cout => Cout);  a<= not a after period;  b<= not b after period\*2;  cin<= not cin after period\*4;  end Behavioral; |

Testbench for 1-bit full adder

|  |
| --- |
| 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 FourBitFullAdder is  Port ( A : in STD\_LOGIC\_VECTOR (4 downto 1);  B : in STD\_LOGIC\_VECTOR (4 downto 1);  Cin : in STD\_LOGIC;  Cout : out STD\_LOGIC;  S : out STD\_LOGIC\_VECTOR (4 downto 1));  end FourBitFullAdder;  architecture Behavioral of FourBitFullAdder is  component OneBitFullAdder is  Port (A,B,Cin:in std\_logic;  S,Cout: out std\_logic);  end component;  Signal C1, C2, C3: std\_logic;  begin  oneOBFA: OneBitFullAdder PORT MAP(A => A(1), B => B(1), Cin => Cin, Cout => C1, S => S(1));  twoOBFA: OneBitFullAdder PORT MAP(A => A(2), B => B(2), Cin => C1, Cout => C2, S => S(2));  threeOBFA: OneBitFullAdder PORT MAP(A => A(3), B => B(3), Cin => C2, Cout => C3, S => S(3));  fourOBFA: OneBitFullAdder PORT MAP(A => A(4), B => B(4), Cin => C3, Cout => Cout, S => S(4));  end Behavioral; |

VHDL code for 4-bit full adder

|  |
| --- |
| 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 FourBitFullAdderTestbench is  -- Port ( );  end FourBitFullAdderTestbench;  architecture Behavioral of FourBitFullAdderTestbench is  Component FourBitFullAdder is  Port (A, B: in std\_logic\_vector (4 downto 1);  Cin:in std\_logic;  S: out std\_logic\_vector (4 downto 1);  Cout: out std\_logic);  End component;  --inputs  Signal a: std\_logic\_vector (4 downto 1):= (others =>'0');  Signal b: std\_logic\_vector (4 downto 1):= (others =>'0');  Signal cin: std\_logic:= '0';  --outputs  Signal s: std\_logic\_vector (4 downto 1);  Signal cout: std\_logic;  Constant period: time := 1ns;  begin  Uut: FourBitFullAdder PORT MAP (a => a, b => b, cin => cin, s => s, cout => cout);  test:process  begin  a<="0110";  b<="0011";  cin<='0';  wait for period;  a<="1010";  b<="0011";  cin<='0';  wait for period;  a<="0100";  b<="0101";  cin<='1';  wait for period;  a<="0101";  b<="0110";  cin<='1';  wait for period;  end process;  end Behavioral; |

Testbench for 4-bit full adder

|  |
| --- |
| 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 FourBitAddSub is  Port ( a : in STD\_LOGIC\_VECTOR (4 downto 1);  b : in STD\_LOGIC\_VECTOR (4 downto 1);  m : in STD\_LOGIC;  s : out STD\_LOGIC\_VECTOR (4 downto 1);  cout : out STD\_LOGIC);  end FourBitAddSub;  architecture Behavioral of FourBitAddSub is  component FourBitFullAdder is  Port ( A : in STD\_LOGIC\_VECTOR (4 downto 1);  B : in STD\_LOGIC\_VECTOR (4 downto 1);  Cin : in STD\_LOGIC;  Cout : out STD\_LOGIC;  S : out STD\_LOGIC\_VECTOR (4 downto 1));  end component;  begin  fulladder: FourBitFullAdder PORT MAP(A => A, B(1) => (B(1) xor m), B(2) => (B(2) xor m), B(3) => (B(3) xor m), B(4) => (B(4) xor m), Cin => m, Cout => cout, S => S);  end Behavioral; |

VHDL code for 4-bit adder/subtractor

|  |
| --- |
| 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 FourBitAddSubTestbench is  -- Port ( );  end FourBitAddSubTestbench;  architecture Behavioral of FourBitAddSubTestbench is  Component FourBitAddSub is  Port (A, B: in std\_logic\_vector (4 downto 1);  m:in std\_logic;  S: out std\_logic\_vector (4 downto 1);  Cout: out std\_logic);  End component;  --inputs  Signal a: std\_logic\_vector (4 downto 1):= (others =>'0');  Signal b: std\_logic\_vector (4 downto 1):= (others =>'0');  Signal m: std\_logic:= '0';  --outputs  Signal s: std\_logic\_vector (4 downto 1);  Signal cout: std\_logic;  Constant period: time := 1ns;  begin  Uut: FourBitAddSub PORT MAP (a => a, b => b, m => m, s => s, cout => cout);  test:process  begin  a<="0110";  b<="0011";  m<='0';  wait for period;  a<="1010";  b<="0011";  m<='0';  wait for period;  a<="0100";  b<="0101";  m<='1';  wait for period;  a<="0101";  b<="0110";  m<='1';  wait for period;  end process;  end Behavioral; |

Testbench for 4-bit adder/subtractor

A screenshot of a computer

Description automatically generated

Waveform for 1-bit adder

A screenshot of a computer

Description automatically generated

Waveform for 4-bit adder

A screenshot of a computer

Description automatically generated

Waveform for 4-bit adder/subtractor

Question: 29 input patterns are required for both a 4-bit adder and 4-bit adder/subtractor because they both have 9 inputs. 4 a inputs, 4 b inputs, and a cin input for the adder and a m input for the adder/subtractor

1. **CONCLUSION**

My results satisfy the requirements, as I get correct outputs for all the given inputs. I don’t think it’s possible to improve my design to get better results. I have learned basic VHDL coding.