****

**EE 4550L/EE6550L**

**IC Hardware Security and Trust LAB**

**SPRING 2025**

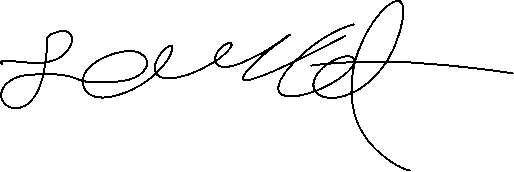
**TA: Kanchan Vissamsetty**

**Lab section: 01**

**Name: Logan Current**

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

**Signature: Date: 1/30/2025**



**Report due date: 2/6/2025**

1. **OBJECTIVE**

The Objective of this lab is to practice writing VHDL code within the Xilinx Vivado IDE and to simulate VHDL code along with the corresponding testbench.

1. **PROCEDURE**

First, I made the truth table for the 1-bit full adder then optimized the Cout and S outputs using a K-map. This made it easier and more efficient to code for in VHDL. Afterwards I made the .vhd file with the testbench for the full adder and confirmed the output waveform matched my truth table. Next, I modified the same code to make my 4-bit ripple carry adder by using generate statements to make it easier for mort map everything together. I then modified the test bench for the full adder to work for the 4-bit fuller adder and made a process called “test\_VECTOR” to make sure the specific test cases worked for my design. Finally, I modified the same file from before and added a XOR2 component within the file and used a generate statement for the XOR2 and fuller adder form before making the port maps for me. The testbench used was the same one in the 4-bit ripple carry adder but modified it to work with the adder/subtractor.

1. **RESULT**

**Handwritten Work:**

A paper with writing on it

AI-generated content may be incorrect.

**1-bit Full Adder Waveform:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**4-bit Ripple Carry Adder Waveform:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Adder Subtractor Waveform:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**VHDL Code:**

**--- FA.vhd---**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity RCA is**

**Port ( A,B,Cin : in STD\_LOGIC;**

**Cout,S : out STD\_LOGIC);**

**end RCA;**

**architecture Behavioral of RCA is**

**begin**

**Cout <= (Cin AND B) OR (Cin AND A) OR (A AND B);**

**S <= (A XOR B) XOR Cin;**

**end Behavioral;**

**--- tb\_FA.vhd---**

**library IEEE,STD,WORK;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**entity tb\_RCA is**

**end tb\_RCA;**

**architecture Behavioral of tb\_RCA is**

**component RCA**

**port (A,B : in std\_logic;**

**Cin : in std\_logic;**

**Cout,S : out std\_logic);**

**end component;**

**signal a,b : std\_logic:= '0';**

**signal Ci : std\_logic:= '0';**

**signal Co,s : std\_logic;**

**constant period : time := 1 ns;**

**begin**

**uut: RCA port map (A=>a, B=>b, Cin=>Ci, Cout=>Co, S=>s);**

**a <= not a after period;**

**b <= not b after period\*2;**

**Ci <= not Ci after period\*4;**

**end Behavioral;**

**--- RCA.vhd ---**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity FA is**

**Port (A,B,Cin : in STD\_LOGIC;**

**Cout,S : out STD\_LOGIC);**

**end FA;**

**architecture FA\_Behavioral of FA is**

**begin**

**Cout <= (Cin AND B) OR (Cin AND A) OR (A AND B);**

**S <= (A XOR B) XOR Cin;**

**end;**

**---**

**---**

**---**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity RCA is**

**generic (N:integer:= 4);**

**port (A,B : in std\_logic\_vector(N-1 downto 0);**

**Cin : in std\_logic;**

**Cout : out std\_logic;**

**S : out std\_logic\_vector(N-1 downto 0));**

**end;**

**architecture RCA\_Behavioral of RCA is**

**component FA**

**port (A,B,Cin : in std\_logic;**

**Cout,S : out std\_logic);**

**end component;**

**signal C : std\_logic\_vector(N downto 0);**

**begin**

**C(0) <= Cin;**

**GI: for I in 0 to N-1 generate**

**GI:FA port map(A=>A(I), B=>B(I), Cin => C(I), Cout => C(I+1), S => S(I));**

**end generate;**

**Cout <= C(N);**

**end;**

**--------tb\_RCA------**

**library IEEE,STD,WORK;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**entity tb\_RCA is**

**generic (N:integer:= 4);**

**end tb\_RCA;**

**architecture Behavioral of tb\_RCA is**

**component RCA**

**generic(N:integer);**

**port (A,B : in std\_logic\_vector(N-1 downto 0);**

**Cin : in std\_logic;**

**Cout : out std\_logic;**

**S : out std\_logic\_vector(N-1 downto 0));**

**end component;**

**signal A,B : std\_logic\_vector(N-1 downto 0);**

**signal Ci : std\_logic:= '0';**

**signal S : std\_logic\_vector(N downto 0);**

**constant period : time := 12 ns;**

**begin**

**uut: RCA generic map(N=>N) port map(A=>A, B=>B, Cin=>Ci, Cout=>S(N), S=>S(N-1 downto 0));**

**test\_VECTOR : process**

**begin**

**----------------------------------**

**A <= "0110";**

**B <= "0011";**

**Ci <= '0';**

**wait for period;**

**----------------------------------**

**A <= "1010";**

**B <= "0011";**

**Ci <= '0';**

**wait for period;**

**----------------------------------**

**A <= "0100";**

**B <= "0101";**

**Ci <= '1';**

**wait for period;**

**----------------------------------**

**A <= "0101";**

**B <= "0110";**

**Ci <= '1';**

**wait for period;**

**----------------------------------**

**end process;**

**end Behavioral;**

**--- RCA.vhd---- (adder\_subtractor code)--**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity FA is**

**Port (A,B,Cin : in STD\_LOGIC;**

**Cout,S : out STD\_LOGIC);**

**end FA;**

**architecture FA\_Behavioral of FA is**

**begin**

**Cout <= (Cin AND B) OR (Cin AND A) OR (A AND B);**

**S <= (A XOR B) XOR Cin;**

**end;**

**---**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity XOR2 is**

**Port (A,B :in std\_logic;**

**Y: out std\_logic);**

**end entity;**

**architecture XOR2\_Behavioral of XOR2 is**

**begin**

**Y <= A XOR B;**

**end;**

**---**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.NUMERIC\_STD.ALL;**

**entity RCA is**

**generic (N:integer:= 4);**

**port (A,B : in std\_logic\_vector(N-1 downto 0);**

**M: in std\_logic;**

**Cout : out std\_logic;**

**S : out std\_logic\_vector(N-1 downto 0));**

**end;**

**architecture RCA\_Behavioral of RCA is**

**component FA**

**port (A,B,Cin : in std\_logic;**

**Cout,S : out std\_logic);**

**end component;**

**component XOR2**

**port (A,B : in std\_logic;**

**Y : out std\_logic);**

**end component;**

**signal OP : std\_logic\_vector(N-1 downto 0);**

**signal C : std\_logic\_vector(N downto 0);**

**begin**

**C(0) <= M;**

**GJ: for I in 0 to N-1 generate**

**GJ:XOR2 port map(A=>B(I), B=>M, Y=>OP(I));**

**end generate;**

**GI: for I in 0 to N-1 generate**

**GI:FA port map(A=>A(I), B=>OP(I), Cin=>C(I), Cout =>C(I+1), S => S(I));**

**end generate;**

**Cout <= C(N);**

**end;**

**----tb\_RCA---(adder/subtractor testbench)---**

**library IEEE,STD,WORK;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**entity tb\_RCA is**

**generic (N:integer:= 4);**

**end tb\_RCA;**

**architecture Behavioral of tb\_RCA is**

**component RCA**

**generic(N:integer);**

**port (A,B : in std\_logic\_vector(N-1 downto 0);**

**M : in std\_logic;**

**Cout : out std\_logic;**

**S : out std\_logic\_vector(N-1 downto 0));**

**end component;**

**signal A,B,S : std\_logic\_vector(N-1 downto 0);**

**signal M,Cout : std\_logic;**

**constant period : time := 10 ns;**

**begin**

**uut: RCA generic map(N=>N) port map (A=>A, B=>B, M=>M, Cout=>Cout, S=>S);**

**test\_VECTOR : 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;**

**Questions to Answer:**

**Question: How many input patterns are required for exhaustive testing on 4-bit adder and**

**adder/subtractor? Why?**

Well, for a 4-bit adder it has an A input that has or 16 possible values, same with the B input. The Cin input has or 2 possible values. To get the total number of combinations, you have to multiply which comes out to be 512 total input combinations.

For an adder/subtractor, depending on how it is made it would also have an A and B input that have or 16 possible values each. The carry-in (or Cin) would also have , 2 possible values, there would be an extra input that denotes adding or subtract which is called M in this case, it would have or 2 possible values. The total input combinations would end up being and equal 1024 as a total input combination.

1. **CONCLUSION**

Overall, this lab wasn’t too bad. I had to brush up on my VHDL skills and remember the proper syntax, I ended up coding the full adder as its own component instead of making it with NAND gates but it’s more optimal that way. The only issue I ran into during this lab was the waveform for making sure the adder/subtractor worked. When it did the 5-4 example was output 15, which is due to 2’s compliment math but I wasn’t sure if we should have had a carry bit added to the S waveform to show that it was 1. I made sure that the subtraction worked properly by also switching the test cases where M=1 to be M=0 and vice versa and got the expected output. Either way, the adder/subtractor works as expected.