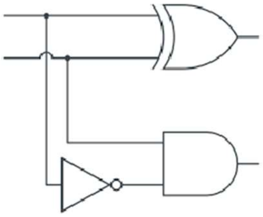
Dylan Liesenfelt
13 JUNE 2024
CDA3203

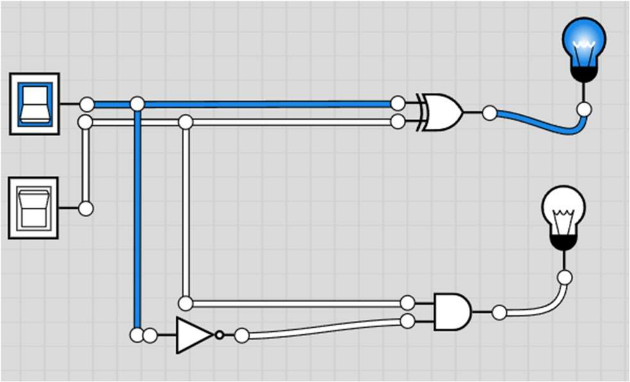
Class Activity 1

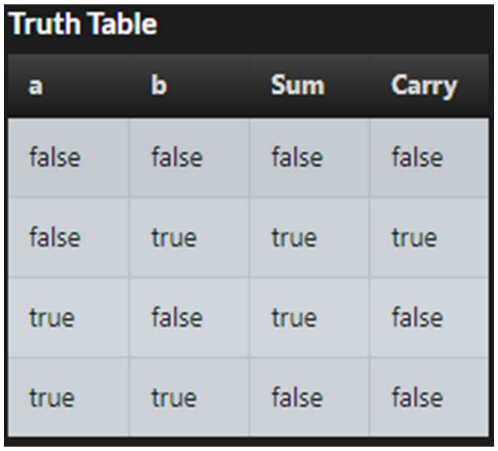
1. Design the below combinational circuits using VHDL and simulate the same in EDA playground with a suitable test case in testbench.



(a)

A.

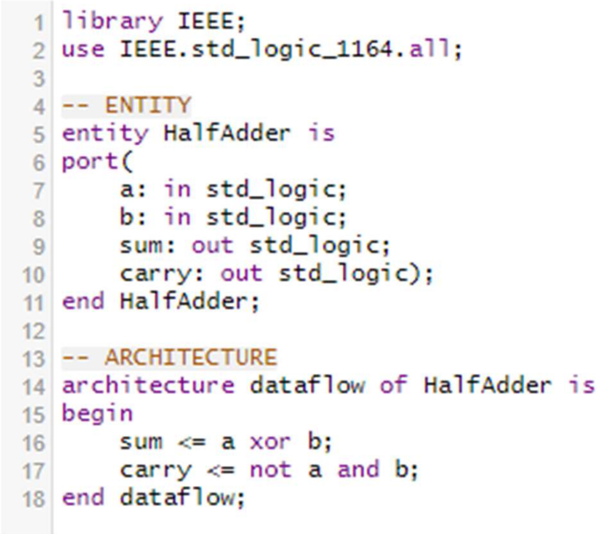




Truth Table

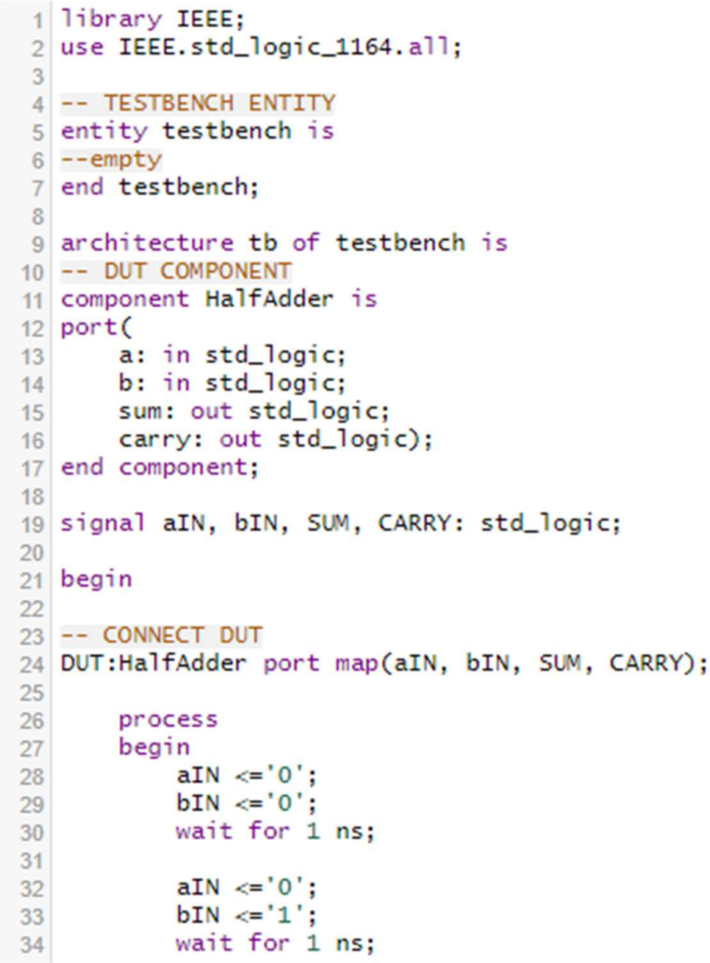
a	b	Sum	Carry
false	false	false	false
false	true	true	true
true	false	true	false
true	true	false	false

Design:

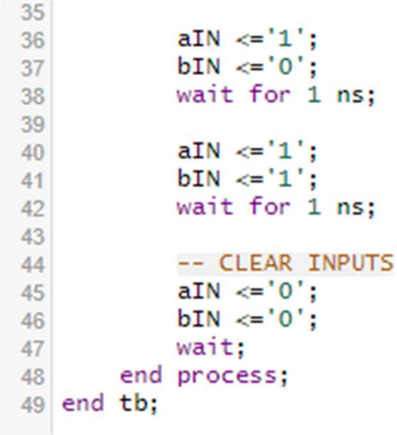


```
library IEEE;
use IEEE.std_logic_1164.all;

-- ENTITY
entity HalfAdder is
port(
    a: in std_logic;
    b: in std_logic;
    sum: out std_logic;
    carry: out std_logic);
end HalfAdder;

-- ARCHITECTURE
architecture dataflow of HalfAdder is
begin
    sum <= a xor b;
    carry <= not a and b;
end dataflow;
```

Testbench:

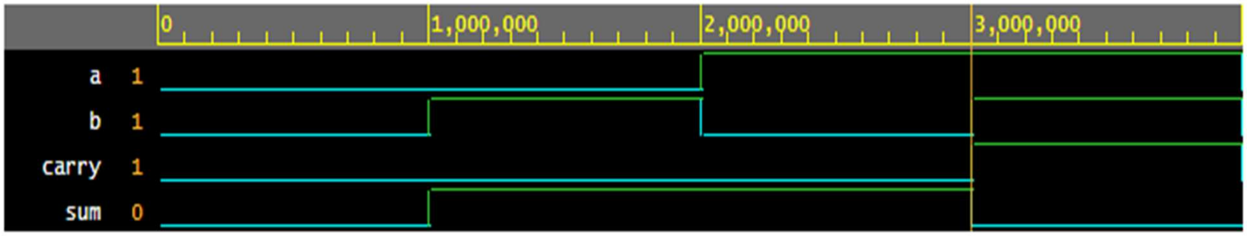


```
library IEEE;
use IEEE.std_logic_1164.all;

-- TESTBENCH ENTITY
entity testbench is
--empty
end testbench;

architecture tb of testbench is
-- DUT COMPONENT
component HalfAdder is
port(
    a: in std_logic;
    b: in std_logic;
    sum: out std_logic;
    carry: out std_logic);
end component;

signal aIN, bIN, SUM, CARRY: std_logic;

begin

-- CONNECT DUT
DUT:HalfAdder port map(aIN, bIN, SUM, CARRY);

    process
    begin
        aIN <='0';
        bIN <='0';
        wait for 1 ns;

        aIN <='0';
        bIN <='1';
        wait for 1 ns;
```



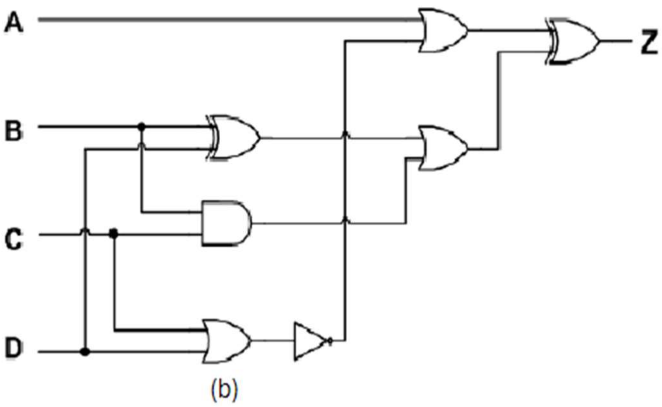
```

        aIN <='1';
        bIN <='0';
        wait for 1 ns;

        aIN <='1';
        bIN <='1';
        wait for 1 ns;

        -- CLEAR INPUTS
        aIN <='0';
        bIN <='0';
        wait;
    end process;
end tb;
```

Wave:



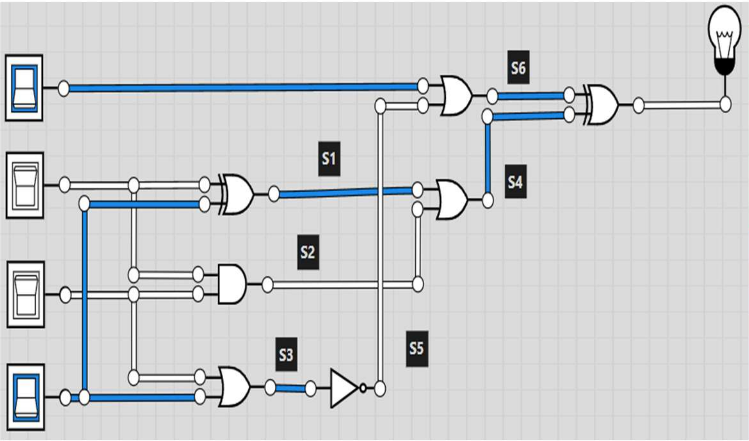
Observation:

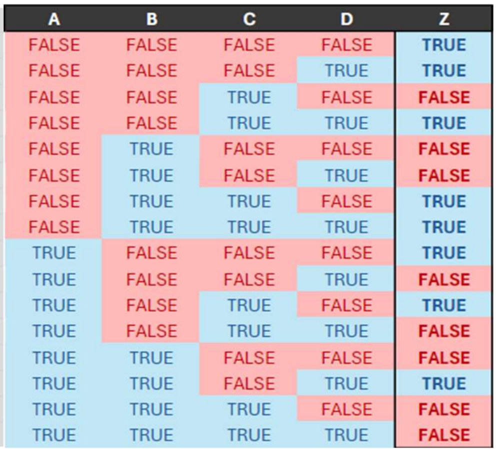
All I did was modify my half adder design with a not function for the carry, but we can see from the wave and truth table that they are the same outputs.



(b)

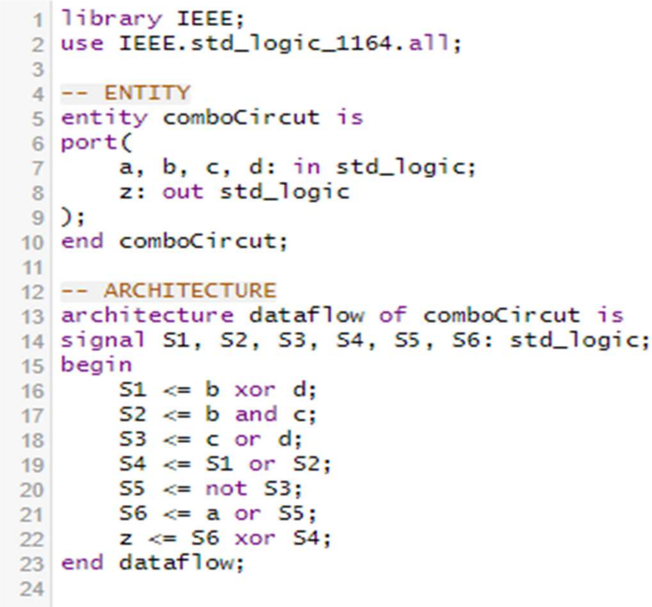
B.





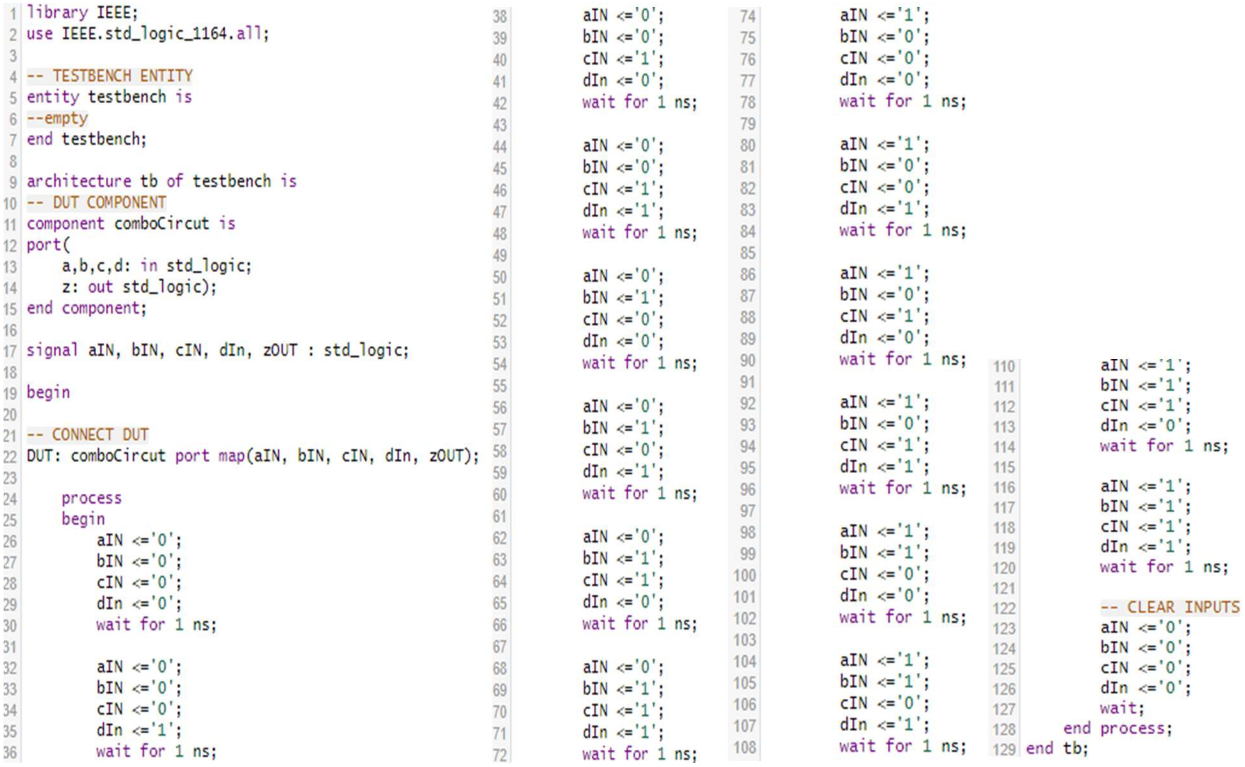
A	B	C	D	Z
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	TRUE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	TRUE	TRUE	FALSE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE

Design:



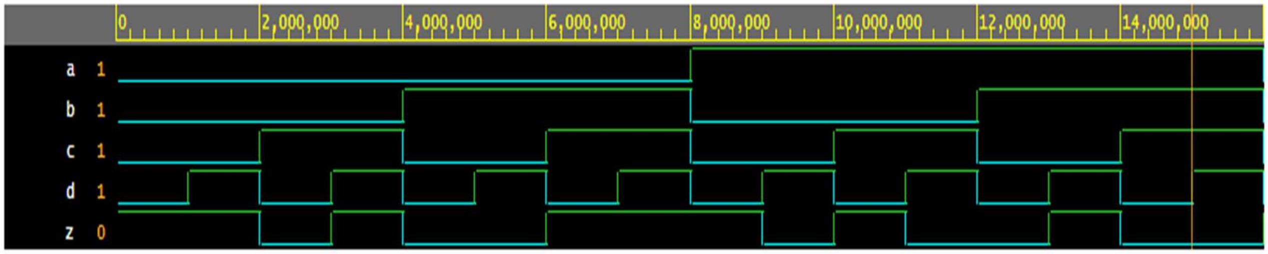
```
library IEEE;
use IEEE.std_logic_1164.all;

-- ENTITY
entity comboCircut is
port(
    a, b, c, d: in std_logic;
    z: out std_logic
);
end comboCircut;

-- ARCHITECTURE
architecture dataflow of comboCircut is
signal S1, S2, S3, S4, S5, S6: std_logic;
begin
    S1 <= b xor d;
    S2 <= b and c;
    S3 <= c or d;
    S4 <= S1 or S2;
    S5 <= not S3;
    S6 <= a or S5;
    z <= S6 xor S4;
end dataflow;

```

Testbench:



```
library IEEE;
use IEEE.std_logic_1164.all;

-- TESTBENCH ENTITY
entity testbench is
--empty
end testbench;

architecture tb of testbench is
-- DUT COMPONENT
component comboCircut is
port(
    a,b,c,d: in std_logic;
    z: out std_logic);
end component;

signal aIN, bIN, cIN, dIn, zOUT : std_logic;

begin

-- CONNECT DUT
DUT: comboCircut port map(aIN, bIN, cIN, dIn, zOUT);

    process
    begin
        aIN <='0';
        bIN <='0';
        cIN <='0';
        dIn <='0';
        wait for 1 ns;

        aIN <='0';
        bIN <='0';
        cIN <='0';
        dIn <='1';
        wait for 1 ns;

        aIN <='0';
        bIN <='0';
        cIN <='1';
        dIn <='0';
        wait for 1 ns;

        aIN <='0';
        bIN <='0';
        cIN <='1';
        dIn <='1';
        wait for 1 ns;

        aIN <='0';
        bIN <='1';
        cIN <='0';
        dIn <='0';
        wait for 1 ns;

        aIN <='0';
        bIN <='1';
        cIN <='0';
        dIn <='1';
        wait for 1 ns;

        aIN <='0';
        bIN <='1';
        cIN <='1';
        dIn <='0';
        wait for 1 ns;

        aIN <='0';
        bIN <='1';
        cIN <='1';
        dIn <='1';
        wait for 1 ns;

        aIN <='1';
        bIN <='0';
        cIN <='0';
        dIn <='0';
        wait for 1 ns;

        aIN <='1';
        bIN <='0';
        cIN <='0';
        dIn <='1';
        wait for 1 ns;

        aIN <='1';
        bIN <='0';
        cIN <='1';
        dIn <='0';
        wait for 1 ns;

        aIN <='1';
        bIN <='0';
        cIN <='1';
        dIn <='1';
        wait for 1 ns;

        aIN <='1';
        bIN <='1';
        cIN <='0';
        dIn <='0';
        wait for 1 ns;

        aIN <='1';
        bIN <='1';
        cIN <='0';
        dIn <='1';
        wait for 1 ns;

        aIN <='1';
        bIN <='1';
        cIN <='1';
        dIn <='0';
        wait for 1 ns;

        aIN <='1';
        bIN <='1';
        cIN <='1';
        dIn <='1';
        wait for 1 ns;

        -- CLEAR INPUTS
        aIN <='0';
        bIN <='0';
        cIN <='0';
        dIn <='0';
        wait;
    end process;
end tb;
```

Wave:



Observation:

This one was a really good learning experience, I feel more comfortable in VHDL now after this exercise. As we can see, the wave and truth table match each other, implementation was once again successful.