ECE 413: Intro to VLSI

Assignment 5: Two-Bit Full Adder Design

Roderick Renwick

DEARBORN

November 20, 2019
Fall 2019

Honor Code:

I have neither given nor received unauthorized assistance on this graded report.

X______________________________Roderick_Renwick_____________________________

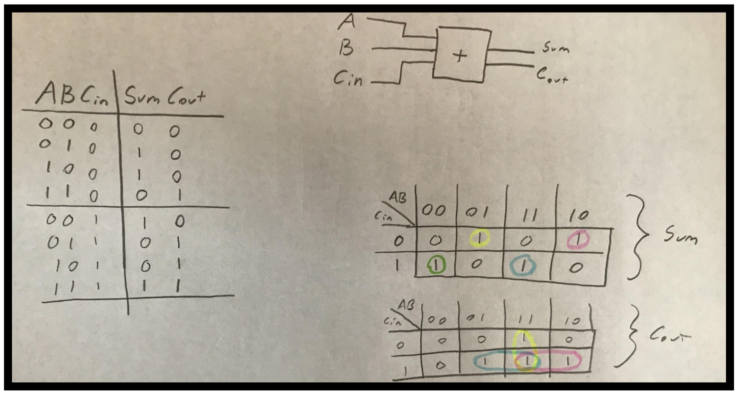
Table of Contents

Introduction

The following laboratory procedure lays out the construction of a two-bit full adder and its transistor layout composed of CMOS inverters that are grouped into NAND and XOR logic gates.

Design: One-Bit-Full-Adder

To realize a two-bit-full-adder, the one-bit-full-adder must be designed first. The following figures provide the truth table, optimized kmap functions, and logic-gate layout for a one-bit-full-adder:


A
B
Cin
+
Sum
Cout

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

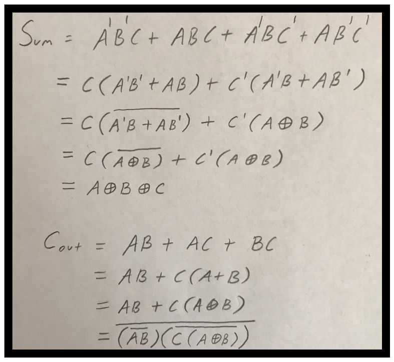
Sum

C_{in} \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C_{out}

C_{in} \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

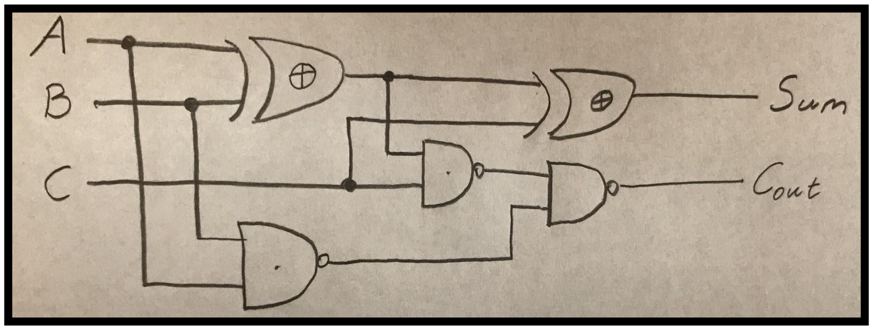
One-Bit-Full-Adder Truth Table & K-Maps



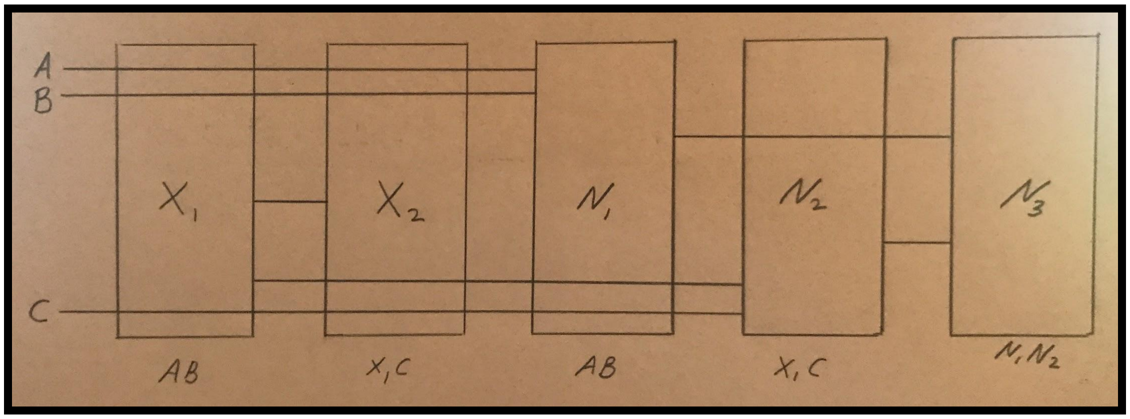
$$
\begin{aligned}
Sum &= A'B'C + ABC + A'BC' + AB'C' \\
&= C(A'B' + AB) + C'(A'B + AB') \\
&= C(\overline{A'B + AB'}) + C'(A \oplus B) \\
&= C(\overline{A \oplus B}) + C'(A \oplus B) \\
&= A \oplus B \oplus C
\end{aligned}
$$

$$
\begin{aligned}
C_{out} &= AB + AC + BC \\
&= AB + C(A+B) \\
&= AB + C(A \oplus B) \\
&= \overline{(\overline{AB})(\overline{C(A \oplus B)})}
\end{aligned}
$$

One-Bit-Full-Adder Boolean Algebra Optimization



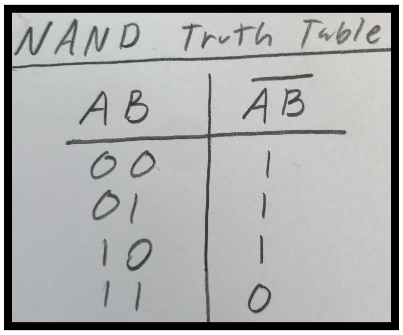
One-Bit-Full-Adder Logic-Gate Design



Candidate Layout for One-Bit Adder Desgin

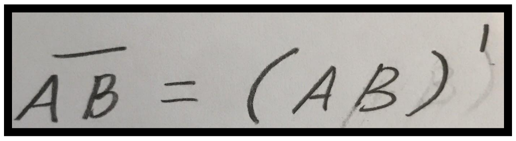
Design: NAND Gate

The truth table, optimized functions, circuit design, and transistor-level layout for a two-input NAND gate are as follows:

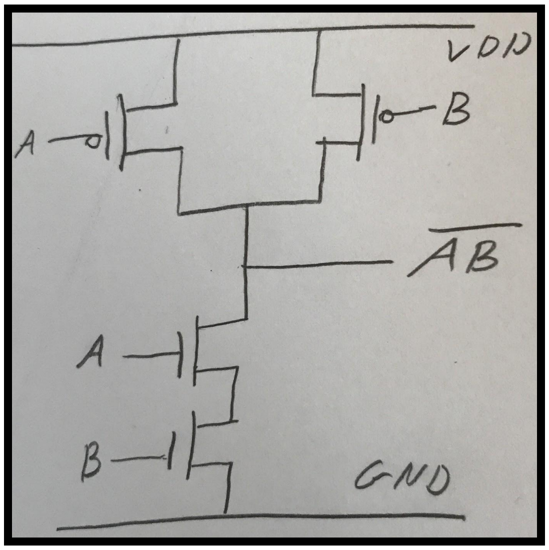


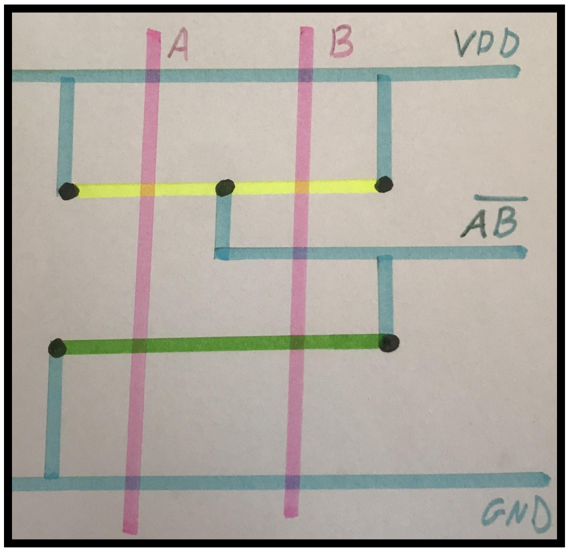
NAND Truth Table

AB	$\overline{AB}$
00	1
01	1
10	1
11	0



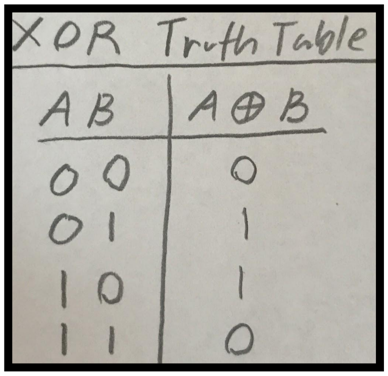
$$\overline{AB} = (AB)'$$





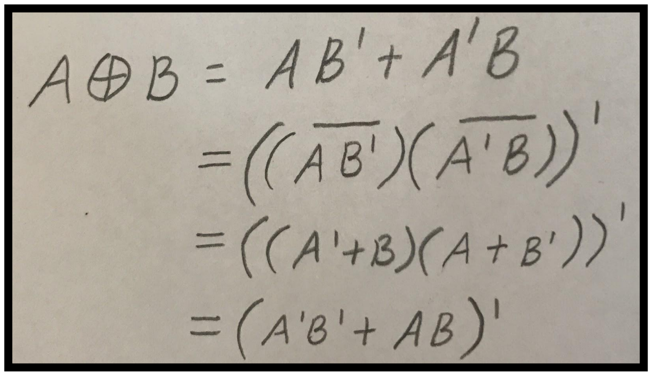
Design: XOR Gate

The truth table, optimized functions, circuit design, and transistor-level layout for a two-input XOR gate are as follows:

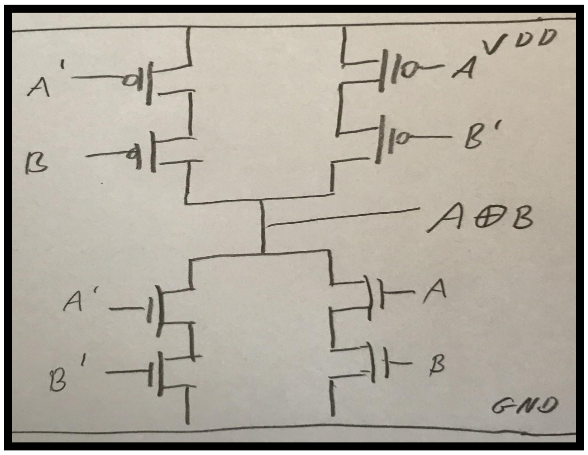


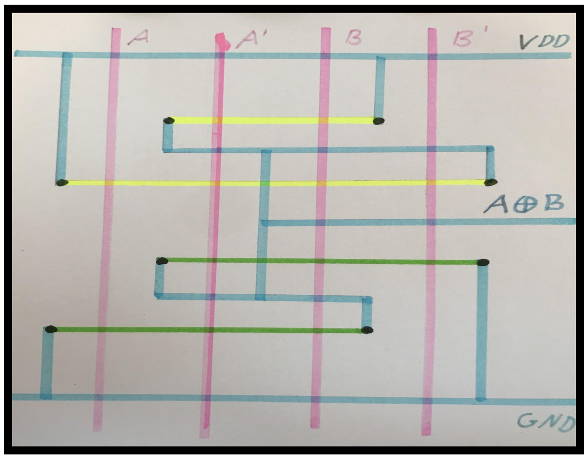
XOR Truth Table

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



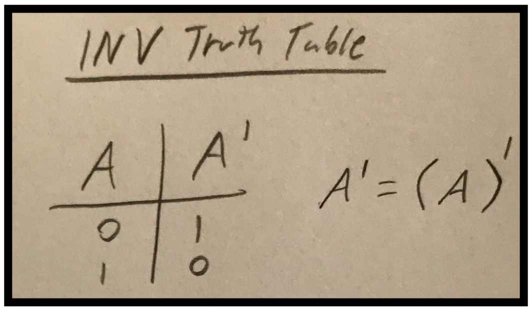
$$
\begin{aligned}
A \oplus B &= AB' + A'B \\
&= ((\overline{AB'})(\overline{A'B}))' \\
&= ((A'+B)(A+B'))' \\
&= (A'B' + AB)'
\end{aligned}
$$





Design: INV Gate

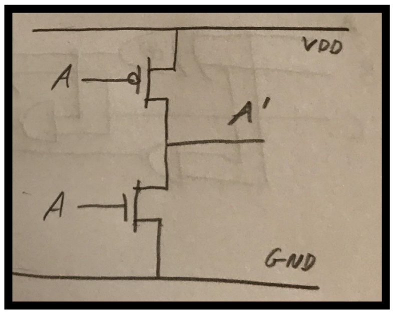
The truth table, optimized functions, circuit design, and transistor-level layout for an Inverter gate is as follows (note that this is done for both inputs A, and B):

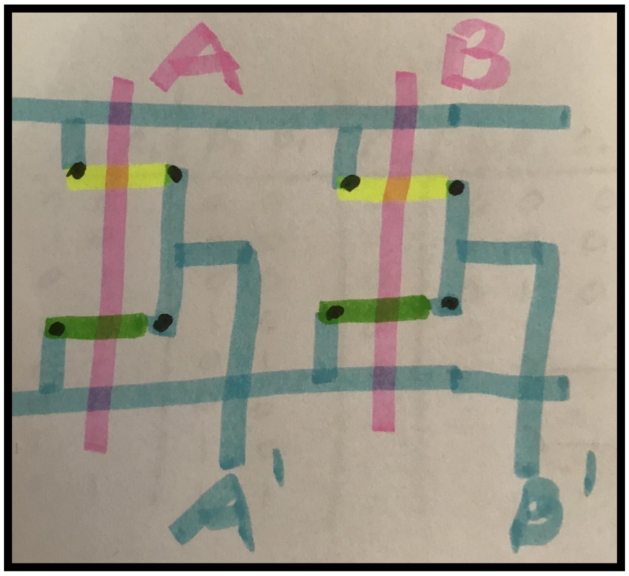


INV Truth Table

A	A'
0	1
1	0

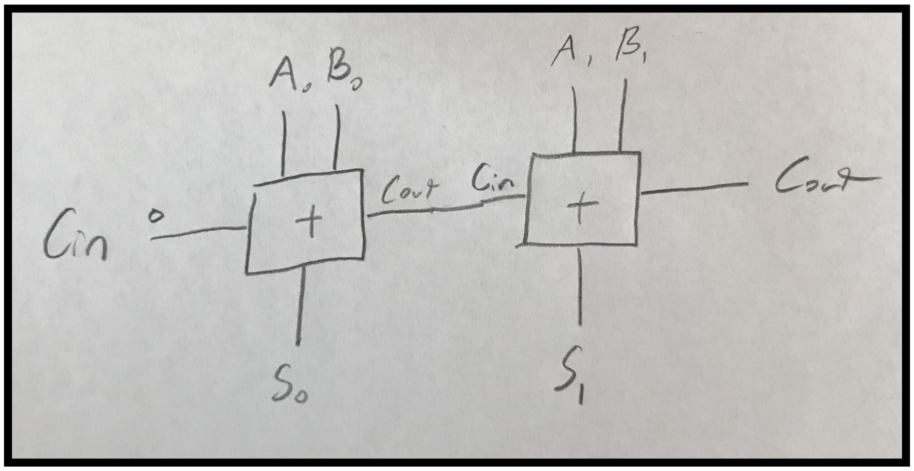
$A' = (A)'$

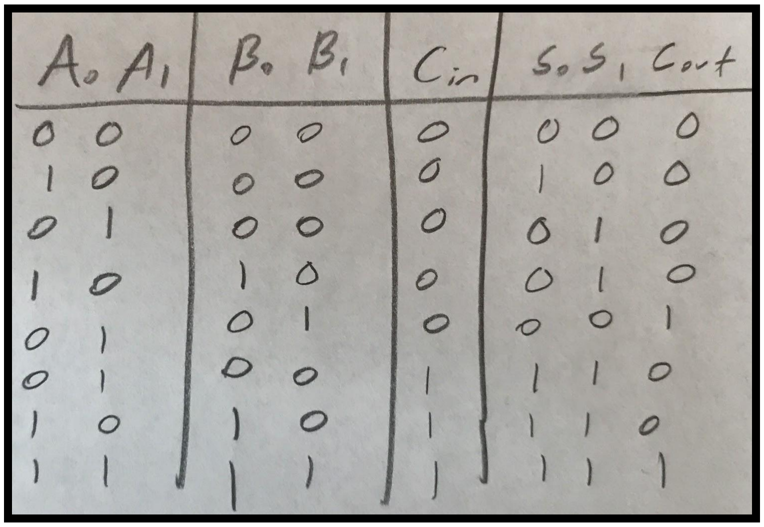




Design: Two-Bit-Full-Adder

Following the design and layout of transistors to construct a NAND gate, Inverter Gate, and an XOR gate, required to realize a one-bit-full-adder, a two-bit-full-adder may then be implemented by connecting two one-bit-full-adders together. This is done by taking the carr-out of the first adder and feeding it into the carry-in of the second adder, while at the same time feeding the compliments of the A and B inputs into the second adder. The design and truth table for the two-bit-full-adder are as follows:

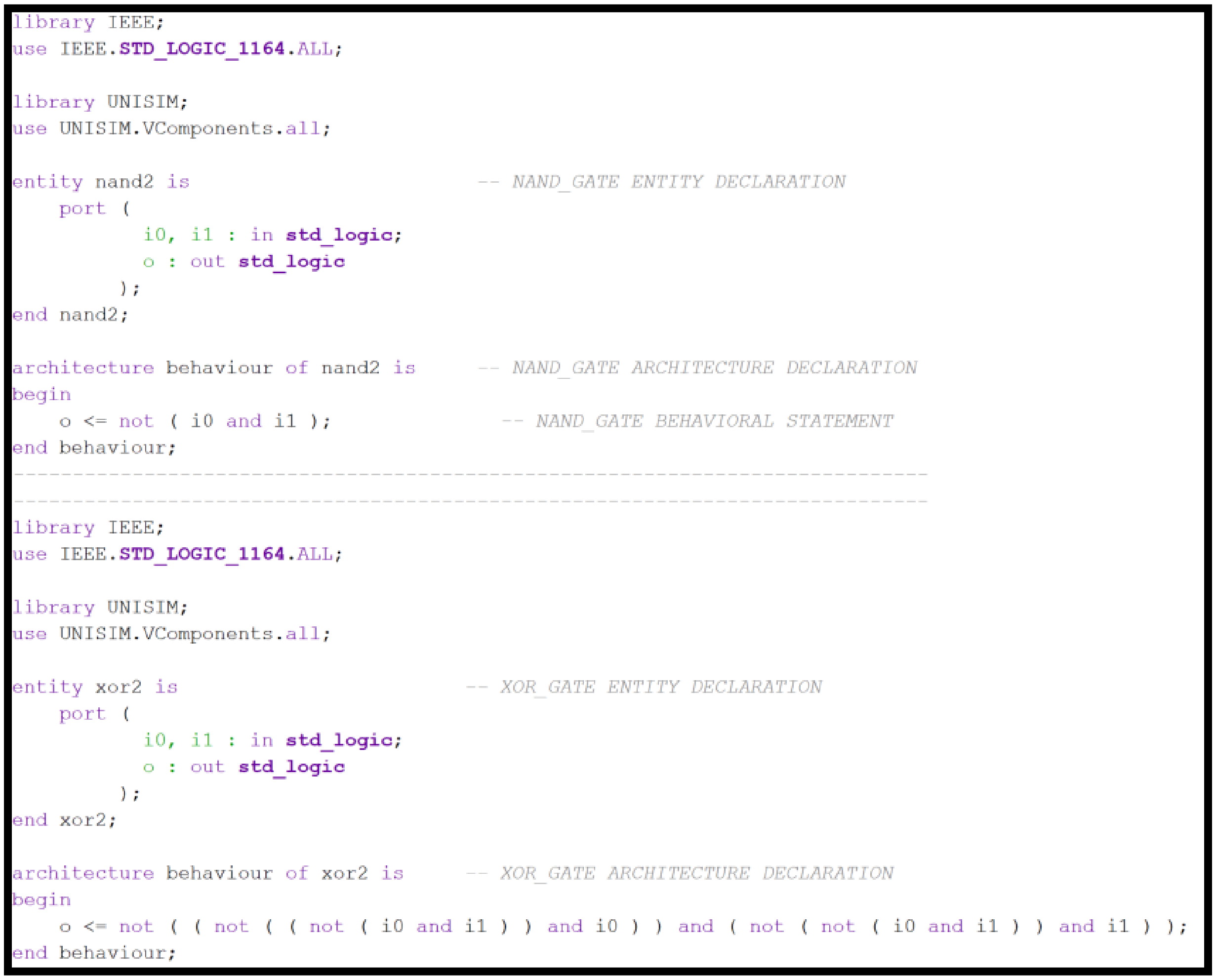




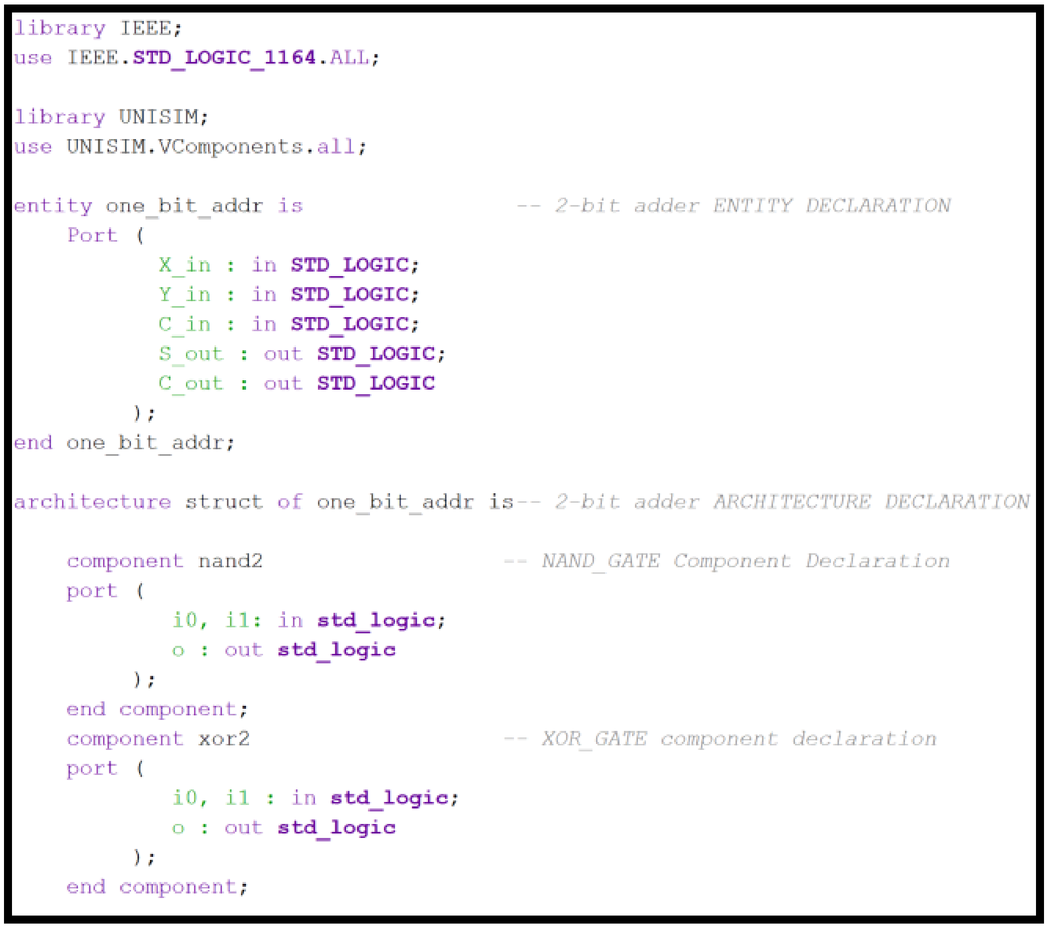
A_0	A_1	B_0	B_1	C_{in}	S_0	S_1	C_{out}
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	0	1	0
0	1	0	1	0	0	0	1
0	1	0	0	1	1	1	0
1	0	1	0	1	1	1	0
1	1	1	1	1	1	1	1

Implementation & Simulation

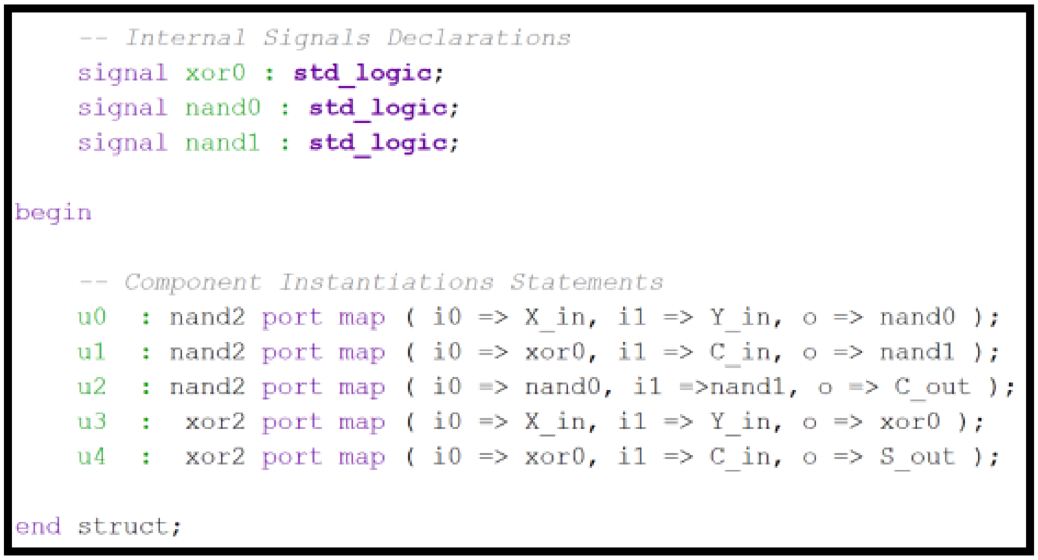
The following figures show the logical simulation of the two-bit-full-adder through valvalo software to verify and validate the design implementation:



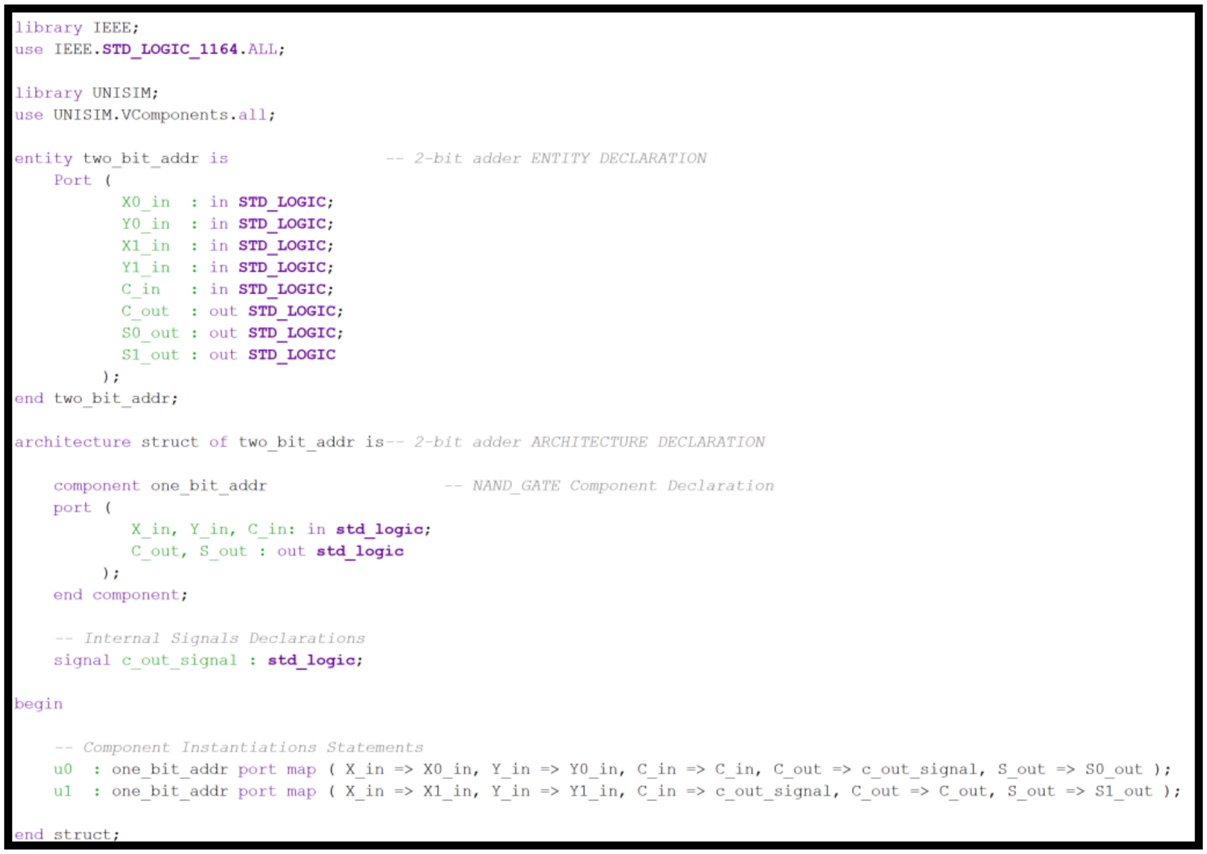
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity nand2 is                        -- NAND_GATE ENTITY DECLARATION
    port (
           i0, i1 : in std_logic;
           o : out std_logic
         );
end nand2;

architecture behaviour of nand2 is     -- NAND_GATE ARCHITECTURE DECLARATION
begin
    o <= not ( i0 and i1 );              -- NAND_GATE BEHAVIORAL STATEMENT
end behaviour;
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity xor2 is                         -- XOR_GATE ENTITY DECLARATION
    port (
           i0, i1 : in std_logic;
           o : out std_logic
         );
end xor2;

architecture behaviour of xor2 is      -- XOR_GATE ARCHITECTURE DECLARATION
begin
    o <= not ( ( not ( ( not ( i0 and i1 ) ) and i0 ) ) and ( not ( not ( i0 and i1 ) ) and i1 ) );
end behaviour;
```



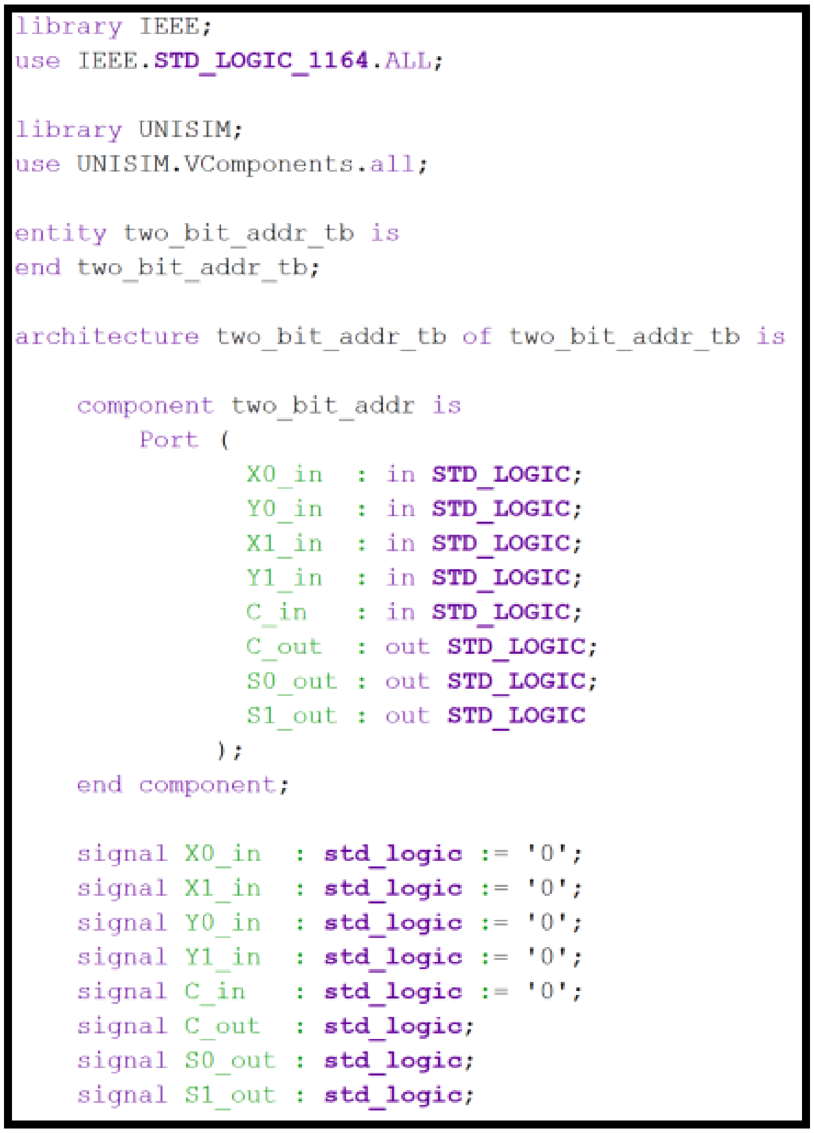
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity one_bit_addr is                    -- 2-bit adder ENTITY DECLARATION
    Port (
           X_in : in STD_LOGIC;
           Y_in : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S_out : out STD_LOGIC;
           C_out : out STD_LOGIC
         );
end one_bit_addr;

architecture struct of one_bit_addr is-- 2-bit adder ARCHITECTURE DECLARATION

    component nand2                       -- NAND_GATE Component Declaration
    port (
            i0, i1: in std_logic;
            o : out std_logic
         );
    end component;
    component xor2                        -- XOR_GATE component declaration
    port (
            i0, i1 : in std_logic;
            o : out std_logic
         );
    end component;
```



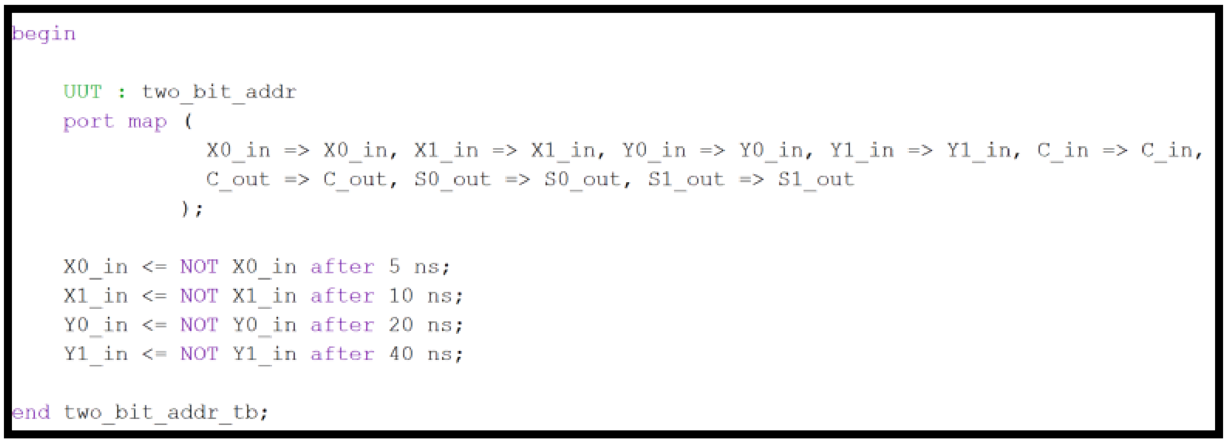
```
    -- Internal Signals Declarations
    signal xor0 : std_logic;
    signal nand0 : std_logic;
    signal nand1 : std_logic;

begin

    -- Component Instantiations Statements
    u0  : nand2 port map ( i0 => X_in, i1 => Y_in, o => nand0 );
    u1  : nand2 port map ( i0 => xor0, i1 => C_in, o => nand1 );
    u2  : nand2 port map ( i0 => nand0, i1 =>nand1, o => C_out );
    u3  :  xor2 port map ( i0 => X_in, i1 => Y_in, o => xor0 );
    u4  :  xor2 port map ( i0 => xor0, i1 => C_in, o => S_out );

end struct;
```



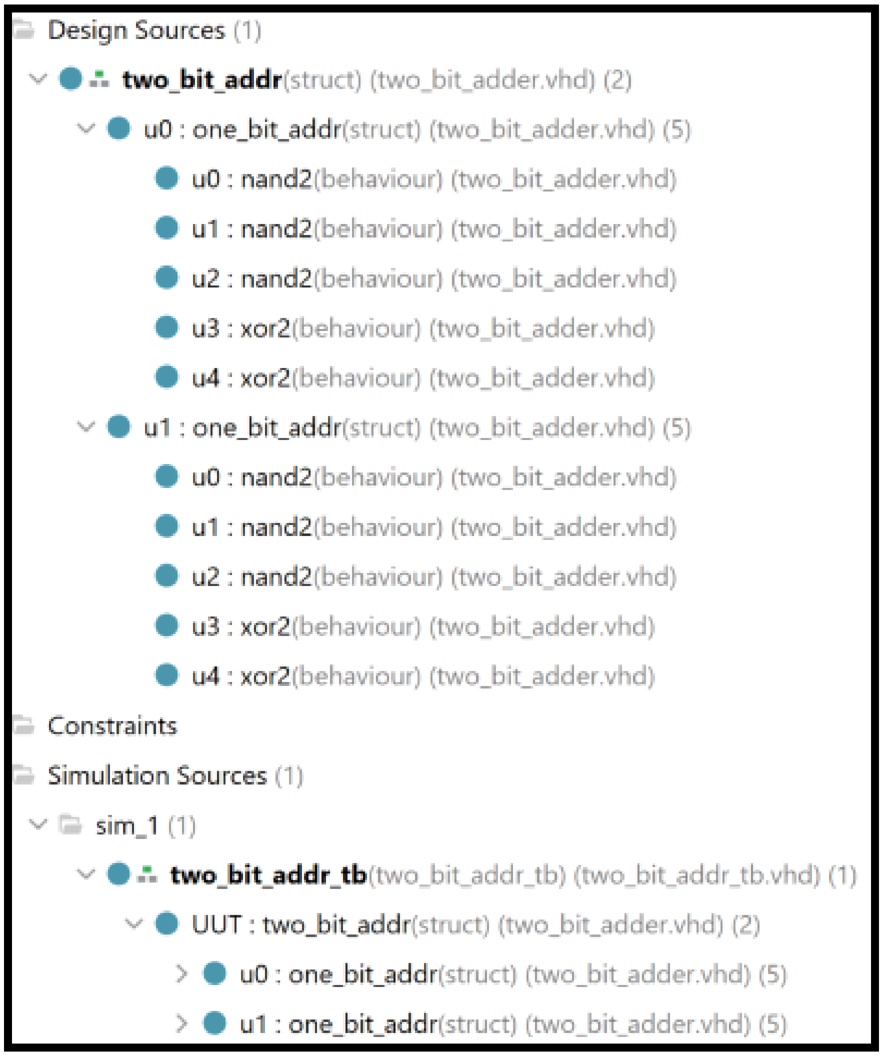
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity two_bit_addr is                  -- 2-bit adder ENTITY DECLARATION
    Port (
           X0_in  : in STD_LOGIC;
           Y0_in  : in STD_LOGIC;
           X1_in  : in STD_LOGIC;
           Y1_in  : in STD_LOGIC;
           C_in   : in STD_LOGIC;
           C_out  : out STD_LOGIC;
           S0_out : out STD_LOGIC;
           S1_out : out STD_LOGIC
         );
end two_bit_addr;

architecture struct of two_bit_addr is-- 2-bit adder ARCHITECTURE DECLARATION

    component one_bit_addr                    -- NAND_GATE Component Declaration
    port (
            X_in, Y_in, C_in: in std_logic;
            C_out, S_out : out std_logic
         );
    end component;

    -- Internal Signals Declarations
    signal c_out_signal : std_logic;

begin

    -- Component Instantiations Statements
    u0  : one_bit_addr port map ( X_in => X0_in, Y_in => Y0_in, C_in => C_in, C_out => c_out_signal, S_out => S0_out );
    u1  : one_bit_addr port map ( X_in => X1_in, Y_in => Y1_in, C_in => c_out_signal, C_out => C_out, S_out => S1_out );

end struct;
```



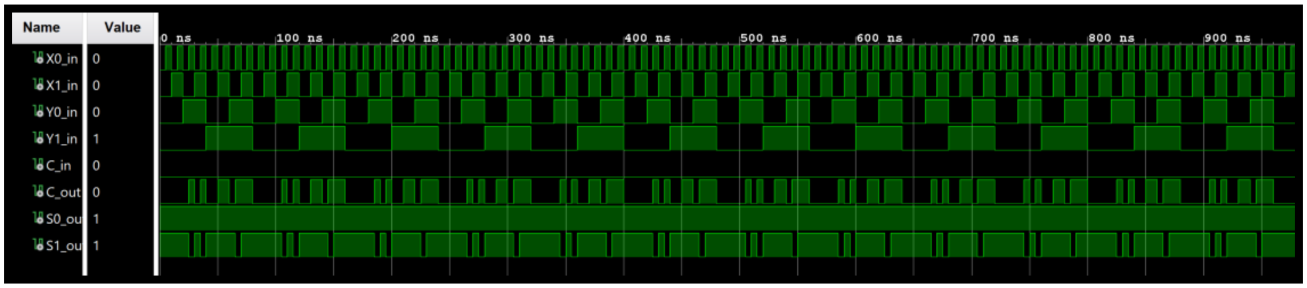
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity two_bit_addr_tb is
end two_bit_addr_tb;

architecture two_bit_addr_tb of two_bit_addr_tb is

    component two_bit_addr is
        Port (
               X0_in  : in STD_LOGIC;
               Y0_in  : in STD_LOGIC;
               X1_in  : in STD_LOGIC;
               Y1_in  : in STD_LOGIC;
               C_in   : in STD_LOGIC;
               C_out  : out STD_LOGIC;
               S0_out : out STD_LOGIC;
               S1_out : out STD_LOGIC
             );
    end component;

    signal X0_in  : std_logic := '0';
    signal X1_in  : std_logic := '0';
    signal Y0_in  : std_logic := '0';
    signal Y1_in  : std_logic := '0';
    signal C_in   : std_logic := '0';
    signal C_out  : std_logic;
    signal S0_out : std_logic;
    signal S1_out : std_logic;
```



```
begin

    UUT : two_bit_addr
    port map (
                X0_in => X0_in, X1_in => X1_in, Y0_in => Y0_in, Y1_in => Y1_in, C_in => C_in,
                C_out => C_out, S0_out => S0_out, S1_out => S1_out
              );

    X0_in <= NOT X0_in after 5 ns;
    X1_in <= NOT X1_in after 10 ns;
    Y0_in <= NOT Y0_in after 20 ns;
    Y1_in <= NOT Y1_in after 40 ns;

end two_bit_addr_tb;
```



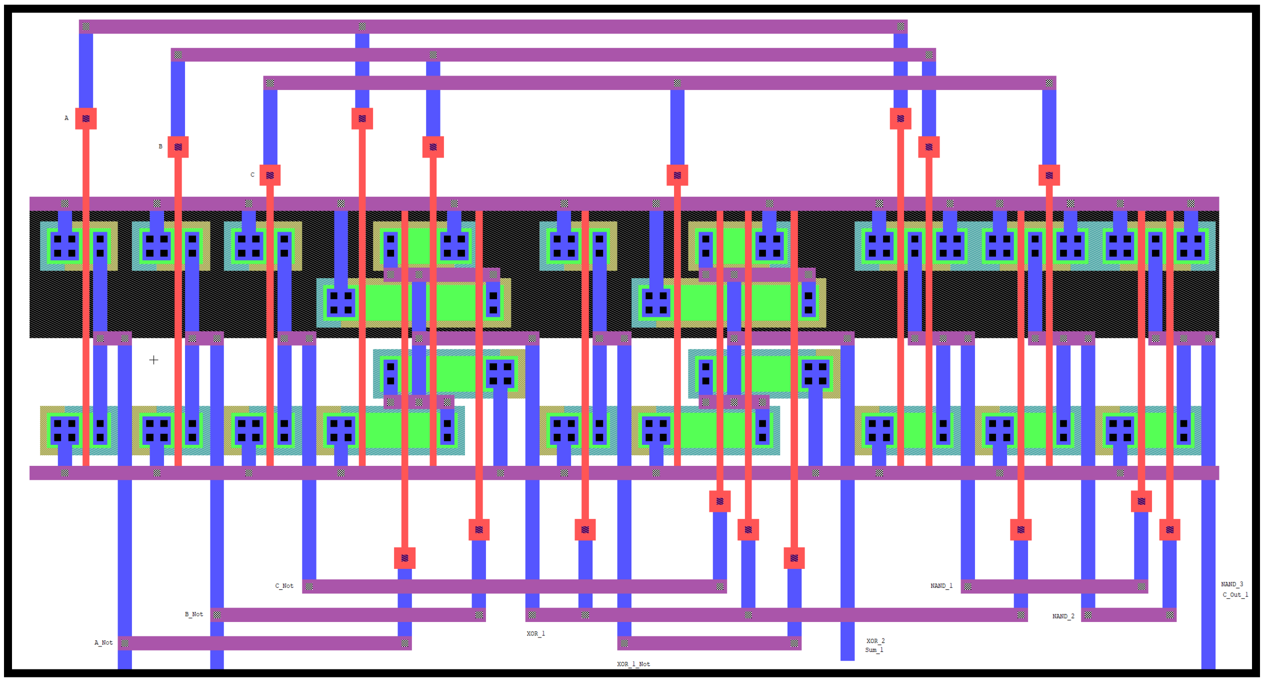
```
Design Sources (1)
  two_bit_addr(struct) (two_bit_adder.vhd) (2)
    u0 : one_bit_addr(struct) (two_bit_adder.vhd) (5)
      u0 : nand2(behaviour) (two_bit_adder.vhd)
      u1 : nand2(behaviour) (two_bit_adder.vhd)
      u2 : nand2(behaviour) (two_bit_adder.vhd)
      u3 : xor2(behaviour) (two_bit_adder.vhd)
      u4 : xor2(behaviour) (two_bit_adder.vhd)
    u1 : one_bit_addr(struct) (two_bit_adder.vhd) (5)
      u0 : nand2(behaviour) (two_bit_adder.vhd)
      u1 : nand2(behaviour) (two_bit_adder.vhd)
      u2 : nand2(behaviour) (two_bit_adder.vhd)
      u3 : xor2(behaviour) (two_bit_adder.vhd)
      u4 : xor2(behaviour) (two_bit_adder.vhd)
Constraints
Simulation Sources (1)
  sim_1 (1)
    two_bit_addr_tb(two_bit_addr_tb) (two_bit_addr_tb.vhd) (1)
      UUT : two_bit_addr(struct) (two_bit_adder.vhd) (2)
        u0 : one_bit_addr(struct) (two_bit_adder.vhd) (5)
        u1 : one_bit_addr(struct) (two_bit_adder.vhd) (5)
```



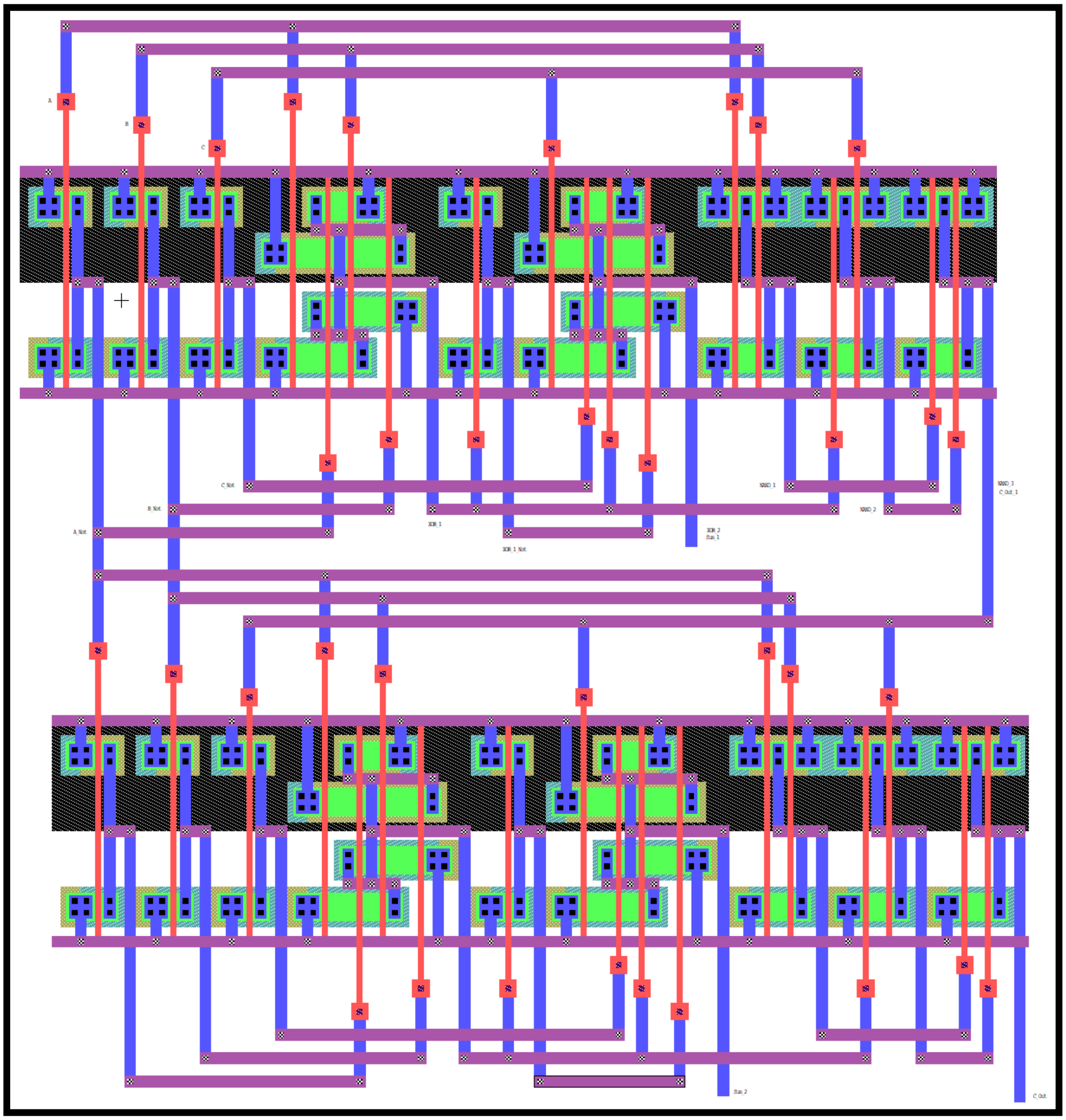
Name	Value
X0_in	0
X1_in	0
Y0_in	0
Y1_in	1
C_in	0
C_out	0
S0_ou	1
S1_ou	1

L-Edit Results

Here is the resulting L-Edit layout of a one-bit-full-adder:

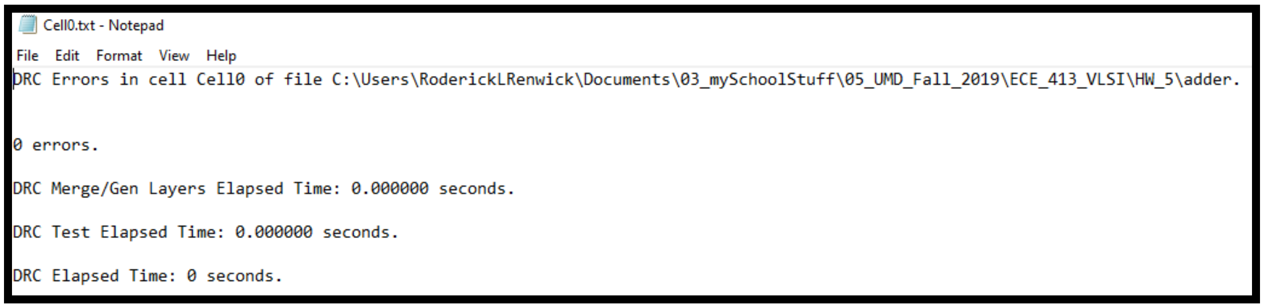


Here is the resulting L-Edit layout of a two-bit-full-adder, after connecting two one-bit-full-adders together as laid out in the design steps above:

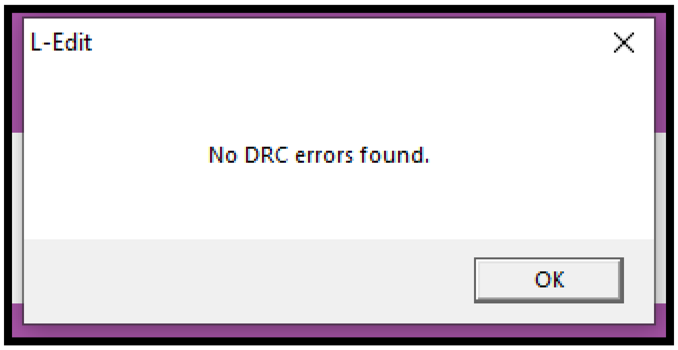


DRC Verification

Here are the resulting DRC check verification messages and file:



```
DRC Errors in cell Cell0 of file C:\Users\RoderickLRenwick\Documents\03_mySchoolStuff\05_UMD_Fall_2019\ECE_413_VLSI\HW_5\adder.


0 errors.

DRC Merge/Gen Layers Elapsed Time: 0.000000 seconds.

DRC Test Elapsed Time: 0.000000 seconds.

DRC Elapsed Time: 0 seconds.
```



No DRC errors found.

Conclusion

The homework was successfully verified and validated from both the software implementation in Valvado, and the hardware layout design in L-Edit. Overall, I learned a lot from this lab. It really challenged me to think carefully and plan ahead for the layout from start to finish.