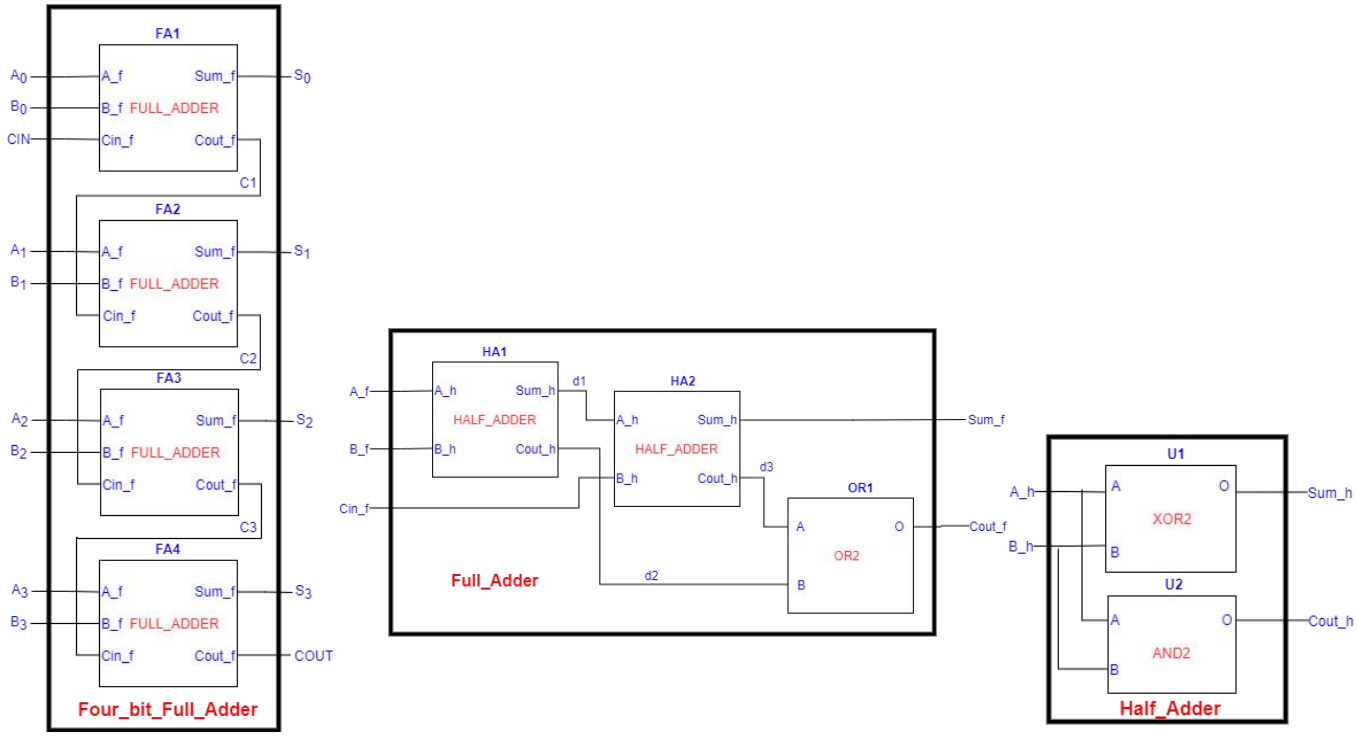
11911214 杨鸿嘉

1. Structure



2. VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Four_bit_Full_Adder is
      Port ( A0,A1,A2,A3,B0,B1,B2,B3,CIN : in std_logic;
             S0,S1,S2,S3,COUT : out std_logic);
end Four_bit_Full_Adder;
architecture STRUCTURE of Four_bit_Full_Adder is
   component FULL_ADDER is
      port (A_f, B_f ,Cin_f: in std_logic; Sum_f, Cout_f : out std_logic);
   end component FULL_ADDER;
signal c1, c2, c3 : std_logic;
begin
   FA1: FULL_ADDER port map (A_f=>A0, B_f=>B0, Cin_f=>CIN,Sum_f=>S0, Cout_F=>C1);
   FA2: FULL_ADDER port map (A_f=>A1, B_f=>B1, Cin_f=>C1,Sum_f=>S1, Cout_f=>C2);
   FA3: FULL_ADDER port map (A_f=>A2, B_f=>B2, Cin_f=>C2,Sum_f=>S2, Cout_f=>C3);
   FA4: FULL_ADDER port map (A_f=>A3, B_f=>B3, Cin_f=>C3,Sum_f=>S3, Cout_f=>COUT);
end STRUCTURE;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Full_Adder is
   port (A_f, B_f, Cin_f : in std_logic;
         Sum_f, Cout_f : out std_logic );
end entity Full_Adder;
architecture STRUCTURE of Full_Adder is
   component HALF_ADDER is
      port (A_h, B_h : in std_logic;
            Sum_h, Cout_h : out std_logic);
   end component HALF_ADDER;
   component OR2 is
      port (A, B : in std_logic; O: out std_logic);
   end component OR2;
signal d1, d2, d3 : std_logic;
begin
   HA1: HALF_ADDER port map (A_h=>A_f, B_h=>B_f, Sum_h=>d1, Cout_h=>d2);
   HA2: HALF_ADDER port map (A_h=>d1, B_h=>Cin_f, Sum_h=>Sum_f, Cout_h=>d3);
   OR1: OR2 port map (A=>d3, B=>d2, O=>Cout_f);
end STRUCTURE;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Half_Adder is
   port (A_h, B_h : in std_logic;
         Sum_h,Cout_h: out std_logic);
end entity Half_Adder;
architecture STRUCTURE of Half_Adder is
   component XOR2 is
      port (A, B: in std_logic; O: out std_logic);
   end component XOR2;
   component AND2 is
      port (A, B : in std_logic; O: out std_logic);
   end component AND2;
begin
   U1: XOR2 port map (A => A_h, B => B_h, O => Sum_h);
   U2: AND2 port map (A => A_h, B => B_h, O => Cout_h);
end STRUCTURE;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR2 is
port (A, B: in std_logic; O: out
std_logic );
end entity OR2;
architecture BHV of OR2 is
begin
   O <= A or B;-- after 10 ns;
end BHV;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND2 is
port (A, B: in std_logic; O: out
std_logic );
end entity AND2;
architecture dataflow of AND2 is
begin
   O <= A and B;
end dataflow;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR2 is
port (A, B: in std_logic; O: out
std_logic );
end entity XOR2;
architecture BHV of XOR2 is
begin
   O <= A xor B; --after 10 ns;
end BHV;
```

In the VHDL code we **set no delay(using delta delay) for the gate to simplify our simulation. We can easily verify the result for there is no obvious delay in waveform.** Further we will add the gate delay and see the result again.

3. Testbench code

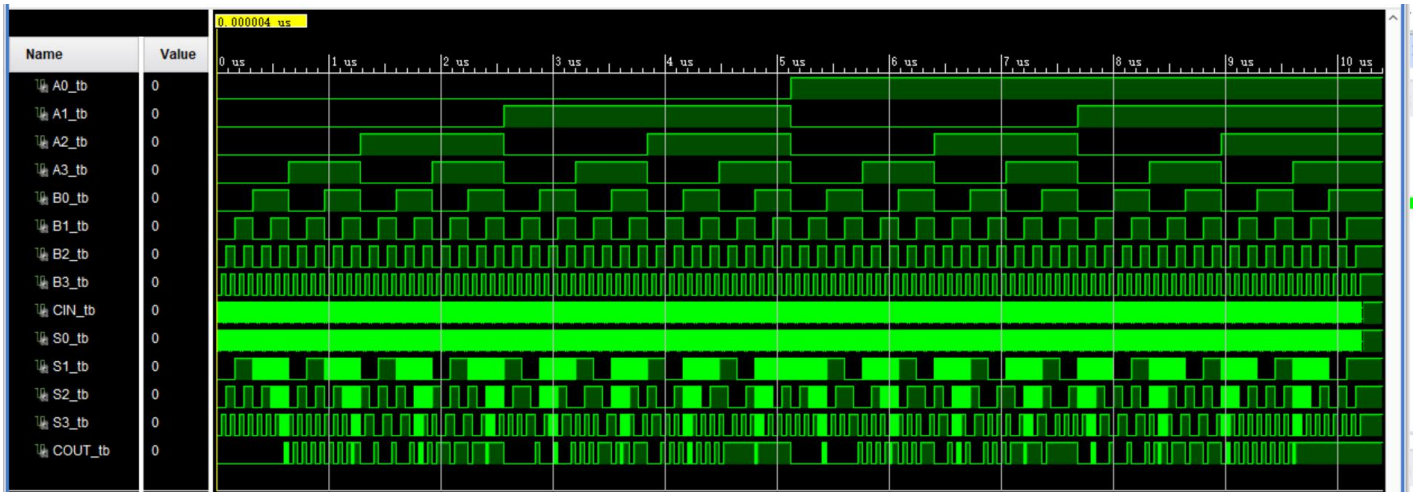
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity Four_bit_Full_Adder_tb is
--    Port ( );
end Four_bit_Full_Adder_tb;
architecture Behavioral of Four_bit_Full_Adder_tb is
  COMPONENT four_bit_full_adder
  PORT(A0,A1,A2,A3,B0,B1,B2,B3,CIN: in std_logic;
       S0,S1,S2,S3,COUT: out std_logic);
  END COMPONENT;
  signal A0_tb,A1_tb,A2_tb,A3_tb,B0_tb,B1_tb,B2_tb,B3_tb,CIN_tb: std_logic;
  signal S0_tb,S1_tb,S2_tb,S3_tb,COUT_tb: std_logic;
begin
  uut: four_bit_full_adder PORT MAP ( A0=>A0_tb,A1=>A1_tb,A2=>A2_tb,A3=>A3_tb,
                                      B0=>B0_tb,B1=>B1_tb,B2=>B2_tb,B3=>B3_tb,CIN=>CIN_tb,
                                     S0=>S0_tb,S1=>S1_tb,S2=>S2_tb,S3=>S3_tb,COUT=>COUT_tb);
tb: process is
    constant PERIOD: time := 20 ns;
    constant n : integer := 9;
    variable i : integer;
  begin
  for i in 0 to 2**n - 1 loop
    (A0_tb,A1_tb,A2_tb,A3_tb,B0_tb,B1_tb,B2_tb,B3_tb,CIN_tb) <= to_unsigned(i, n);
     wait for PERIOD;
  end loop;
  wait;
end process;

end Behavioral;
```

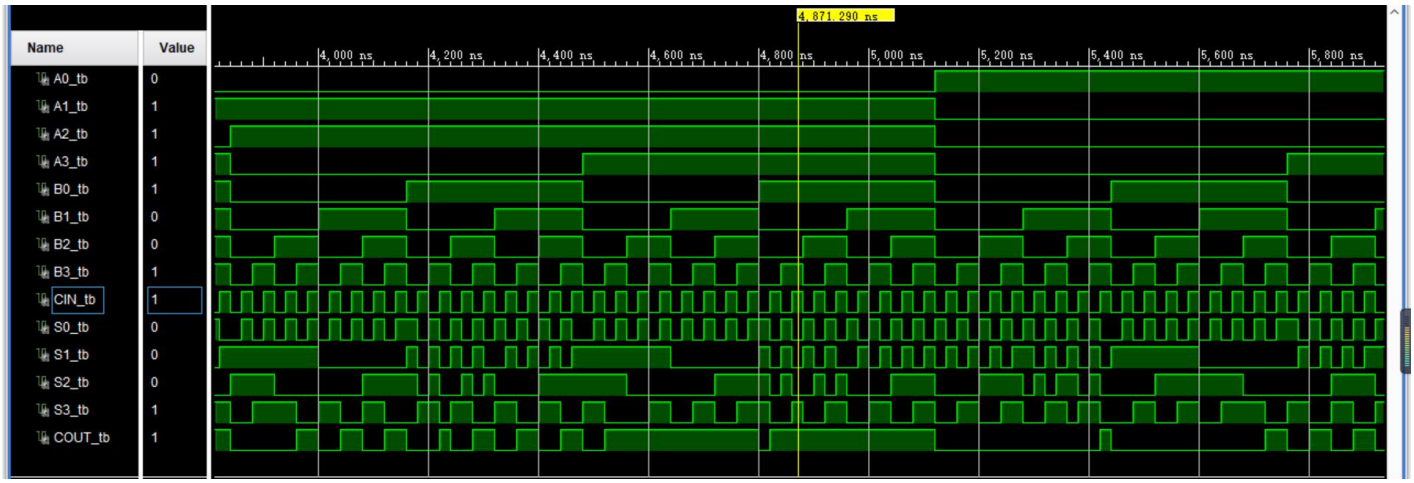
4. Simulation result

Behavior result:



Behavior simulation

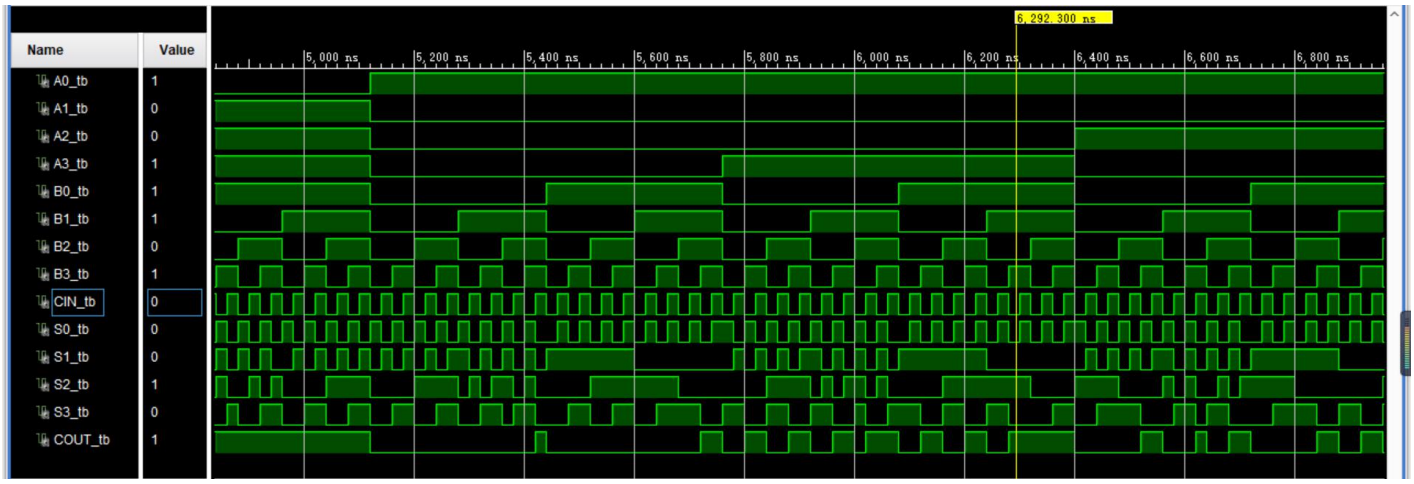
The result is correct, **for the logic expression for the output of 4-bit full-adder is complex to write out, we do not write the 'assert' and 'report' in testbench**. We can choose several output to verify, for example:



A:1110 B:1001 Cin:1

Output Sum:1000 Cout:1 which is correct

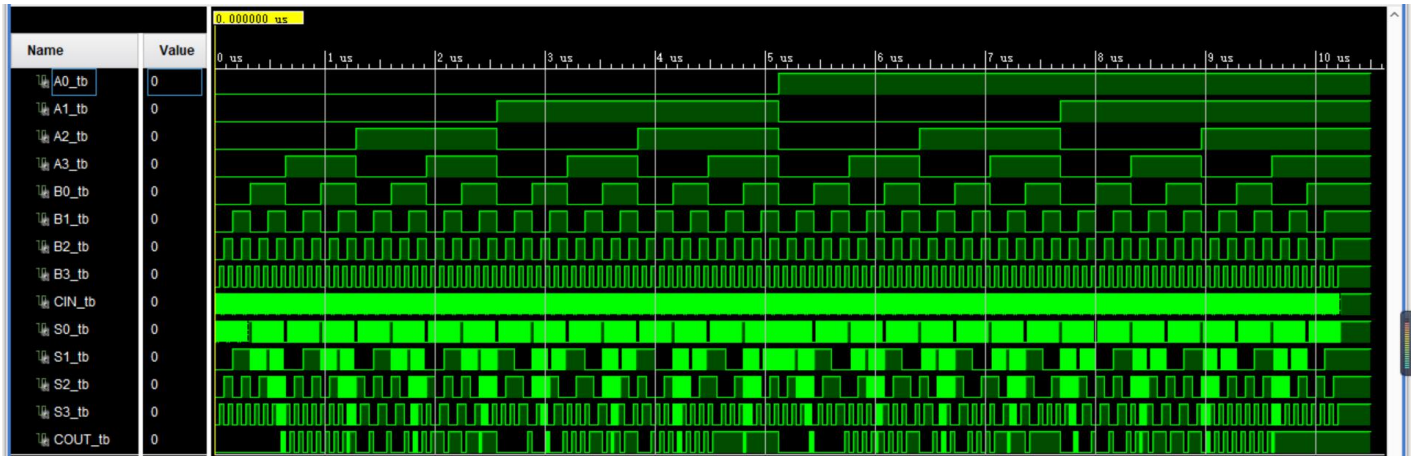
Also, another verify:



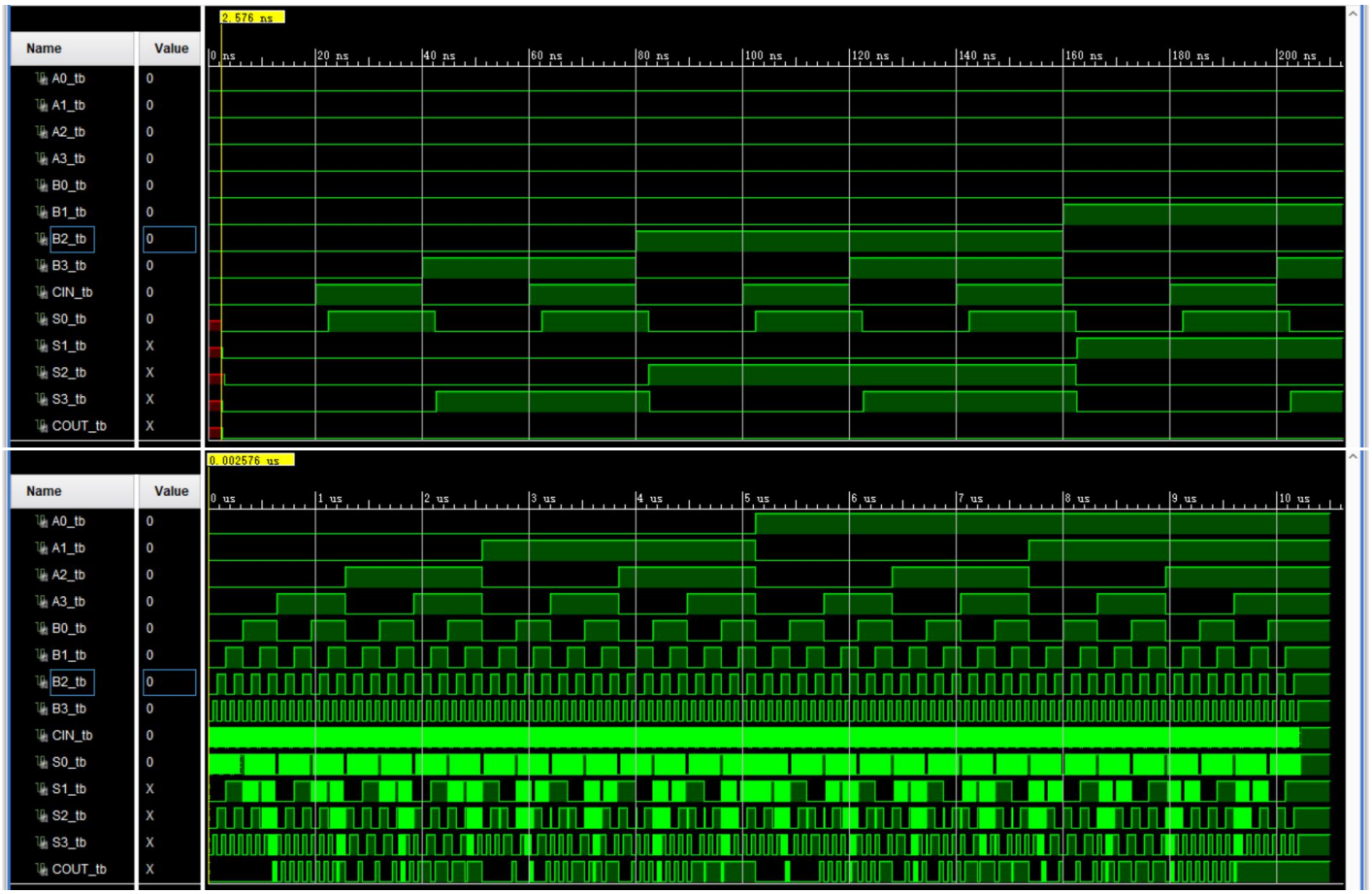
A:1001 B:1011 Cin:0

Output Sum:0100 Cout:1 which is also correct

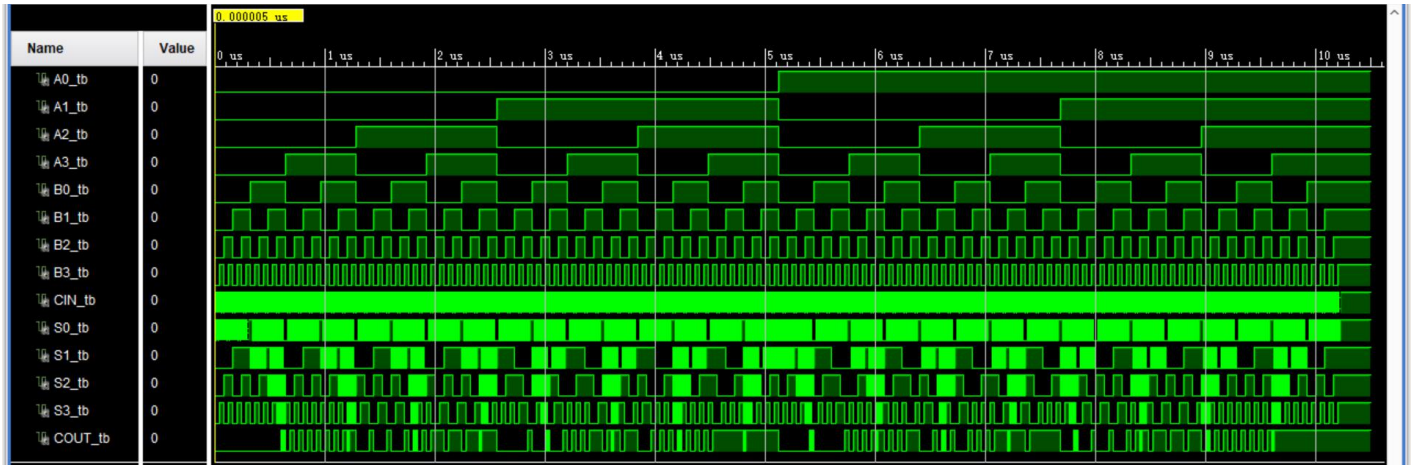
Post-synthesis functional simulation:



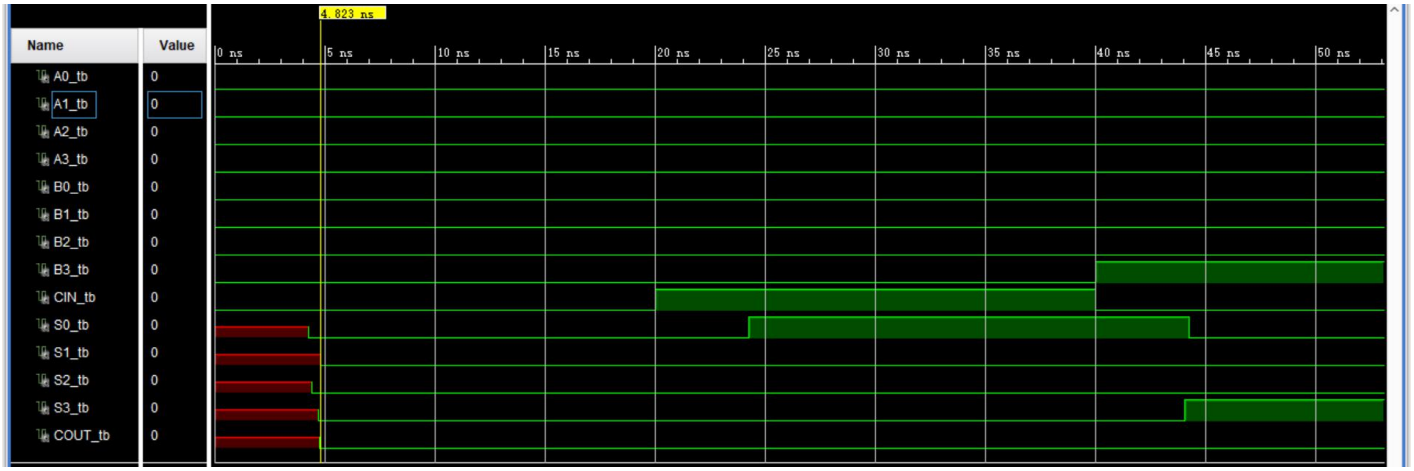
Post-synthesis timing simulation:

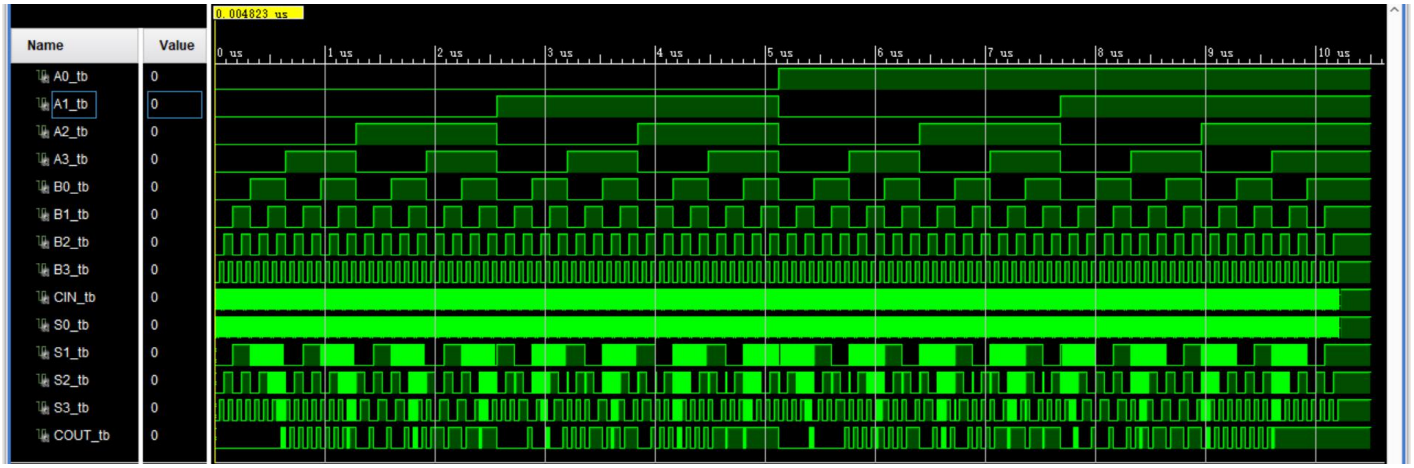


Post-Implementation functional simulation:



Post-Implementation timing simulation:





Further discussion: add the delay of the gate

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR2 is
port (A, B: in std_logic; O: out std_logic );
end entity OR2;
architecture BHV of OR2 is
begin
  O <= A or B after 10 ns;
end BHV;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND2 is
port (A, B: in std_logic; O: out std_logic );
end entity AND2;
architecture dataflow of AND2 is
begin
  O <= A and B;
end dataflow;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR2 is
port (A, B: in std_logic; O: out std_logic );
end entity XOR2;
architecture BHV of XOR2 is
begin
  O <= A xor B after 10 ns;
end BHV;
```

Behavior simulation result: