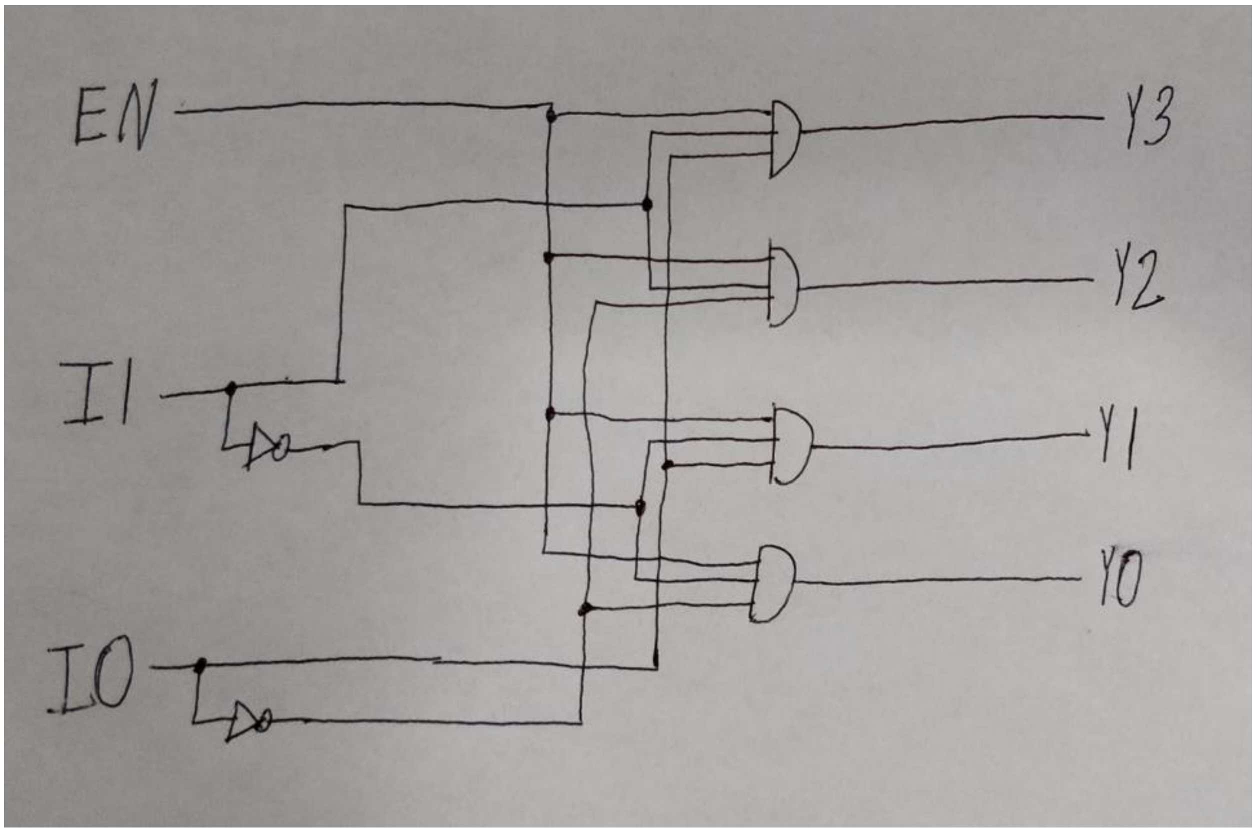
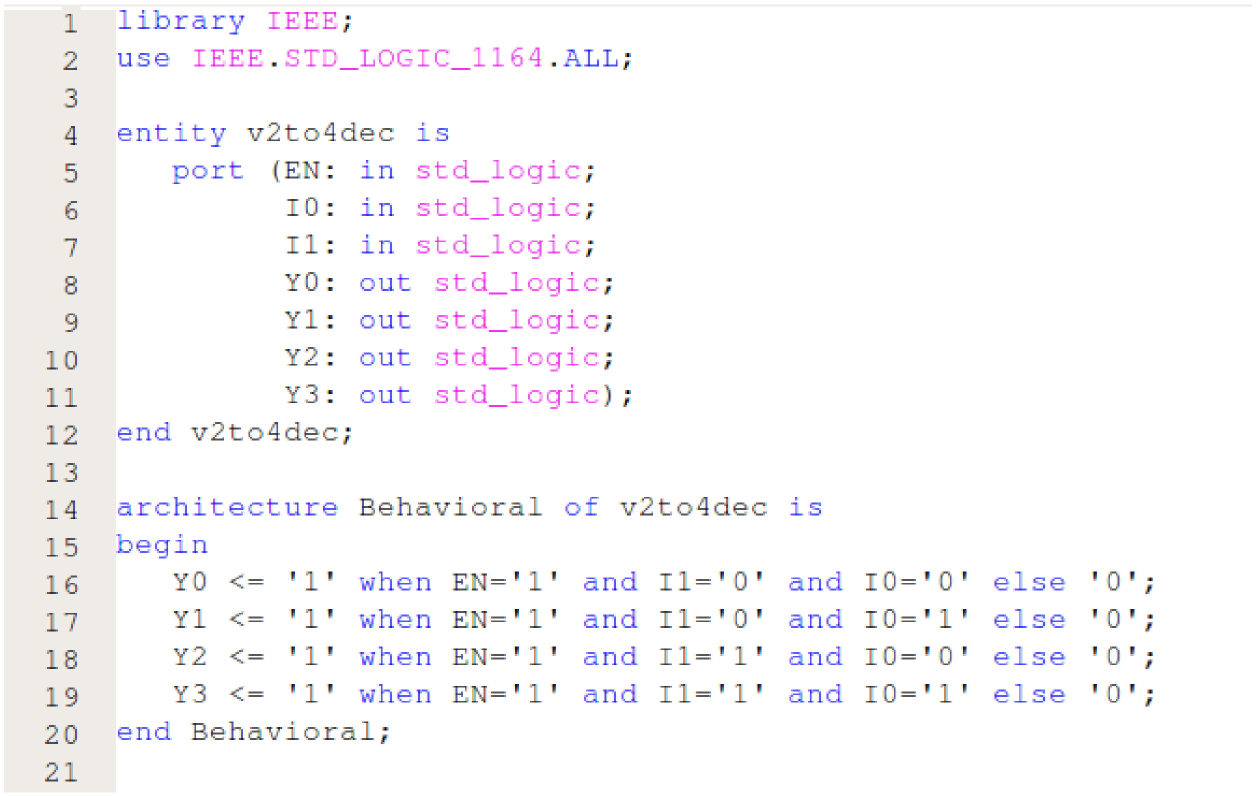
Task 1: 2-to-4 Decoder

The 2x4 decoder is a decoder that takes in 2 inputs and outputs 1 of 4 different outputs. Deciding which one to output in based on the 2 inputs and the logic written in the function. In this example of a 2x4 decoder, our inputs are I0 and I1, and our outputs are Y0, Y1, Y2, and Y3. The logic of the module itself follows as such:

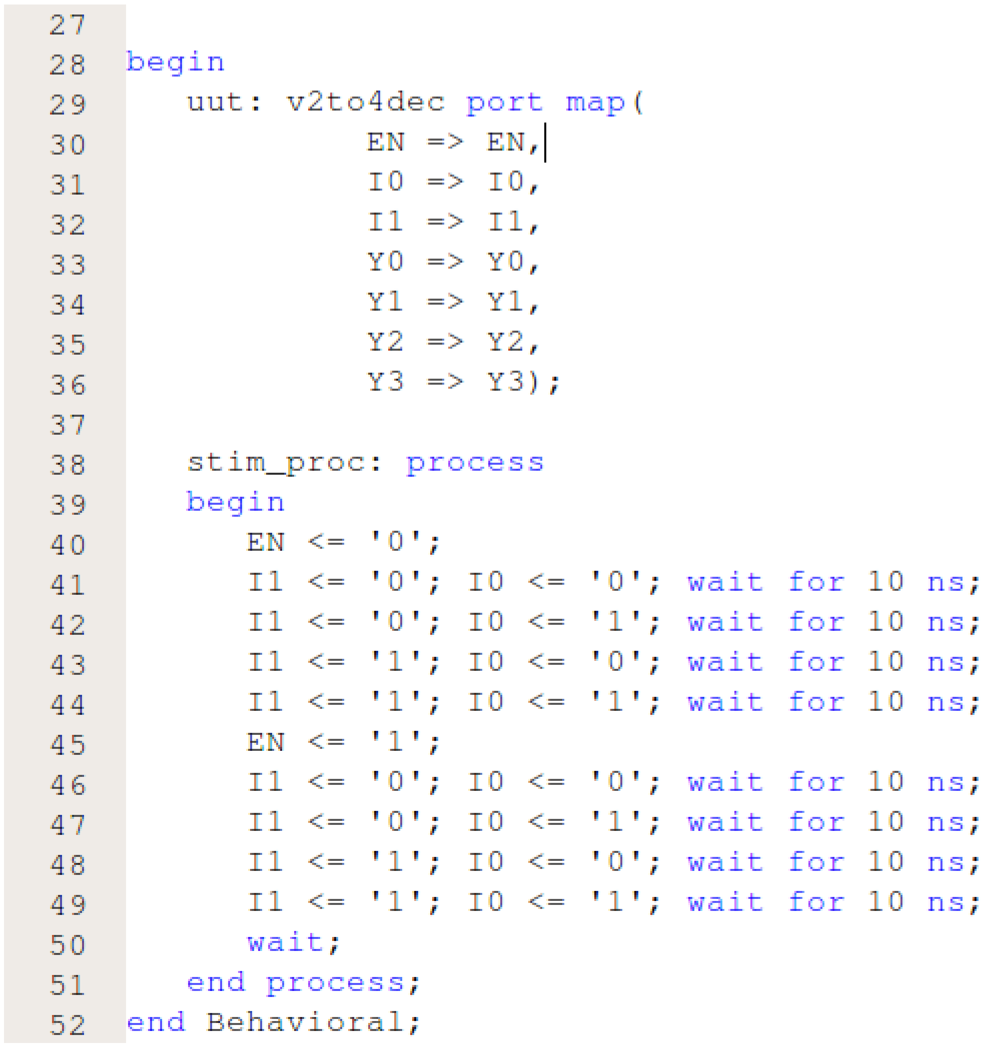
Inputs			**Outputs**			
EN	**I1**	**I0**	**Y0**	**Y1**	**Y2**	**Y3**
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



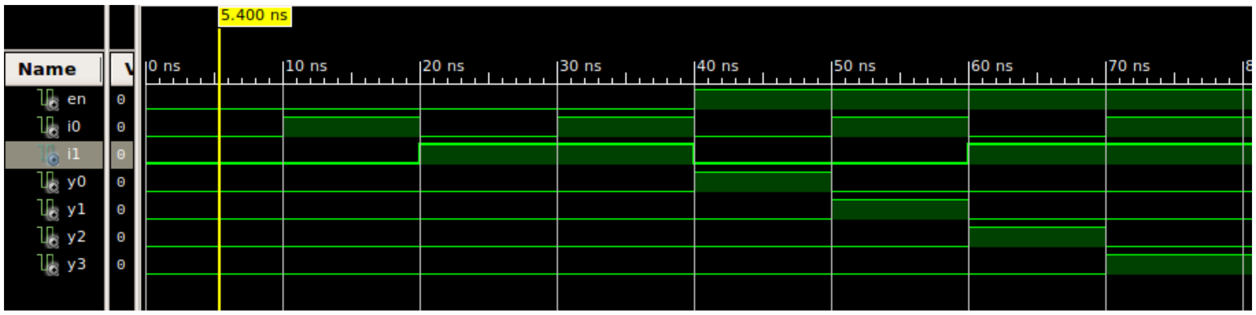


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity v2to4dec is
   port (EN: in std_logic;
         I0: in std_logic;
         I1: in std_logic;
         Y0: out std_logic;
         Y1: out std_logic;
         Y2: out std_logic;
         Y3: out std_logic);
end v2to4dec;

architecture Behavioral of v2to4dec is
begin
   Y0 <= '1' when EN='1' and I1='0' and I0='0' else '0';
   Y1 <= '1' when EN='1' and I1='0' and I0='1' else '0';
   Y2 <= '1' when EN='1' and I1='1' and I0='0' else '0';
   Y3 <= '1' when EN='1' and I1='1' and I0='1' else '0';
end Behavioral;

```

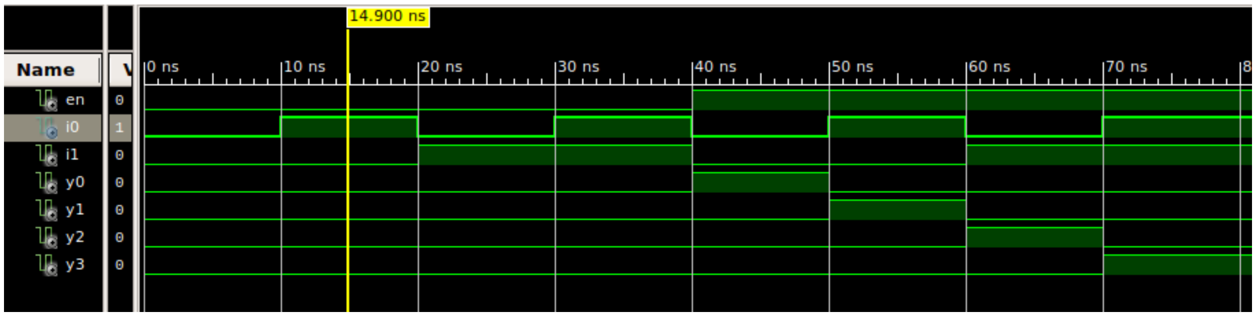
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity testbench is
end testbench;

architecture Behavioral of testbench is
   component v2to4dec
   port (EN: in std_logic;
         I0: in std_logic;
         I1: in std_logic;
         Y0: out std_logic;
         Y1: out std_logic;
         Y2: out std_logic;
         Y3: out std_logic);
   end component;

   --inputs
   signal EN : std_logic := '0';
   signal I0 : std_logic := '0';
   signal I1 : std_logic := '0';
   --outputs
   signal Y0 : std_logic;
   signal Y1 : std_logic;
   signal Y2 : std_logic;
   signal Y3 : std_logic;

```



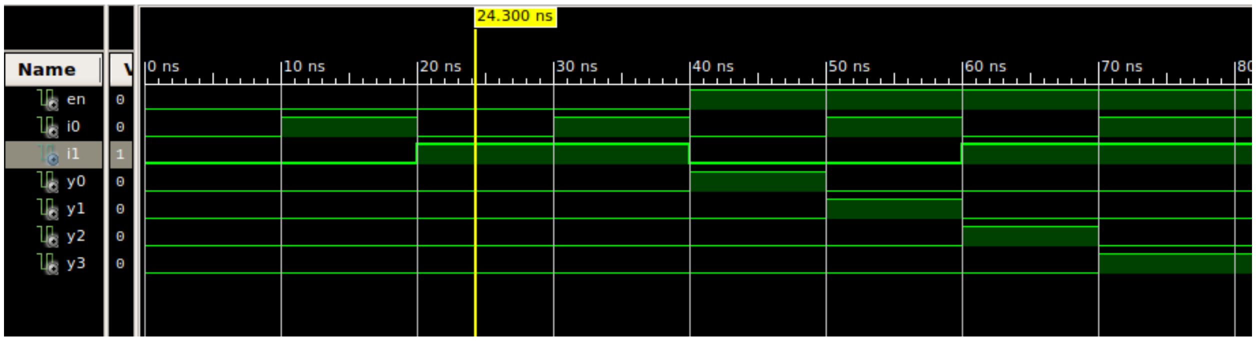
```

begin
   uut: v2to4dec port map(
            EN => EN,
            I0 => I0,
            I1 => I1,
            Y0 => Y0,
            Y1 => Y1,
            Y2 => Y2,
            Y3 => Y3);

   stim_proc: process
   begin
      EN <= '0';
      I1 <= '0'; I0 <= '0'; wait for 10 ns;
      I1 <= '0'; I0 <= '1'; wait for 10 ns;
      I1 <= '1'; I0 <= '0'; wait for 10 ns;
      I1 <= '1'; I0 <= '1'; wait for 10 ns;
      EN <= '1';
      I1 <= '0'; I0 <= '0'; wait for 10 ns;
      I1 <= '0'; I0 <= '1'; wait for 10 ns;
      I1 <= '1'; I0 <= '0'; wait for 10 ns;
      I1 <= '1'; I0 <= '1'; wait for 10 ns;
      wait;
   end process;
end Behavioral;
```



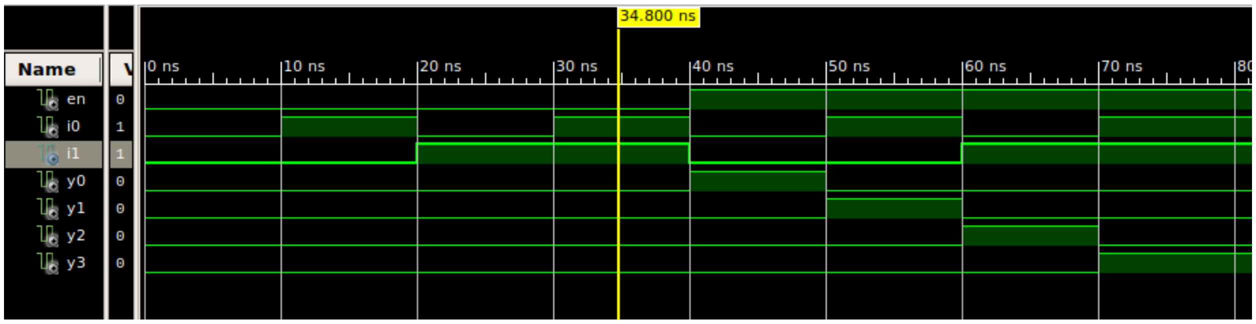
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



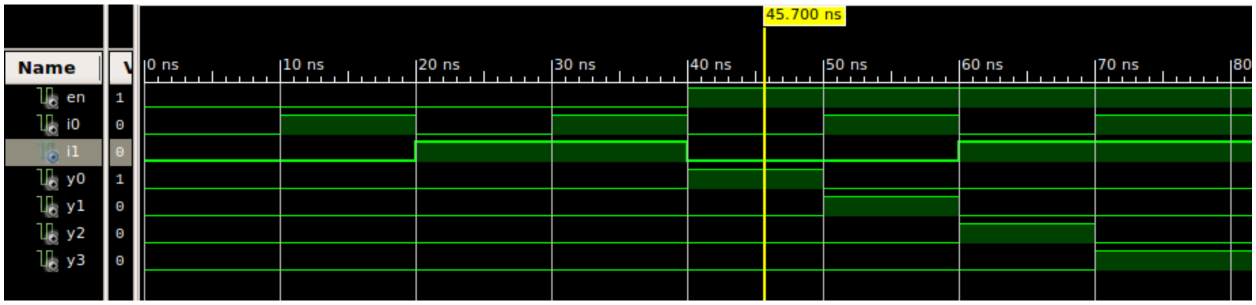
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



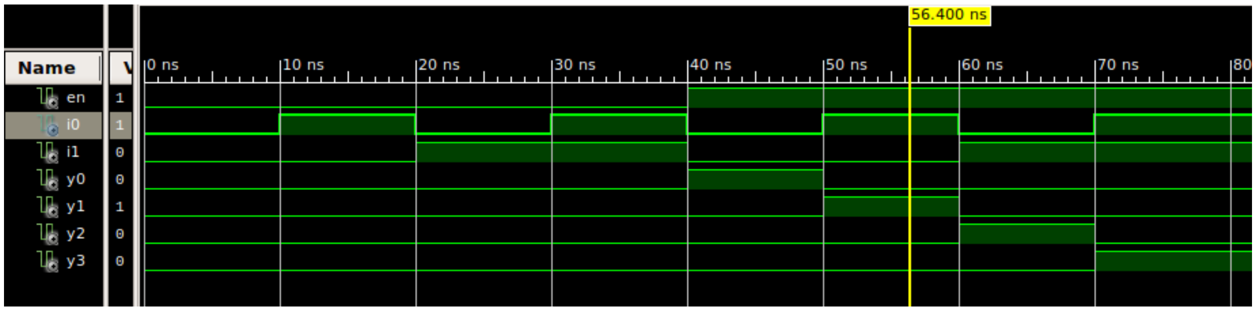
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



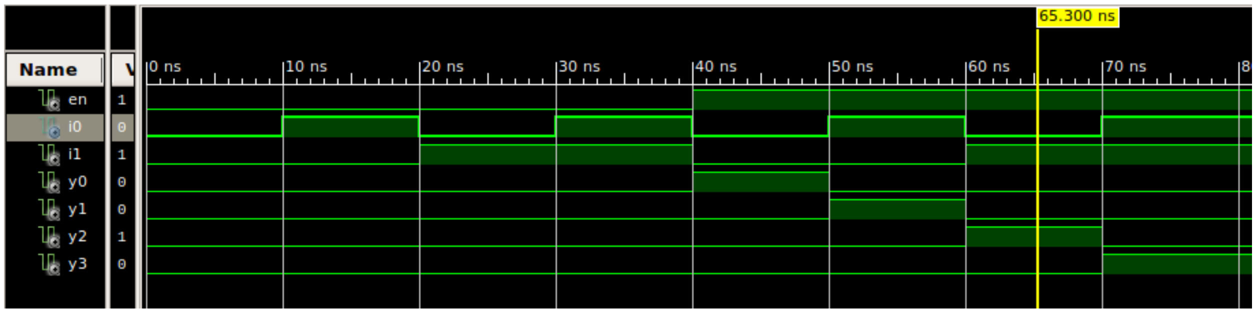
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



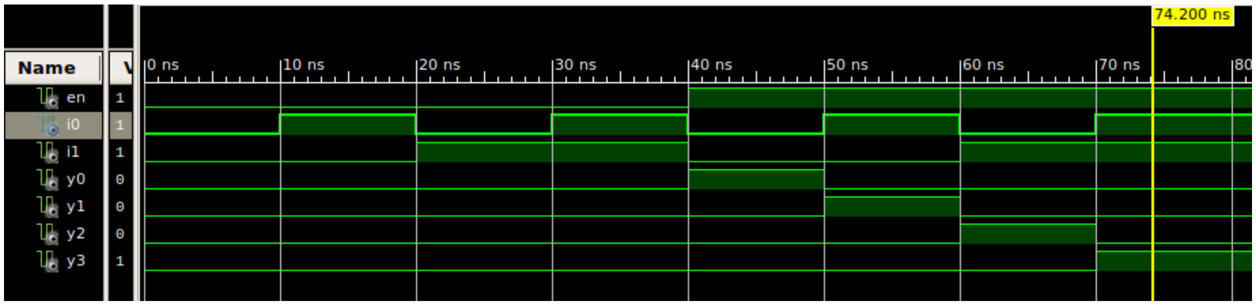
This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since I0 and I1 are negated, Y0 should be asserted which can be seen above.



This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since I0 is asserted and I1 is negated, Y1 should be asserted which can be seen above.



This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since I0 is negated and I1 is asserted, Y2 should be asserted which can be seen above.

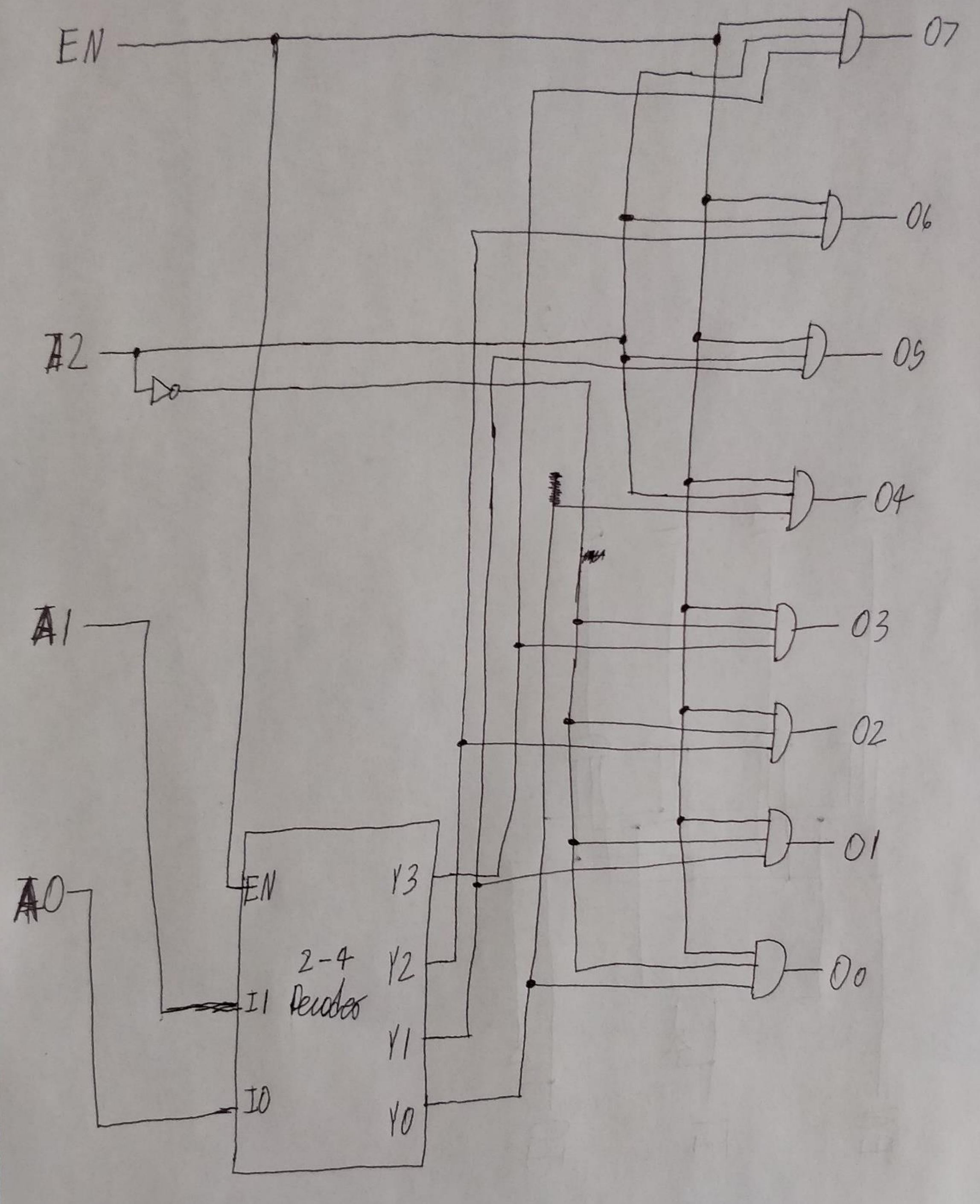


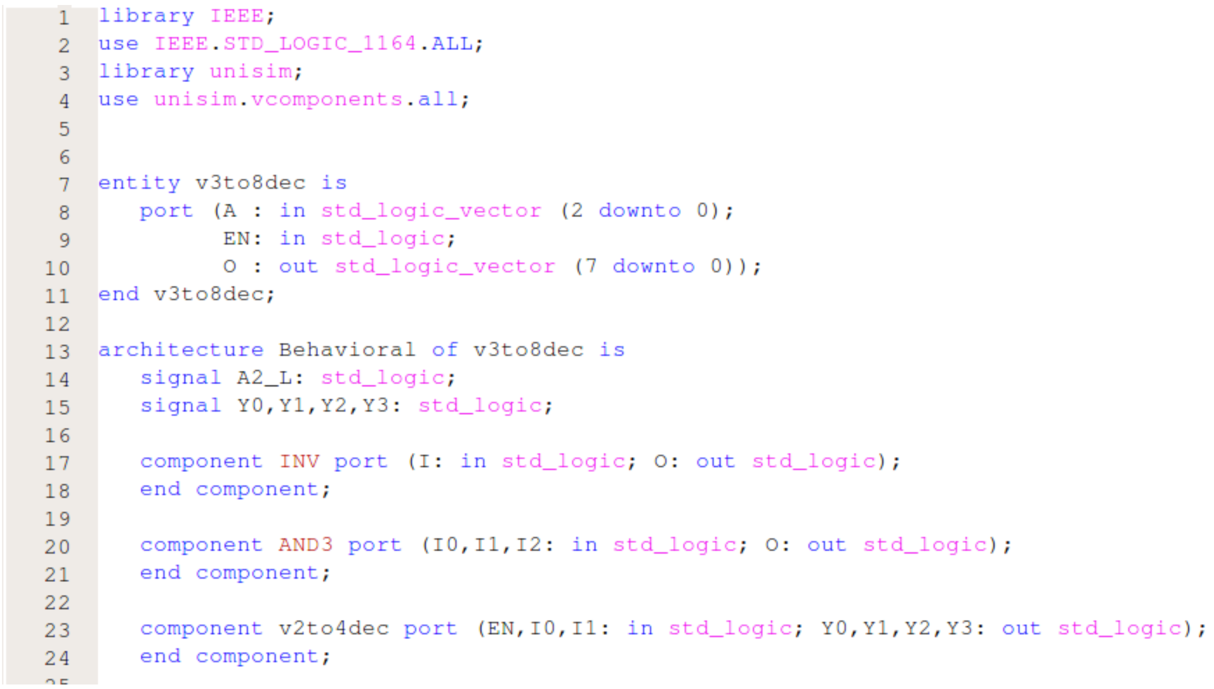
This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since I0 and I1 are asserted, Y3 should be asserted which can be seen above.

Task 2: 3-to-8 Decoder

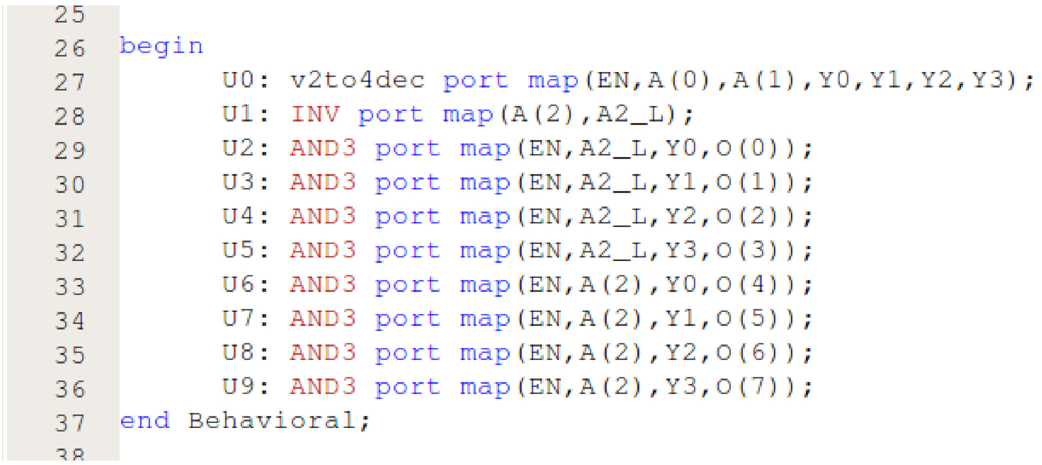
The 3x8 decoder is a decoder that takes in 3 inputs and outputs 1 of 8 different outputs. Deciding which one to output in based on the 3 inputs and the logic written in the function. In this example of a 3x8 decoder, our inputs are A0, A1, and A2, and our outputs are O0, O1, O2, O3, O4, O5, O6, and O7. The logic of the module itself follows as such:

Inputs				Outputs							
EN	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

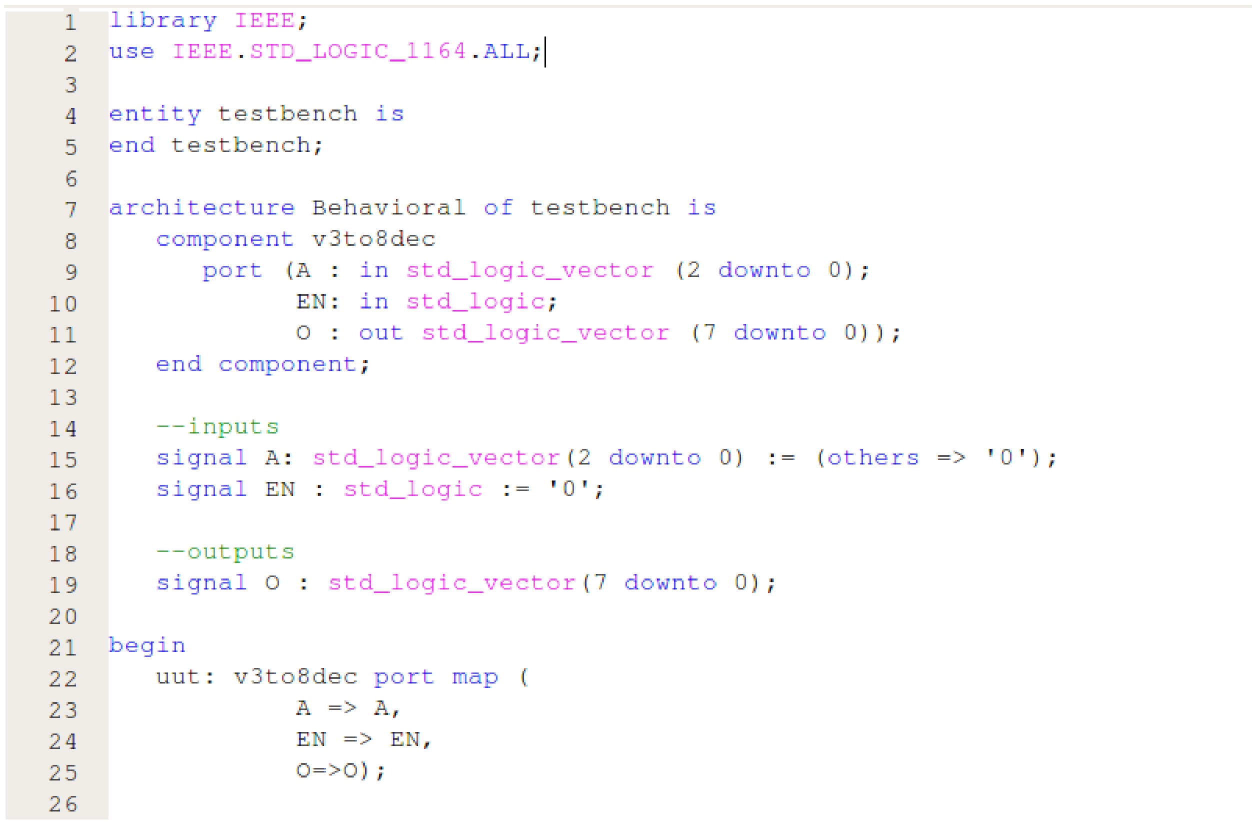

EN
A2
A1
A0
EN
I1
I0
2-4 Decoder
Y3
Y2
Y1
Y0
O7
O6
O5
O4
O3
O2
O1
O0



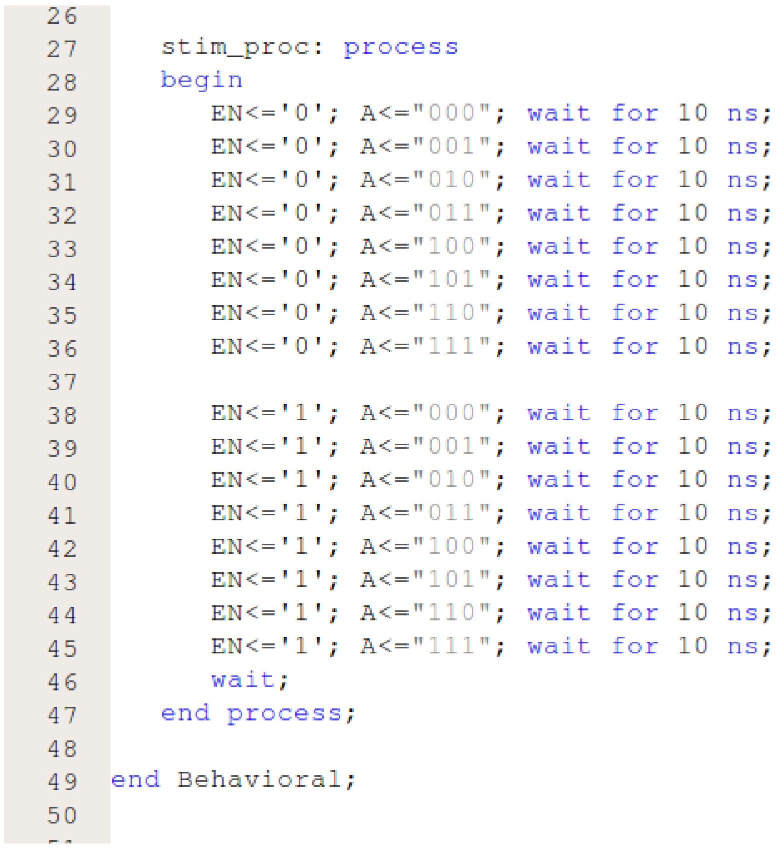
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library unisim;
use unisim.vcomponents.all;


entity v3to8dec is
   port (A : in std_logic_vector (2 downto 0);
         EN: in std_logic;
         O : out std_logic_vector (7 downto 0));
end v3to8dec;

architecture Behavioral of v3to8dec is
   signal A2_L: std_logic;
   signal Y0,Y1,Y2,Y3: std_logic;

   component INV port (I: in std_logic; O: out std_logic);
   end component;

   component AND3 port (I0,I1,I2: in std_logic; O: out std_logic);
   end component;

   component v2to4dec port (EN,I0,I1: in std_logic; Y0,Y1,Y2,Y3: out std_logic);
   end component;
```



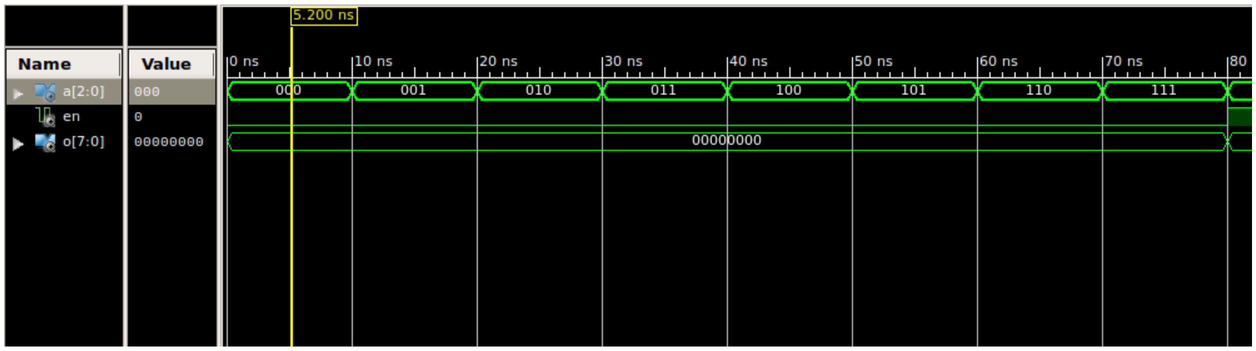
```vhdl

begin
      U0: v2to4dec port map(EN,A(0),A(1),Y0,Y1,Y2,Y3);
      U1: INV port map(A(2),A2_L);
      U2: AND3 port map(EN,A2_L,Y0,O(0));
      U3: AND3 port map(EN,A2_L,Y1,O(1));
      U4: AND3 port map(EN,A2_L,Y2,O(2));
      U5: AND3 port map(EN,A2_L,Y3,O(3));
      U6: AND3 port map(EN,A(2),Y0,O(4));
      U7: AND3 port map(EN,A(2),Y1,O(5));
      U8: AND3 port map(EN,A(2),Y2,O(6));
      U9: AND3 port map(EN,A(2),Y3,O(7));
end Behavioral;
```



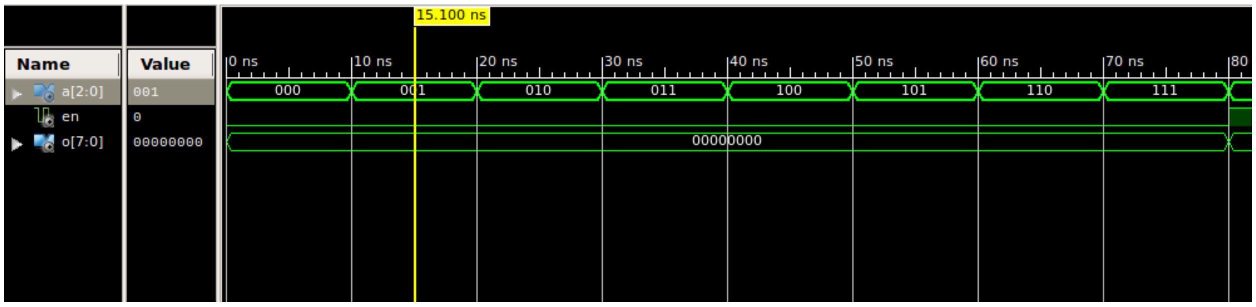
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity testbench is
end testbench;

architecture Behavioral of testbench is
   component v3to8dec
      port (A : in std_logic_vector (2 downto 0);
            EN: in std_logic;
            O : out std_logic_vector (7 downto 0));
   end component;

   --inputs
   signal A: std_logic_vector(2 downto 0) := (others => '0');
   signal EN : std_logic := '0';

   --outputs
   signal O : std_logic_vector(7 downto 0);

begin
   uut: v3to8dec port map (
            A => A,
            EN => EN,
            O=>O);

```



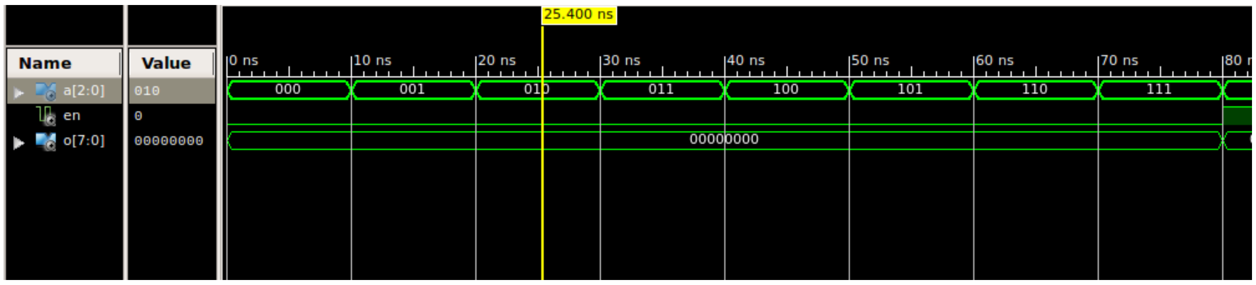
```vhdl

   stim_proc: process
   begin
      EN<='0'; A<="000"; wait for 10 ns;
      EN<='0'; A<="001"; wait for 10 ns;
      EN<='0'; A<="010"; wait for 10 ns;
      EN<='0'; A<="011"; wait for 10 ns;
      EN<='0'; A<="100"; wait for 10 ns;
      EN<='0'; A<="101"; wait for 10 ns;
      EN<='0'; A<="110"; wait for 10 ns;
      EN<='0'; A<="111"; wait for 10 ns;

      EN<='1'; A<="000"; wait for 10 ns;
      EN<='1'; A<="001"; wait for 10 ns;
      EN<='1'; A<="010"; wait for 10 ns;
      EN<='1'; A<="011"; wait for 10 ns;
      EN<='1'; A<="100"; wait for 10 ns;
      EN<='1'; A<="101"; wait for 10 ns;
      EN<='1'; A<="110"; wait for 10 ns;
      EN<='1'; A<="111"; wait for 10 ns;
      wait;
   end process;

end Behavioral;

```



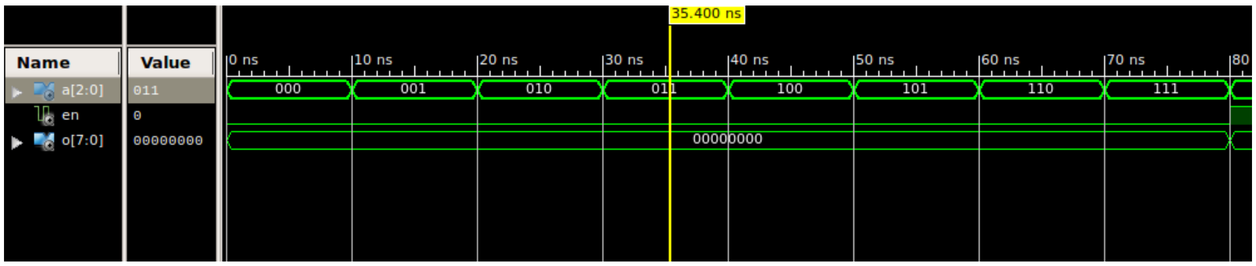
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



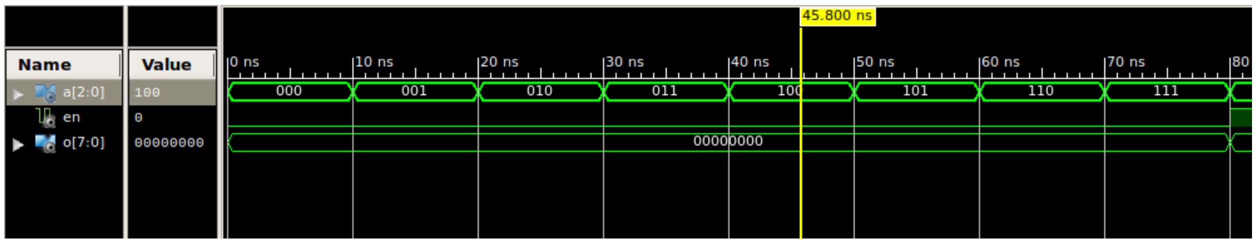
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



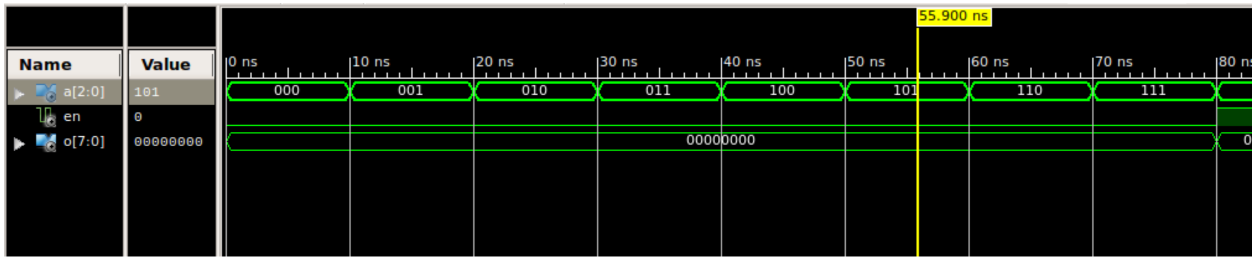
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



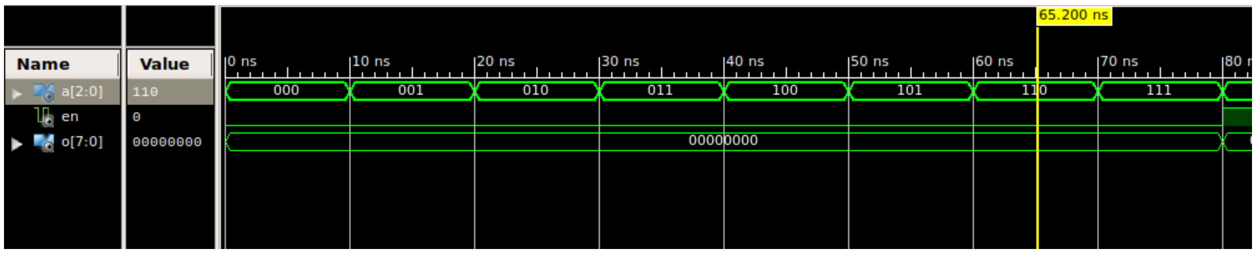
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



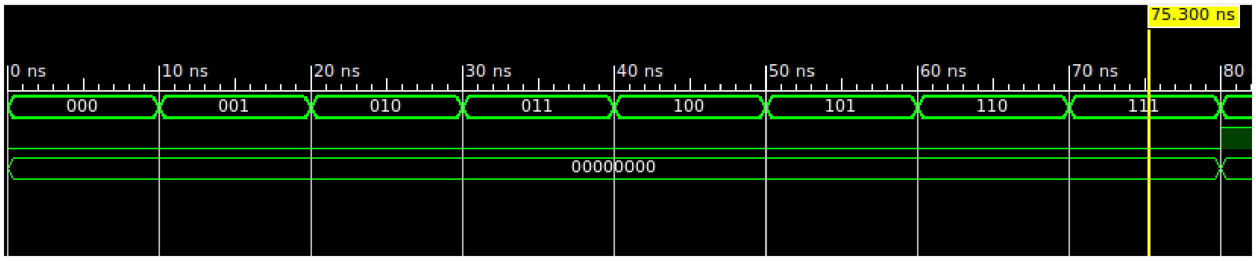
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



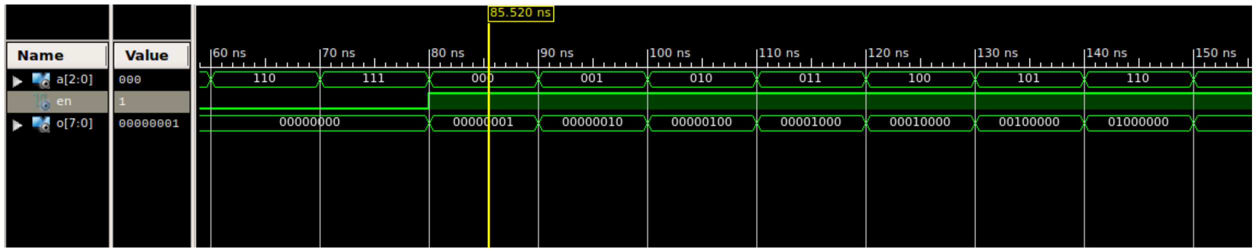
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



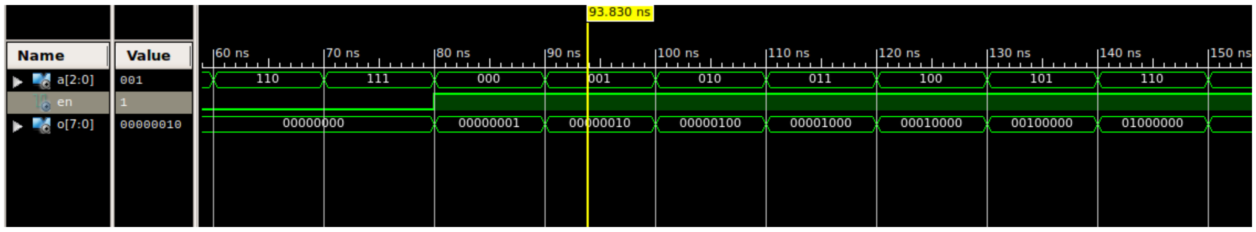
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



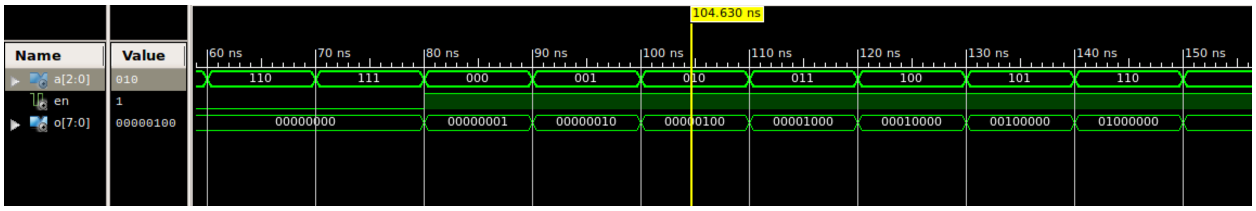
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



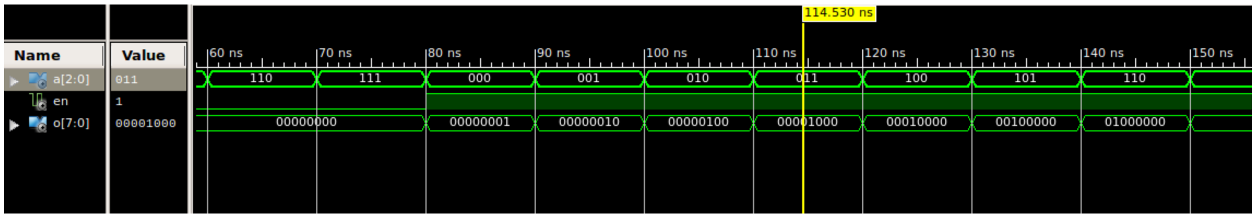
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0, A1, and A2 are negated, O0 should be asserted, which can be seen above.



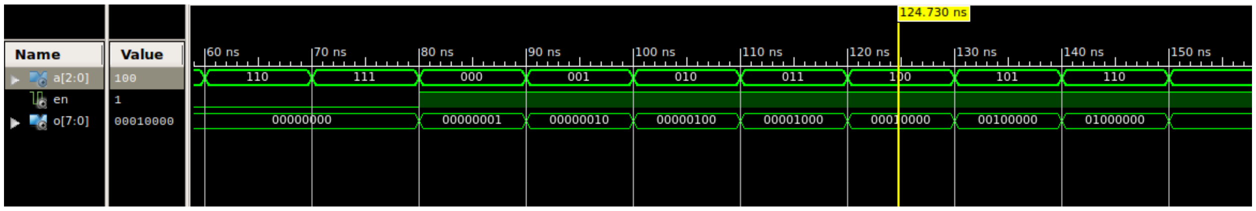
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0 is asserted and A1 and A2 are negated, O1 should be asserted, which can be seen above.



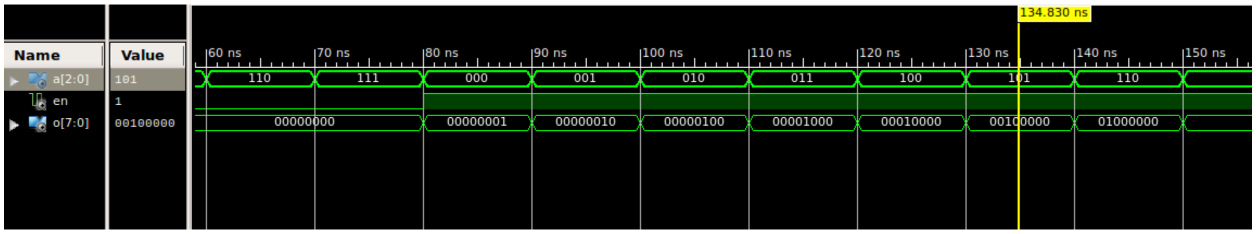
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A1 is asserted and A0 and A2 are negated, O2 should be asserted, which can be seen above.



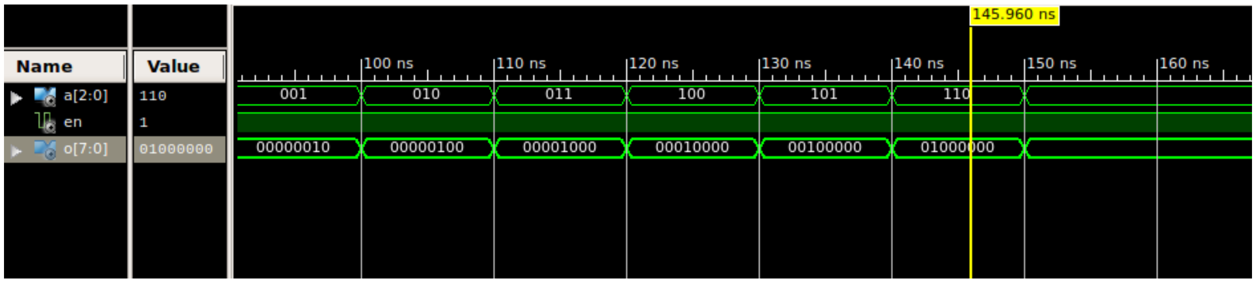
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0 and A1 are asserted and A2 is negated, O3 should be asserted, which can be seen above.



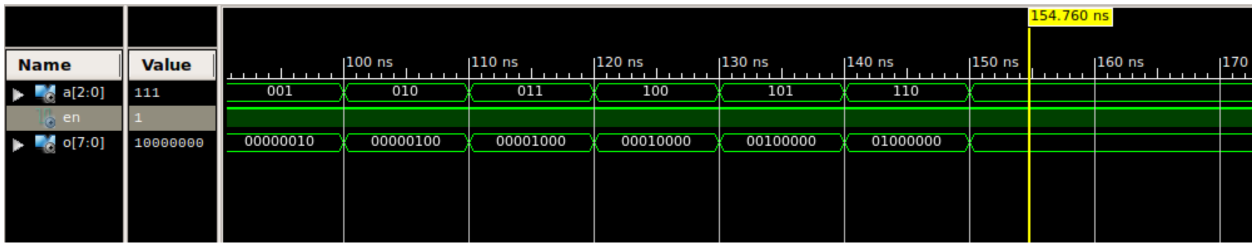
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A3 is asserted and A0 and A1 are negated, O4 should be asserted, which can be seen above.



This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0 and A2 are asserted and A1 is negated, O5 should be asserted, which can be seen above.



This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A1 and A2 are asserted and A0 is negated, O6 should be asserted, which can be seen above.



This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0, A1, and A2 are asserted, O7 should be asserted, which can be seen above.