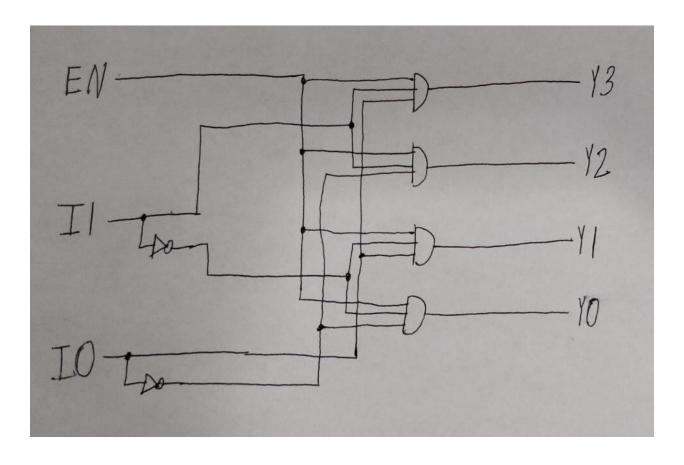
## Task 1: 2-to-4 Decoder

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 IO and I1, and our outputs are Y0, Y1, Y2, and Y3. The logic of the module itself follows as such:

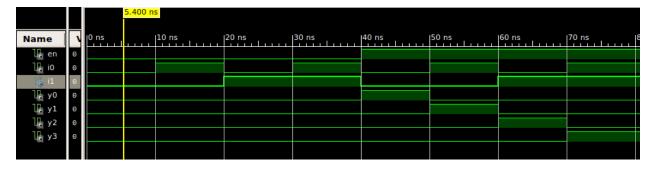
	Inputs		Outputs							
EN	I1	10	Y0	Y1	Y2	Y3				
0	Х	Х	0	0	0	0				
1	0	0	1	0	0	0				
1	0	1	0	1	0	0				
1	1	0	0	0	1	0				
1	1	1	0	0	0	1				



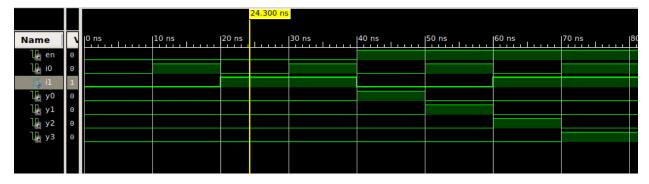
```
1 library IEEE;
   use IEEE.STD_LOGIC_1164.ALL;
2
 3
 4 entity v2to4dec is
     port (EN: in std_logic;
 5
            I0: in std_logic;
 6
            I1: in std_logic;
 7
            Y0: out std_logic;
8
            Y1: out std_logic;
9
            Y2: out std_logic;
10
            Y3: out std_logic);
11
12 end v2to4dec;
13
14 architecture Behavioral of v2to4dec is
15 begin
      Y0 <= '1' when EN='1' and I1='0' and I0='0' else '0';
16
      Y1 <= '1' when EN='1' and I1='0' and I0='1' else '0';
17
      Y2 <= '1' when EN='1' and I1='1' and I0='0' else '0';
18
19
      Y3 <= '1' when EN='1' and I1='1' and I0='1' else '0';
20 end Behavioral;
21
```

```
1 library IEEE;
 2 use IEEE.STD_LOGIC_1164.ALL;
 3
 4 entity testbench is
 5 end testbench;
 6
   architecture Behavioral of testbench is
 7
       component v2to4dec
 8
       port (EN: in std_logic;
 9
             I0: in std_logic;
10
             I1: in std_logic;
11
             Y0: out std_logic;
12
             Y1: out std_logic;
13
             Y2: out std_logic;
14
             Y3: out std_logic);
15
       end component;
16
17
       --inputs
18
       signal EN : std_logic := '0';
19
       signal I0 : std_logic := '0';
20
       signal I1 : std_logic := '0';
21
       --outputs
2.2
       signal Y0 : std_logic;
23
       signal Y1 : std_logic;
24
       signal Y2 : std_logic;
25
       signal Y3 : std_logic;
26
27
```

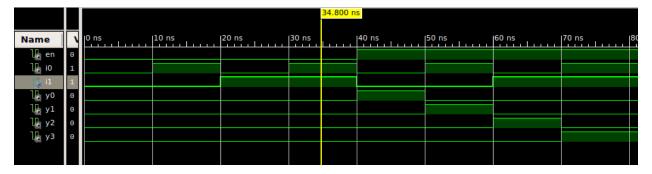
```
27
    begin
28
29
       uut: v2to4dec port map(
                 EN => EN,
30
                  I0 \Rightarrow I0,
31
32
                  I1 => I1,
                 Y0 \Rightarrow Y0,
33
                 Y1 \Rightarrow Y1,
34
                 Y2 \Rightarrow Y2
35
36
                 Y3 => Y3);
37
38
       stim_proc: process
       begin
39
           EN <= '0';
40
           I1 <= '0'; I0 <= '0'; wait for 10 ns;
41
           I1 <= '0'; I0 <= '1'; wait for 10 ns;
42
           I1 <= '1'; I0 <= '0'; wait for 10 ns;
43
           I1 <= '1'; I0 <= '1'; wait for 10 ns;
44
           EN <= '1';
45
           I1 <= '0'; I0 <= '0'; wait for 10 ns;
46
           I1 <= '0'; I0 <= '1'; wait for 10 ns;
47
           I1 <= '1'; I0 <= '0'; wait for 10 ns;
48
           I1 <= '1'; I0 <= '1'; wait for 10 ns;
49
           wait;
50
51
       end process;
    end Behavioral;
52
```



				14.900 n	5					
Name	N	0 ns	10 ns		20 ns	30 ns	40 ns	50 ns	60 ns	70 ns   8
∏ <sub>d</sub> en	0									
₹ i0	1									
V₀ i1	0									
$\mathbb{T}_{\mathbb{Q}}$ y0	0									
\[ y1	0									
∏ <sub>0</sub> y2	0									
7.0	0									

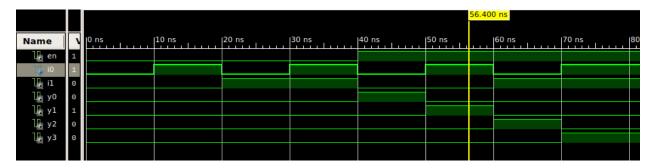


This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.

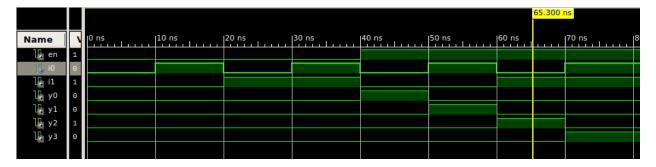


							45.700	ns			
Name	N	0 ns	10 ns	20 ns	30 ns	40 ns		50 ns	60 ns	70 ns	80
1046	1										
୍ୟା io	0										
₩ i1	0										
1046	1										
₩ y1	0										
∏ <sub>0</sub> y2	0										
Т₀ уз	0										
											ı

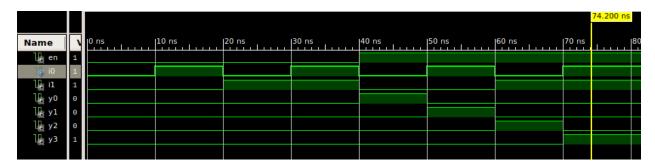
This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since IO and I1 are negated, YO should be asserted which can be seen above.



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

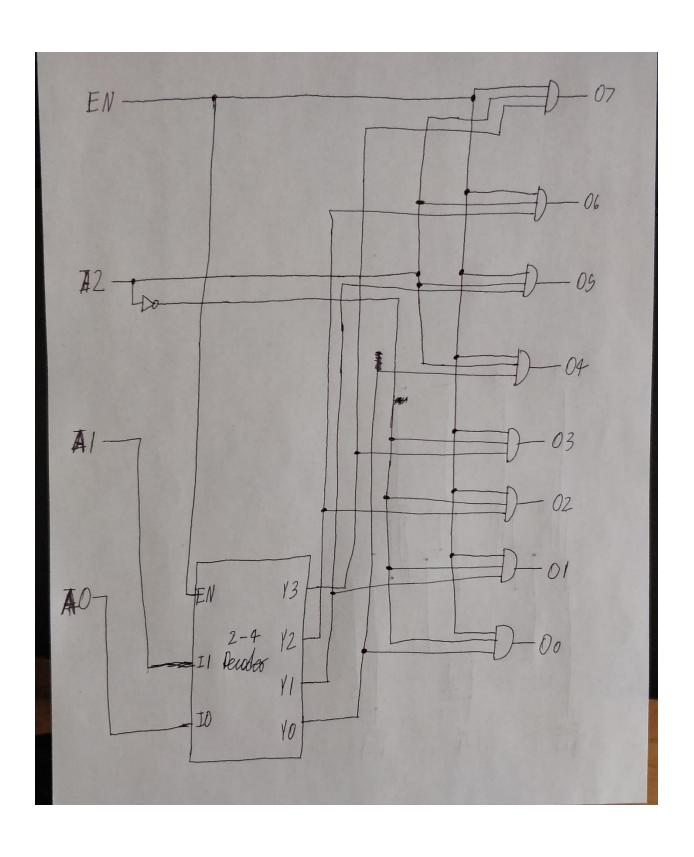


This is the correct outcome because the enable bit is asserted, therefore the circuit is active, and since IO and I1 are asserted, Y3 should be asserted which can be seen above.

## Task 2: 3-to-8 Decoder

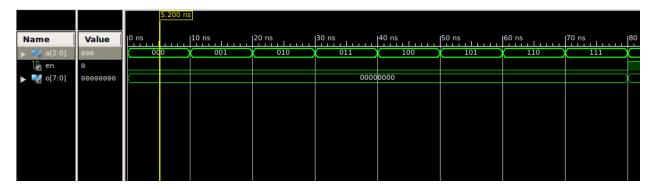
The 3x8 decoder is a decoder that takes in 3 inputs and outputs 1 of 8 different outputs. Deciding which one to output in based on the 3 inputs and the logic written in the function. In this example of a 3x8 decoder, our inputs are A0, A1, and A2, and our outputs are O0, O1, O2, O3, O4, O5, O6, and O7.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

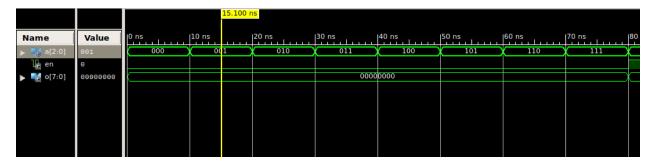
	Inp	uts		Outputs									
EN	A2	A1	A0	00	01	02	03	04	05	06	07		
0	Х	Х	Х	0	0	0	0	0	0	0	0		
1	0	0	0	1	0	0	0	0	0	0	0		
1	0	0	1	0	1	0	0	0	0	0	0		
1	0	1	0	0	0	1	0	0	0	0	0		
1	0	1	1	0	0	0	1	0	0	0	0		
1	1	0	0	0	0	0	0	1	0	0	0		
1	1	0	1	0	0	0	0	0	1	0	0		
1	1	1	0	0	0	0	0	0	0	1	0		
1	1	1	1	0	0	0	0	0	0	0	1		



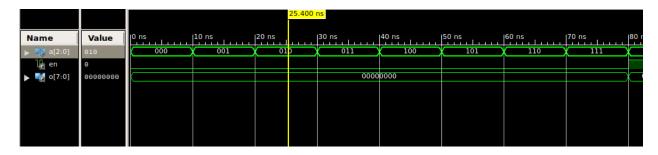
```
1 library IEEE;
  2 use IEEE.STD_LOGIC_1164.ALL;
  3 library unisim;
  4 use unisim.vcomponents.all;
  6
  7 entity v3to8dec is
      port (A : in std_logic_vector (2 downto 0);
  8
             EN: in std_logic;
  9
             O : out std_logic_vector (7 downto 0));
  10
  11 end v3to8dec;
  12
  13 architecture Behavioral of v3to8dec is
        signal A2_L: std_logic;
  14
        signal Y0, Y1, Y2, Y3: std_logic;
  15
  16
  17
        component INV port (I: in std_logic; O: out std_logic);
        end component;
  18
  19
        component AND3 port (I0, I1, I2: in std_logic; 0: out std_logic);
  20
  21
        end component;
  22
        component v2to4dec port (EN,I0,I1: in std_logic; Y0,Y1,Y2,Y3: out std_logic);
  23
        end component;
  24
  25
  26 begin
              U0: v2to4dec port map(EN,A(0),A(1),Y0,Y1,Y2,Y3);
  27
              U1: INV port map(A(2), A2_L);
  28
  29
              U2: AND3 port map(EN, A2_L, Y0, O(0));
              U3: AND3 port map(EN, A2_L, Y1, O(1));
  30
              U4: AND3 port map(EN, A2_L, Y2, O(2));
  31
              U5: AND3 port map(EN, A2_L, Y3, O(3));
  32
             U6: AND3 port map(EN, A(2), Y0, O(4));
  33
              U7: AND3 port map(EN, A(2), Y1, O(5));
  34
              U8: AND3 port map(EN, A(2), Y2, O(6));
  35
              U9: AND3 port map(EN, A(2), Y3, O(7));
  36
  37 end Behavioral;
 3.8
```

```
1 library IEEE;
 2 use IEEE.STD_LOGIC_1164.ALL;
 3
 4 entity testbench is
 5 end testbench;
 6
   architecture Behavioral of testbench is
 7
       component v3to8dec
 8
          port (A : in std_logic_vector (2 downto 0);
 9
                EN: in std_logic;
10
                O : out std_logic_vector (7 downto 0));
11
       end component;
12
13
       --inputs
14
       signal A: std_logic_vector(2 downto 0) := (others => '0');
15
       signal EN : std_logic := '0';
16
17
18
       --outputs
19
       signal 0 : std_logic_vector(7 downto 0);
20
21 begin
       uut: v3to8dec port map (
22
                A => A
23
                EN => EN,
24
                0=>0);
25
26
26
       stim_proc: process
27
       begin
28
          EN<='0'; A<="000"; wait for 10 ns;
29
30
          EN<='0'; A<="001"; wait for 10 ns;
          EN<='0'; A<="010"; wait for 10 ns;
31
          EN<='0'; A<="011"; wait for 10 ns;
32
          EN<='0'; A<="100"; wait for 10 ns;
33
          EN<='0'; A<="101"; wait for 10 ns;
34
          EN<='0'; A<="110"; wait for 10 ns;
35
          EN<='0'; A<="111"; wait for 10 ns;
36
37
          EN<='1'; A<="000"; wait for 10 ns;
38
          EN<='1'; A<="001"; wait for 10 ns;
39
          EN \le '1'; A \le ''010"; wait for 10 ns;
40
41
          EN<='1'; A<="011"; wait for 10 ns;
          EN<='1'; A<="100"; wait for 10 ns;
42
          EN<='1'; A<="101"; wait for 10 ns;
43
          EN<='1'; A<="110"; wait for 10 ns;
44
45
          EN<='1'; A<="111"; wait for 10 ns;
          wait;
46
       end process;
47
48
49 end Behavioral;
50
```

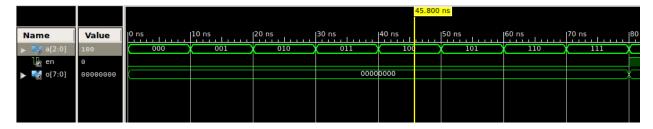




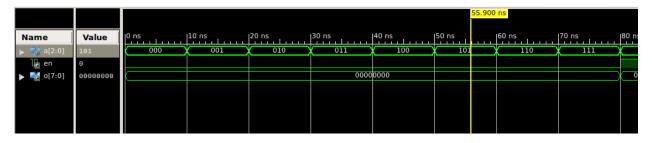
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



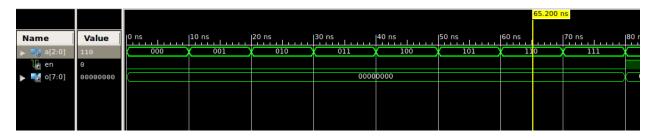
						35.400	ns				
Name	Value	0 ns	10 ns	20 ns	30 ns		40 ns	50 ns	60 ns	70 ns	80
▶ <b>■</b> a[2:0]	011	000	001	010	01	1	100	101	110	111	
U₀ en	0										
▶ 🌄 o[7:0]	00000000					0000	0000			X	

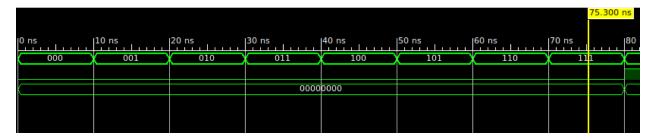


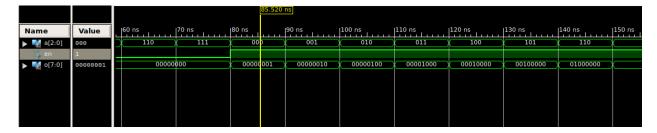
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



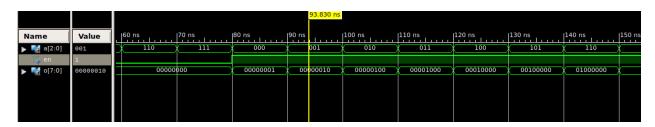
This is the correct outcome because the enable bit is negated, therefore the circuit is off and none of the input combinations will matter, as none of the outcomes will be asserted.



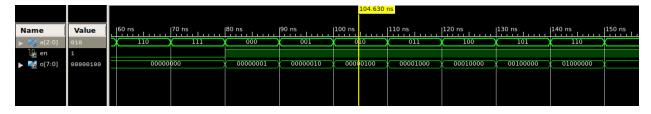




This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0, A1, and A2 are negated, O0 should be asserted, which can be seen above.



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.



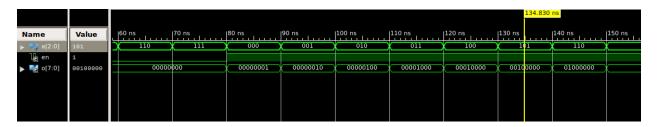
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A1 is asserted and A0 and A2 are negated, O2 should be asserted, which can be seen above.

							114.530	ns ns			
Name	Value	60 ns	70 ns	180 ns	190 ns	100 ns	110 ns	120 ns	130 ns	140 ns	150 ns
▶ <b>■</b> a[2:0]	011	110	111	000	001	010	011	100	101	110	Х
V₀ en	1										
▶ ■ o[7:0]	00001000	00000	000	00000001	00000010	00000100	00001000	00010000	00100000	01000000	X

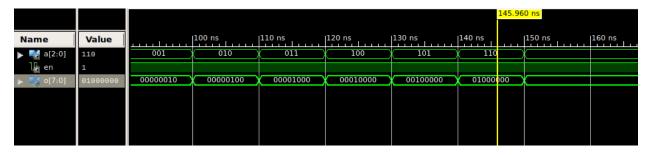
This is the correct outcome because, the enable bit is asserted so the circuit is on, and since A0 and A1 are asserted and A2 is negated, O3 should be asserted, which can be seen above.



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.

									154.760	ns.	
Name	Value		100 ns	110 ns	120 ns	130 ns	140 ns	150 ns	1	160 ns	170
▶ <b>■</b> a[2:0]	111	001	010	011	100	101	110	<b>*</b>			
퉪 en	1										
▶ <b>№</b> o[7:0]	10000000	00000010	00000100	00001000	00010000	00100000	01000000	<b>k</b>			

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