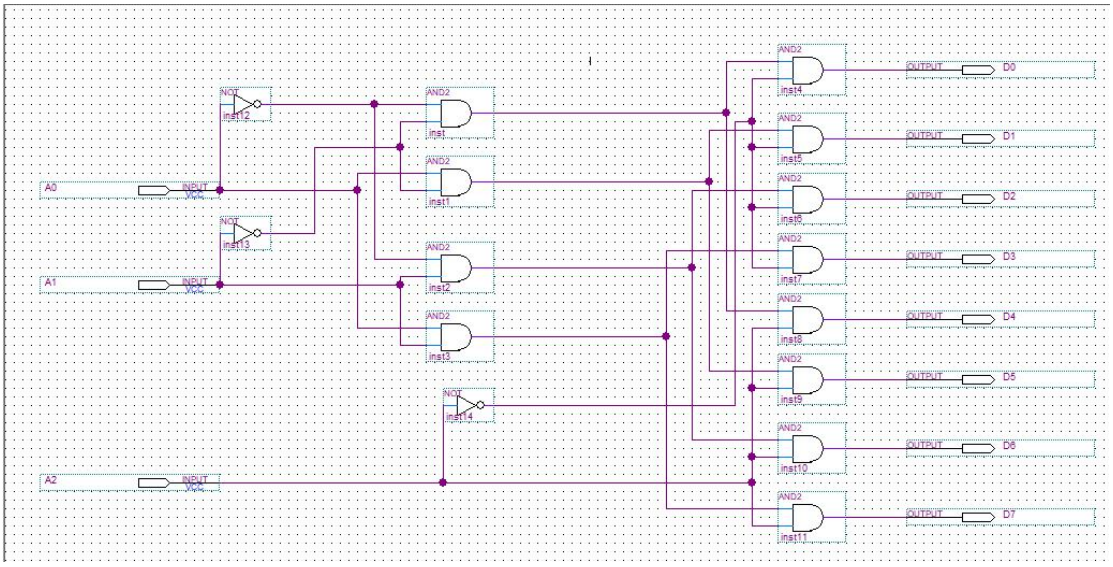
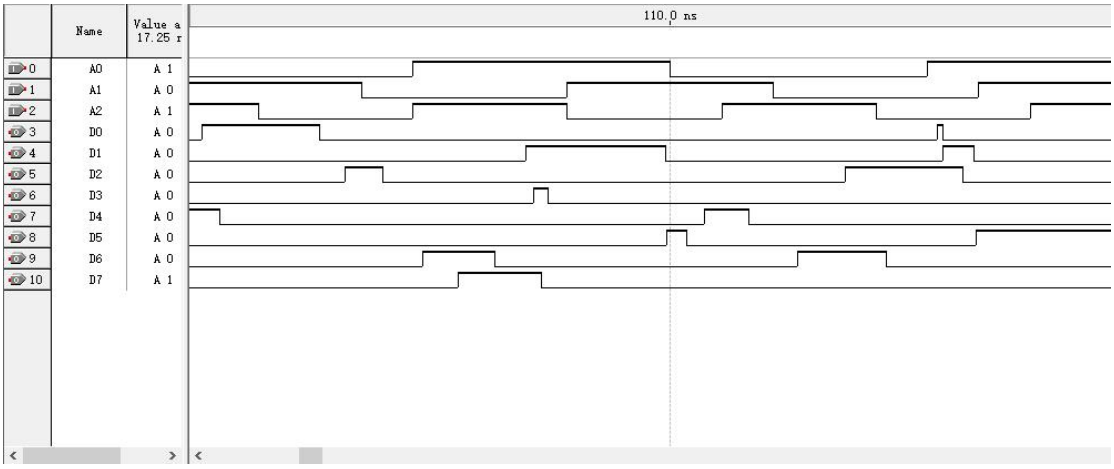
3-8 译码器的原理图



输入：A0,A1,A2

输出：D0,D1,D2,D3,D4,D5,D6,D7

仿真结果：



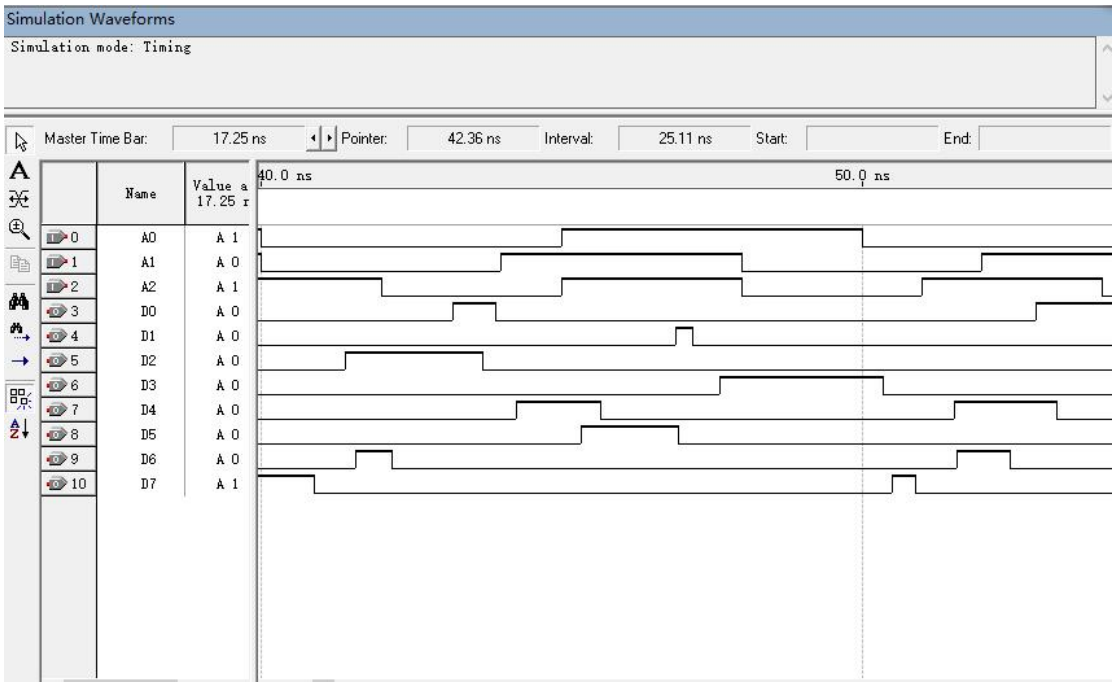
3-8 译码器 VHDL 程序

```
1  library lcdf_vhdl,ieee;
2  use ieee.std_logic_1164.all,lcdf_vhdl.func_prims.all;
3
4  entity zjw2018 is
5  port (A:in std_logic_vector(0 to 2);
6        D:out std_logic_vector(0 to 7));
7  end zjw2018;
8
9  architecture structural of zjw2018 is
10
11 signal A0_n,A1_n,A2_n,and00_out,and01_out,and02_out,and03_out,and10_out,and11_out,
12 and12_out,and13_out,and14_out,and15_out,and16_out,and17_out:std_logic;
13
14 begin
15  inv_0:not1 port map(in1 => A(0),out1 => A0_n);
16  inv_1:not1 port map(A(1),A1_n);
17  inv_2:not1 port map(A(2),A2_n);
18
19  and_00:and_2 port map(A0_n,A1_n,and00_out);
20  and_01:and_2 port map(A(0),A1_n,and01_out);
21  and_02:and_2 port map(A0_n,A(1),and02_out);
22  and_03:and_2 port map(A(0),A(1),and03_out);
23
24  and_10:and_2 port map(and10_out,A2_n,D(0));
25  and_11:and_2 port map(and11_out,A2_n,D(1));
26  and_12:and_2 port map(and12_out,A2_n,D(2));
27  and_13:and_2 port map(and13_out,A2_n,D(3));
28  and_14:and_2 port map(and10_out,A(2),D(4));
29  and_15:and_2 port map(and11_out,A(2),D(5));
30  and_16:and_2 port map(and12_out,A(2),D(6));
31  and_17:and_2 port map(and13_out,A(2),D(7));
32
33 end structural;
```

输入：A0,A1,A2

输出：D0,D1,D2,D3,D4,D5,D6,D7

仿真结果：



模型机指令译码器的 VHDL 程序

```
library ieee;
use ieee.std_logic_1164.all;
entity command_decoder is
    port(EN:in std_logic;
    IR: in std_logic_vector(7 downto 0);
    order: out std_logic_vector(3 downto 0);
    RA,RB: out std_logic_vector(1 downto 0);
    MOVA,MOVB,MOVC,ALU1,ALU2,FL,FR,JMP,JZ,JC,IN0,OUT0,NOP,HEAL: out std_logic);
end command_decoder;

architecture dec of command_decoder is
signal instruct: std_logic_vector(3 downto 0);
signal R1, R2: std_logic_vector(1 downto 0);
begin
    order <= instruct;
    RA <= R2;
    RB <= R1;
    instruct <= IR(7 downto 4);
    R1 <= IR(3 downto 2);
    R2 <= IR(1 downto 0);
    MOVA <= '1' when instruct & EN = "11111" and (R1 /= "11" and R2 /= "11") else '0';
    MOVB <= '1' when instruct & R1 & EN = "1111111" else '0';
    MOVC <= '1' when instruct & R2 & EN = "1111111" and instruct & R1 & EN /= "1111111" else '0';
    ALU1 <= '1' when instruct & EN = "10011" or instruct & EN = "01101" else '0';
    ALU2 <= '1' when instruct & EN = "01011" or instruct & EN = "10111" else '0';
    FL <= '1' when instruct & EN & R2= "1010111" else '0';
    FR <= '1' when instruct & EN & R2= "1010100" else '0';
    JMP <= '1' when IR & EN = "000100001" else '0';
    JZ <= '1' when IR & EN = "000100011" else '0';
    JC <= '1' when IR & EN = "000100101" else '0';
    IN0 <= '1' when instruct & EN = "00101" else '0';
    OUT0 <= '1' when instruct & EN = "01001" else '0';
    NOP <= '1' when instruct & EN = "01111" else '0';
    HEAL <= '1' when instruct & EN = "10001" else '0';
end dec;
```