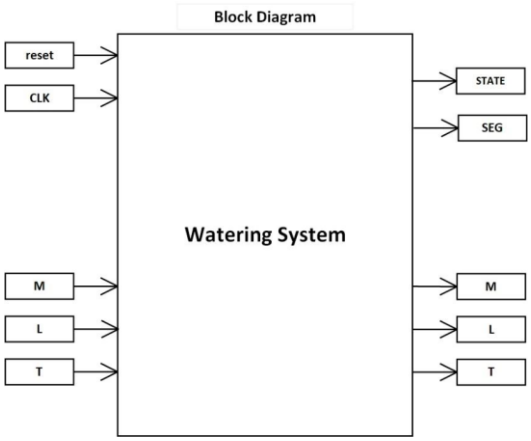
طراحی سیستم آبیاری هوشمند:

ماژول مد نظر به این صورت است:



ماژول دارای ورودی های زیر است:

۱- clk: این همان کلاک روی برد است که میتواند توسط کریستال پالس ساعت تامین شود. این ورودی یک بیتی است.

۲- reset: با یک شدن این ورودی، سیستم در هر حالتی که باشد، متوقف می شود و وارد حالت St۲ می شود و تا زمانی که start یک نشده باشد، در همان حالت می ماند. این ورودی یک بیتی است. ریست با کلاک سنکرون است.

۳- start: اگر مدار در حالت St۲ باشد، با زدن دکمه استارت، وارد حالت St۰ می شود و عملیات کاری مدار آغاز می شود. لازم به ذکر است این ورودی در بلوک دیاگرام شکل فوق در نظر نگرفته شده است ولی برای اینکه رفتار مدار دارای منطق بهتری باشد، وجود آن ضروری است. این ورودی یک بیتی است. استارت با کلاک سنکرون است.

٤- M: این ورودی، سطح رطوبت محیط را نشان میدهد و از سنسور رطوبت گرفته می شود. این سیگنال سه بیتی است یعنی میتواند مقادیر ۰ تا ۷ را پوشش دهد.

۵- L: این ورودی ، سطح نور محیط را نشان میدهد و از سنسور نورسنج گرفته می شود. این سیگنال یک بیتی است. مقدار ایده آل این سنسور، ۰ است.

۶- T: این ورودی ، سطح دمای محیط را نشان میدهد و از سنسور دماسنج گرفته می شود. این سیگنال یک بیتی است. مقدار ایده آل این سنسور، ۰ است.

ماژول دارای خروجی های زیر است:

۱- STATE: مدار دارای سه حالت است. (St۰ و St۱ و St۲). ما میخواهیم در هر لحظه اینکه مدار در چه حالتی است را در خروجی نشان دهیم. چون مدار دارای سه حالت است، پس برای نشان دادن این حالات، نیاز به دو بیت داریم پس این خروجی دو بیتی است.

۲- SEG: وقتی سیستم در حالت آبیاری است، روی سون سگمنت عبارت H و وقتی سیستم در حالت آبیاری نمیباشد، روی سون سگمنت عبارت – نوشته می شود. ورودی سون سگمنت هشت بیتی است پس این سیگنال نیز باید هشت بیتی باشد.

۳- M: این خروجی، سطح رطوبت محیط را نشان میدهد و همان ورودی ماژول است که از سنسور رطوبت گرفته می شود. این سیگنال سه بیتی است یعنی میتواند مقادیر ۰ تا ۷ را پوشش دهد.

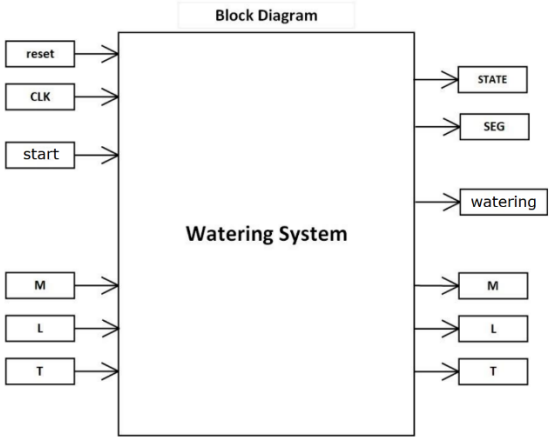
٤- L: این خروجی ، سطح نور محیط را نشان میدهد و همان ورودی ماژول است که از سنسور نورسنج گرفته می شود. این سیگنال یک بیتی است. مقدار ایده آل این سنسور، ۰ است.

۵- T: این خروجی ، سطح دمای محیط را نشان میدهد و همان ورودی ماژول است که از سنسور دماسنج گرفته می شود. این سیگنال یک بیتی است. مقدار ایده آل این سنسور، ۰ است.

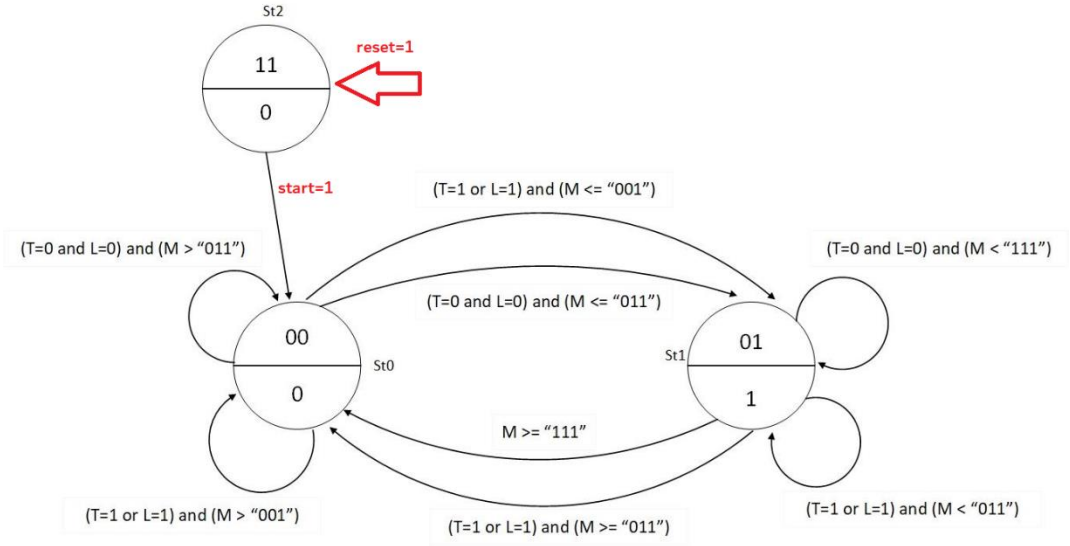
۶- Watering: این خروجی، نشان دهنده این است که در حالت حاضر سیستم در حالت آبیاری می باشد یا خیر. لازم به ذکر است که در صورت مساله این خروجی در نظر گرفته نشده است ولی برای اینکه مدار رفتار منطقی تری داشته باشد، ما آن را برای مدار در نظر گرفته ایم.

ما برای اینکه ورودی ها و خروجی ها را بهتر در کد نشان دهیم، ابتدای ورودی ها (به جز کلاک) _i و ابتدای خروجی ها، _o قرار داده ایم. به عنوان مثال، به جای اینکه خروجی SEG داشته باشیم، جهت نامگذاری بهتر، خروجی را o_SEG نامگذاری کرده ایم.

پس بلوک دیاگرام مداری که ما پیاده سازی کرده ایم، به این صورت است:



با توجه به توضیحات فوق، ماشین حالت پیاده سازی شده به این صورت است:



قسمت های قرمز را خودمان به ماشین حالت اولیه اضافه کرده ایم.

توضیح ماشین حالت فوق:

ماشین حالت فوق، سه حالت دارد:

۰- St۰: در این حالت، آبیاری اتفاق نمی افتد و خروجی Watering صفر است.

زمانی که در حالت صفر ST۰ قرار داشته باشیم امکان رخ دادن چهار حالت وجود دارد که عبارتند از:

- **شرایط ایده آل باشد و سطح رطوبت بیشتر از ۳ باشد:** سیستم در حالت صفر باقی میماند.
- **شرایط ایده آل باشد و سطح رطوبت کمتر یا مساوی با ۳ باشد:** سیستم از حالت صفر به حالت یک وارد میشود.
- **شرایط ایده آل نباشد و سطح رطوبت کمتر یا مساوی با ۱ باشد:** سیستم از حالت صفر به حالت یک وارد میشود.
- **شرایط ایده آل نباشد و سطح رطوبت بیشتر از ۱ باشد:** سیستم در حالت صفر باقی میماند.

۱- St۱: در این حالت، آبیاری اتفاق می افتد و خروجی Watering یک است.

زمانی که در حالت یک (ST۱ زمانی که آبیاری اتفاق میافتد) قرار داشته باشیم امکان رخ دادن چهار حالت وجود دارد که عبارتند از:

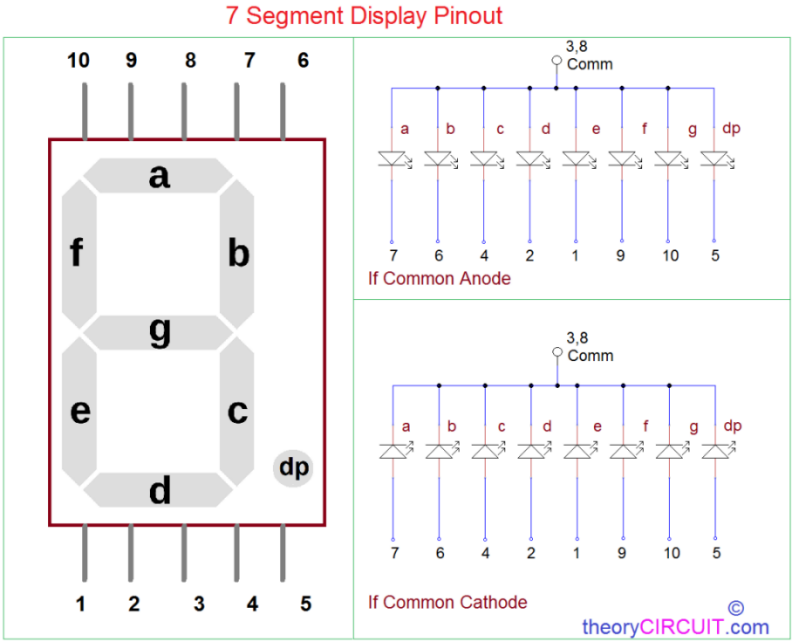
- **شرایط ایده آل باشد و سطح رطوبت کمتر از ۷ باشد:** سیستم در حالت یک باقی میماند.
- **سطح رطوبت بزرگتر یا مساوی با ۷ باشد:** سیستم از حالت یک به حالت صفر وارد میشود.
- **شرایط ایده آل نباشد و سطح رطوبت کمتر از ۳ باشد:** سیستم در حالت یک باقی میماند.

- **شرایط ایده آل نباشد و سطح رطوبت بیشتر یا مساوی با ۳ باشد:** سیستم از حالت یک به حالت صفر وارد میشود.

۲- St۲: در این حالت، آبیاری اتفاق نمی افتد و خروجی Watering صفر است. این حالت وقتی رخ میدهد که ورودی reset برابر با یک شده باشد و تا زمانی که start یک نشده باشد، در این حالت می مانیم.

توضیحات در رابطه با سون سگمنت:

سون سگمنت به دو صورت آند مشترک یا کاتد مشترک موجود است:



همانطور که مشخص است، سون سگمنت از ۸ عدد LED تشکیل شده است. ما فرض میکنیم از حالت کاتد مشترک استفاده شده است. در این حالت، پایه مشترک یا Comm باید به ولتاژ LOW و پایه هایی که میخواهیم LED متصل به آن ها روشن باشد، به ولتاژ HIGH متصل شوند.

ما قرار داد میکنیم که سیگنال متصل به سون سگمنت به این ترتیب باشد:

dp.g.f.e.d.c.b.a

dp همواره در این پروژه خاموش است پس بیت هشتم خروجی SEG همواره صفر است. ما در این پروژه، نیاز داریم که وقتی سیستم در حالت آبیاری است، روی سون سگمنت H و وقتی سیستم در حالت آبیاری نیست، روی سون سگمنت – بنویسیم. با توجه به توضیحات گفته شده، داریم:

H = ۰۱۱۱۰۱۱۰ on Y segment

- = ۰۱۰۰۰۰۰۰ on Y segment

کد VHDL:

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

entity watering is
    Port ( clk : in  STD_LOGIC;
           i_reset : in  STD_LOGIC;
           i_start : in  STD_LOGIC;
           i_M : in  STD_LOGIC_VECTOR (2 downto 0);
           i_T : in  STD_LOGIC;
           i_L : in  STD_LOGIC;
           o_state : out  STD_LOGIC_VECTOR (1 downto 0);
           o_SEG : out  STD_LOGIC_VECTOR (7 downto 0);
           o_watering : out  STD_LOGIC;
           o_M : out  STD_LOGIC_VECTOR (2 downto 0);
           o_T : out  STD_LOGIC;
           o_L : out  STD_LOGIC);
end watering;

architecture Behavioral of watering is

   SUBTYPE Tstate IS std_logic_vector (1 downto 0);

       constant St0 : Tstate := "00";
       constant St1 : Tstate := "01";
       constant St2 : Tstate := "11";

   SIGNAL state: Tstate;
   SIGNAL next_state: Tstate;

begin

o_M <= i_M;
o_T <= i_T;
o_L <= i_L;
o_state <= state;

   PROCESS(clk)
   BEGIN
            IF rising_edge(clk) THEN
                   IF i_reset ='1' then
           state <= St2;
                   ELSE
           state <= next_state;
                   END IF;
      END IF;
   END PROCESS;


PROCESS (state, i_M, i_T , i_L , i_start)
```

```
   BEGIN
       next_state <= St2;

      CASE state IS

         WHEN St0 =>
                          o_watering <= '0';
                          o_SEG <= "01000000";

                          if (i_T='0' and i_L='0' and i_M>"011") then
                                 next_state <= St0;
                          elsif ((i_T='1' or i_L='1') and i_M>"001") then
                                 next_state <= St0;
                          elsif (i_T='0' and i_L='0' and i_M<="011") then
                                 next_state <= St1;
                          elsif ((i_T='1' or i_L='1') and i_M<="001") then
                                 next_state <= St1;
                          end if;

         WHEN St1 =>

                          o_watering <= '1';
                          o_SEG <= "01110110";

                          if (i_T='0' and i_L='0' and i_M<"111") then
                                 next_state <= St1;
                          elsif ((i_T='1' or i_L='1') and i_M<"011") then
                                 next_state <= St1;
                          elsif (i_T='0' and i_L='0' and i_M>="011") then
                                 next_state <= St0;
                          elsif ((i_T='1' or i_L='1') and i_M>="111") then
                                 next_state <= St0;
                          end if;

         WHEN St2 =>
                          o_watering <= '0';
                          o_SEG <= "01000000";

                          if (i_start='1') then
                                 next_state <= St0;
                          else
                                 next_state <= St2;
                          end if;

         WHEN OTHERS =>
                          next_state <= St2;
                          o_watering <= '0';
                          o_SEG <= "01000000";
                   END CASE;

   END PROCESS;
end Behavioral;
```

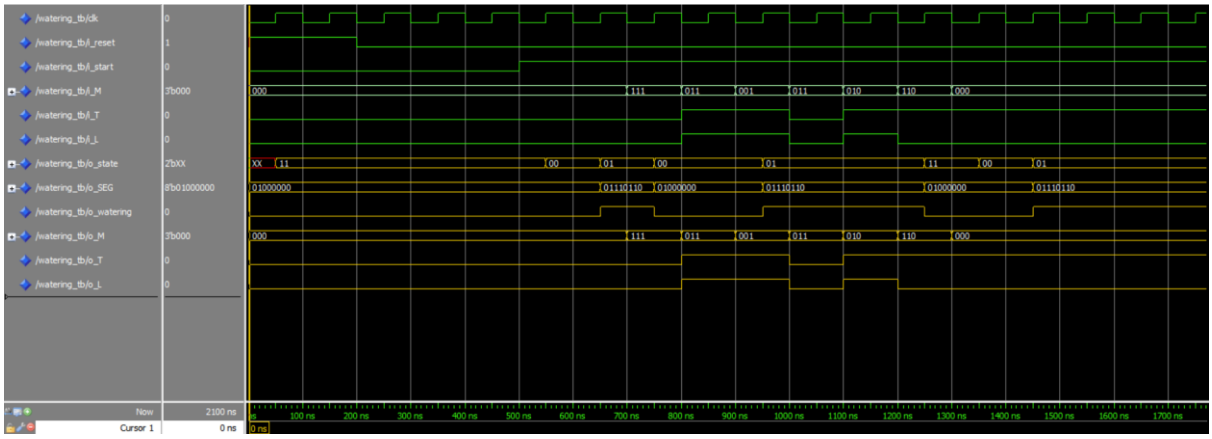
تست بنچ:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY watering_tb IS
END watering_tb;

ARCHITECTURE behavior OF watering_tb IS

    COMPONENT watering
    PORT(
         clk : IN  std_logic;
         i_reset : IN  std_logic;
         i_start : IN  std_logic;
         i_M : IN  std_logic_vector(2 downto 0);
         i_T : IN  std_logic;
         i_L : IN  std_logic;
         o_state : OUT  std_logic_vector(1 downto 0);
         o_SEG : OUT  std_logic_vector(7 downto 0);
         o_watering : OUT  std_logic;
         o_M : OUT  std_logic_vector(2 downto 0);
         o_T : OUT  std_logic;
         o_L : OUT  std_logic
        );
    END COMPONENT;

   signal clk : std_logic := '0';
   signal i_reset : std_logic := '0';
   signal i_start : std_logic := '0';
   signal i_M : std_logic_vector(2 downto 0) := (others => '0');
   signal i_T : std_logic := '0';
   signal i_L : std_logic := '0';

   signal o_state : std_logic_vector(1 downto 0);
   signal o_SEG : std_logic_vector(7 downto 0);
   signal o_watering : std_logic;
   signal o_M : std_logic_vector(2 downto 0);
   signal o_T : std_logic;
   signal o_L : std_logic;

BEGIN

   uut: watering PORT MAP (
          clk => clk,
          i_reset => i_reset,
          i_start => i_start,
          i_M => i_M,
          i_T => i_T,
          i_L => i_L,
          o_state => o_state,
          o_SEG => o_SEG,
```

```
        o_watering => o_watering,
        o_M => o_M,
        o_T => o_T,
        o_L => o_L
      );

   process
   begin
            clk <= '0';
            wait for 50 ns;
            clk <= '1';
            wait for 50 ns;
   end process;


   process
   begin
            i_start <= '0';
            i_reset <= '1';
            i_M <= "000";
            i_T <= '0';
            i_L <= '0';
      wait for 200 ns;
            i_reset <= '0';
      wait for 300 ns;
            i_start <= '1';
      wait for 200 ns;
            i_M <= "111";       i_T <= '0';         i_L <= '0';
      wait for 100 ns;
            i_M <= "011";       i_T <= '1';         i_L <= '1';
      wait for 100 ns;
            i_M <= "001";       i_T <= '0';         i_L <= '1';
      wait for 100 ns;
            i_M <= "011";       i_T <= '0';         i_L <= '0';
         wait for 100 ns;
            i_M <= "010";       i_T <= '1';         i_L <= '1';
         wait for 100 ns;
            i_M <= "110";       i_T <= '1';         i_L <= '0';
         wait for 100 ns;
            i_M <= "000";       i_T <= '1';         i_L <= '0';
      wait;
   end process;

END;
```

تست بنچ طوری نوشته شده است که همه حالات را در بر بگیرد.

خروجی به این صورت است:



در این مدار همه حالات و ترنزیسشن ها تست شده است.

https://github.com/ZahraAllameh/Smartirragation