MH

Masoud Heidary

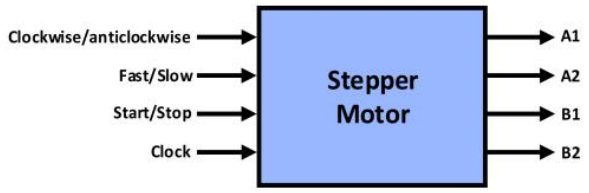
think outside the box

CA Lab – EX8

MasoudHeidaryMH@gmail.com

راه اندازی موتور پله‌ای

- راه‌اندازی استپ موتور با قابلیت تنظیم جهت گردش، دارای دو سرعت مختلف
- موتور موجود از نوع موتور پله‌ای دوفاز دوقطبی است.



برای طراحی این درایور، میتوانیم به این متصور باشیم که، یک Machine State هشتایی به صورت زیر داریم.

حال زمانی که میخواهیم به صورت Full Step کار کنیم، باید بین حالت های D0, D2, D4, D6 جا به جا شویم، و اگر میخواهیم به صورت Half Step کار کنیم، باید بین حالت های D0,...,D7 جا به جا شویم.

جهت حرکت نیز، به توجه به جهت حرکت ما بین حالت ها مشخص میشود، که اگر از بالا به پایین حرکت کنیم، موتور ما ساعت گرد خواهد بود، و اگر با پایین به بالا حرکت کنیم، موتور ما پاد ساعت گرد خواهد بود.

Half Step, State Machine

Full Step

1	1	0	0	D0
0	1	0	0	D1
0	1	1	0	D2
0	0	1	0	D3
0	0	1	1	D4
0	0	0	1	D5
1	0	0	1	D6
1	0	0	0	D7

*** زمانی که درایور ما از حالت Half Step به Full Step جا به جا میشود، ابتدا باید وارد حالت های D0, D2, D4, D6 شویم، سپس دو تا دو تا جا به جا شویم.

*** برای راحتی در شبیه سازی و مشاهده خروجی ها، از سیگنال out_t به صورت ۴ بیتی استفاده شده است، اما در واقع نیازی به آن نبوده.

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

entity StepperMotor is
    Port ( AntiClockWise : in  STD_LOGIC;
           FullStep : in  STD_LOGIC;
           En : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           A1 : out  STD_LOGIC;
           A2 : out  STD_LOGIC;
           B1 : out  STD_LOGIC;
           B2 : out  STD_LOGIC);
end StepperMotor;

architecture Behavioral of StepperMotor is
    type state_type is (D0, D1, D2, D3, D4, D5, D6, D7);
    signal state: state_type := D0;
    signal out_t: STD_LOGIC_VECTOR (3 downto 0);
begin
    A1 <= out_t(0);
    A2 <= out_t(2);
    B1 <= out_t(1);
    B2 <= out_t(3);

    -- STATE process
    process (clk, En)
    begin
        -- change state by clk
        if rising_edge(clk) and En='1' then
            case state is
                when D0 =>
                    if FullStep = '1' then
                        if AntiClockWise = '0' then
                            state <= D2;
                        else
                            state <= D6;
                        end if;
                    else
                        if AntiClockWise = '0' then
                            state <= D1;
```

```
                    else
                        state <= D7;
                    end if;
                end if;

            when D1 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D2;
                    else
                        state <= D0;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D2;
                    else
                        state <= D0;
                    end if;
                end if;

            when D2 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D4;
                    else
                        state <= D0;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D3;
                    else
                        state <= D1;
                    end if;
                end if;

            when D3 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D4;
                    else
                        state <= D2;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D4;
```

```
                    else
                        state <= D2;
                    end if;
                end if;

            when D4 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D6;
                    else
                        state <= D2;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D5;
                    else
                        state <= D3;
                    end if;
                end if;

            when D5 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D6;
                    else
                        state <= D4;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D6;
                    else
                        state <= D4;
                    end if;
                end if;

            when D6 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D0;
                    else
                        state <= D4;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D7;
```

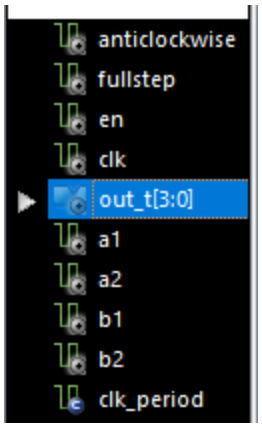
```
                    else
                        state <= D5;
                    end if;
                end if;

            when D7 =>
                if FullStep = '1' then
                    if AntiClockWise = '0' then
                        state <= D0;
                    else
                        state <= D6;
                    end if;
                else
                    if AntiClockWise = '0' then
                        state <= D0;
                    else
                        state <= D6;
                    end if;
                end if;

        end case;
    end if;
end process;

-- change output based on state
process (state)
begin
    case state is
        when D0 =>
            out_t <= "1100";
        when D1 =>
            out_t <= "0100";
        when D2 =>
            out_t <= "0110";
        when D3 =>
            out_t <= "0010";
        when D4 =>
            out_t <= "0011";
        when D5 =>
            out_t <= "0001";
        when D6 =>
            out_t <= "1001";
        when D7 =>
            out_t <= "1000";
    end case;
```

```
    end process;

end Behavioral;
```

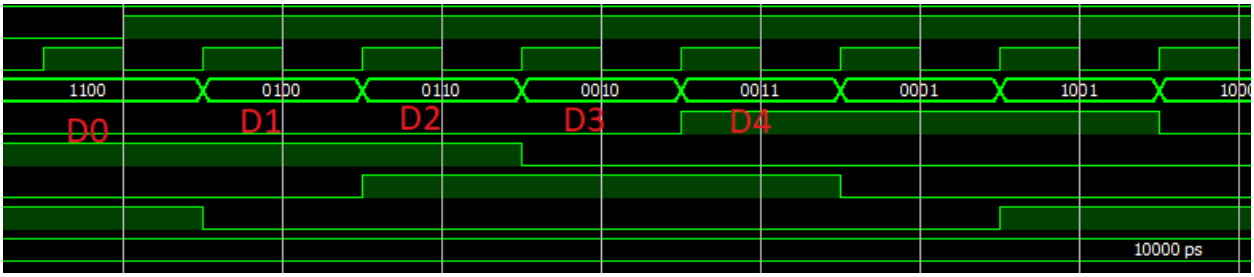
Test

```
      En <= '0';
      wait for clk_period*5;

      En <= '1';
      wait for clk_period*50;

      AntiClockWise <= '1';
      wait for clk_period*50;

      FullStep <= '1';
      AntiClockWise <= '0';
      wait for clk_period*50;

      AntiClockWise <= '1';
      wait for clk_period*50;
```

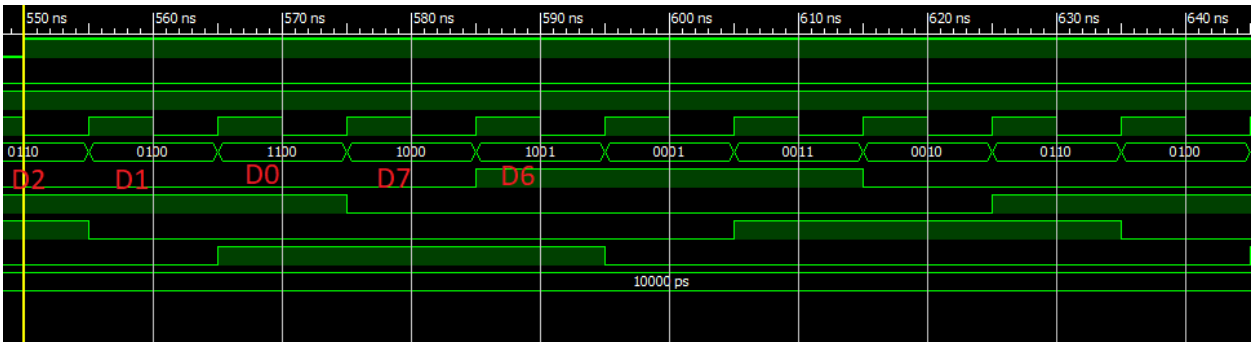
نمودارهای خروجی به صورت زیر میباشند



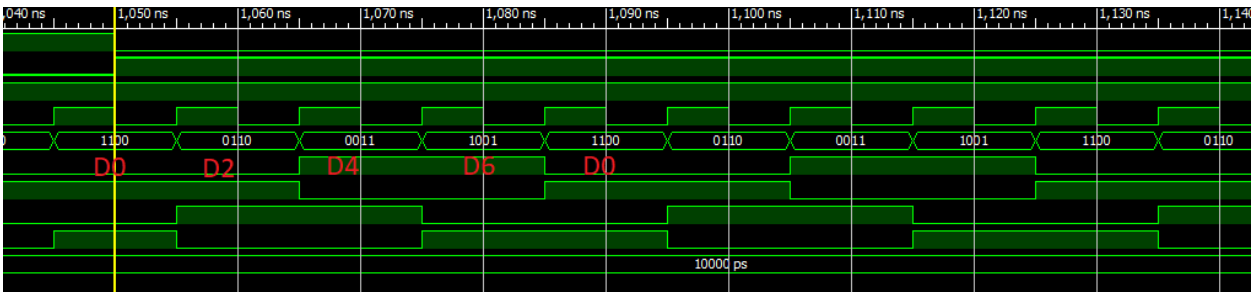
Rotate: Clock Wise, Step: Half Step



Rotate: Anti Clock Wise, Step: Half Step



Rotate: Clock Wise, Step: Full Step



Rotate: Anti Clock Wise, Step: Full Step