|  |
| --- |
|  |
| CA Lab – EX6 |
| |  |  |  | | --- | --- | --- | |  | MasoudHeidaryMH@gmail.com |  | |

****

**شمارنده BCD**

برای طراحی این شماره از machine state استفاده کرده و با آمدن هر کلاک بالارونده به حالت بعدی میرویم، واضح است با هر تغییر حالت خروجی را نیز باید آبدیت نماییم.

برای پایه ریست نیز، به این صورت عمل میکنیم که هر زمانی فعال شد حالت مدار به صفر تغییر میدهیم و تا زمانی که ریست فعال است از تغییر حالت مدار جلوگیری میکنیم.

Code:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity FSM\_BCD is

    Port ( clk : in  STD\_LOGIC;

           reset : in  STD\_LOGIC;

           output : out  STD\_LOGIC\_VECTOR (3 downto 0));

end FSM\_BCD;

architecture Behavioral of FSM\_BCD is

    type state\_type is (D0, D1, D2, D3, D4, D5, D6, D7, D8, D9);

    signal state: state\_type := D0;

begin

    -- STATE process

    process (clk, reset)

    begin

        -- reset pin config

        if reset = '1' then

            state <= D0;

        -- change state by clk

        elsif rising\_edge(clk) then

            case state is

                when D0 =>

                    state <= D1;

                when D1 =>

                    state <= D2;

                when D2 =>

                    state <= D3;

                when D3 =>

                    state <= D4;

                when D4 =>

                    state <= D5;

                when D5 =>

                    state <= D6;

                when D6 =>

                    state <= D7;

                when D7 =>

                    state <= D8;

                when D8 =>

                    state <= D9;

                when D9 =>

                    state <= D0;

            end case;

        end if;

    end process;

    -- change output based on state

    process (state)

    begin

        case state is

            when D0 =>

                output <= "0000";

            when D1 =>

                output <= "0001";

            when D2 =>

                output <= "0010";

            when D3 =>

                output <= "0011";

            when D4 =>

                output <= "0100";

            when D5 =>

                output <= "0101";

            when D6 =>

                output <= "0110";

            when D7 =>

                output <= "0111";

            when D8 =>

                output <= "1000";

            when D9 =>

                output <= "1001";

        end case;

    end process;

end Behavioral;

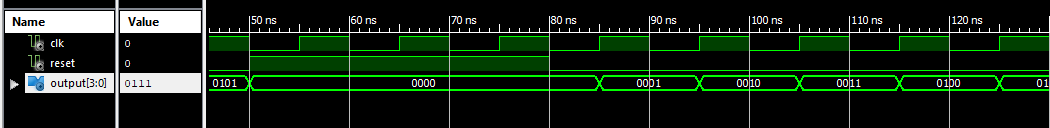
نوع پیاده سازی شده machine state نیز از یکی از مطالب شرکت اینتل ایده گرفته شده است

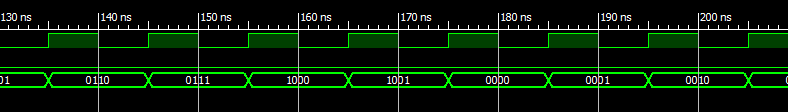
[link](https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/hdl/vhdl/vhdl_pro_state_machines.htm#:~:text=A%20state%20machine%20is%20a,and%20the%20next%2Dstate%20logic.)

(if link didn’t work)

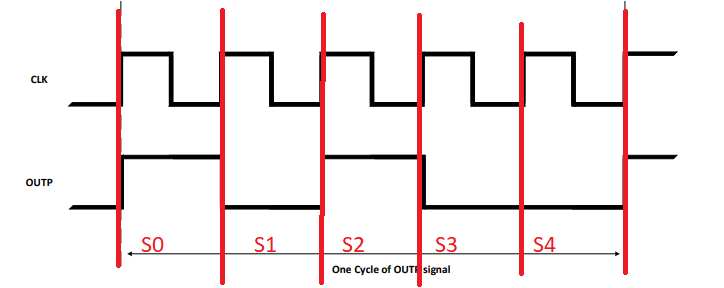
<https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/hdl/vhdl/vhdl_pro_state_machines.htm#:~:text=A%20state%20machine%20is%20a,and%20the%20next%2Dstate%20logic>.

Test Bench:

Reset Pin

Counting

**مولد سیگنال**

اگر سیگنال خروجی را به صورت state های مختلفی از یک state machine بدانیم، میتوانیم سیگنال را به صورت زیر تجسم کنیم

حال میتوانیم مانند مثال قبلی، با آمدن کلاک تنها به state بعدی برویم و خروجی را نیز update کنیم.

Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity FSM\_Signal is

    Port ( clk : in  STD\_LOGIC;

           output : out  STD\_LOGIC);

end FSM\_Signal;

architecture Behavioral of FSM\_Signal is

    type state\_type is (S0, S1, S2, S3, S4);

    signal state: state\_type := S4;

begin

    process (clk)

    begin

        if rising\_edge(clk) then

            case state is

                when S0 =>

                    state <= S1;

                when S1 =>

                    state <= S2;

                when S2 =>

                    state <= S3;

                when S3 =>

                    state <= S4;

                when S4 =>

                    state <= S0;

            end case;

        end if;

    end process;

    -- change output based on state

    process (state)

    begin

        case state is

            when S0 =>

                output <= '1';

            when S1 =>

                output <= '0';

            when S2 =>

                output <= '1';

            when S3 =>

                output <= '0';

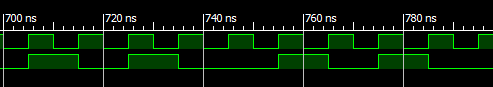
            when S4 =>

                output <= '0';

            end case;

    end process;

end Behavioral;

Test Bench: