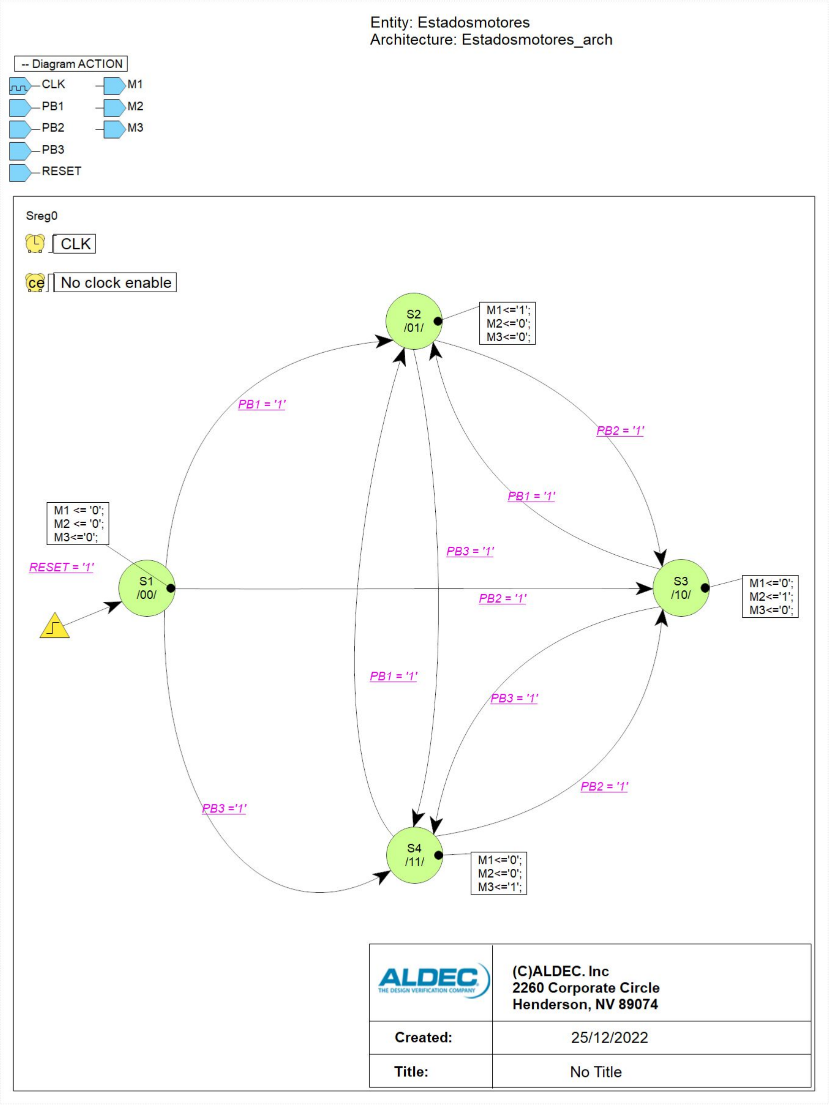
1 Estadosmotores.asf



1.1 Estadosmotores.vhd

```
-----------------------------------------------------------------------------
--
--
-- Title       : No Title
-- Design      :
-- Author      : albert0127
-- Company     : s
--
-----------------------------------------------------------------------------
--
--
-- File        : c:/Users/amf01/Documents/DLP/maquinaestadosmotores/compile/E
stadosmotores.vhd
-- Generated   : Sun Dec 25 14:51:27 2022
-- From        : C:/Users/amf01/Documents/DLP/maquinaestadosmotores/src/Estad
osmotores.asf
-- By          : Active-HDL Student Edition FSM Code Generator ver. 6.0
--
-----------------------------------------------------------------------------
--
--
-- Description :
--
-----------------------------------------------------------------------------
--

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_signed.all;

entity Estadosmotores is
    port (
        CLK: in STD_LOGIC;
        PB1: in STD_LOGIC;
        PB2: in STD_LOGIC;
        PB3: in STD_LOGIC;
        RESET: in STD_LOGIC;
        M1: out STD_LOGIC;
        M2: out STD_LOGIC;
        M3: out STD_LOGIC
);
end Estadosmotores;

architecture Estadosmotores_arch of Estadosmotores is

-- BINARY ENCODED state machine: Sreg0
attribute ENUM_ENCODING: string;
type Sreg0_type is (
    S1, S2, S3, S4
);
attribute ENUM_ENCODING of Sreg0_type: type is
    "00 " &     -- S1
    "01 " &     -- S2
    "10 " &     -- S3
    "11" ;      -- S4

signal Sreg0, NextState_Sreg0: Sreg0_type;

begin
```

```
-- FSM coverage pragmas
-- Aldec enum Machine_Sreg0 CURRENT=Sreg0
-- Aldec enum Machine_Sreg0 NEXT=NextState_Sreg0
-- Aldec enum Machine_Sreg0 INITIAL_STATE=S1
-- Aldec enum Machine_Sreg0 STATES=S2,S3,S4
-- Aldec enum Machine_Sreg0 TRANS=S1->S2,S1->S3,S1->S4,S2->S3,S2->S4,S3->S2,S
3->S4,S4->S2,S4->S3

-- User statements

-- Diagram ACTION

----------------------------------------------------------------------
-- Machine: Sreg0
----------------------------------------------------------------------
------------------------------------
-- Next State Logic (combinatorial)
------------------------------------
Sreg0_NextState: process (PB1, PB2, PB3, Sreg0)
begin
    NextState_Sreg0 <= Sreg0;
    -- Set default values for outputs and signals
    M1 <= '0';
    M2 <= '0';
    M3 <= '0';
    case Sreg0 is
        when S1 =>
            M1 <= '0';
            M2 <= '0';
            M3<='0';
            if PB1 = '1' then
                NextState_Sreg0 <= S2;
            elsif PB2 = '1' then
                NextState_Sreg0 <= S3;
            elsif PB3 ='1' then
                NextState_Sreg0 <= S4;
            end if;
        when S2 =>
            M1<='1';
            M2<='0';
            M3<='0';
            if PB2 = '1' then
                NextState_Sreg0 <= S3;
            elsif PB3 = '1' then
                NextState_Sreg0 <= S4;
            end if;
        when S3 =>
            M1<='0';
            M2<='1';
            M3<='0';
            if PB1 = '1' then
                NextState_Sreg0 <= S2;
            elsif PB3 = '1' then
                NextState_Sreg0 <= S4;
            end if;
        when S4 =>
            M1<='0';
            M2<='0';
            M3<='1';
            if PB2 = '1' then
                NextState_Sreg0 <= S3;
            elsif PB1 = '1' then
                NextState_Sreg0 <= S2;
            end if;
--vhdl_cover_off
        when others =>
            null;
--vhdl_cover_on
    end case;
end process;

------------------------------------
-- Current State Logic (sequential)
------------------------------------
Sreg0_CurrentState: process (CLK)
begin
    if CLK'event and CLK = '1' then
        if RESET = '1' then
            Sreg0 <= S1;
        else
            Sreg0 <= NextState_Sreg0;
        end if;
    end if;
end process;

end Estadosmotores_arch;
```