```vhdl
----------------------------------------------------------------------------------
-- Company: 
-- Engineer: 
-- 
-- Create Date: 10/12/2018 12:44:03 PM
-- Design Name: 
-- Module Name: mcdecoder - Behavioral
-- Project Name: Morse Code Decoder for Homework 1
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mcdecoder is
    Port ( dot : in STD_LOGIC;
           dash : in STD_LOGIC;
           lg : in STD_LOGIC;
           wg : in STD_LOGIC;
           valid : in STD_LOGIC;
           dout : out STD_LOGIC_VECTOR (7 downto 0);
           dvalid : out STD_LOGIC;
           error : out STD_LOGIC;
           clr : in STD_LOGIC;
           clk : in STD_LOGIC);
end mcdecoder;

architecture Behavioral of mcdecoder is

   --Use descriptive names for the states, like st1_reset, st2_search
   type state_type is (st1_<name_state>, st2_<name_state>, ...);
   signal state, next_state : state_type;

   --Declare other internal signals here
   signal <name>_i : std_logic;  -- example signal

begin

   --A clock process for state register
   proc_statereg: process (<clock>, clr)
   begin
      if (clr = '1') then
         -- jump to reset state here
         state <= <your_reset_state>;
      if (<clock>'event and <clock> = '1') then
         state <= next_state;
      end if;
   end process;
   --MEALY State-Machine - Outputs based on state and inputs
   proc_output: process (state, <input1>, <input2>, ...)
   begin
      --insert statements to decode internal output signals
      --below is simple example
      if (state = <name> and <input1> = '1') then
         <output> <= '1';
      else
         <output> <= '0';
      end if;
   end process;
   -- Next State Logic.  This corresponds to your next state logic table
   proc_ns: process (state, <input1>, <input2>, ...)
   begin
      --declare default state for next_state to avoid latches
      next_state <= state;  --default is to stay in current state
      --insert statements to decode next_state
      --below is a simple example
      case (state) is
         when st1_<name> =>
            if <input_1> = '1' then
               next_state <= st2_<name>;
            end if;
         when st2_<name> =>
            if <input_2> = '1' then
               next_state <= st3_<name>;
            end if;
         when st3_<name> =>
            next_state <= st1_<name>;
         when others =>
            next_state <= st1_<name>;
      end case;
   end process;

end Behavioral;
```