CNT_DPRAM

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cnt_dpram is
  port (
    CLK      : in  std_logic;
    RST      : in  std_logic;
    DIR      : in  std_logic_vector (7 downto 0);
    DIR_VLD  : in  std_logic;
    DATO     : in  std_logic_vector (7 downto 0);
    DATO_VLD : in  std_logic;
    ADDRESS  : out std_logic_vector(7 downto 0);
    DATA     : out std_logic_vector(7 downto 0);
    WE_DP1   : out std_logic;
    WE_DP2   : out std_logic);
end cnt_dpram;

architecture RTL of cnt_dpram is
  constant dir_dpram1 : std_logic_vector(7 downto 0) := x"A1";
  constant dir_dpram2 : std_logic_vector(7 downto 0) := x"A2";

  signal dir_ant : std_logic_vector(7 downto 0);
  signal CEB     : std_logic;
  signal REC : std_logic;
  signal CEC : std_logic;

  type MEF is (REP, ESP, ESW, ESD, RES, ESC);
  signal std_act, prox_std : MEF;

begin

  process (CLK, RST) is
  begin  -- process
    if RST = '1' then
      dir_ant <= (others => '0');
    elsif CLK'event and CLK = '1' then
      if CEB = '1' then
        dir_ant <= DIR;
      end if;
    end if;
  end process;

  process (CLK, RST, REC) is
    variable cnt : std_logic_vector(7 downto 0);
  begin  -- process
    if RST = '1' then
      ADDRESS <= (others => '0');
      cnt     := x"00";
    elsif REC='0' then
      ADDRESS <= (others => '0');
    cnt     :=  x"00";
  elsif CLK'event and CLK = '1' then
    if CEC = '1' then
      if cnt = x"FF" then
        cnt     := x"00";
        ADDRESS <= std_logic_vector(unsigned(cnt));
      else
        cnt     :=  std_logic_vector(unsigned(cnt)+1);
        ADDRESS <= std_logic_vector(unsigned(cnt));
      end if;
    end if;
  end if;
end process;

DATA <= DATO;

process (DIR_VLD, DATO_VLD, std_act, DIR, dir_ant) is
begin
  case std_act is
    when REP =>
      if DIR_VLD = '1' and (DIR = dir_dpram1 or DIR = dir_dpram2) then
        prox_std <= ESP;
      else
        prox_std <= REP;
      end if;
    when ESP =>
      if DATO_VLD = '1' then
        prox_std <= ESW;
      else
        prox_std <= ESP;
      end if;
    when ESW => prox_std <= ESC;
    when ESC => prox_std <= ESD;
    when ESD =>
      if DIR_VLD = '1' and DIR = dir_ant then
        prox_std <= ESP;
      elsif DIR_VLD = '1' and DIR /= dir_ant then
        prox_std <= RES;
      elsif DIR_VLD = '1' and DIR /= dir_dpram1 and DIR /= dir_dpram2 then
        prox_std <= REP;
      else
        prox_std <= ESD;
      end if;
    when RES => prox_std <= ESP;
  end case;
end process;

process (CLK, RST) is
begin  -- process
  if RST = '1' then
    std_act <= REP;
  elsif CLK'event and CLK = '1' then
```

```
      std_act <= prox_std;
    end if;
  end process;

  CEC <= '1' when std_act=ESC else '0';
  CEB <= '1' when std_act=ESP else '0';
  REC <= '0' when (std_act=RES or std_act=REP) else '1';
  WE_DP1 <= '1' when std_act = ESW and DIR = dir_dpram1 else '0';
  WE_DP2 <= '1' when std_act = ESW and DIR = dir_dpram2 else '0';

end RTL;
```