Ö5.1

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

entity E5_1 is
    port (
    r    : in  std_logic_vector(1 downto 0);
    clk  : in  std_logic;
    dout : out std_logic_vector(1 downto 0));
end E5_1;

architecture rtl of E5_1 is
signal cnt_int : std_logic_vector (1 downto 0) := "00";
begin
  process (r, clk, cnt_int)
  begin
    if (clk'event and clk = '1' and clk'last_value = '0') then
      case r is
        when "00" => cnt_int <= cnt_int;
        when "10" => cnt_int <= cnt_int +1;
        when "01" => cnt_int <= cnt_int -2;
        when "11" => cnt_int <= "00";
        when others => null;
      end case;
    end if;
  end process;

dout <= cnt_int;
end rtl;
```

Ö5.2

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.all;

entity D5_2 is
  port (
    clk, reset: in  std_logic;
    upper_limit, lower_limit, measured_value: in std_logic_vector(7 downto
0);
    count: buffer std_logic_vector(7 downto 0));
end D5_2;

architecture rtl of D5_2 is
  begin
  process (clk, reset)

  begin  -- process
    if reset='1' then
      count <= (others => '0');
    elsif clk'event and clk='1' then
      if (measured_value >= lower_limit) and (measured_value <=
upper_limit)  then
        count <= count +1;
      end if;
    end if;
  end process;
end rtl;
```

Ö5.3

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity E5_3 is
  port (
    clk, reset : in  std_logic;
    s          : out std_logic_vector(2 downto 0));
end E5_3;

architecture rtl of E5_3 is

  type direction_type is (up, down);

  signal count : intefger range 0 to 7;
  signal direction : direction_type;

  constant max_value : integer := 7;
  constant min_value : integer := 0;

begin
  process (clk, reset)
  begin  -- process
    if reset='1' then
      count <= 0;
      direction <= up;
    elsif clk'event and clk='1' then
      if count = max_value-1 then
        direction <= down;
      elsif count = min_value+1 then
        direction <= up;
        else
          direction <= direction;
      end if;

      if direction = up then
        count <= count +1;
      else
        count <= count -1;
      end if;
    end if;
  end process;

  output_logic : process (count)
  begin
    case count is
      when 0 => s <= "000";
      when 1 => s <= "001";
      when 2 => s <= "010";
      when 3 => s <= "011";
      when 4 => s <= "100";
      when 5 => s <= "101";
      when 6 => s <= "110";
      when 7 => s <= "111";
    end case;
  end process output_logic;
end rtl;
```