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

entity projeto01_MV is
   port (clock_27, KEY2, KEY3: in std_logic;
         SW: in std_logic_vector(7 downto 0);
         LEDR0,LEDR1: out std_logic;
         LHEX2,LHEX1,LHEX0,LHEX5,LHEX4: out std_logic_vector(6 downto 0));
end projeto01_MV;

architecture ckt of projeto01_MV is
   component maquinaVendas is
      port (CLK_mv, C, reset: in std_logic;
            S, A: in std_logic_vector(7 downto 0);
            D: out std_logic;
            saida_mv: out std_logic_vector(1 downto 0);
            Tot_mv: out std_logic_vector(7 downto 0));
   end component;
   component romMV IS
      PORT(address: IN STD_LOGIC_VECTOR (5 DOWNTO 0);
           clock: IN STD_LOGIC  := '1';
           q: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
   END component;
   component divisorClock is
      port ( clk_in : in std_logic ;
             clk_out : out std_logic );
   end component ;
   component contador6Bits is
      port ( clk_c6B,ld_c6B,clr_c6B: in std_logic;
             out_c6B: out std_logic_vector(5 downto 0));
   end component;
   component botaoSincrono is
      port (clk, b_in: in std_logic ;
            b_out: out std_logic );
   end component;
   component bitToBcdToD7Seg is
      port (SW_in: in std_logic_vector(7 downto 0);
            HEX0,HEX1,HEX2: out std_logic_vector(6 downto 0));
   end component;

   signal CLK_1, saida_botao, saida_MV, ld_reverso, clr_reverso: std_logic;
   signal saida_SS: std_logic_vector(1 downto 0);
   signal saida_contador: std_logic_vector(5 downto 0);
   signal entrada_A_mv, total_sum_mv: std_logic_vector(7 downto 0);
   signal saida_hex2, saida_hex1, saida_hex0, saida_hex5, saida_hex4,lixo:
std_logic_vector(6 downto 0);

   begin

      ld_reverso <= not KEY2;
      clr_reverso <= not KEY3;

      DivClock: divisorClock port map(
         clk_in => clock_27,
         clk_out => CLK_1);

      BS: botaoSincrono port map(
         clk => CLK_1,
         b_in => ld_reverso,
         b_out => saida_botao);

      ROM: romMV port map (
         clock => CLK_1,
         address => saida_contador,
         q => entrada_A_mv);

```

```vhdl
        Cont: contador6Bits port map(
           clk_c6B => CLK_1,
           ld_c6B => saida_botao,
           clr_c6B => clr_reverso,
           out_c6B => saida_contador);

        MV: maquinaVendas port map(
           CLK_mv => CLK_1,
           C => saida_botao,
           reset => clr_reverso,
           S => SW,
           A => entrada_A_mv,
           D => saida_MV,
           saida_mv => saida_SS,
           Tot_mv => total_sum_mv);

        LedTotalMV: bitToBcdToD7Seg port map(
           SW_in => total_sum_mv,
           HEX0 => saida_hex0,
           HEX1 => saida_hex1,
           HEX2 => saida_hex2);

        LedROM: bitToBcdToD7Seg port map(
           SW_in => entrada_A_mv,
           HEX0 => saida_hex4,
           HEX1 => saida_hex5,
           HEX2 => lixo);

        LEDR0 <= saida_MV;
        LEDR1 <= saida_botao;
        LHEX2 <= saida_hex2;
        LHEX1 <= saida_hex1;
        LHEX0 <= saida_hex0;
        LHEX4 <= saida_hex4;
        LHEX5 <= saida_hex5;
  end ckt ;
```

```vhdl
library ieee ;
use ieee.std_logic_1164.all;
entity divisorClock is
   port ( clk_in : in std_logic ;
          clk_out : out std_logic );
end divisorClock ;
architecture ckt of divisorClock is
   signal ax : std_logic ;
   begin
      process ( clk_in )
         variable cnt: integer range 0 to 13500000 := 0;
         begin
            if ( rising_edge ( clk_in )) then
               if (cnt =13500000) then
                  cnt := 0;
                  ax <= not ax;
               else
                  cnt := cnt +1;
               end if;
            end if;
      end process ;
      clk_out <= ax;
end ckt;
```

```vhdl
library ieee ;
use ieee.std_logic_1164.all;
entity botaoSincrono is
   port (clk, b_in: in std_logic ;
         b_out: out std_logic );
end botaoSincrono ;

architecture ckt of botaoSincrono is
   type state_type is (E1, E2, E3);
   signal y_present , y_next : state_type ;
   begin
      process (b_in, y_present )
      begin
         case y_present is
            when E1 =>
               if b_in = '0' then y_next <= E1;
               else y_next <= E2; end if;
            when E2 =>
               if b_in = '0' then y_next <= E1;
               else y_next <= E3; end if;
            when E3 =>
               if b_in = '0' then y_next <= E1;
               else y_next <= E3; end if;
         end case ;
      end process ;
      process (clk)
      begin
         if (clk'event and clk = '1') then
            y_present <= y_next;
         end if;
      end process;
      b_out <= '1' when y_present = E2 else '0';
end ckt;
```

```vhdl
-- megafunction wizard: %ROM: 1-PORT%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altsyncram

-- ============================================================
-- File Name: romMV.vhd
-- Megafunction Name(s):
--          altsyncram
--
-- Simulation Library Files(s):
--          altera_mf
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 13.0.0 Build 156 04/24/2013 SJ Web Edition
-- ************************************************************


--Copyright (C) 1991-2013 Altera Corporation
--Your use of Altera Corporation's design tools, logic functions
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Altera Program License
--Subscription Agreement, Altera MegaCore Function License
--Agreement, or other applicable license agreement, including,
--without limitation, that your use is for the sole purpose of
--programming logic devices manufactured by Altera and sold by
--Altera or its authorized distributors.  Please refer to the
--applicable agreement for further details.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY romMV IS
   PORT
   (
      address     : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
      clock    : IN STD_LOGIC  := '1';
      q     : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
   );
END romMV;


ARCHITECTURE SYN OF rommv IS

   SIGNAL sub_wire0  : STD_LOGIC_VECTOR (7 DOWNTO 0);



   COMPONENT altsyncram
   GENERIC (
      clock_enable_input_a    : STRING;
      clock_enable_output_a      : STRING;
      init_file      : STRING;
      intended_device_family     : STRING;
      lpm_hint    : STRING;
      lpm_type    : STRING;
      numwords_a     : NATURAL;
```

```vhdl
        operation_mode    : STRING;
        outdata_aclr_a    : STRING;
        outdata_reg_a     : STRING;
        widthad_a      : NATURAL;
        width_a     : NATURAL;
        width_byteena_a      : NATURAL
    );
    PORT (
            address_a   : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
            clock0   : IN STD_LOGIC ;
            q_a   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
    END COMPONENT;

BEGIN
    q    <= sub_wire0(7 DOWNTO 0);

    altsyncram_component : altsyncram
    GENERIC MAP (
        clock_enable_input_a => "BYPASS",
        clock_enable_output_a => "BYPASS",
        init_file => "romMV.mif",
        intended_device_family => "Cyclone II",
        lpm_hint => "ENABLE_RUNTIME_MOD=NO",
        lpm_type => "altsyncram",
        numwords_a => 64,
        operation_mode => "ROM",
        outdata_aclr_a => "NONE",
        outdata_reg_a => "UNREGISTERED",
        widthad_a => 6,
        width_a => 8,
        width_byteena_a => 1
    )
    PORT MAP (
        address_a => address,
        clock0 => clock,
        q_a => sub_wire0
    );



END SYN;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"
-- Retrieval info: PRIVATE: AclrAddr NUMERIC "0"
-- Retrieval info: PRIVATE: AclrByte NUMERIC "0"
-- Retrieval info: PRIVATE: AclrOutput NUMERIC "0"
-- Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"
-- Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"
-- Retrieval info: PRIVATE: BlankMemory NUMERIC "0"
-- Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"
-- Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"
-- Retrieval info: PRIVATE: Clken NUMERIC "0"
-- Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"
-- Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"
-- Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"
-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
-- Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "0"
-- Retrieval info: PRIVATE: JTAG_ID STRING "NONE"
-- Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"
-- Retrieval info: PRIVATE: MIFfilename STRING "romMV.mif"
-- Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "64"
-- Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"
-- Retrieval info: PRIVATE: RegAddr NUMERIC "1"
-- Retrieval info: PRIVATE: RegOutput NUMERIC "0"
-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
-- Retrieval info: PRIVATE: SingleClock NUMERIC "1"
-- Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"
-- Retrieval info: PRIVATE: WidthAddr NUMERIC "6"
-- Retrieval info: PRIVATE: WidthData NUMERIC "8"
-- Retrieval info: PRIVATE: rden NUMERIC "0"
-- Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
-- Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"
-- Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"
-- Retrieval info: CONSTANT: INIT_FILE STRING "romMV.mif"
-- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
-- Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=NO"
-- Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"
-- Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "64"
-- Retrieval info: CONSTANT: OPERATION_MODE STRING "ROM"
-- Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"
-- Retrieval info: CONSTANT: OUTDATA_REG_A STRING "UNREGISTERED"
-- Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "6"
-- Retrieval info: CONSTANT: WIDTH_A NUMERIC "8"
-- Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
-- Retrieval info: USED_PORT: address 0 0 6 0 INPUT NODEFVAL "address[5..0]"
-- Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
-- Retrieval info: USED_PORT: q 0 0 8 0 OUTPUT NODEFVAL "q[7..0]"
-- Retrieval info: CONNECT: @address_a 0 0 6 0 address 0 0 6 0
-- Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
-- Retrieval info: CONNECT: q 0 0 8 0 @q_a 0 0 8 0
-- Retrieval info: GEN_FILE: TYPE_NORMAL romMV.vhd TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL romMV.inc FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL romMV.cmp FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL romMV.bsf FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL romMV_inst.vhd FALSE
-- Retrieval info: LIB_FILE: altera_mf

```

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

entity maquinaVendas is
   port (CLK_mv, C, reset: in std_logic;
         S, A: in std_logic_vector(7 downto 0);
         D: out std_logic;
         saida_mv: out std_logic_vector(1 downto 0);
         Tot_mv: out std_logic_vector(7 downto 0));
end maquinaVendas;

architecture ckt of maquinaVendas is
   component blocoDeControleMV is
      port (clk_mv , rst_mv , c_mv, tot_ld: in std_logic ;
            d_mv, load_tot, clr_tot: out std_logic;
            saida: out std_logic_vector(1 downto 0));
   end component;
   component datapathMV is
      port (clk_dmv , clr_total , ld_total: in std_logic;
            S_mv, A_mv: in std_logic_vector(7 downto 0);
            total_ld: out std_logic;
            total_mv: out std_logic_vector(7 downto 0));
   end component;

   signal result_total: std_logic_vector(7 downto 0);
   signal result_comp_lt, result_despacho, sinal_somar_tot, clear_tot: std_logic;
   signal saida_mc: std_logic_vector(1 downto 0);
   begin
      BlocodeControle: blocoDeControleMV port map(
         clk_mv => CLK_mv,
         rst_mv => reset,
         c_mv => C,
         tot_ld => result_comp_lt,
         d_mv => result_despacho,
         load_tot => sinal_somar_tot,
         clr_tot => clear_tot,
         saida => saida_mc);

      Datapath: datapathMV port map(
         clk_dmv => CLK_mv,
         clr_total => clear_tot,
         ld_total => sinal_somar_tot,
         S_mv => S,
         A_mv => A,
         total_ld => result_comp_lt,
         total_mv => result_total);

      D <= result_despacho;
      Tot_mv <= result_total;
      saida_mv <= saida_mc;
end ckt ;
```

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

entity datapathMV is
   port (clk_dmv , clr_total , ld_total: in std_logic;
         S_mv, A_mv: in std_logic_vector(7 downto 0);
         total_ld: out std_logic;
         total_mv: out std_logic_vector(7 downto 0));
end datapathMV;

architecture ckt of datapathMV is
   component Comparador_8Bits is
     port (eA8,eB8: in std_logic_vector(7 downto 0);
           AeqB8,AltB8,AgtB8: out std_logic);
   end component;
   component reg8Bits is
      port (clk,preSet,clr,load: in std_logic;
            d: in std_logic_vector(7 downto 0);
            q : out std_logic_vector(7 downto 0));
   end component;
   component SUM_8Bits is
       port (A8_in,B8_in: in std_logic_vector(7 downto 0);
             C8_in: in std_logic;
             S8_out: out std_logic_vector(7 downto 0);
             C8_out: out std_logic);
   end component;

   signal result_somador, result_tot: std_logic_vector(7 downto 0);
   signal lixo_sum, result_comp_lt: std_logic;
   signal lixo_comp: std_logic_vector(1 downto 0);
   begin
      Reg: reg8Bits port map(
         clk => clk_dmv,
         preSet => '1',
         clr => not clr_total,
         load => ld_total,
         d => result_somador,
         q => result_tot);

      SUM: SUM_8Bits port map(
         A8_in => result_tot,
         B8_in => A_mv,
         C8_in => '0',
         S8_out => result_somador,
         C8_out => lixo_sum);

      Comp: Comparador_8Bits port map(
         eA8 => result_tot,
         eB8 => S_mv,
         AeqB8 => lixo_comp(1),
         AltB8 => result_comp_lt,
         AgtB8 =>lixo_comp(0));

      total_mv <= result_tot;
      total_ld <= result_comp_lt;
end ckt ;
```

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

entity blocoDeControleMV is
   port (clk_mv , rst_mv , c_mv, tot_ld: in std_logic ;
         d_mv, load_tot, clr_tot: out std_logic;
         saida: out std_logic_vector(1 downto 0));
end blocoDeControleMV;

architecture ckt of blocoDeControleMV is
   type st is (E1, E2, E3, E4);
   signal estado : st;
   begin
      process (clk_mv , rst_mv)
      begin
         if rst_mv = '1' then
            estado <= E1 ;
         elsif (clk_mv'event and clk_mv ='1') then
            case estado is
               when E1 =>
                  estado <= E2;
               when E2 =>
                  if c_mv='1' then estado <= E3;
                  elsif tot_ld='0' then estado <= E4;
                  else estado <= E2;
                  end if;
               when E3 =>
                  estado <= E2;
               when E4 =>
                  estado <= E1;
            end case ;
         end if;
      end process;
      d_mv <= '1' when estado = E4 else '0';
      load_tot <= '1' when estado = E3 else '0';
      clr_tot <= '1' when estado = E1 else '0';
      with estado select
         saida <= "00" when E1 ,
         "01" when E2 ,
         "11" when E4 ,
         "10" when E3 ;
end ckt ;
```

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

entity Comparador is
  port (in_gt, in_eq, in_lt,a,b: in std_logic;
        out_eq,out_lt,out_gt: out std_logic);
end Comparador;

architecture ckt of Comparador is
  begin
    out_gt <= in_gt OR (in_eq AND a AND (NOT b));
    out_lt <= in_lt OR (in_eq AND (NOT a) AND b);
    out_eq <= in_eq AND (a XNOR b);
end ckt;
```

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

entity Comparador_4Bits is
  port (eA,eB: in std_logic_vector(3 downto 0);
        gt,lt,eq: in std_logic;
        AeqB,AltB,AgtB: out std_logic);
end Comparador_4Bits;

architecture ckt of Comparador_4Bits is

  component Comparador is
    port (in_gt, in_eq, in_lt,a,b: in std_logic;
          out_eq,out_lt,out_gt: out std_logic);
   end component;

  signal saida_gt,saida_lt,saida_eq:std_logic_vector(3 downto 0);
  begin
  Comp1:Comparador port map(
        in_gt => gt,
        in_lt => lt,
        in_eq => eq,
        a => eA(3),
        b => eB(3),
        out_eq => saida_eq(3),
        out_gt => saida_gt(3),
        out_lt => saida_lt(3));

  Comp2:Comparador port map(
        in_gt => saida_gt(3),
        in_lt => saida_lt(3),
        in_eq => saida_eq(3),
        a => eA(2),
        b => eB(2),
        out_eq => saida_eq(2),
        out_gt => saida_gt(2),
        out_lt => saida_lt(2));

  Comp3:Comparador port map(
        in_gt => saida_gt(2),
        in_lt => saida_lt(2),
        in_eq => saida_eq(2),
        a => eA(1),
        b => eB(1),
        out_eq => saida_eq(1),
        out_gt => saida_gt(1),
        out_lt => saida_lt(1));

  Comp4:Comparador port map(
        in_gt => saida_gt(1),
        in_lt => saida_lt(1),
        in_eq => saida_eq(1),
        a => eA(0),
        b => eB(0),
        out_eq => saida_eq(0),
        out_gt => saida_gt(0),
        out_lt => saida_lt(0));

  AeqB <= saida_eq(0);
  AltB <= saida_lt(0);
  AgtB <= saida_gt(0);
end ckt;
```

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

entity Comparador_8Bits is
  port (eA8,eB8: in std_logic_vector(7 downto 0);
        AeqB8,AltB8,AgtB8: out std_logic);
end Comparador_8Bits;

architecture ckt of Comparador_8Bits is

  component Comparador_4Bits is
    port (eA,eB: in std_logic_vector(3 downto 0);
          gt,lt,eq: in std_logic;
          AeqB,AltB,AgtB: out std_logic);
   end component;

  signal saida_gt,saida_lt,saida_eq: std_logic_vector(1 downto 0);
  begin
      Comp1:Comparador_4Bits port map(
           gt => '0',
           lt => '0',
           eq => '1',
           eA(3 downto 0) => eA8(7 downto 4),
           eB(3 downto 0) => eB8(7 downto 4),
           AeqB => saida_eq(1),
           AgtB => saida_gt(1),
           AltB => saida_lt(1));

      Comp2: Comparador_4Bits port map(
           gt => saida_gt(1),
           lt => saida_lt(1),
           eq => saida_eq(1),
           eA(3 downto 0) => eA8(3 downto 0),
           eB(3 downto 0) => eB8(3 downto 0),
           AeqB => saida_eq(0),
           AgtB => saida_gt(0),
           AltB => saida_lt(0));

      AeqB8 <= saida_eq(0);
      AltB8 <= saida_lt(0);
      AgtB8 <= saida_gt(0);

end ckt;
```

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

entity reg8Bits is
   port (clk,preSet,clr,load: in std_logic;
   d: in std_logic_vector(7 downto 0);
   q : out std_logic_vector(7 downto 0));
end reg8Bits;
architecture ckt of reg8Bits is

   signal qs: std_logic_vector(7 downto 0);

   begin
      process (clk ,preSet,clr)
      begin
         if preSet = '0' then qs <= "11111111";
         elsif clr = '0' then qs <= "00000000";
         elsif clk ='1' and clk ' event then
            if load = '1' then
               qs <= d;
            end if;
         end if;
      end process ;
      q <= qs;
end ckt;
```

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

entity SUM_1Bit is
  port (A_in,B_in,C_in: in std_logic;
        S_out, C_out: out std_logic);
end SUM_1Bit;

Architecture ckt of SUM_1Bit is

Begin
  S_out <= ((B_in and ((C_in nor A_in) or (C_in and A_in))) or ((not B_in) and (((not C_in
) and A_in) or (C_in and (not A_in)))));
  C_out <= ((C_in and (A_in or B_in)) or (A_in and B_in));
end ckt;
```

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

entity SUM_2Bits is
  port (A2_in,B2_in: in std_logic_vector(1 downto 0);
        C2_in: in std_logic;
        S2_out: out std_logic_vector(1 downto 0);
        C2_out: out std_logic);
end SUM_2Bits;

architecture ckt of SUM_2Bits is
  component SUM_1Bit is
    port (A_in,B_in,C_in: in std_logic;
          S_out, C_out: out std_logic);
  end component;

  signal Sum_01_out : std_logic;

  begin
    SUM01: SUM_1Bit port map(
          A_in => A2_in(0),
          B_in => B2_in(0),
          C_in => C2_in,
          S_out => S2_out(0),
          C_out => Sum_01_out);

    SUM02: SUM_1Bit port map(
          A_in => A2_in(1),
          B_in => B2_in(1),
          C_in => Sum_01_out,
          S_out => S2_out(1),
          C_out => C2_out);

end ckt;
```

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

entity SUM_4Bits is
  port (A4_in,B4_in: in std_logic_vector(3 downto 0);
        C4_in: in std_logic;
        S4_out: out std_logic_vector(3 downto 0);
        C4_out: out std_logic);
end SUM_4Bits;

architecture ckt of SUM_4Bits is
  component SUM_2Bits is
    port (A2_in,B2_in: in std_logic_vector(1 downto 0);
          C2_in: in std_logic;
          S2_out: out std_logic_vector(1 downto 0);
          C2_out: out std_logic);
  end component;

  signal Sum_01_out : std_logic;

  begin
    SUM01: SUM_2Bits port map(
          A2_in => A4_in(1 downto 0),
          B2_in => B4_in(1 downto 0),
          C2_in => C4_in,
          S2_out => S4_out(1 downto 0),
          C2_out => Sum_01_out);

    SUM02: SUM_2Bits port map(
          A2_in => A4_in(3 downto 2),
          B2_in => B4_in(3 downto 2),
          C2_in => Sum_01_out,
          S2_out => S4_out(3 downto 2),
          C2_out => C4_out);

end ckt;
```

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

entity SUM_8Bits is
  port (A8_in,B8_in: in std_logic_vector(7 downto 0);
        C8_in: in std_logic;
        S8_out: out std_logic_vector(7 downto 0);
        C8_out: out std_logic);
end SUM_8Bits;

architecture ckt of SUM_8Bits is
  component SUM_4Bits is
    port (A4_in,B4_in: in std_logic_vector(3 downto 0);
          C4_in: in std_logic;
          S4_out: out std_logic_vector(3 downto 0);
          C4_out: out std_logic);
  end component;

  signal Sum_01_out : std_logic;

  begin
    SUM01: SUM_4Bits port map(
          A4_in => A8_in(3 downto 0),
          B4_in => B8_in(3 downto 0),
          C4_in => C8_in,
          S4_out => S8_out(3 downto 0),
          C4_out => Sum_01_out);

    SUM02: SUM_4Bits port map(
          A4_in => A8_in(7 downto 4),
          B4_in => B8_in(7 downto 4),
          C4_in => Sum_01_out,
          S4_out => S8_out(7 downto 4),
          C4_out => C8_out);

end ckt;
```

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

entity ciBitToBcd is
  port (BtB_in: in bit_vector(3 downto 0);
        BtB_out: out bit_vector(3 downto 0));
end ciBitToBcd;

architecture ckt of ciBitToBcd is
begin
  BtB_out(3) <= (BtB_in(3) or (BtB_in(2) and (BtB_in(1) or BtB_in(0))));
  BtB_out(2)<= ((BtB_in(2) and (not BtB_in(1)) and (not BtB_in(0))) or (BtB_in(3) and 
BtB_in(0)));
  BtB_out(1)<= ((BtB_in(3) and (not BtB_in(0))) or (BtB_in(1) and ((not BtB_in(2)) or 
BtB_in(0))));
  BtB_out(0)<= (((not BtB_in(3)) and (not BtB_in(2)) and BtB_in(0)) or ((not BtB_in(0)) 
and (BtB_in(3) or (BtB_in(2) and BtB_in(1)))));
end ckt;

```

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

entity bitToBcd is
  port (bit_in: in std_logic_vector(7 downto 0);
        bcd_out: out std_logic_vector(11 downto 0));
end bitToBcd ;

architecture ckt of bitToBcd is
  component ciBitToBcd is
    port (BtB_in: in std_logic_vector(3 downto 0);
          BtB_out: out std_logic_vector(3 downto 0));
  end component;

  signal ciBtB_01_out, ciBtB_02_out, ciBtB_03_out, ciBtB_04_out, ciBtB_05_out,
ciBtB_06_out, ciBtB_07_out: std_logic_vector(3 downto 0);

  begin
      ciBtB01: ciBitToBcd port map(
           BtB_in(3) => '0',
           BtB_in(2 downto 0) => bit_in(7 downto 5),
           BtB_out => ciBtB_01_out);

      ciBtB02: ciBitToBcd port map(
           BtB_in(3 downto 1) => ciBtB_01_out(2 downto 0),
           BtB_in(0) => bit_in(4),
           BtB_out => ciBtB_02_out);

      ciBtB03: ciBitToBcd port map(
           BtB_in(3 downto 1) => ciBtB_02_out(2 downto 0),
           BtB_in(0) => bit_in(3),
           BtB_out => ciBtB_03_out);

      ciBtB04: ciBitToBcd port map(
           BtB_in(3) => '0',
           BtB_in(2) => ciBtB_01_out(3),
           BtB_in(1) => ciBtB_02_out(3),
           BtB_in(0) => ciBtB_03_out(3),
           BtB_out => ciBtB_04_out);

      ciBtB05: ciBitToBcd port map(
           BtB_in(3 downto 1)=> ciBtB_03_out(2 downto 0),
           BtB_in(0) => bit_in(2),
           BtB_out => ciBtB_05_out);

      ciBtB06: ciBitToBcd port map(
           BtB_in(3 downto 1)=> ciBtB_04_out(2 downto 0),
           BtB_in(0) => ciBtB_05_out(3),
           BtB_out => ciBtB_06_out);

      ciBtB07: ciBitToBcd port map(
           BtB_in(3 downto 1)=> ciBtB_05_out(2 downto 0),
           BtB_in(0) => bit_in(1),
           BtB_out => ciBtB_07_out);

      bcd_out(11) <= '0';
      bcd_out(10) <= '0';
      bcd_out(9) <= ciBtB_04_out(3);
      bcd_out(8 downto 5) <= ciBtB_06_out;
      bcd_out(4 downto 1) <= ciBtB_07_out;
      bcd_out(0) <= bit_in(0);

end ckt;
```

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

entity d7Seg is
  port(S_in: in std_logic_vector(3 downto 0);
       D_out: out std_logic_vector(6 downto 0));
end d7Seg;

architecture display of d7Seg is
  begin
    D_out(0) <= not ((S_in(3) or S_in(1) or (S_in(2) and S_in(0)) or (S_in(2) nor S_in(0
))));

    D_out(1) <= not (((not S_in(2)) or (S_in(1) nor S_in(0)) or (S_in(1) and S_in(0))));

    D_out(2) <= not (((not(S_in(3)) and ((S_in(1) and S_in(0)) or S_in(2))) or (not(S_in(2
) or S_in(1)))));

    D_out(3) <= not((S_in(3) or (S_in(1) and (S_in(2) nand S_in(0))) or (S_in(2) nor S_in(
0)) or (S_in(2) and (not S_in(1)) and S_in(0))));

    D_out(4) <= not (((S_in(1) and (not S_in(0))) or (S_in(2) nor S_in(0))));

    D_out(5) <= not ((S_in(3) or (S_in(1) nor S_in(0)) or (S_in(2) and (S_in(1) nand S_in(
0)))));

    D_out(6) <= not ((S_in(3) or (S_in(1) and (not S_in(2))) or (S_in(2) and (S_in(1) nand
 S_in(0)))));

end display;
```

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

entity bitToBcdToD7Seg is
  port (SW_in: in std_logic_vector(7 downto 0);
        HEX0,HEX1,HEX2: out std_logic_vector(6 downto 0));
end bitToBcdToD7Seg ;

architecture ckt of bitToBcdToD7Seg is
  component bitToBcd is
    port (bit_in: in std_logic_vector(7 downto 0);
          bcd_out: out std_logic_vector(11 downto 0));
  end component;
  component d7Seg is
     port(S_in: in std_logic_vector(3 downto 0);
          D_out: out std_logic_vector(6 downto 0));
  end component;

  signal bcd_ax_out: std_logic_vector(11 downto 0);
  signal hex0_ax_out,hex1_ax_out,hex2_ax_out : std_logic_vector(6 downto 0);

  begin
      BtB: BitToBcd port map(
           bit_in => SW_in,
           bcd_out => bcd_ax_out);

      D7S2: d7Seg port map(
            S_in => bcd_ax_out(11 downto 8),
            D_out => hex2_ax_out);

      D7S1: d7Seg port map(
            S_in => bcd_ax_out(7 downto 4),
            D_out => hex1_ax_out);

      D7S0: d7Seg port map(
            S_in => bcd_ax_out(3 downto 0),
            D_out => hex0_ax_out);
      HEX0 <= hex0_ax_out;
      HEX1 <= hex1_ax_out;
      HEX2 <= hex2_ax_out;
end ckt;
```

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

ENTITY ffjk IS
  port ( clk ,J ,K ,P ,C : in std_logic;
          q : out std_logic );
END ffjk ;
ARCHITECTURE ckt OF ffjk IS
SIGNAL qS : std_logic;
BEGIN
  PROCESS ( clk ,P ,C )
  BEGIN
    IF P = '0' THEN qS <= '1';
    ELSIF C ='0' THEN qS <= '0';
    ELSIF clk = '1' AND clk ' EVENT THEN
      IF J = '1' AND K = '1' THEN qS <= NOT qS ;
      ELSIF J = '1' AND K = '0' THEN qS <= '1';
      ELSIF J = '0' AND K = '1' THEN qS <= '0';
      END IF;
    END IF;
  END PROCESS ;
q <= qS ;
END ckt ;
```

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

entity contador6Bits is
  port ( clk_c6B,ld_c6B,clr_c6B: in std_logic;
         out_c6B: out std_logic_vector(5 downto 0));
end contador6Bits ;
architecture ckt of contador6Bits is
   component ffjk is
      port (clk ,J ,K ,P ,C: in std_logic;
            q: out std_logic);
   end component;

   signal q0_out,q1_out,q2_out,q3_out,q4_out,q5_out, clr_reverso: std_logic;
   signal resp_and: std_logic_vector(4 downto 0);

   begin

      clr_reverso <= not clr_c6B;
      Q0: ffjk port map(
           clk => clk_c6B,
           J => ld_c6B,
           K => ld_c6B,
           P => '1',
           C => clr_reverso,
           q => q0_out);

      resp_and(0) <= q0_out and ld_c6B;

      Q1: ffjk port map(
           clk => clk_c6B,
           J => resp_and(0),
           K => resp_and(0),
           P => '1',
           C => clr_reverso,
           q => q1_out);

      resp_and(1) <= q1_out and resp_and(0);

      Q2: ffjk port map(
           clk => clk_c6B,
           J => resp_and(1),
           K => resp_and(1),
           P => '1',
           C => clr_reverso,
           q => q2_out);

      resp_and(2) <= q2_out and resp_and(1);

      Q3: ffjk port map(
           clk => clk_c6B,
           J => resp_and(2),
           K => resp_and(2),
           P => '1',
           C => clr_reverso,
           q => q3_out);

      resp_and(3) <= q3_out and resp_and(2);

      Q4: ffjk port map(
           clk => clk_c6B,
           J => resp_and(3),
           K => resp_and(3),
           P => '1',
           C => clr_reverso,
           q => q4_out);
```

```vhdl

        resp_and(4) <= q4_out and resp_and(3);

        Q5: ffjk port map(
             clk => clk_c6B,
             J => resp_and(4),
             K => resp_and(4),
             P => '1',
             C => clr_reverso,
             q => q5_out);

        out_c6B(0) <= q0_out;
        out_c6B(1) <= q1_out;
        out_c6B(2) <= q2_out;
        out_c6B(3) <= q3_out;
        out_c6B(4) <= q4_out;
        out_c6B(5) <= q5_out;


end ckt ;
```